CONTIN (Version 2) Users Manual

(March, 1984)

Part 1

Technical Report EMBL-DA07 (March 1984)

Part 1: Users Manual

Part 2: Output from Test Runs

Users manual for CONTIN — A portable Fortran IV program for the regularized solution of linear algebraic and linear integral equations of the first kind, with options for linear equality and inequality constraints.

Stephen W. Provencher

sp@S-provencher.COM

http://S-provencher.COM

This manual was typeset with *D. E. Knuth*'s text formatting system TEX.

[Knuth, D.E., *The TEXbook*, (Addison-Wesley Publishing Company, Reading, Massachusetts, 1984)]

TABLE OF UPDATES

0.	AUG 82	Original version
1.	MAR 84	Hollow sphere form factors corrected Output of (STD. DEV.)/MEAN for each peak Diagnostics improved
2.	APR 91	Splice Applications Package

TABLE OF CONTENTS

MAR 84

PAGE UPDATE SUMMARY

All pages are the original version, AUG 82, except where changes are marked in the left margin with vertical bars (|). Deletions from the AUG 82 version are marked in the left margin with a small diamond (◇).

Update 2 (April 1991)

1. PREFACE AND OVERVIEW

1.1 Purpose of CONTIN and Related Programs

CONTIN uses a generalized version of the algorithm in [1], and it is described in detail in [2, 19]. It is a general purpose program for solving linear integral equations of the first kind and systems of (possibly ill-conditioned) linear algebraic equations. It is called CONTIN because it is often applied to solving integral equations of the first kind for effectively CONTINuous distributions of diffusion coefficients [1], molecular weights [3], relaxation times [4], electron densities [5], etc. When Laplace's integral equation is being solved, it is often useful [1, 3, 4] to compare the solution with that of DISCRETE [6-8], which analyzes the data as a discrete sum (rather than an integral over a continuous distribution) of exponentials. To date, we have sent out over 200 copies of DISCRETE. We also intend to distribute (hopefully in 1984) SPLMOD, a program for analyzing sums of general one-parameter functions (e.g., convoluted exponentials as in fluorescence decay) [6].

1.2 Distribution of CONTIN

The main source of CONTIN is now the Computer Physics Communications (CPC) Program Library, Dept. of Applied Mathematics and Theoretical Physics, Queen's University of Belfast, BT7 INN, Northern Ireland. Order forms are available in any issue of CPC. The program is also available from me (for non-profit institutions only) and will probably be available from the NEA Data Bank and the Quantum Chemistry Program Exchange. At present, the Applications Packages (described in Sec 6) are only available from me.

As a new user, you should not attempt to install a second-hand version of CONTIN at a new computer center. You should obtain the original Version 2 from me. This will protect you against altered versions. In any case, as a new user you should send me the date in the heading of the output, so that any outdated versions can be replaced and your name can be put on a mailing list for possible updates, etc.

1.3 For Users of previous versions

1.3.1 Corrections to Version 1

Three corrections should be made to the code of Version 1:

(1) Remove line `ANLYZ 75`. (This could have led to an error if `NEQ=NORDER=0` and `NLINF>1`.)

(2) Change line `STRIN 23` from `IFINT(RIN)=INT(RIN*1.001)` to `IFINT(RIN)=INT(RIN+SIGN(.5,RIN))`. (This will prevent large input values for integers from being incorrectly converted.)

(3) Change line `SEGRD 64` from `GO TO 500` to `GO TO 800`. (This will prevent a serious error in the quadrature weights from occurring when `IGRID=IQUAD=2`.)

1.3.2 Differences between Versions 1 and 2

With the corrections in Sec 1.3.1, Version 1 can still be used. However, Version 2 offers the following advantages:

(1) automatic internal scaling to increase numerical stability and avoid overflow, especially in the computation of moments. This is especially useful when `NLINF>0`, since the complications described in Sec 3.6.5.7 of the Version 1 Users Manual do not occur and the precautions described there are not necessary. For users of the CD package, this is unimportant.

(2) A much more efficient search of the α-grid is automatically made, and `NQPROG` and `RSVMNX` usually can be left at their default values.

(3) Error estimates are made [2].

(4) More control over the amount and detail of the output is available.

(5) `USEREX`, `USERIN`, `USERK`, `USEROU`, and `USERSI` have been generalized for more convenience, especially for photon correlation.

The input format [19] has purposely been changed to prevent data sets for Version 1 from being accidentally used with Version 2. For the Control Variables, all values are input in the `E15.6` field (including `+1.` for `.TRUE.` and `-1.` for `.FALSE.`) and the `I8` field has been reduced to `I5`. The actions of `RSVMNX`, `IPRINT`, `IPLRES`, and elements of `RUSER`, `IUSER`, `LUSER` (in `USEREX`, `USERIN`, `USERK`, and `USERSI`) have been slightly changed. The Control Variables `DFMIN`, `DOCHOS`, `IUSROU`, `MPKMOM`, and `NEWPG1` have been added, and `DOUSOU` has been deleted.

1.3.3 Differences between Version 2 (AUG 82) and (MAR 84)

The only serious error in Version 2 (AUG 82) that has been discovered and corrected in (MAR 84) affects only those users of the photon correlation package who are using the (erroneous) hollow sphere form factors. This and a few additional minor improvements are described in CONTIN Update 1 (MAR 84). However this manual is only consistent with Version 2 (MAR 84), and, as a new user, you should implement this version. Users of Version 2 (AUG 82) need only implement the (MAR 84) version if they are using the hollow sphere form factors for light scattering or have encountered any of the other rare problems described in Update 1. You should use the corresponding Users Manual, however.

1.4 Design of this Users Manual

1.4.1 What you must read

This Users Manual has also been designed to offer ease of use for users interested only in a single application and more detailed information for those who want to explore various new applications and approaches. Sections 1 and 3 are essential reading for everyone, except for users of the CD Package (Sec 6.5), who can also skip Sec 3. If CONTIN has already been installed on your computer, then you can skip Sec 2. If you are only interested in an Applications Package, then, after reading Secs 1 and 3, you can jump to the description of the Applications Package in Sec 6, where you will be referred to any parts of Sec 4 that are necessary reading. I have attempted to make the subsections in Sec 4 self-contained enough to be read separately. Very few users will have to read all of Sec 4. However, you should at least read the Table of Contents to become aware of all the possibilities of CONTIN.

Section 5 is only for when you have difficulties. Section 6 is only necessary if you are using one of the Applications Packages. However, Sec 6 provides useful examples of how you can utilize the possibilities of CONTIN, and scanning it may give you new ideas on how to solve your own problems with CONTIN. Section 7 is the list of references. This is far from complete and is mainly to provide examples of applications of CONTIN. However some of the publications referenced in Sec 7 have useful bibliographies.

Section 8 contains useful summaries and indexes of important equations, Control Variables, and other terms. To save space, each term is not cross-referenced every time it appears in the text. Therefore, when you see a term for the first time, you should simply use Sec 8 to find out where it is defined.

I have attempted to write this manual so that only a minimal knowledge of mathematics and computers is required for its use, even at the risk of using very elementary terminology and notation. A major part of [2] involves heuristic discussions of what ill-posed problems are and how to interpret the results from CONTIN. This is much more important than with most program packages, because these concepts are essential and may be new to many users. This is particularly so in fields like molecular biology, where many ill-posed problems have not been recognized as such and have been traditionally handled with unstable and subjective inversion procedures.

References [2] and [19] are strongly recommended reading. You should at least read the PROGRAM SUMMARY of [19] and the abstract of [2] now. You should also read Sec 1 of [2] now, unless you are only using the CD package. You will be referred throughout this manual to other parts of [2] and [19]. It is not necessary to read the appendix of [2].

1.4.2 If questions or problems arise

The great flexibility of CONTIN could have two consequences:

(1) It may not be apparent to you how to best set up the problem or what combination of options is best or even if your problem can be handled by CONTIN. If such questions arise, please contact me; we are interested in new applications of CONTIN.

(2) We will probably never be able to test CONTIN with all possible combinations of options and obviously not with all possible USER subprograms. Difficulties may arise for certain combinations, either due to your improper usage or to errors in CONTIN. If you still have not resolved the difficulties after trying the suggestions in Sec 5, then send me the information specified in Sec 5.2. This will help us maintain CONTIN.

1.4.3 Conventions and notation

Fortran variables (e.g., NY) and terms and phrases output by CONTIN (e.g., LINEAR COEFFICIENTS) are printed in all capitals with a teletype font in this manual. Some terms with special meanings in this manual are capitalized; the first time that they are defined they are also italicized. The definitions of the Fortran Control Variables in Table 4 of [19] are set in boxes. The locations of the definitions of many terms can be found in Sec 8.

The term "*Page*" refers exclusively to the hand-numbered pages of the printed output from test runs in Part 2 of this manual. All other references to locations in this manual are to section and subsection, not page.

"Eq (3.2.2-1)" refers to the equation numbered (1) in Sec 3.2.2. The term "Eq (1)" refers to the equation numbered (1) in the same section that the term "Eq (1)" occurs.

"[2]" refers to the reference numbered 2 in Sec 7.

In CONTIN itself, comment cards set off with cards with "C" in column 1 and "*" in columns 2-72 are messages of possible interest to you. In the USER subprograms, all comments are for you. The rest of the comments are mainly for me. They will only give you a rough idea of what is going on; e.g., arguments of the subprograms are usually not redefined in the comments of every subprogram. It is only intended that you be able to optionally modify the main, BLOCK DATA, and USER subprograms.

1.5 Acknowledgements

CONTIN has been under development and testing since early 1976. I thank the users at EMBL and elsewhere; their problems stimulated many extensions and added options. I am especially grateful to Jürgen Glöckner for much help in testing and preparing CONTIN and this manual for distribution and to Robert Vogel and Leo De Maeyer for many useful discussions.

2. INSTALLING CONTIN ON YOUR COMPUTER

2.1 Reading the Magnetic Tape

2.1.1 Tape format

All tapes are nine-track, unlabeled, odd-parity, 1600-bits-per-inch with 80 characters per logical record (i.e., Fortran card images) and 6 records (480 characters) per block. The tape reels are marked to show whether the tapes are in ASCII or EBCDIC code. When requesting a tape, you should specify one of these codes. Do not send us a blank tape of your own. We have written a large number of tapes (once and for all) and we can send you one that you can copy and return.

2.1.2 Tape contents

The files are ordered on the tape as follows:

File 1	CONTIN Version 2SP
File 2	» » 2DP
File 3	Application Package 1, Version 2SP
File 4	» » » » 2DP
File 5	Test data for Application Package 1 (for both versions)
File 6	Application Package 2, Version 2SP
etc.	

Since we intend to continually add Application Packages, a table of tape contents will be sent with each tape. You always need either File 1 or File 2 (see Sec 2.2 to decide which). In addition, if you are planning to use an Application Package, then you will need an Application Package file and its test data.

It is a good idea to make a copy of the entire tape. Versions 2SP and 2DP are identical except that comment cards have been changed to Fortran statements and vice versa and the markers "!SP" and "!DP" have been inserted and deleted. Thus the pairs of files can be used to check each other in case copying errors are suspected. More important, you may decide later to use the other version. The Application Packages can provide useful examples when you are modifying the USER subprograms for your own application. (Sometimes the "!" is translated as a "]", but "!" only occurs in COMMENT cards.)

2.1.3 File contents

Files 1 and 2 are in the same format as AAOC and AAOB of the CPC Program

library. That is, they are complete compile-load-go files for the *VAX* with control cards, Fortran source, and test data [19]. The Fortran subprograms are ordered as follows: main subprogram, BLOCK DATA, 13 USER subprograms, and the other 51 subprograms. The USER and other subprograms are in alphabetical order. The 5883 "cards" are numbered sequentially in columns 77–80. The contents of the files for the Applications Packages are given in Sec 6.

The last card of each subprogram has the characters "END" in columns 7–9 and blanks in columns 1–6 and 10–72. This card only comes at the end of a subprogram. The first card in each subprogram also has a unique string that makes it easy to separate the subprograms: it has the characters "C++++" in columns 1–5. There are no separator cards between subprograms; the last card of one is followed immediately by the first card of the next.

The first card also has "VERSION 2DP (MAR 1984)" (or with DP replaced by SP); if your File 1 or File 2 has first cards with a date other than MAR 1984, then either your copy of CONTIN or this Users Manual is outdated. Send me the date on your first cards and a photocopy of the Table of Updates at the front of this manual. Note that this date does not apply to subprograms from applications packages. Their first cards have in addition the characters "PACKAGE", and their required dates are given in Sec 6.

2.2 Version 2SP or 2DP?

I have attempted to use numerically stable algorithms [9]. Nevertheless, the single-precision Version 2SP is only recommended for general use on machines (e.g., *CDC 7600*) that have single-precision words (i.e., Fortran REAL variables) of 60 bits or more. Otherwise, Version 2DP, which has key parts in double precision, should be used.

2.3 Possible necessary changes to CONTIN

I believe that CONTIN is in *1966 ANSI* standard Fortran IV with the exception that some dummy arrays have a "1" in their rightmost DIMENSION specification. This is a popular exception even with libraries that are intended to be portable. Therefore, it is intended that CONTIN can be run with no changes, except for the machine-dependent changes listed below and the problem-dependent changes described in Sec 3.5. This has been the case with test runs on the *DEC 1090* at Max-Planck-Institut für Kernphysik in Heidelberg and on the *Cray-1*, *IBM 360/91*, and *Amdahl 470* at Max-Planck-Institut für Plasma-

physik in Garching bei München using the G, H, and WATFIV compilers, as well as the *VAX 11/780* at EMBL.

2.3.1 Set the four machine-dependent parameters

These must be set (once and for all) in the DATA statement in Line 206 (in Version 2SP) or Line 207 (in Version 2DP) in the BLOCK DATA subprogram. See Sec 3.1 of [19].

2.3.2 Underflow

In many parts of CONTIN, underflows are expected to occur and be replaced by zero, which is the normal action of most compilers. However, if you happen to have a compiler that stops execution at underflows or prints a diagnostic at every underflow, no matter how many, then you will have to prevent this, e.g., by using a compilation option or switch or calling the WATFIV subprogram:

```
CALL TRAPS (0,0,99999,0,0).
```

Subprogram USERK (Lines 671–674, 787, 843) illustrates how you can avoid underflow in the library function EXP; this is actually necessary with a few compilers.

2.3.3 WATFIV

WATFIV was useful in testing CONTIN, but is extremely slow in execution and not recommended for general usage. If you nevertheless want to test CONTIN with WATFIV, you may have to make further changes. In addition to the CALL TRAPS in Sec 2.3.2, some versions of WATFIV require that even intrinsic DOUBLE PRECISION functions be explicitly typed. You can do this by simply inserting the statement

```
DOUBLE PRECISION DABS, DMAX1, DSIN, DSQRT, DBLE, DMOD
```

in every subprogram.

2.3.4 Other Possible changes

There may be other required changes peculiar to your system. For example, you may be required to open your own input, output, and (only if you input IUNIT ≥ 0) scratch files. Instructions on where to do this are given in the main subprogram in Lines 108–116.

Other possible problem-dependent changes are discussed elsewhere: adjusting DIMENSION specifications (Sec 3.5), the default values of the Control Variables (Sec 3.4), the array IAPACK (Sec 3.2 of [19]), or the USER subprograms (Sec 4.1). However, after making the changes discussed here in Sec 2.3, you should be ready to compile and run the test data.

2.4 Run with the Test Data

Compilation on the *DEC 1090*, *VAX 11/780*, and with the G, H, and WATFIV compilers on the *IBM/Amdahl* produced no diagnostics, except for 32 WATFIV extensions due to a "1" in the rightmost DIMENSION specification of dummy arrays.

The test run using Version 2DP with the two test data sets in File 2 took a total of about 360 CPU seconds (not counting compilation and loading) on the *VAX 11/780*.

Selected output from the test run on the *VAX 11/780* is shown in Part 2 of this manual and in [19]. However, before you compare your output too closely with this, you should read Sec 6 of [19].

Optimizing compilers are the weakest parts of many systems. If the test run aborts with a seemingly meaningless diagnostic and you have compiled with an optimization switch on (often the default setting), see point (8) of Sec 5.2.

3. ESSENTIAL INFORMATION ON USING CONTIN

3.1 Outline of Problems that CONTIN Can Handle

CONTIN analyzes data that can be represented (or approximated) by a set of linear equations:

$$y_k \approx \sum_{j=1}^{N_x} A_{kj} x_j\,, \quad k = 1, \ldots, N_y \tag{1}$$

where the data y_k generally contain experimental noise (and hence the $\approx$), the A_{kj} are known, and the x_j are to be estimated. CONTIN can handle cases where the equations are ill-conditioned or linearly dependent or even where there are more unknowns than equations (i.e., $N_x > N_y$).

This includes Fredholm (and Volterra) integral equations of the first kind:

$$y_k \approx \int_a^b K(g, t_k)\, s(g)\, dg + \sum_{i=1}^{N_L} \beta_i L_i(t_k)\,, \quad k = 1, \ldots, N_y \tag{2}$$

where the function $K(g,t)$ and the values of the independent variable, t_k, are known, and $s(g)$ is to be estimated. The optional sum over the N_L known $L_i(t)$ and unknown β_i permits, for example, an additional constant term, β_1, to be included by setting $N_L = 1$ and $L_1(t) = 1$. CONTIN has options to automatically convert Eq (2) to Eq (1) using the trapezoidal or Simpson's rule:

$$y_k \approx \sum_{m=1}^{N_g} c_m K(g_m, t_k)\, s(g_m) + \sum_{i=1}^{N_L} \beta_i L_i(t_k)\,, \quad k = 1, \ldots, N_y \tag{3}$$

where c_m are the coefficients in the quadrature (i.e., numerical integration) formula. The solution will then be $N_x = N_g + N_L$ values of x_j: the N_g values of the $s(g_m)$ and the N_L values of the β_i. Thus the solution $s(g)$ is represented by its values at the N_g grid points g_m. There are many optional control variables and USER subprograms that allow you, for example, to specify the grid points, another quadrature formula, or a continuous (e.g., spline) representation of $s(g)$.

The most important options allow you to use *a priori* knowledge by imposing linear inequality and equality constraints on the solution x_j:

$$\sum_{j=1}^{N_x} D_{ij} x_j \geq d_i\,, \quad i = 1, \ldots, N_{ineq} \tag{4}$$

$$\sum_{j=1}^{N_x} E_{ij} x_j = e_i\,, \quad i = 1, \ldots, N_{eq} \tag{5}$$

where the arrays D, E, d and e are specified by you. This allows you, for example, to constrain the solution to be nonnegative.

There are other options that limit the number of extrema in the sequence $s(g_m)$, $m = 1, \ldots, N_g$. This can help to answer questions like "Does there exist a unimodal solution that is consistent with the data or is at least a bimodal solution necessary?" These *Peak Constrained* solutions are described in Sec 4.1.6.4.

Options to statistically weight the data in the regularized least-squares analysis of Eq (1) are described in Sec 4.1.2.7. Options to generate simulated data with pseudorandom noise are described in Sec 4.1.6.2.

There are many other options and USER subprograms described in Secs 3.4 and 4.1. You can get some idea of the possibilities by scanning the Table of Contents.

3.2 Outline of Methods Used

I will only give a brief and somewhat heuristic outline to enable you to use CONTIN; see [2] and the references therein for more complete discussions.

3.2.1 Ill-Posed problems

Solving Eq (3.1-2) is generally an ill-posed problem. This means that, even for arbitrarily small (but nonzero) noise levels in the y_k, there still exists a large (typically infinite) set of solutions $s(g)$ that all fit the y_k in Eq (3.1-2) to within the noise level. Even worse, solutions in this set can differ from each other by arbitrarily large amounts; i.e., the errors in the solutions are unbounded. Similarly, when Eq (3.1-2) is approximated by Eq (3.1-1), there is generally a large set (call it S) of very different solutions that all satisfy Eq (3.1-1) to within the noise levels of the y_k.

One member of S is the ordinary least squares solution of Eq (3.1-1), i.e., the set of x_j that satisfies

$$\text{VAR} \equiv \sum_{k=1}^{N_y} w_k \left(y_k - \sum_{j=1}^{N_x} A_{kj} x_j \right)^2 = \text{minimum} \tag{1}$$

where the w_k are optional weights that you can assign. However, there is no reason to prefer this solution over any other members of S. In fact, it is extremely unlikely that this solution (or any other randomly chosen one) will be close to the true solution; on the contrary, in view of the unboundedness of the errors, it is very likely that this solution will be a very poor estimate.

3.2.2 Constrained regularization

CONTIN computes a *constrained regularized solution*, which is the set of x_j that satisfies

$$\text{VAR} + \alpha^2 \sum_{i=1}^{N_{reg}} \left(r_i - \sum_{j=1}^{N_x} R_{ij} x_j \right)^2 = \text{minimum} \qquad (1)$$

subject to the constraints in Eqs (3.1–4) and (3.1–5). The term added to VAR is called the *regularizor*. Its form is determined by specifying the arrays r and R; its strength is determined by specifying α, the *regularization parameter*. The regularizor penalizes a solution for deviations from behavior expected on the basis of statistical *a priori* knowledge or on the basis of the principle of parsimony, as explained below. [If the matrix A in Eq (3.1–1) is well-conditioned and you simply want an ordinary least squares solution subject to the constraints in Eqs (3.1–4) and (3.1–5), then you can still use CONTIN by setting α so small that it is effectively zero.]

Equations (1), (3.1–4), and (3.1–5) result from a combination of two strategies that attempt to select a better estimate from S: (1) incorporation of *a priori* information and (2) *parsimony*. Absolute *a priori* knowledge [e.g., nonnegativity of $s(g)$] can often be incorporated into the constraints in Eqs (3.1–4) and (3.1–5) and is very important because it can eliminate a large subset of S, leading to increased accuracy and resolution. Statistical *a priori* information (e.g., on the expectation values and variances of the x_j) can be incorporated into the regularizor, e.g., using a Bayesian approach (see [10] and Sec 6.5).

In the usual case that statistical *a priori* information is insufficient or missing completely, the regularizor is specified by parsimony. The *principle of parsimony* says, of all solutions in S that have not been eliminated by the constraints, choose the simplest one, i.e., the solution that reveals the least amount of detail or information that was not already known or expected. This is almost certainly not the true solution, but it does tend to protect you from artifacts; i.e., while it probably does not have all the detail of the true solution, the detail that it does have is necessary to fit the data and therefore more likely to be real and not artifact. The appropriate definition of "simplest" obviously

depends on the problem, the *a priori* information, and what you are going to do with the solution. However, by far the most common definition of "simplest" is "smoothest", and, with NORDER=2, CONTIN automatically sets R and r to penalize deviations from smoothness. In addition to smoothness it is often appropriate to include "minimum number of peaks in $s(g)$" in the definition of "simplest" and CONTIN can do this (see Sec 4.1.6.4).

3.2.3 Choosing the regularization parameter

The criterion for choosing α is given in Sec 3.6 of [2]. (See also Sec 5 of [19].) However, it should not be used blindly; you should make sure that the CHOSEN SOLUTION is consistent with all your other knowledge, particularly of the noise levels in the data. You should be especially careful when you expect systematic errors (as indicated by the plot of residuals and the RANDOM RUNS PROB discussed in Secs 3.6.5 and 3.6.6), when $N_y - N_{DF}$ is less than about 10, or (worst of all) when you have artificially pre-smoothed data; then you must rely on your own judgement to decide what is consistent with the data.

Error estimates are discussed in Sec 3.7 of [2]. You should beware of unqualified error estimates for solutions of inverse problems. For example, it has been popular to "stabilize" an inverse problem by analyzing the data in terms of a (often oversimplified) model function with only a few parameters (e.g., a molecular weight distribution as a spline with a small number of knots or as a histogram with a small number of bins). The standard confidence regions for these parameters obtained from least squares are often optimistically small. However, these confidence regions are based on the assumption that the model is correct, or at least an adequate approximation, and this is often not the case, because to stabilize the problem only a small number of parameters (degrees of freedom) are used. Others use too many parameters (e.g., a histogram or spline representation with 9 or more parameters when the data are inadequate to reliably estimate so many parameters) with no regularization and no confidence-region estimate, and they are then often surprised to find solutions with extra peaks. CONTIN attempts to automatically select the α with the number of degrees of freedom that provides the optimum compromise between stability and an adequate model, without fixing the number of degrees of freedom and the detailed form of the model beforehand.

3.3 Input Data Deck

3.3.1 Test data

The test data are shown at the end of [19] under TEST RUN INPUT.

3.3.2 Composition of a Data Set

This is described in Sec 3.3 of [19]. Some further comments:

Card Set 4 is a convenient alternative for inputting the t_k in Eq (3.1–2) when the t_k can be input in NINTT groups of equally spaced t_k.

For example, if the t_k were

1,2, 4,6,8,10, 20,30,40,50, 100,200,300,400,500

then you could input the Control Variable NINTT=4 and

NT	TSTART	TEND
2	1.	2.
4	4.	10.
4	20.	50.
5	100.	500.

If you are simply solving Eq (3.1–1) rather than Eq (3.1–2) and if the array A is simply given by a set of constants, then the t_k may not have any meaning for you. You must nevertheless input the t_k, and it is perhaps easiest to let $t_k = k$ by inputting NINTT=1, TSTART=1., and TEND=N_y.

Card Sets 8–15b are for possible input to USER subprograms and are usually not needed. They are described in Sec 4.2. However, you only have to read Sec 4.2 if you are using a Control Variable that leads to input in a USER subprogram (and the description of the Control Variables in Sec 4.1 will tell you this) or if you are making your own modifications of USER subprograms that result in input.

3.4 Essential Control Variables

The more than 40 Control Variables make CONTIN very flexible. They are all set to default values in the BLOCK DATA subprogram. Therefore, it is only necessary to input a Control Variable in Card Set 2 if you wish to change its value. Card Set 2 can be empty.

You can reset the default values in the BLOCK DATA subprogram to values that

you will most often use. This can save you from inputting these in Card Set 2 every time. You should keep a listing of the current version of your BLOCK DATA subprogram so that you know what the default values are. However, CONTIN also prints out the final values of the Control Variables that will be used for each data set; so this gives you a final check on their values (Sec 3.6.2).

In the descriptions of the Control Variables below and in Sec 4.1, the name of the Control Variable is followed by its DIMENSION specification (if it is an array) followed by the FORMAT and Fortran input list specification for the second card (if the Control Variable requires two cards) and finally instructions for its use. The definition of the Control Variable is enclosed in a box.

Only the essential Control Variables are listed in this section. If you are only using an Applications Package, you may be able to get away without reading Sec 4. Otherwise you will probably have to read parts of Sec 4.1, where more Control Variables are described.

3.4.1 Input control

LAST = T	if this is the last data set in the input data deck,
= F	if one or more data sets follow this one.

A *Data Set* always has Card 1 as its first card and contains all the cards and card sets from the range Card 1–Card Set 15b that are necessary for an analysis of a set of y_k. An input data deck may consist of one or more Data Sets. After CONTIN has completed the analysis of a Data Set, it stops if LAST=.TRUE. in the Data Set just analyzed; if LAST=.FALSE., it reads in the next Data Set and analyzes it.

The settings of all Control Variables are preserved from the analysis of one Data Set to the next. In this way, a lot of input in Card Set 2 is often avoided in Data Sets after the first one. Note that this preservation of values only applies to the Control Variables. The necessary data on Card 1 and Card 3–Card Set 15b must be in every Data Set.

NINTT ≤ 0	when N_y and the t_k in Eq (3.1–2) are to be input with Card 5a and Card Set 5b.
> 0	when N_y and the t_k are to be computed from the input in Card Set 4. The t_k must be in NINTT groups of equally spaced t_k.

See Sec 3.3 for a discussion of NINTT and Card Sets 4 and 5b.

IFORMY(70)	FORMAT(1X,70A1) (IFORMY(J),J=1,70) The second card must contain a Hollerith string of characters making up a legal Fortran FORMAT specification (enclosed in parentheses) for the y_k in Card Set 6.
IFORMT(70)	FORMAT(1X,70A1) (IFORMT(J),J=1,70) This is defined in the same way as IFORMY above, except that it specifies the FORMAT for the input of the t_k in Card Set 5b.
IFORMW(70)	FORMAT(1X,70A1) (IFORMW(J),J=1,70) This is defined in the same way as IFORMY above, except that it specifies the FORMAT for the input of the w_k in Card Set 7.

Note that the FORMAT specifications for the above three Control Variables are incorrect in Table 4 of [19]; the 1X is missing.

3.4.2 Setting up the quadrature grid for Eq (3.1-2)

NG = N_g	in Eq (3.1-3), the number of quadrature grid points, if Eq (3.1-2) is being solved by quadrature;
= N_x	if Eq (3.1-1) is being solved directly.

NG must be at least 2 and must satisfy Eqs (3.5-1), (3.5-2), and (3.5-6). [If for some reason you wanted only one grid point, you could do this by setting NG=2, IQUAD=1, and constraining the solution at one of the two grid points to be zero (see Sec 4.1.2.6).]

When Eq (3.1-2) is being solved by quadrature, the choice of NG is not so obvious. NG must be large enough to make the quadrature grid sufficiently fine (1) so that the quadrature approximation in Eq (3.1-3) is accurate to within the experimental noise level and (2) so that $s(g)$ can be represented in sufficient detail by its values at only the NG grid points. If $s(g)$ contains no more than two or three resolvable peaks, then an NG of the order of 40 is usually sufficient to satisfy both requirements. If $K(g,t)$ were an extremely rapidly varying function of g, then this might not satisfy requirement (1), but changes of integration variable (see Sec 4.1.2.1) or GMNMX (see below) can often eliminate this problem. If not, Sec 3.1 and Eq (3.4) of [2] show how the quadrature approximation can be completely eliminated. We have never found this necessary, even with Fourier and Fourier-Bessel kernels, but this will depend on their frequencies, and some

very rapidly varying kernels (e.g., Mie scattering kernels) might require that you avoid quadrature as discussed in Sec 3.1 of [2].

In principle, you can make NG as large as you wish, the larger, the better. You can have NG > N_y, the number of data points. However, a large part of the computation time is approximately proportional to N_g^3 (and much of the storage to N_g^2). This usually makes an NG $\gtrsim$ 50 expensive and an NG $\gtrsim$ 100 impractical.

It is very easy to test if NG is large enough. You simply change NG and see if the solution, $s(g)$, changes significantly. If not, then NG is very likely large enough.

GMNMX(2)	GMNMX(1)=g_1 and GMNMX(2)=g_{N_g} in Eq (3.1-3); i.e., the first and last grid points in the quadrature.

If you are solving Eq (3.1-1) directly, then there is no quadrature, and the grid points, g_m, have no meaning. It is nevertheless necessary to specify them, and a convenient choice is GMNMX(1)=1 and GMNMX(2)=N_g.

If you are solving Eq (3.1-2) by quadrature, you may want to set GMNMX(1) > a and GMNMX(2) < b. The only requirement is

$$\left| \int_{a}^{\text{GMNMX(1)}} K(g,t_k)\, s(g)\, dg \right| \ll \sigma_k \qquad (1)$$

$$\left| \int_{\text{GMNMX(2)}}^{b} K(g,t_k)\, s(g)\, dg \right| \ll \sigma_k \quad , \qquad k = 1, \ldots, N_y \qquad (2)$$

where σ_k is the standard deviation of the noise at data point k. This is often obvious by the way $K(g,t_k)$ rapidly approaches zero with increasing or decreasing g. By restricting the range of the grid in this way, you make the grid finer and therefore help to satisfy requirements (1) and (2) in the above discussion of NG. However, you should be careful not to be too restrictive with GMNMX. You should have a few extra grid points at each end of the grid, i.e., Eqs (1) and (2) should hold if you replace GMNMX(1) with g_3 and GMNMX(2) with g_{N_g-2}. This is because boundary conditions (Sec 4.1.2.8) can affect a few grid points at each end of the grid. As with NG, a very easy test is to change GMNMX and see if the solution changes significantly. If not, then GMNMX was probably not too restrictive.

Obviously, if you have *a priori* knowledge that $s(g)$ is non-zero only over a restricted range, then GMNMX should be restricted accordingly. Parsimony may also dictate that you try extra analyses with a restricted GMNMX. For example, if the analysis with the unrestricted GMNMX led to secondary peaks or other surprising features at the edges of the grid, then you could repeat the analysis with a shrunken grid that did not contain these extremes to see if a more parsimonious solution could be found that was still consistent with the data. You should not hesitate to try extra analyses using such restrictions or the many other possible constraints described in Sec 4.1 (especially Sec 4.1.6.4) that might help impose parsimony. Furthermore, the variations in the solutions that are consistent with the data will give you a better idea of what features of the solution are invariant and therefore apparently demanded by the data and what parts of the solution are unstable and poorly determined by the data.

Most Applications Packages give specific guidelines for setting up the quadrature grid. If you are not just using an Application Package, then you should also read Sec 4.1.2.1, which tells you how to specify the quadrature formula and the rest of the grid points between g_1 and g_{N_g}.

3.4.3 Lines per page of output

LINEPG =	the number of lines per page available on your output device.

It is used to plan certain plotted output (of the residuals) so that plots are not broken up in going from one page to the next.

3.5 Changing DIMENSION Specifications

If your problem is large, it may be necessary to increase the DIMENSION specifications of some arrays before you can use CONTIN. Throughout CONTIN there are checks to make sure that these maximum allowable dimensions are not exceeded; so one simple way to see if your problem is too large is to try running CONTIN and see if it stops with an error message telling you this. If CONTIN doesn't stop with an error message, then you don't have to read Sec 3.5. However, it is probably worthwhile to first check to make sure that at least the obvious requirements below (e.g., MY) are satisfied. Furthermore, if high-speed storage is expensive, then it would be worthwhile to check below to see if your problem will always be small enough to permit you to reduce some of the DIMENSION specifications.

The DIMENSION specifications are in terms of the eight *DIMENSION Parameters* MA, MG, MY, MINEQ, MEQ, MREG, MWORK, and MDONE. They must satisfy the following requirements:

MA	$\geq$	NG + NLINF + 2	(1)
MG	$\geq$	NG + NLINF + 2	(2)
MY	$\geq$	max {MG, NY}	(3)
MINEQ	$\geq$	NINEQ	(4)
MEQ	$\geq$	NEQ	(5)
MREG	$\geq$	max {NREG, NG + NLINF} + 1	(6)
MWORK	$\geq$	max {(MINEQ + 2)(MG + 1) − 4, MG(MG − 2), 4 * MG}	(7)
MDONE	$\geq$	MQPITR is reasonable if you are doing Peak-Constrained solutions (Sec 4.1.6.4). If you never will do these, then MDONE=1 will save about (7 * MDONE) INTEGER storage locations.	

By max $\{J, K\}$ is meant the maximum of J and K. All other quantities on the right-hand sides can be found in the index in Sec 8. However, all but MQPITR (Sec 4.1.6.4) are simply equal to quantities in Eqs (3.1-1)-(3.2.2-1): NG = N_g if Eq (3.1-2) is being solved; NG = N_x if Eq (3.1-1) is being solved; NLINF = N_L in Eq (3.1-2); NY = N_y in Eq (3.1-1); NINEQ = N_{ineq} in Eq (3.1-4); NEQ = N_{eq} in Eq (3.1-5); NREG = N_{reg} in Eq (3.2.2-1).

CONTIN is designed so that you can change the DIMENSION specifications in two easy steps in the MAIN subprogram: First change the values of the eight DIMENSION Parameters in the DATA statement in Lines 100-101. Then change the DIMENSION statements in Lines 56-65 according to the specifications shown in Lines 69-78. No other changes in the MAIN subprogram or any of the other subprograms are needed.

3.6 Interpreting the output

3.6.1 Output from run with the test data

The beginning of Part 2 of this manual lists selected output from the test run in [19]. Before you compare this too closely with your output, read Sec 3.6. It indicates the many parts that are machine-dependent. Section 3.6 also uses examples from this output to illustrate important general guidelines and points to check in all output. For the purposes of Sec 3.6, it is not necessary that you know the meanings of the Control Variables in the test data or that you know the problem being solved; Sec 6.1 discusses these aspects for those interested

in photon correlation spectroscopy. Only the essential points are discussed here (although, if you are only using an Applications Package with the Version recommended in Sec 2.2, then you might be able to skim over Secs 3.6.3 and 3.6.4). See Sec 4.1 for more details. Section 4.1.5 describes useful Control Variables that can increase or decrease the amount and detail of the output.

3.6.2 Output of the input data

The top of *Page 1* gives the version and date of CONTIN, the package name (from IAPACK as discussed in Sec 3.2 of [19]), and the heading from Card 1 (see Sec 3.3.2). Then Card Set 2, Card 3, and Card Set 4 are printed out as they are read in.

Page 2 contains the final values of the Control Variables, after the changes specified in Card Set 2 and in USERIN have been made. These are the values that will be used throughout the analysis of Data Set 1. It is always a good idea to check and make sure that these values are correct. They are output in six groups (in alphabetical order): undimensioned and dimensioned REAL, INTEGER, and LOGICAL, respectively.

Finally, at the top of *Page 3*, are the values of t_k and y_k that will be used in Eqs (3.1-2) and (3.1-1).

3.6.3 Machine-dependent parameters - PRECIS

Next on *Page 3* come three machine-dependent parameters, SRANGE and RANGE, which you previously set in BLOCK DATA (see Sec 3.1 of [19]), and PRECIS, which is computed in CONTIN as approximately 10 times the relative machine precision; i.e., PRECIS is approximately the smallest positive number such that 1.0+0.1*PRECIS is computed as greater than 1.0 (in DOUBLE PRECISION for Version 2DP and REAL for Version 2SP). Thus PRECIS=1.86D-16 means that there are approximately 16 significant figures in the DOUBLE PRECISION arithmetic. If PRECIS is greater than 1.00E-12, then you should switch from Version 2SP to Version 2DP.

The table and the scale factor for ALPHA are summarized in the first paragraph in Sec 5 of [19].

3.6.4 Singular values

Next on *Page 3* comes the number of unregularized variables, i.e., those which are not explicitly in the regularizor. These are typically the N_L β_i in Eq (3.1-2).

The regularizor is then not full rank, but CONTIN automatically makes it full rank, as described following Eq (A.7) in [2]. However, this means that the scaled singular values, printed at the bottom of *Page 3* will depend on PRECIS, which is machine dependent. Therefore you should not expect your singular values to agree with those on *Page 3*. Even when there are no unregularized variables, only the largest singular values should agree.

3.6.5 Solutions

This output is described in Sec 5 of [19]. You should check the following points:

(1) The VARIANCE of the Reference Solution, i.e., the last solution with an asterisk at the beginning of the second line (see *Page 5*) should agree with OBJ. FCTN. in the first few figures. If not, α is not small enough, and you should read Sec 4.1.3.

(2) Scaling is done automatically in Version 2, but very poor choices of scaling for the t_k or g_m could lead to underflows or overflows. For example the solutions for Test Data Set 2 are of the order of 10^{20}. While this did not lead to overflow, a change in units in the t_k and g_m (e.g., from cm to microns) could avoid this if there were such a danger.

(3) If the CHOSEN SOLUTION, printed at the end of the analysis, does not have a PROB1 between about 0.05 and 0.95, see Sec 4.1.3.

Note also the following points:

(1) The first 6 solutions are for increasing values of α on a coarse grid in equal intervals of $\log(\alpha)$ from a very small to a very large α. Next come 6 solutions for α on a finer grid concentrated in the interesting transition region of PROB1, where it goes from a small to a large value. (Section 4.1.3 explains how this is controlled.)

(2) Sometimes numerical problems prevent the error estimates for the solution from being computed. This is clearly indicated, since they are then *all* set to zero.

(3) When the quadrature grid is not in equal intervals of g, the relative widths of peaks can be misleading, and the actual areas under the peaks, MOMENT(0), can help put things in proper perspective. For example, on the last page of [19], the MOMENT(0) values show that PEAK 2 actually has a slightly greater area than PEAK 1, although the logarithmic spacing of the grid points tends to obscure this.

(4) When MOMENT(J),J=1,2,3 are computed, then the standard deviation of each peak divided by its mean is also printed next to (STD. DEV.)/MEAN. This is a useful dimensionless measure of the width of a peak. Note that these values are only directly useful when the solution is nonnegative (or nonpositive); in the other cases this quantity sometimes cannot even be computed and a zero is output instead.

(5) The *residuals* are the N_y values of the left-hand sides of Eq (3.1–1) or (3.1–3) minus the right-hand sides.

(5a) Of the 5 values printed after PUNCOR, the first ones (i.e., those with the shortest lags) are the most sensitive to correlations in the residuals. When the number of data points is less than about 100, both PRUNS and PUNCOR are relatively insensitive tests. In fact, unless there are at least 10 positive and 10 negative residuals, the approximation [11, p. 97] to PRUNS becomes so poor that a -1.0 is printed in its place.

(5b) These tests are only meaningful if the ordering of the residuals is, e.g., if t_k in Eq (3.1–2) varies monotonically with k.

(5c) Few sets of real data are completely free of systematic errors. For example, while digitally acquired photon correlation or fluorescence decay data usually give reasonably large PRUNS and PUNCOR, fiber diffraction data collected on film usually pick up enough systematic errors during scanning, averaging, disorientation correction, background subtraction, and phasing, that PRUNS and PUNCOR are very small.

3.6.6 Plot of the weighted residuals

The N_y weighted residuals are simply the residuals (Sec 3.6.5) multiplied by $w_k^{1/2}$; so the sum of their squares is just VAR in Eq (3.2.1–1). The weighted residuals are plotted on *Page 16* for the CHOSEN SOLUTION, i.e., the solution with PROB1 TO REJECT closest to 0.5. (Do not expect exact agreement with your results because the grid of α values is machine-dependent.) As indicated on the third line of *Page 16*, this is the solution with ALPHA/S(1)=2.52E-12; its PRUNS and PUNCOR are also printed. Also printed on the third line are the maximum (8.1E-3) and minimum (-5.6E-3) values of the weighted residuals, and the top and bottom dashed lines of the plot correspond to these values. The dashed line in between corresponds to zero. The abscissa is simply the subscript of the residuals, k.

The plot can be very useful for several reasons:

(1) You can easily spot outliers, which may have been caused by gross errors in inputting your data.

(2) Together with PRUNS, PUNCOR, and the plot of the fit (Sec 3.6.7), you can recognize a systematic lack of fit of your model to the data.

(3) What PRUNS and PUNCOR cannot detect is a systematic trend in the magnitudes of the residuals, as on *Page 16*, where the magnitude systematically increases with k. This is due to the improper weighting ($w_k = 1$) in this PRELIMINARY UNWEIGHTED ANALYSIS, and the final weighted analysis on *Pages 17–31* results in residuals (*Page 30*) with more uniform magnitudes.

3.6.7 Plot of the fit to the data

Also on *Page 16* is a plot of the data, y_k, and the fit to the data, i.e., the right-hand side of Eq (3.1–1) or (3.1–3). This also corresponds to the CHOSEN SOLUTION, which is identified by ALPHA/S(1)=2.52E-12 on the third line of *Page 16*. Asterisks are plotted when the "X" and "O" characters coincide. (If a least squares weight, w_k, happens to be zero, then, for plotting purposes only, the fit value is arbitrarily set to y_k and an asterisk will be plotted.)

This plot helps you judge the fit from a more natural perspective, but is not as sensitive as the plot of the residuals at seeing trends or outliers. Each line in the plot has a resolution of 108 spaces. Therefore, each space separating the "X" and "O" on a line corresponds to a difference of about 1% of the total range of the y_k.

You should always look at this plot to make sure that the CHOSEN SOLUTION has not resulted in an inadequate fit or an overfit to your data. This is especially important when the possibility of systematic errors or presmoothed data could make the PROB1 TO REJECT criterion useless. In this case, you might want to plot the fit and the residuals for every solution, and Sec 4.1.5.3 tells you how to do this. It also tells you how to suppress these plots completely if a very large N_y would cause too much printout (but the residuals are plotted compactly, and you should always plot them).

3.6.8 Final weighted analysis - CHOSEN SOLUTION

On *Page 17* ERRFIT and the SQUARE ROOTS OF THE LEAST SQUARES WEIGHTS are printed. (ERRFIT is an optional safety margin that is used to compute the weights; it is discussed in Sec 4.1.2.7). The weights are computed from the

fit to the data for the CHOSEN SOLUTION from the PRELIMINARY UNWEIGHTED ANALYSIS on *Page 12*. In fact, the sole purpose of the PRELIMINARY UNWEIGHTED ANALYSIS is to determine these weights, which are then used for the weighted analysis on *Pages 17–31*. If you had specified the weights in the input data, i.e., if IWT=1 or 4 (see Sec 4.1.2.7), then the entire preliminary analysis on *Pages 4–16* would be missing. With IPRINT (Sec 4.1.5.3) you can reduce the output on *Pages 4–15* to 1 page. Section 4.1.4 mentions a more primitive way of estimating the weights. This could avoid the PRELIMINARY UNWEIGHTED ANALYSIS and cut the computation time in half.

It is always a good idea to compare the CHOSEN SOLUTION for the unweighted (*Page 12*) and weighted (*Page 31*) analyses. It is reassuring when they do not differ drastically from one another, since this indicates that they are at least relatively stable to changes in weighting.

The final weighted analysis is on *Pages 17–31*. (You should not expect exact agreement with your output.) The format is the same as that discussed for *Pages 4–16*, except that PRELIMINARY UNWEIGHTED ANALYSIS is missing from the heading, and the CHOSEN SOLUTION is printed again at the end of the analysis, on *Page 31*.

You should not blindly accept the CHOSEN SOLUTION on the last page and look at nothing else. For example, all solutions on *Pages 18–21* and *24–27* are possibilities. Usually the solutions with PROB1 TO REJECT less than about 0.01 are wildly oscillating or have many peaks and can be rejected on the grounds of parsimony. However, in this example, they are also reasonable single-peaked solutions. (This is because of the nonnegativity constraints; if you want to see the disastrous results without these constraints, run with NONNEG=F.) This is just a set of (relatively poor) data from [3], where it was shown using DISCRETE (Sec 1.1) that a delta-function (i.e., monodisperse) distribution could fit the data just as well. Since this monodisperse model is also parsimonious (i.e., simple and physically reasonable), it must also be considered as a possibility. Note that this uncertainty has nothing to do with the method used to analyze the data; the fact is that all of these solutions fit the data to within experimental error, and there is no way around this except to get more accurate data. It is better to be aware of these possibilities than to blindly fit to a single oversimplified model with a few parameters, or, much worse, to an unregularized model with a large number of subjectively chosen parameters (e.g., specifying the bins of a histogram or the knots of a spline). In these cases, what you get out (as a solution) depends very strongly on what you put in (as a model).

Sometimes it even happens that a solution with an α smaller than that of

the CHOSEN SOLUTION is more parsimonious, e.g., has fewer peaks. Obviously you should give preference to this solution, even if it has a small PROB1 TO REJECT. This has only occurred with oscillatory kernels as in the Fourier-Bessel transform, and only very rarely, but you should look at all solutions, not just the CHOSEN SOLUTION.

All the solutions on *Pages 18–21* and *24–27* are concentrated on a small number of grid points. When this happens, you should rerun the data with the grid restricted to the region of interest in order to permit better resolution and accuracy, in this case with GMNMX(1) about 10^4 and GMNMX(2) about 10^6.

Note that, although the CHOSEN SOLUTION has 7 non-zero parameters, the effective number of free parameters (DEG FREEDOM) is only 3.137; the regularizor and nonnegativity keep the problem stabilized while permitting a large number, 32, of possible free parameters. On *Page 18*, where the regularizor is practically zero, DEG FREEDOM=3.000 is equal to the number of non-zero parameters, as expected; nonnegativity constraints alone have kept the number of parameters small and therefore stabilized the problem. (This is unusual; nonnegativity constraints alone are usually insufficient.)

Pages 32–65 contain the analysis of Test Data Set 2. This illustrates the analysis of a simulated radius distribution. Two pages of output from this analysis are at the end of [19]. Only selected pages are reproduced here, to illustrate the output of the squared form factors (*Page 33*) and the simulated data (*Page 34*).

4. MORE INFORMATION ON USING CONTIN

If you are only using an Applications Package you may be able to skip this section, or at least many parts of it (see Sec 6 for advice on what parts you should read). However, it is useful to at least look through this section to see what possibilities are available. (Remember that names of Control Variables, etc. are generally not cross-referenced and to use the indexes in Sec 8.)

4.1 USER Subprograms and More Control Variables

Section 4.1 contains descriptions of all the USER subprograms and all the Control Variables not described in Sec 3.4. The first four characters in the names of all the USER subprograms are "USER"; none of the Control Variables begin with "USER". The USER subprograms are designed for you to change easily to suit your needs. They are quite well documented with comment cards; so the explanations in Sec 4.1 usually complement, rather than repeat, the information in these comment cards. The descriptions of the Control Variables here are in the same format as explained in Sec 3.4. Section 4 of [2] also gives a good overview of the USER subprograms and Control Variables.

4.1.1 User arrays

You can use these Control Variables to input data that will be used in any of the USER subprograms or to store results that are produced in the USER subprograms for later use. Their use is illustrated in some of the USER subprograms described below. There is one array for REAL, INTEGER, and LOGICAL types:

RUSER(100)

IUSER(50)

LUSER(30) This is a LOGICAL array.

These, like all the other Control Variables, are in three COMMON blocks (see the BLOCK DATA listing) that are shared by all the USER subprograms and many others. So be careful not to use the same array element [e.g., IUSER(2)] for two different purposes (even in different subprograms), since the first value will be overwritten.

If, for some reason, you ever want to change the DIMENSION specification of any of these three Control Variables, then you must not only change the DIMENSION specification in the COMMON block in all the subprograms, but also make the change explained in the comment cards in subprogram STORIN and change the corresponding DATA specification in BLOCK DATA so that the full array is initialized.

4.1.2 Specifying the problem

4.1.2.1 Specifying the quadrature grid

IQUAD = 1	for the direct solution of Eq (3.1-1) with no quadrature.
= 2	for approximating Eq (3.1-2) with (3.1-3) using the trapezoidal rule.
= 3	for approximating Eq (3.1-2) with (3.1-3) using Simpson's rule (with the trapezoidal rule for grid points NG − 1 and NG if NG is even).

IGRID = 1	will put the g_m in Eq (3.1-3) on an equally spaced grid with g_1 = GMNMX(1), g_{N_g} = GMNMX(2), and $g_{j+1} - g_j$ equal for $j = 1, \ldots, N_g - 1$.
= 2	as for IGRID=1, except that the grid is equally spaced in $H(g)$; i.e., $H(g_{j+1}) - H(g_j)$ are equal. The function $H(g)$ is defined in USERTR.
= 3	will call USERGR to define the c_m and g_m in Eq (3.1-3).

CONTIN can properly handle IQUAD=3 and IGRID=2. This was done with the Test Data (see *Pages 2* and *33* of the output).

USERTR	When IGRID=2 Fortran statements 210, 220, and 230 must define, respectively, the transformation, $H(g)$, the inverse transformation, $H^{-1}(g)$, and dH/dg.

In the version of USERTR supplied in Files 1 and 2, is the most common case of putting a grid in equal logarithmic intervals: $H(g) = \ln(g)$, $H^{-1}(g) = \exp(g)$, and $dH/dg = 1/g$. Note that it only makes sense to have $H(g)$ a differentiable monotonic function of g. When IGRID=1, USERTR is already programmed to give the identity transformation: $H(g) = H^{-1}(g) = g$, $dH/dg = 1$.

A logarithmic or similar transformation is often helpful (or essential) in providing a grid for an adequate representation of the solution, especially if g ranges

over several orders of magnitude. For example, logarithmic intervals were necessary for the Test Data (note that IGRID=2 on *Pages 2* and *33*). With equal intervals in g we would have had $g_1 = 500$ and $g_2 = 2.5 * 10^5$, which is already over half of the logarithmic grid. Any detail for small g would therefore have been lost.

Usually, the choice of transformation is clear from the application, usually logarithmic or none (i.e., IGRID=1). However, the choice of the transformation can affect the meaning of the regularizor if NORDER is positive (see Sec 4.1.2.8). In this case deviations from smoothness are penalized according to difference approximations to derivatives of $s(g)$ with respect to $H(g)$ at the grid points. Thus solutions are favored that are smooth when $s(g)$ is plotted against $H(g)$ rather than g. For example, the log-normal distribution is very asymmetric and has a very sharp ascent on the low-g side of its peak. This will be less favored than if the logarithmic transformation were used where it is then a smooth and symmetric normal distribution. Thus, you should obviously choose a transformation and regularizor that will not strongly penalize solutions with parsimonious shapes or with shapes that you expect on the basis of *a priori* knowledge. This may sound very dishonest and manipulative, but it is the same as choosing the regularizor on the basis of *a priori* knowledge and parsimony as discussed in Sec 3.2.2; except for the practical problems of grid-point spacing, you could achieve the same effect as a transformation by transforming the regularizor. Furthermore, the choice of a simple but adequate transformation is usually obvious. For example, with molecular weight distributions or broad relaxation spectra where g is nonnegative and covers many orders of magnitude, the logarithmic transformation is usually natural. In estimating electron density distributions, IGRID=1 is usually natural.

USERGR	This is only called when IGRID=3. You can then compute your own special quadrature coefficients and grid, c_m and g_m, in Eq (3.1-3).

The g_m must be monotonic; i.e., you must have either $g_1 > g_2 > \ldots > g_{N_g}$ or $g_1 < g_2 < \ldots < g_{N_g}$.

In the version of USERGR in Files 1 and 2, the g_m are simply read in, and the c_m (called CQUAD) are then computed for the trapezoidal rule.

4.1.2.2 Specifying the kernel

USERK	evaluates $K(g_m, t_k)$ in Eq (3.1-3).

The Fortran arguments of USERK in the notation of Eq (3.1-3) are USERK = $K(g_m, t_k)$, JT = k, T = t, JG = m, and G = g. When IQUAD = 1, the c_m in Eq (3.1-3) are simply set to 1. This is equivalent to Eq (3.1-1) with $K(g_m, t_k) = A_{km}$, $s(g_m) = x_m$, and $N_L = 0$ in Eq (3.1-3). So when IQUAD = 1, you simply evaluate USERK = A_{km}. (CONTIN does permit a positive N_L when IQUAD = 1, but these extra functions are evaluated in USERLF, not USERK; see Sec 4.1.2.4.)

The version of USERK in Files 1 and 2 evaluates Eq (2) of [19]. It is also described in Secs 6.5.1 and 6.5.2. Lines 666-670 explain how you can easily modify USERK for your kernel. LINE 681-702

4.1.2.3 Avoiding the quadrature in Eq (3.1-3)

Sections 3.1 and 3.3.2 of [2] describe how you can avoid quadrature by expanding the solution as a sum of basis functions, as in Eq (3.4) of [2]. Equation (3.5) of [2] must then be evaluated in USERK. As mentioned in Sec 3.1 of [2], this can be useful if the kernel varies very rapidly with g, and a very large N_g would be necessary for a sufficiently accurate quadrature. We have not yet encountered such a case.

In Eq (3.13) of [2], note that B_i and B_j should read B_i'' and B_j''.

4.1.2.4 Specifying the N_L terms in Eq (3.1-2)

NLINF = N_L	in the second sum in Eq (3.1-2).
= 0	if there is no second sum in Eq (3.1-2).

NLINF must not be negative and must satisfy Eqs (3.5-1), (3.5-2), and (3.5-6). You can also use a positive NLINF with IQUAD=1, since in this case CONTIN still uses Eq (3.1-3) (with all the $c_m = 1$).

USERLF	(only called when NLINF is positive) evaluates the $L_i(t_k)$ in Eq (3.1-2).

The Fortran arguments of USERLF in the notation of Eq (3.1-2) are USERLF = $L_i(t_k)$, JY = k, JLINF = i, and T = t. (You needn't worry about NYDIM; it is only used in a DIMENSION statement for T.)

The versions of USERLF in Files 1 and 2 illustrate the use of the user array IUSER. In the usual case when NLINF=1 and IUSER(2)=0 (see *Page 2* of the test output), USERLF evaluates $L_1(t) = 1$; i.e., a simple additive constant (often called "background" or "baseline") is provided for. However, when NLINF=2, two background constants are provided for; the first is only present for the first IUSER(2) data points, i.e., for $k = 1, 2, \ldots,$ IUSER(2), and the second only present for $k =$ IUSER(2)+1, ..., N_y. This allows you to combine two sets of data and to allow each of them to have their own background.

4.1.2.5 Specifying the inequality constraints

DOUSNQ = T	if USERNQ is to be called to specify inequality constraints.
= F	if USERNQ is not to be called.

USERNQ	(only called when DOUSNQ=T) sets the inequality constraints in Eq (3.1-4).

In USERNQ, you must set the Fortran variable NINEQ = N_{ineq} in Eq (3.1-4) and AINEQ(I,J) = D_{ij} for I = 1, ..., NINEQ and J = 1, ..., NGL, where

$$\text{NGL} = \text{NG} + \text{NLINF} \quad (= N_g + N_L = N_x) \tag{1}$$

is the total number of unknowns in Eq (3.1-3), I = i, and J = j. You must also set AINEQ(I,NGLP1) = d_i in Eq (3.1-4) for I = 1, ..., NINEQ, where

$$\text{NGLP1} = \text{NGL} + 1 \tag{2}$$

The version of USERNQ in Files 1 and 2 constrains the N_L linear coefficients β_i in Eq (3.1-3) to be nonnegative.

If the constraints include the nonnegativity of all the $s(g_m)$ in Eq (3.1-3), then you should set these with NONNEG and not in USERNQ.

NONNEG = T	to constrain $s(g_m)$ in Eq (3.1-3) to be nonnegative for $m = 1, \ldots, N_g$.
= F	to take no such action.

When NONNEG=T, CONTIN adds these constraints to any that have been set in USERNQ [and resets NINEQ = $N_{ineq} + N_g$. In addition, peak constraints can further increase NINEQ as described in Sec 4.1.6.4, but CONTIN checks for violations of Eq (3.5-4); so you do not have to worry about this unless an error message and abort occurs.]

4.1.2.6 Specifying the equality constraints

NEQ = N_{eq}	in Eq (3.1-5)
= 0	for no equality constraints.

NEQ must satisfy Eq (3.5-5).

USEREQ	(only called if NEQ is positive) sets the equality constraints in Eq (3.1-5).

In USEREQ you must set the Fortran variable AEQ(I,J) = E_{ij} and AEQ(I,NGLP1) = e_i in Eq (3.1-5) for I = 1, ..., NEQ and J = 1, ..., NGL, where I = i and J = j. The rank of E must be N_{eq}; i.e., the equality constraints must be linearly independent and consistent with each other.

The version of USEREQ in Files 1 and 2 constrains $s(g_{N_y})$ = RUSER(1) if NEQ ≥ 1, $s(g_1)$ = RUSER(2) if NEQ ≥ 2, and the integral over the solution [using the first sum on the right of Eq (3.1-3) as an approximation] to be RUSER(6) if NEQ = 3. NEQ must be 0, 1, 2, or 3 for this version.

It is useful to fix one or both of the endpoints $s(g_1)$ and $s(g_{N_y})$, if you know that the solution must approach certain values at the endpoints. Even if you only know that the solution is practically indeterminate at the endpoints (e.g., because the endpoint regions hardly contribute to the data), constraining the endpoints to zero can help find a parsimonious solution when there are no nonnegativity constraints [4, p. 317].

4.1.2.7 Specifying the least-squares weights

The w_k in Eq (3.2-1) should be proportional to $1/\sigma_k^2$, where σ_k is the standard deviation of the noise at data point k. If the noise level is independent of k, then IWT=1 (an unweighted analysis) is appropriate. With IWT=4, you can input the w_k directly. You can also use IWT=1 and USERIN (see Sec 4.1.4) to compute the w_k from the input data. If one of these cases apply, then you can skip the

rest of Sec 4.1.2.7.

Often, however, σ_k is a function of $y(t_k)$, the (exact) value that y_k would take if it were noise-free. For example, for Poisson statistics, $\sigma_k = [y(t_k)]^{1/2}$ and you should use $w_k = 1/y(t_k)$. Unfortunately, $y(t_k)$ is unknown, and using y_k in place of $y(t_k)$ can dangerously bias the analysis, because y_k that happen to have negative noise components [i.e., y_k that are smaller than $y(t_k)$] will be given too large w_k, and y_k with positive noise will be given too small w_k. This is especially dangerous in the case of Poisson statistics and very small $y(t_k)$, where y_k could be nearly (or exactly) zero and w_k could then become disastrously large.

Therefore CONTIN has the option (with IWT=2,3, or 5) of doing a PRELIMINARY UNWEIGHTED ANALYSIS to get better estimates of the $y(t_k)$. These estimates are simply YFIT_k, the fit to the y_k [i.e., the $y(t_k)$ computed from the solution] obtained in the PRELIMINARY UNWEIGHTED ANALYSIS (as on *Page 16* of the test output). Then, for example,

$$\text{YSAFE}_k = \max\{|\text{YFIT}_k|, \text{ERRFIT}\} \qquad (1)$$

could be used in place of the unknown $y(t_k)$ for computing the w_k. ERRFIT is an added safety margin to prevent a disastrously large w_k if a $|\text{YFIT}_k|$ happens to be very small and w_k happens to be proportional to a negative power of $|\text{YFIT}_k|$, as when IWT=2 or 3. It is computed by first finding the k for which $|\text{YFIT}_k|$ is minimum and then setting ERRFIT to the root-mean-square of the residuals, $y_k - \text{YFIT}_k$, for the NERFIT k centered at this minimum. Thus ERRFIT is a rough estimate of the scatter of the y_k near this minimum.

If there is no danger of w_k blowing up for small $|\text{YFIT}_k|$, then ERRFIT may not be necessary. In fact it may be possible to avoid the PRELIMINARY UNWEIGHTED ANALYSIS completely and to use the y_k data directly to compute the w_k (e.g., in USERIN), *provided* that the w_k are insensitive to the noise in the y_k.

If negative values of $y(t_k)$ are meaningful, then the sign of YFIT_k could be appended to the right-hand side of Eq (1).

This PRELIMINARY UNWEIGHTED ANALYSIS strategy for determining least-squares weights has been widely used [e.g., 8, 6, 5], and Price [13] has pointed out that for Poisson statistics it yields approximately maximum-likelihood estimates. Although it doubles the computation time, it was mentioned in Sec 3.6.8 that it is useful to compare the results of the final weighted analysis with the preliminary unweighted one to see if the results are very sensitive to the weighting. You might even find evidence (e.g., in the residuals) that the unweighted

analysis is more appropriate. There is much evidence that, when you are uncertain whether a weighted or unweighted analysis is more appropriate, it is generally safer to favor the unweighted one.

IWT = 1	for $w_k = 1$ for $k = 1, \ldots, N_y$ (i.e., an unweighted analysis, which is appropriate when σ_k is independent of k).
= 2	for $w_k = 1/\text{YSAFE}_k$, where YSAFE_k is given by Eq (1). This is appropriate when σ_k is proportional to $\|y(t_k)\|^{1/2}$, as with Poisson statistics.
= 3	for $w_k = \text{YSAFE}_k^{-2}$. This is appropriate when σ_k is proportional to $\|y(t_k)\|$, i.e. for a constant *relative* error.
= 4	for inputting the w_k in Card Set 7 (see Sec 3.3.2).
= 5	for computing the w_k in USERWT. This is only necessary if none of the other IWT values are appropriate.

You may also input IWT=1 and then compute the w_k from the input data in USERIN (see Sec 4.1.4). When IWT=2, 3, or 5 a PRELIMINARY UNWEIGHTED ANALYSIS, as discussed above, is done. IWT=3 weights small YSAFE_k very strongly; you should be certain that it is appropriate, and you should use NERFIT.

NERFIT =	the number of residuals used to compute ERRFIT, as discussed above. NERFIT = $O(10)$ is usually reasonable.
= 0	for setting ERRFIT = 0.

USERWT	(only called when IWT=5) computes the w_k after the PRELIMINARY UNWEIGHTED ANALYSIS.

In USERWT you are supplied with ERRFIT, Y (the array of the y_k), and YLYFIT (the array of the residuals, $y_k - \text{YFIT}_k$). From these you must compute and store the *square roots* of the w_k in SQRTW(K), K = 1,...,NY. You can evaluate YSAFE_k in Eq (1) as AMAX1(ABS(Y(K)-YLYFIT(K)),ERRFIT) with K = k. You can then use YSAFE_k in place of the unknown $y(t_k)$ to estimate the $1/\sigma_k$ and store them in the SQRTW(K).

In the version of USERWT in Files 1 and 2,

$$\sigma_k^2 = \frac{y^2(t_k)+1}{4By^2(t_k)}\,, \tag{2}$$

where B is a proportionality constant, independent of k, and can therefore be ignored when computing SQRTW. Also note that there is no danger of $1/\sigma_k$ (and hence SQRTW) becoming disastrously large if the approximation to $y(t_k)$ becomes very small; therefore ERRFIT is not necessary and NERFIT=0 can be used (see *Pages 2* and *32* of test output). Equation (2) arises in photon correlation spectroscopy; it is derived in Eqs (9), (10), (13) and (14) of [1].

Equation (13) of [1] is a very useful general approximation (due to Bartlett) for σ_k when the y_k have been obtained from a transformation of the original data, whose statistics are known.

4.1.2.8 Specifying the regularizor

NORDER < 0	for calling USERRG to set a special user-defined regularizor.
= n	$n = 0, 1, \ldots, 5$ for setting the regularizor [the sum that is muliplied by α^2 in Eq (3.2.2-1)] to be the sums of the squares of the n^{th} differences of the $N_g - n$ sets $(x_1, x_2, \ldots, x_{n+1})$, $(x_2, x_3, \ldots, x_{n+2}), \ldots, (x_{N_g-n}, x_{N_g-n+1}, \ldots, x_{N_g})$.

The integer n is called the order of the regularizor. When NORDER=0, the regularizor is simply the sum of the x_j^2 for $j = 1, \ldots, N_g$. This tends to find the "smallest" solution that is consistent with the data, in that it tends to keep the x_j^2 small. Using this zero-order regularizor is also referred to as "ridge regression" and there are sometimes good statistical grounds for using this, e.g., when the x_j are uncorrelated and have equal *a priori* probability distributions [10].

Most common is NORDER=2, where the regularizor is the sum of the squares of the second differences of the first N_g x_j. These differences are approximations to the second derivatives and therefore NORDER=2 tends to give the "smoothest" solution that is consistent with the data. Here "smoothest" is in the same sense as the smoothness of cubic splines, which minimize an integral over second derivatives squared. An NORDER=2 is reasonable in the common case that a smooth solution without sharp changes in slope can be considered parsimonious.

An NORDER=3 usually produces results similar to NORDER=2, but it has a greater

tendency to produce extra side lobes and therefore is not recommended. Similarly, it is hard to imagine why you would ever want to use NORDER=1, 4, or 5. Usually a positive NORDER is only sensible if the quadrature grid is uniformly spaced, as with IGRID=1 or 2.

Often when you are solving Eq (3.1-2) with (3.1-3), you know that $s(g) = 0$ outside the range of the grid, $g_1 \le g \le g_{N_g}$, or you have effectively forced this by truncating the integral in Eq (3.1-2) at $g_1 > a$ or $g_{N_g} < b$. You can include this information by including in the regularizor extra n^{th} differences over zeroes outside the grid. For example, you can include differences over $(0, x_1, x_2, \ldots, x_n)$, $(0, 0, x_1, x_2, \ldots, x_{n-1}), \ldots$, $(0, 0, \ldots, 0, x_1)$ and similarly over $(x_{N_g-n-1}, \ldots, x_{N_g}, 0), \ldots$, $(x_{N_g}, 0, \ldots, 0)$. This tends to make the solution approach zero more smoothly at one or both ends of the grid, and this is often more natural and parsimonious. You can do this with:

NENDZ(2)	NENDZ(1) = the number of extra zeroes to be placed before x_1.
	NENDZ(2) = the number of extra zeroes to be placed after x_{N_g}.

This is only done when NORDER is positive. Then NENDZ(1) + NENDZ(2) extra differences will be included in the regularizor. When NENDZ(J)=0, no extra differences will be added for the end of the grid corresponding to J. You must have:

$$0 \le \text{NENDZ(J)} \le \text{NORDER}, \quad \text{J=1,2} \tag{1}$$

USERRG	(only called when NORDER < 0)
	sets a special user-defined regularizor in Eq (3.2.2-1).

In USERRG, you must set the Fortran variables NREG = N_{reg}, REG(I,J) = R_{ij}, and REG(I,NGLP1) = r_i in Eq (3.2.2-1) for I = 1, ..., NREG and J = 1, ..., NGL, where I = i, J = j, and NGL (= $N_g + N_L = N_x$) and NGLP1 (= NGL + 1) are supplied to you in a COMMON block in USERRG.

The version of USERRG in Files 1 and 2 penalizes deviations of the solution from an expected one by placing the $N_g \times N_g$ identity matrix in R and the expected solution in $\mathbf{r}$. The expectation values of $x_1, \ldots, x_{N_g}$ are read in [FORMAT(5E15.6)] once and stored in RUSER for possible future use. The comment cards explain how this is done by setting LUSER(1)=F in Card Set 2 and using it as a flag to tell whether the expected solution has already been read in. [Note that LUSER(1) must be reset to F in subsequent Data Sets.] [Note that only N_g, not N_x, values of the expected solution are specified; nothing is

said about the N_L expected values of the β_i in Eq (3.1-2).] Such a regularizor can be useful in testing for deviations from an expected solution [3. p. 4275]. For example, deviations from an expected molecular weight distribution might be due to postulating an incorrect chemical kinetic mechanism, running the polymerization incorrectly, or to contamination.

4.1.3 Specifying the range of the regularization parameter

If you are using the PROB1 TO REJECT criterion, then Version 2 with the default settings has an efficient way of first scanning a wide range of α with a coarse grid and then concentrating on the interesting region of α with a fine grid. Thus, it should not be necessary to change the Control Variables described in this section. Only if you are not using PROB1 or if you are referred to this section by points (1) or (3) of Sec 3.6.5, it is necessary to read this section.

CONTIN computes a series of solutions for α in (optionally) two grids in equal intervals of $\log(\alpha)$.

NQPROG(2)	NQPROG(J) = the number of α values (and hence solutions) in grid J, J=1,2. NQPROG(J) = 0 for no grid J.

RSVMNX(2,2)	FORMAT(4E10.3) The first α in grid J is α = RSVMNX(1,J)*S(1)*PRECIS, and the last α in grid J is α = RSVMNX(2,J)*S(1).
RSVMNX(J,2) = 0	(for J=1 or 2) automatically produces a grid of NQPROG(2) α values concentrated in the interesting transition region for PROB1.

Typically RSVMNX(J,1)=1, RSVMNX(J,2)=0, and NQPROG(J)=6 for J=1 and 2. With RSVMNX(1,1)=1, the coarse grid starts with a very small α, and with RSVMNX(2,1)=1 the coarse grid ends with a very large α. (Note that what "large" and "small" α means depends on the scaling of the problem, while the meanings of RSVMNX are scale-independent.)

If you were referred to this section by point (1) of Sec 3.6.5, then use the default values of RSVMNX. If this is already the case, then you should switch to Version 2DP or reduce RSVMNX(1,1) to about 0.01. If this still does not help, then you might try reformulating the problem to a better scaled one where the rows in the matrix A and the singular values (*Page 3* of the test output) do not have such a wide range of values. For example, in light scattering you might solve for a weight- rather than an number-fraction distribution. If this does not help, then the PROB1 TO REJECT criterion will tend to give an over-regularized solution.

If you were referred to this section by point (3) of Sec 3.6.5, then use the default values of RSVMNX. If this is already the case, then find in the output two values of ALPHA/S(1), a_1 and a_2, that lie as closely as possible on each side of the transition region where PROB1 TO REJECT goes from less than 0.05 to more than 0.95. Setting RSVMNX(1,2)=a_1/PRECIS and RSVMNX(2,2)=a_2 will concentrate grid 2 in this interesting region.

If you know that the regularized solution must have a minimum number of degrees of freedom, then DFMIN can be useful:

DFMIN > 0	will reduce RSVMNX(2,J), J=1,2, so that there will be approximately at least DFMIN degrees of freedom in the regularized solution (not counting the unregularized parameters or binding inequality constraints).
≤ 0	will cause no such action.

Typically DFMIN=2 is appropriate. If DFMIN is accidentally input with an unreasonably large value (e.g., greater than the number of regularized variables), it will be ignored.

If you know (e.g., from a previous run) exactly the α that you want to use, then you can specify it:

ALPST(2)	ALPST(1) > 0 will cause grid 2 during the PRELIMINARY UNWEIGHTED ANALYSIS (e.g., the analysis on *Pages 3–16* of the test output) to be composed of only one grid point: α = ALPST(1), and will cause the corresponding solution to be used to compute the least squares weights. ALPST(2) > 0 will cause grid 2 during the final analysis (e.g., the analysis on *Pages 17–31*) to be composed of only α = ALPST(2), and will cause the corresponding solution to be taken as the CHOSEN SOLUTION. ALPST(J) > 0 for J = 1 or 2 acts as described above, regardless of the values of NQPROG(2), RSVMNX, or PROB1 TO REJECT. ALPST(J) $\leq$ 0 will cause no such action.

ALPST is often used with Peak-Constrained Analyses (see Sec 4.1.6.4).

4.1.4 Modifying the input data

USERIN provides a convenient way of transforming your raw input data into the data required by CONTIN, or for computing special least squares weights from the input data.

DOUSIN = T	to call USERIN
= F	to take no such action

USERIN	(only called when DOUSIN=T) is called after the input of Card 1–Card Set 7 (see Sec 3.3.2) and allows you to modify the t_k, y_k, and w_k in Eqs (3.1–2) and (3.2.1–1) or the Control Variables.

The important Fortran arguments of USERIN in terms of Eqs (3.1–2) and (3.2.1–1) are T(K) = t_k, Y(K) = y_k, SQRTW(K) = w_k, and K = k. Note that you put the w_k (*not* their square roots, which are automatically computed by CONTIN later) in SQRTW.

USERIN is called before the output shown on *Page 2*. Thus changes that you have made in USERIN will be in this output.

The version of USERIN in Files 1 and 2 is described in Sec 4.2 of [19] and Sec 6.1.

As explained in Sec 4.1.2.7, the computation of the least-squares weights often requires estimates of the noise-free values of the data. The PRELIMINARY UNWEIGHTED ANALYSIS described there is generally the safest and best way of doing this. However, if CPU time is critical, then one could attempt to get an estimate of $y(t_k)$ by fitting an empirical smooth curve through the data in USERIN, say using Eq (7.13.3) or (7.13.4) of [20]. The danger of overfitting or underfitting is then much greater, but the CPU time will be cut in half. Note that the empirically smoothed values should only be used in USERIN for YFIT_k in Eq (4.1.2.7–1) in computing the weights; they should *never* be used in the analysis to replace the raw data, y_k.

4.1.5 Output control

4.1.5.1 Special user-programmed output

IUSROU(2)	controls when USEROU will be called. IUSROU(1) controls this for the PRELIMINARY UNWEIGHTED ANALYSIS, and IUSROU(2) for the final analysis.
IUSROU = 0	for never calling USEROU.
= 1	for calling USEROU only after the Peak-Constrained solution.
= 2	for also calling USEROU after the CHOSEN SOLUTION.
= 3	for calling USEROU after every solution.
= 4	for also starting a new page for each solution.

USEROU	(calling controlled by IUSROU) can be called after each plot of a solution and before the output of the moments. It gives you the opportunity to use the solution to compute further output for your special purposes.

The arrays supplied to you as arguments in USEROU are described in the COMMENT cards in USEROU.

The version of USEROU in Files 1 and 2 is specialized to circular dichroism (see Sec 6.5). The solution vector is multiplied by a matrix to obtain the 3 quantities that are actually of interest, and these are then normalized and output.

4.1.5.2 Moments of the solution

These are described in the discussion of Eqs (7) and (8) of [19].

DOMOM = T	to compute and output the moments every time a solution is printed.
= F	to take no such action.

MOMNMX(2)	If DOMOM=T, then M_j in Eq (7) of [19] is evaluated for $j =$ MOMNMX(1),...,MOMNMX(2).

If MOMNMX(1) > MOMNMX(2), then the moment computation is skipped.

MPKMOM =	the number of individual peaks for which moments are also to be computed.

If there are more than MPKMOM peaks, then peak MPKMOM and all following peaks along the g-axis are considered together as peak MPKMOM. If MPKMOM ≤ 1, then only the moments of the entire solution are computed.

4.1.5.3 Controlling the quantity and spacing of the output

NEWPG1 = T	to start a new page at the start of a run.
= F	to take no such action.

Some systems automatically jump to a new page at the start of an execution and NEWPG1=F can prevent an extra blank page.

MIOERR =	the number of messages indicating errors in your input format that will be printed before the run is aborted.

MIOERR=5 is reasonable. CONTIN automatically sets MIOERR to at least 2.

PRY = T	to output the T and Y arrays as on *Page 4* of the test output. If IWT=4, the square roots of the w_k in Eq (3.2.1-1) are also output. If SIMULA=T, then EXACT, the exact noise-free values, and the noise, Y-EXACT, used in the simulation are also printed.
= F	to suppress this output.

Unless N_y is excessively large, PRY=T is strongly recommended.

PRWT = T	to output the SQUARE ROOTS OF THE LEAST SQUARES WEIGHTS (only when IWT=2, 3, or 5) as on *Page 17* of the test output.
= F	to suppress this output.

Unless N_y is excessively large, PRWT=T is strongly recommended.

IPRINT(2)	has exactly the same allowed values and effect as IUSROU (Sec 4.1.5.1), except that it controls when the plots of the solution will be printed.

IPLRES(2)	controls when the weighted residuals (Sec 3.6.6) will be plotted. IPLRES(1) controls this for the PRELIMINARY UNWEIGHTED ANALYSIS, and IPLRES(2) for the final analysis.
IPLRES = 0	for never plotting them.
= 1	for plotting them only after the Peak-Constrained Solutions.
= 2	for also plotting them before the CHOSEN SOLUTION (as in the test output).
= 3	for plotting them after every solution.

IPLRES=2 is recommended. However, it is often useful to use IPLRES=3 to see how the plots change as α increases, especially if you are uncertain about the applicability of PROB1 TO REJECT to decide the CHOSEN SOLUTION.

IPLFIT(2)	has exactly the same meaning and allowed values as IPLRES, except that it controls when the plots of the fit to the data (Sec 3.6.7) will be made.

IPLFIT(J)=2 for J=1 and 2 is recommended. However, as with IPLRES, IPLFIT(2)=3 can be very useful in visually assessing when α is getting so large that the data are no longer being fit to within the experimental noise level. IPLFIT(2)=3 is especially recommended for the first few data sets that you analyze; it can give you a reliable idea of the resolving power of your data and of the uncertainties in the solution.

DOCHOS = T	to output the CHOSEN SOLUTION once again at the end of the analysis (as on *Page 31*).
= F	to take no such action.

4.1.6 Miscellaneous options

4.1.6.1 Scratch file - to save computing time

You can save computer time, especially if the computation in USERK of $K(g_m, t_k)$ in Eq (3.1-3) is time-consuming, by specifying that $K(g_m, t_k)$ be computed only once and then written onto a temporary scratch file for later use. Otherwise, $K(g_m, t_k)$ is recomputed every time (twice for each solution).

IUNIT $\geq$ 0	is the device number of the scratch file; i.e., $K(g_m, t_k)$ will be written and read WRITE(IUNIT) ... and READ(IUNIT) ....
$<$ 0	to use no scratch file and recompute $K(g_m, t_k)$ every time.

If you use a scratch file, the comment cards in Lines 113-116 point out that you may have to open the scratch file numbered IUNIT right after Line 126 if your system does not do this automatically for you. This scratch file must be of the type that will allow unformatted WRITEs, REWINDs, and unformatted READs (iterated in that order) to be executed (in subprogram GETROW). The file requires enough space for $(N_y + N_L + 1)N_y = (N_x + 1)N_y$ words (DOUBLE PRECISION for Version 2DP and REAL for Version 2SP), where N_y, N_L, N_x, and N_y are defined in Sec 3.1.

4.1.6.2 Simulating data

SIMULA = T	to call USERSI and USEREX to compute simulated data, y_k, and put them in the array Y.
= F	to input the array Y in Card Set 6 (Sec 3.3 of [19]).

USERSI	(only called when SIMULA=T) computes the simulated data, y_k.

In USERSI you must use USEREX to put the noise-free simulated "data" in the array EXACT(J) for J = 1,...,NY. You then add pseudo-random normal deviates (furnished by the subprogram RGAUSS) to EXACT, and store this simulated (noisy) data in Y(J) for J = 1,...,NY. You must have set IUSER(3) to an integer between 1 and 2147483646, since it is used by the pseudo-random number generator. Otherwise, USERSI sets it to the default value of 30171. As a Fortran argument in USERSI, you are supplied with the array T, which contains the t_k in Eq (3.1-3) for $k = 1, \ldots, N_y$.

The version of USERSI in Files 1 and 2 is discussed in Sec 4.3 of [19] and the COMMENT cards.

USEREX	(only called when SIMULA=T) computes the noise-free value of a simulated data point.

As Fortran arguments in USEREX you are supplied with T, the array of the t_k in Eq (3.1-3), and IROW, the subscript of the data point to be evaluated. You must set USEREX to the noise-free value of the data point at T(IROW).

The version of USEREX in Files 1 and 2 is described in Sec 4.3 of [19] and the COMMENT cards. Section 3.7 and 4.6 of [2] describe some useful applications of simulations.

4.1.6.3 Plotting a second curve with the solutions

ONLY1 = F	to call USERSX to compute a second array that will be plotted with the solutions.
= T	to take no such action, i.e., to plot only the solution (as in the test output).

When ONLY1=F, the two curves are plotted in the same way that the data and the fit to the data are plotted on *Page 16* of the test output. The characters "X" and "O" are used for the solution and the second curve, respectively.

The most common use of ONLY1=F would be with SIMULA=T, where the second curve could be the exact solution that was used to simulate the data. However, this is sometimes impractical, e.g., when the exact solution is a delta-function. It is not necessary that SIMULA=T to use ONLY1=F; it is conceivable that, even without simulated data, you might want to compare your solutions with a second curve.

USERSX	(only called when ONLY1=F) computes the second curve to be plotted with the solutions.

As a Fortran argument in USERSX, you are furnished with the array G of the g_m in Eq (3.1-3) for $m = 1, \ldots, N_g$. You must put the second curve to be plotted in EXACT(J) for J = 1,...,NG.

In the version of USERSX in Files 1 and 2, the second curve to be plotted is $g_m^r \exp(-g_m)/r!$, where you must have specified RUSER(8) = r with $1 \leq r \leq 20$ (but r need not be a whole number) and where none of the g_m are negative.

4.1.6.4 Peak-Constrained Solutions

Usually the number of extrema in a solution, $s(g)$, is a good measure of its complexity. The appearance of an extra peak in $s(g)$ is often very significant. In this case, the principle of parsimony requires that you find the solution with the fewest extrema that still is consistent with the data. This section describes how you can perform Peak-Constrained Solutions, where $s(g)$ is constrained to have no more than a specified number of extrema. For example, you can perform a series of Peak-Constrained Analyses, progressively allowing more extrema with each analysis until you find a solution that is consistent with the data (e.g., one that has a PROB1 TO REJECT less than about 0.9) [4. p. 317]. This tends to protect you against artifacts. It allows you to say with more confidence that the data *require* this many extrema, since solutions with fewer extrema that were consistent with the data could not be found.

Test Data Set 3 was constructed to illustrate Peak-Constrained Analyses. The input data are shown at the end of Sec. 4.1.6.4; the output is shown in Part 2 of this manual. Suppose that you had previously performed an analysis without Peak-Constraints and that the solution on *Page 5* of the test output had been the CHOSEN SOLUTION. An important question is whether a more parsimonious

solution consistent with the data exists without the side peaks at g = 1.26E3 and g = 5.E6, or at least without the peak at g = 5.E6. You could then make a second run with Test Data Set 3 to answer this. You now use NQPROG(1)=3 and a very restricted RSVMNX to concentrate on the region of small α to get a reference solution. (If your test run aborts with ERROR ANALYZ 4, switch to Version 2DP or delete the input of values for RSVMNX and NQPROG in the test data to leave them at their default values.) You then use ALPST(2) to specify the single α that you are interested in. As explained in Sec 4.1.3, the solution with α = ALPST(2) is always taken as the CHOSEN SOLUTION. The Peak-Constrained Analyses follow and use this α of the CHOSEN SOLUTION.

If you have not previously analyzed the data and still want to perform a Peak-Constrained Analysis, then you can set ALPST=0.0. The α of the CHOSEN SOLUTION (the one with PROB1 TO REJECT closest to 0.5) will then be used for the Peak-Constrained Analysis.

As mentioned in Sec 3.6.8, a solution with a smaller α than that of the CHOSEN SOLUTION can in rare cases have fewer peaks than the CHOSEN SOLUTION. In this case it is a good idea to specify with ALPST that the Peak-Constrained Solution be performed with this small α. As described below, you can specify with NFLAT that, if this Peak-Constrained Solution is not smooth enough, the analysis be repeated with larger values of α.

A Peak-Constrained Analysis divides the grid g_m, $m = 1, \ldots, N_g$, into a fixed number of *Monotonic Regions*, in which it is specified that $s(g_m)$ must be either non-increasing or non-decreasing. A total of $N_g - 1$ extra inequality constraints of the form $s(g_m) \leq s(g_{m+1})$ or $s(g_m) \geq s(g_{m+1})$ are introduced to enforce this. Thus a single-peaked analysis with a maximum at the peak is specified by two Monotonic Regions, the first with $s(g_m)$ non-decreasing and the second with $s(g_m)$ non-increasing; the peak is at the common boundary of the two Monotonic Regions. Similarly, a double-peaked analysis is specified by four Monotonic Regions: non-decreasing, non-increasing, non-decreasing, non-increasing (or up, down, up, down). The analysis proceeds by systematically moving the common boundaries of the Monotonic Regions (i.e., the internal extrema), one grid point at a time, until a local minimum with respect to the extrema positions in the OBJ. FCTN. given by Eq (3.2.2-1) has been found. This is safe in that it guarantees a local minimum, but it becomes very inefficient as the number of regions increases, and a maximum of four regions is allowed. (The efficiency could probably be improved by using improved combinatorial methods or nonlinear or parametric programming.) A monotonic analysis specifies the entire grid as a single Monotonic Region and there are no internal extrema to move around; the number of internal extrema is simply

the number of Monotonic Regions minus one.

NNSGN(2)	NNSGN(ISTAGE) = the number of Peak-Constrained Analyses to be performed at *Stage* ISTAGE, where Stage 1 is the PRELIMINARY UNWEIGHTED ANALYSIS and Stage 2 is the rest of the analysis. You must have NNSGN(ISTAGE) ≤ 4 for ISTAGE $= 1, 2$.

In Test Data Set 1, NNSGN(ISTAGE)=0 for ISTAGE=1 and 2; so no Peak-Constrained Analyses are performed at either Stage. In Test Data Set 3, NNSGN(2)=2 (see *Page 2*); so two Peak-Constrained Analyses are performed (on *Pages 6-9* and on *Pages 10-13*). With Test Data Set 3, there is no Stage 1; so NNSGN(1)=0 is not even referenced.

NSGN(4)	NSGN(J) = the total number of Monotonic Regions for the J^{th} Peak-Constrained Analysis. You must have $1 \leq$ NSGN(J) ≤ 4 for J $= 1, \ldots,$ NNSGN(ISTAGE) and ISTAGE $= 1, 2$.

In Test Data Set 3, NSGN(1)=2 and NSGN(2)=4 (see *Page 2*). Since the number of internal extrema [i.e., not counting the endpoints $s(g_1)$ and $s(g_{N_g})$] is NSGN(J)-1, the first Peak-Constrained Analysis on *Pages 6-9* has the heading "1-EXTREMA-CONSTRAINED ANALYSIS" and the second on *Pages 10-13* "3-EXTREMA-CONSTRAINED ANALYSIS".

For each Peak-Constrained Analysis, you must specify starting positions for the extrema:

LSIGN(4,4)	FORMAT(16I5) LSIGN(J,K) = $\pm$ (the starting value of the subscript of the grid point of the J^{th} extremum for the K^{th} Peak-Constrained Analysis). LSIGN(J,K) is positive if the extremum is a maximum and negative if it is a minimum. For the purpose of specifying LSIGN, the first extremum is always the external extremum at g_1; so you must always have LSIGN(1,K) $= \pm 1$. You must then specify the rest of the internal extrema; so you must specify LSIGN(J,K) for J $= 1, \ldots,$ NSGN(K). The external extremum at g_{N_g} is obvious and is not specified.

There is one convenient exception where you do not have to specify all of

LSIGN. This is if NSGN(K)=2. You can then specify LSIGN(2,K)=0, and CONTIN will set LSIGN(2,K) for you. CONTIN uses the CHOSEN SOLUTION. It sets LSIGN(2,K) to correspond to the maximum [if LSIGN(1,K)=-1] or the minimum [if LSIGN(1,K)=1] in this solution with the largest magnitude.

This special case is illustrated in Test Data Set 3, where NSGN(1)=2 and LSIGN(2,1)=0. LSIGN(2,1) for the first Peak-Constrained Analysis (*Page 6*) is obtained from the CHOSEN SOLUTION, which is the one with $\alpha =$ ALPST(2) (*Page 5*). Since LSIGN(1,1)=-1 corresponds to a minimum, LSIGN(2,1) must correspond to a maximum, and the largest value of the solution on *Page 5* occurs at g_{12}; so LSIGN(2,1)=12. This can be seen on *Page 6*. The first iteration (ITER.=1) corresponds to this starting position of the extrema, and under the heading "EXTREMA INDICES" are printed -1 and 12, the values of LSIGN(1,1) and LSIGN(2,1). The EXTREMA INDICES show the positions of the extrema as the internal extrema are systematically moved at each iteration, positive and negative signs indicating maxima and minima, respectively.

An asterisk at the far left of an iteration line means that OBJ. FCTN. has decreased. Therefore the last iteration with an asterisk corresponds to the Peak-Constrained Solution, which is plotted. This is ITER. 1 on *Page 6* and ITER. 7 on *Page 8*. (The reason for the second 1-EXTREMA analysis on *Page 8* is given below in the discussion of NFLAT and SRMIN.)

In Test Data Set 3, NSGN(2)=4 and LSIGN(J,2)=-1, 3, -7, 13 for J $= 1, \ldots, 4$ (*Page 2*). These values of LSIGN are printed as the EXTREMA INDICES for ITER. 1 on *Page 10*. The character "X" at the far left of an iteration line means that this combination of EXTREMA INDICES has already been analyzed and the value of OBJ. FCTN. stored from the first time will be used. For example, ITER. 12 is identical to ITER. 1. The solution from ITER. 3 is the Peak-Constrained Solution.

The 1-EXTREMA analysis on *Page 6* had no trouble cleanly eliminating the peak at $g = 1260$ in the solution on *Page 5*. Therefore this peak is not demanded by the data. [It was purposely induced by using NORDER=3 rather than NORDER=2, as in Test Data Set 1.] Of course the other peak at $g =$ 5.E6 is also formally gone, in that the constraints have forced $s(g_m)$ to be non-increasing for $m \geq 12$. However, the peak has been replaced with a *Flat Spot* or plateau with $s(g_{15}), \ldots, s(g_{21}) =$ 1.553E-14. Flat Spots are usually more unrealistic and unparsimonious than extra peaks. Flat Spots can occur when α is too small or when the peak being forced out is really demanded by the data. To see if a larger α can eliminate the Flat Spot, you can specify that, if a Flat Spot is detected, CONTIN repeat the Peak-Constrained Analysis with α increased (by

the same factor that it was increased in the analyses without Peak-Constraints, as on *Pages 3-4*):

NFLAT(4,2)	FORMAT(8I5) NFLAT(J,K) = the maximum number of Peak-Constrained Solutions that will be attempted in the J^{th} Peak-Constrained Analysis at Stage K if Flat Spots are detected. (Stage K for K = 1,2 is defined with NNSGN). NFLAT(J,K)=0 or 1 means that no extra solutions will be attempted if a Flat Spot is detected.

SRMIN	A Flat Spot is defined by two successive binding constraints with $s(g_m) = s(g_{m+1}) = s(g_{m+2})$. However, if NONNEG=T, then Flat Spots with $s(g_m) <$ SRMIN*SMAX, where SMAX is the maximum of $s(g_j)$, $j = 1,\ldots,N_g$, are considered insignificant and are not detected. An SRMIN = $O(10^{-2})$ or $O(10^{-3})$ is usually reasonable.

Obviously, when NONNEG=T a Flat Spot at $s(g_m) = 0$ (or nearly zero) is generally reasonable, and SRMIN can prevent the analysis from being repeated just because of this. When NONNEG=F, Flat Spots can sometimes be reasonable. For example, if, instead of nonnegativity constraints, you had $s(g_m) \geq r$ or $s(g_m) \leq r$ for $m = 1,\ldots,N_g$, a Flat Spot at $s(g_m) = r$ would probably be reasonable. For this reason, it is best to transform $s(g)$ in Eq (3.1-2) so that these constraints become nonnegativity constraints; you can then use NONNEG=T and a positive SRMIN. Otherwise you would probably have to use NFLAT(J,K)=0. Another important reason to use NONNEG=T whenever possible is that CONTIN has special provisions for temporarily replacing the nonnegativity constraints with the monotonicity constraints during the Peak-Constrained Analyses. Otherwise N_g extra monotonicity constraints are added and much more storage is required.

In Test Data Set 3, NFLAT(1,2)=3 and SRMIN=5.E-4 (*Page 2*). This causes the Flat Spot on *Page 6* to be detected, and a second solution with a larger α is tried on *Page 8*. This also has a Flat Spot, but 1.452E-14 < SRMIN*SMAX; so no third solution is performed. On *Page 11*, there is also a Flat Spot that would be detected, but NFLAT(2,2)=0; so no second solution is performed.

The Peak-Constrained Analyses were not able to cleanly eliminate the peak at g = 5.E6. Flat Spots would not disappear, even when α was so large that PROB1 TO REJECT exceeded 0.999. Thus the peak is demanded by the data. This is not surprising, since this peak was purposely induced by setting NLINF=0, and the peak at g = 5.E6 attempts to compensate for the missing "background" term that was in Test Data Set 1. Although the magnitude of the peak is small, its effect is substantial, as can be seen by comparing the MOMENTS for the TOTAL CURVE on *Page 12* or *31* for Test Data Set 1 with those on *Page 4*.

Each Peak-Constrained Solution is followed by plots of the residuals and the fit to the data (on *Pages 7, 9, 12*). The corresponding plots for the CHOSEN SOLUTION are on *Page 13*. All of these plots, as well as PRUNS and PUNCOR, have correlated residuals and fits inferior to those with Test Data Set 1 (*Page 16* and *30*). Thus the constant background term (or "dust" term in the context of photon correlation spectroscopy) is essential.

It is usually a waste of computer time to use the Peak-Constrained Solutions during the PRELIMINARY UNWEIGHTED ANALYSIS just to fit a smooth curve through the data to compute least squares weights. Therefore NNSGN(1)=0 is recommended. If for some reason you do use a positive NNSGN(1), then the Peak-Constrained Solution with PROB1 TO REJECT closest to 0.5 will be taken as the CHOSEN SOLUTION and used to compute the weights; i.e., the preceding solutions without Peak-Constraints are not considered.

The quadratic programming algorithm is guaranteed to find the unique global minimum in OBJ. FCTN. for the given constraints. However, the systematic variation of the extrema indices only guarantees that a local minimum with respect to the extrema indices is found. Thus a 3-extrema solution with a lower OBJ. FCTN. than that on *Pages 10-11* might be found by using a different LSIGN to start with a different set of extrema indices. This is less likely to happen in a 1-extremum analysis, and it is in any case easy to check by making a few runs with different LSIGNs. However, 3-extrema analyses are too expensive to exhaustively perform. This prevents you from making absolute statements like, "There exists no parsimonious double-peaked solution consistent with the data", but analyses from a few carefully chosen LSIGNs can allow you to say that it is unlikely that such a solution exists.

2-extrema and 3-extrema analyses can be prohibitively inefficient and expensive if you have used a poor LSIGN. Each iteration in a Peak-Constrained Analysis is about as expensive as a solution without Peak Constraints. To protect yourself against excessive iterations:

MQPITR =	the maximum allowed number of iterations in any Peak-Constrained Analysis. A MQPITR=50 or 100 can be necessary for a 3-extrema analysis. For a 1-extremum analysis, MQPITR=N_y is sufficient.

The analysis will be interrupted with an error message if more than MQPITR iterations are needed, and the run proceeds to the next analysis as if a Flat Spot had been found. When the character "X" appears at the far left of an iteration line, this iteration is skipped and therefore not counted as far as MQPITR is concerned. Information on each iteration is stored in arrays with DIMENSION specification MDONE (see Sec 3.5); therefore MDONE=MQPITR is sufficient.

In Test Data Set 3, MQPITR=90 (*Page 2*). An MQPITR=46-9=37 would have been just enough to get through the analysis on *Page 10*.

```
TEST DATA SET 3 (FOR PEAK-CONSTRAINED SOLUTIONS)
NG                    21.
GMNMX     1         5.E+2
GMNMX     2         5.E+6
NINTT                  3.
IFORMY
(6F8.6)
IUSER    10            1.
RUSER    15         1.43
RUSER    16         488.
RUSER    17          60.
RUSER    18       1.37E-4
RUSER    22          -.5
RUSER    10          -1.
RSVMNX
    1.E4    1.E-10
NQPROG    1            3.
NORDER                 3.
NENDZ     1            1.
ALPST     2       7.962E-6
LSIGN
  -1    0    0    0   -1    3   -7   13
NFLAT
   0    0    0    0    3
SRMIN                5.E-4
NNSGN     2            2.
NSGN      2            4.
NSGN      1            2.
MQPITR                90.
END
NSTEND   17         5.E-6         85.E-6
```



```
NSTEND   16         95.E-6        245.E-6
NSTEND    4        265.E-6        325.E-6
.450999 .410113 .372522 .340069 .310318 .283569
.258853 .236028 .216811 .199376 .181524 .165491
.153746 .139687 .128724 .117704 .109878 .094114
.080559 .068725 .058679 .053363 .045275 .039581
.033519 .031586 .027971 .023976 .021711 .021533
.020312 .016487 .017212 .016077 .011657 .013386
.010805
```

4.1.6.5 Selection criterion for the CHOSEN SOLUTION

You should never change the two Control Variables described below. They are included here only for completeness.

ICRIT(2)	ICRIT(K)=1 to decide the CHOSEN SOLUTION using PROB1 TO REJECT at Stage K. =2 to decide the CHOSEN SOLUTION using PROB2 TO REJECT at Stage K. (Stage K for K = 1, 2 is defined with NNSGN). You should always use ICRIT(K)=1.

PLEVEL(2,2)	FORMAT(4F5.2) If ICRIT(K)=1, then the CHOSEN SOLUTION at Stage K will be the one with PROB1 TO REJECT closest to PLEVEL(1,K). If ICRIT(K)=2, then the CHOSEN SOLUTION at Stage K will be the one with PROB2 TO REJECT closest to PLEVEL(2,K). You should always use PLEVEL(J,K)=0.5 for J,K = 1, 2.

Equation (3.23) of [2] defines

$$\text{PROB1 TO REJECT} = P(F_1; \text{NDFZ}, N_y - \text{NDFZ}), \tag{1}$$

where $P(F; n_1, n_2)$ is Fisher's F-distribution function with n_1 and n_2 degrees of freedom, NDFZ is the effective number of degrees of freedom in the reference solution [the solution with the minimum VAR in Eq (3.2.1-1)], and

$$F_1 = \frac{\text{VAR} - \text{VARZ}}{\text{VARZ}} \frac{N_y - \text{NDFZ}}{\text{NDFZ}} \tag{2}$$

where VARZ is the VAR of the reference solution.

$$\text{PROB2 TO REJECT} = P(F_2; \text{NDFZ} - \text{NDF}, N_y - \text{NDFZ}), \tag{3}$$

where NDF is the effective number of degrees of freedom in the solution being tested and

$$F_2 = \frac{\text{VAR} - \text{VARZ}}{\text{VARZ}} \frac{N_y - \text{NDFZ}}{\text{NDFZ} - \text{NDF}} \tag{4}$$

If NDFZ − NDF < 0.1, then PROB2 TO REJECT is arbitrarily set to 1.0.

PROB2 TO REJECT actually seems like a more natural criterion than PROB1. It is analogous to testing a linear hypothesis in linear regression [11, p. 74] that the VAR of the solution with NDF degrees of freedom is significantly larger than the reference, VARZ. The problem is that when there are binding constraints, the definition of degrees of freedom is not clear (although some progress seems to have been made recently in choosing α when there are constraints [14]). Our definition of NDF does not lead to a PROB2 that monotonically increases with α, and it can even happen that NDFZ − NDF is negative. When there are no constraints, PROB2 and PROB1 usually lead to similar results; so it is recommended that you always use ICRIT(K)=1 and ignore PROB2 TO REJECT. See Sec 3.6 of [2] for further reasons for using PROB1.

4.2 More on the Composition of a Data Set

Some of the USER subprograms may require input data. Below is listed the order in which any input data must occur in the input data deck.

Card 1 - Card Set 7 (standard input data - see Table 3 of [19])		
Card Set 8	USERIN	
Card Set 10	USERGR	
Card Set 11	USERSI	
Card Set 12	USERSX	
Card Set 13	USERNQ	
Card Sets 14a,b	USERRG	
Card Set 15a	USEREQ	
Card Set 16	USERWT	
Card Sets 14c,d	USERRG	(only if IWT=2, 3, or 5)
Card Set 15b	USEREQ	(only if IWT=2, 3, or 5)

Obviously you only need a Card Set if the corresponding USER subprogram is called and if it requires input data. There is no Card Set 9.

When IWT=1 or 4, USERRG is called twice. When IWT=2, 3, or 5, USERRG is called 4 times and USEREQ twice. You can save inputting the same data 2 or 4 times by storing the data as Card Set 14a or 15a is read and using a flag to remember this and to skip the input of Card Set 14b,c, and d and 15b. This

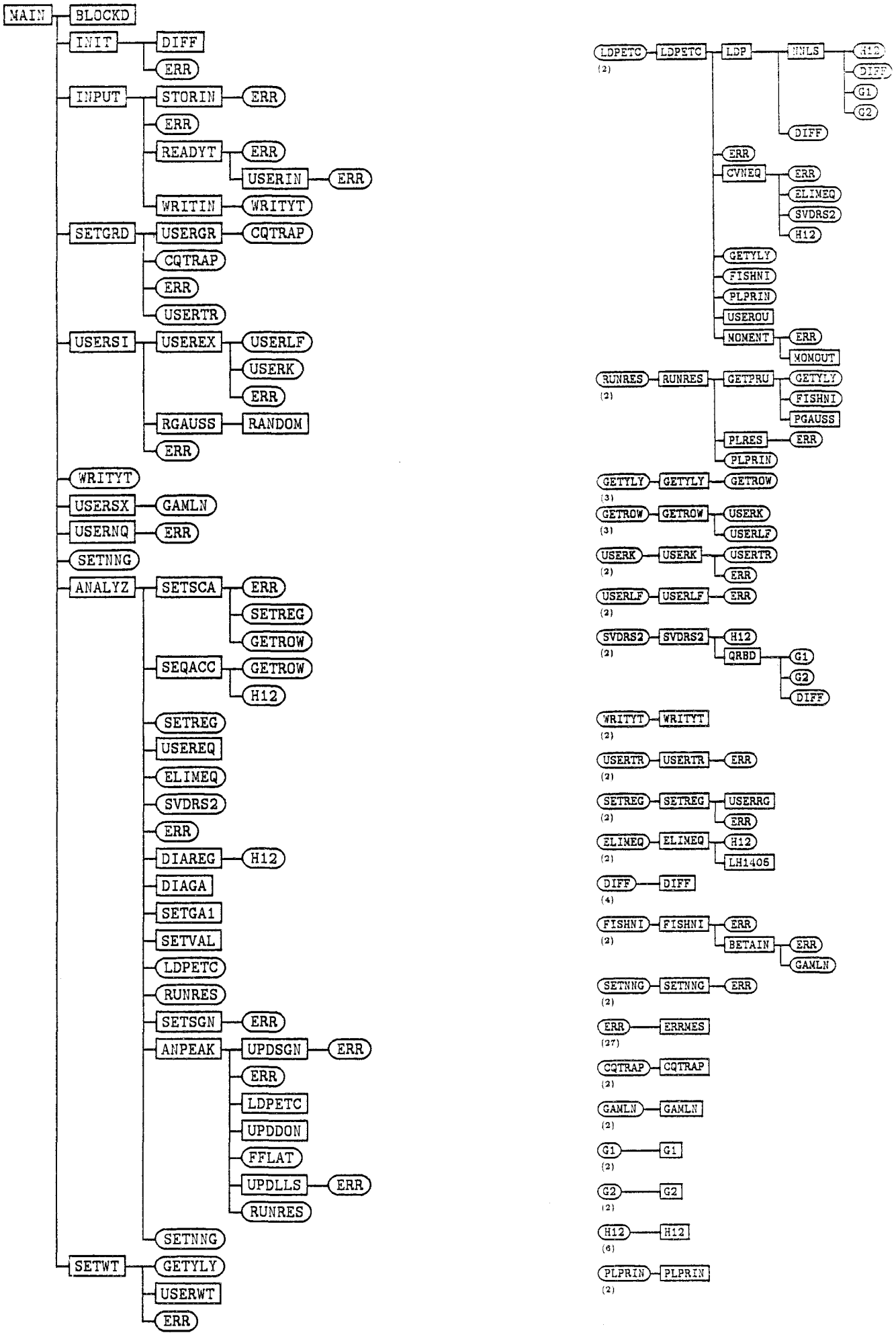
is illustrated in the version of USERRG in Files 1 and 2. You can also put the necessary data into a USER array with Card Set 2 or in USERIN and thereby avoid using Card Set 14a or 15a completely. This is illustrated in the version of USEREQ in Files 1 and 2.

4.3 CALLing Diagram for the Subprograms

It is not necessary to read this section to use CONTIN. With the diagram below, you can trace the calling sequences for all the subprograms for the version of CONTIN in Files 1 and 2. In Files 1 and 2, the comment cards at the beginning of each subprogram usually give you at least a rough idea of the tasks being performed there. See also Table 1 of [19].

In the diagram below, MAIN and BLOCKD represent the main subprogram and the BLOCK DATA subprogram, respectively. All the other names in rectangles are the actual names of subprograms, and each name only appears once. Connecting lines run from the rectangle of a subprogram indicating all the subprograms that it calls. The calling sequence is approximately from top to bottom.

Each oval is a block of one or more subprograms that is called in more than one place in the diagram. Each oval is defined by a separate diagram to the right or below where the oval is called. In the diagram of an oval, the number in parentheses below the oval is the number of places in the other diagrams above and to the left that the oval appears.


MAIN
BLOCKD
INIT
DIFF
ERR
INPUT
STORIN
ERR
ERR
READYT
ERR
USERIN
ERR
WRITIN
WRITYT
SETGRD
USERGR
CQTRAP
CQTRAP
ERR
USERTR
USERSI
USEREX
USERLF
USERK
ERR
RGAUSS
RANDOM
ERR
WRITYT
USERSX
GAMLN
USERNQ
ERR
SETNNG
ANALYZ
SETSCA
ERR
SETREG
GETROW
SEQACC
GETROW
H12
SETREG
USEREQ
ELIMEQ
SVDRS2
ERR
DIAREG
H12
DIAGA
SETGA1
SETVAL
LDPETC
RUNRES
SETSGN
ERR
ANPEAK
UPDSGN
ERR
ERR
LDPETC
UPDDON
FFLAT
UPDLLS
ERR
RUNRES
SETNNG
SETWT
GETYLY
USERWT
ERR
LDPETC
(2)
LDPETC
LDP
NNLS
H12
DIFF
G1
G2
DIFF
ERR
CVNEQ
ERR
ELIMEQ
SVDRS2
H12
GETYLY
FISHNI
PLPRIN
USEROU
MOMENT
ERR
MOMOUT
RUNRES
(2)
RUNRES
GETPRU
GETYLY
FISHNI
PGAUSS
PLRES
ERR
PLPRIN
GETYLY
(3)
GETYLY
GETROW
GETROW
(3)
GETROW
USERK
USERLF
USERK
(2)
USERK
USERTR
ERR
USERLF
(2)
USERLF
ERR
SVDRS2
(2)
SVDRS2
H12
QRBD
G1
G2
DIFF
WRITYT
(2)
WRITYT
USERTR
(2)
USERTR
ERR
SETREG
(2)
SETREG
USERRG
ERR
ELIMEQ
(2)
ELIMEQ
H12
LH1406
DIFF
(4)
DIFF
FISHNI
(2)
FISHNI
ERR
BETAIN
ERR
GAMLN
SETNNG
(2)
SETNNG
ERR
ERR
(27)
ERRMES
CQTRAP
(2)
CQTRAP
GAMLN
(2)
GAMLN
G1
(2)
G1
G2
(2)
G2
H12
(6)
H12
PLPRIN
(2)
PLPRIN

5. DIAGNOSTICS AND HINTS FOR TROUBLE-SHOOTING

5.1 Nonstandard Diagnostics and Aborts

CONTIN performs many tests for error conditions, and there are more than 50 diagnostics. Most of these take a standard form, "ERROR Name n. (CHECK USERS GUIDE.)", where Name is the name of the subprogram where the error occurred and n is an integer. If one of these messages occurs, see Sec 5.3, where they are explained.

However, certain errors, especially during input, can occur before there is a chance for a diagnostic. For example if IFORMY, IFORMT, or IFORMW do not contain a legal Fortran specification enclosed in parentheses, then the run may abort with no error message.

If an end-of-file is encountered while trying to read input data, then check the following:

(1) You must have an END card (Card 3 in Table 3 of [19]).

(2) Has a Control Variable been set such that a USER subprogram requiring input data has been inadvertently called?

(3) The input and output files NIN and NOUT must be properly opened (Sec 2.3.4) if your system does not do this automatically for you. Similarly, if you are using a scratch file, then file IUNIT must be properly opened (Sec 4.1.6.1).

As discussed in Sec 2.3.2, you must insure that your computer will replace underflows with zero; many underflows are expected to occur.

There are a few diagnostics that do not take the standard form. Several of these are in the USER subprograms, but they are self-explanatory. You should include ample diagnostics in your versions of USER subprograms. One fairly common nonstandard diagnostic is "MAX. ITERATIONS IN NNLS FOR ALPHA/S(1)=..." (as on Page 3 for Test Data Set 3). This can be expected to occasionally occur for very small α values, especially in Version 2DP. This solution is then skipped and the next larger value of α is tried.

5.2 General Hints for Trouble-Shooting

Usually the cause of the diagnostic is apparent. Often it is due to incorrect input data or to a forbidden combination of Control Variables or USER subprograms. Sometimes, however, the cause is not obvious. There may not even

be a diagnostic or an abort, just meaningless results. One possibility is that there is an error in CONTIN. It is obviously impossible to test all combinations of Control Variables and USER subprograms. You should send us your output, as well as compilation listings of any subprograms that you have modified, information on the version, date, and Application Package used, and excessive amounts of data in machine-readable form. This will help us maintain CONTIN.

First, however, you should make sure that you have not caused the error. Some suggestions:

(1) Check all changes that you have made. To be sure, you might want to make the changes again to a copy of your original file copy of CONTIN and rerun.

(2) In your versions of the USER subprograms, have you exceeded any of the DIMENSION specifications of any arrays? The actual DIMENSION specifications are given in the MAIN subprogram (see also Sec 3.5).

(3) In your versions of the USER subprograms, have you changed any of the COMMON variables to illegal values? It is not recommended that you change these ever, except in the few cases where you are explicitly instructed to do so in a USER subprogram. Extensive tests on these variables are made right after the input, but this does not help if you change them to illegal values later.

(4) Do the actual DIMENSION specifications in Lines 56–65 of the main subprogram correspond to those in the DATA statement in Lines 100–101, as specified in Lines 69–78? This is of course crucial, since the DIMENSION specifications passed to the subprograms would otherwise be wrong.

(5) Grossly erroneous y_k, t_k, constraints, etc. can cause problems, sometimes without diagnostics. For example, if all your y_k were negative, but your constraints implied that they could only be positive, then all the analyses might fail to converge.
Check that GMNMX is reasonable. An excessively broad range can cause numerical problems, which are often apparent by the reference solution having a very small DEG FREEDOM or a poor fit to the data. An excessively narrow range of GMNMX can cause serious cutoff errors in evaluating the integral in Eq (3.1–2), which is sometimes, but not always, apparent by a poor fit to the data.

(6) A run with very simple simulated data (e.g., with a high signal-to-noise ratio and a delta-function solution) could be helpful.

(7) A run with a good debugging compiler like WATFIV (Sec 2.3.3) could be helpful.

(8) If you obtain errors that you cannot explain and have compiled with an optimization switch on (often the default setting), try compiling at least the subprogram where the error occurred with the optimization switch off. For example, with all versions of the *VAX 11/780* Fortran compiler that we have had in the past two years, an abort occurs occasionally in the subprogram MOMENT with a diagnostic that can only be due to incorrect compilation. When this single subprogram is compiled with the /NOOPTIMIZE switch, the program runs correctly. These types of problems have occurred with other programs and other optimizing compilers.

5.3 Standard Diagnostics

The standard diagnostics are of the form:

ERROR Name *n*. (CHECK USERS GUIDE)...,

where Name is the name of the subprogram where the error occurred and *n* is an integer that uniquely identifies the error in Name. These diagnostics are listed below (with Name in alphabetical order). Following Name *n* is one of the following three abbreviations in parentheses:

(F) means a fatal error. The run is aborted.

(NF) means a non-fatal error. The run continues, possibly after taking corrective action. Usually the corrective action is to skip part of the analysis immediately following the error and jump to a new part.

(W) means a warning. CONTIN has detected a peculiar condition, but it is not necessarily an error. The run continues.

The error or warning is then explained, and *possible remedies* are given after the abbreviation "PR:". Several possible remedies numbered (1), (2), etc. are alternatives in descending order of priority; generally only one is necessary to correct the problem, not all of them.

It may happen that the PR do not help or that they seem irrelevant. For example, the PR may suggest checking your modification to USERRG when you never even modified it or called it. If this happens, then try the suggestions in Sec 5.2. If these suggestions do not help, then send me the information listed in Sec 5.2. Also send me this information if the error is described as an *Illogical Stop*. This indicates a condition that is apparently due to a programming error of mine. Of course, if you make certain wide-ranging errors in your

modifications of CONTIN, like those described in points (2), (3), and (4) in Sec 5.2, then nearly anything can happen, perhaps including an Illogical Stop. So you should first check the points in Sec 5.2 when the PR do not help or an Illogical Stop occurs.

Cross references are usually not made to the equations in Secs 3.1 and 3.2 when symbols defined there are used or to other sections when terms indexed in Sec 8 are used. So you should use these indexes.

ANALYZ 0 (F) E in Eq (3.1–5) has a row of all zeroes.

PR: Correct your version of USEREQ or input NEQ=0.

ANALYZ 1 (F) NREG (= N_{reg}) or NGLE (= NG + NLINF − NEQ) is negative or the singular value decomposition of the regularizor matrix R failed to converge.

PR: If IGRID=3, then be sure that in USERRG you have set NREG positive; if so, then try slightly modifying your definition of the regularizor in USERRG, e.g., to make the terms on and near the main diagonal of the matrix R more dominant and of similar magnitude. Check that any equality constraints are consistent and that you have not changed NEQ in USEREQ. (NGLE has already been verified as positive.)

ANALYZ 2 (F) NGLY [= min (NG + NLINF, NY)] or NGLE (= NG + NLINF − NEQ) is negative or the singular value decomposition that would have produced the SINGULAR VALUES (Sec 3.6.4) failed to converge.

PR: Check for gross errors or inconsistencies in the y_k, the regularizor or the constraints. If nothing is found, then try Version 2DP or minor changes to some of these. (NGLY and NGLE have already been verified as positive.)

ANALYZ 3 (F) None of NQPROG(1), NQPROG(2), or ALPST is positive. Therefore no analyses are to be performed.

PR: Set at least one of these three positive.

ANALYZ 4 (F) RSVMNX(2,J)/(RSVMNX(1,J)*PRECIS), the ratio of the largest α to the smallest α on the α-grid is 1.0 or less for J=1 or 2.

PR: Increase RSVMNX(2,J), reduce RSVMNX(1,J), or set NQPROG(J) to 1 or 0.

ANALYZ 5 (F) All NQPROG(1) and NQPROG(2) quadratic programming analyses failed to converge.

PR: **(1)** Increase RSVMNX(2,J) to about 1.0 for J=1 or 2. **(2)** Check points (1)–(7) in Sec 5.2.

ANALYZ 6 (F at Stage 1; NF at Stage 2) There is not enough storage for the extra inequality constraints needed for the J^{th} Peak-Constrained Analysis. The DIMENSION Parameter MINEQ must satisfy MINEQ $\geq$ N_{ineq} + NG − 1, if NONNEG=F, and MINEQ $\geq$ N_{ineq} + NSGN(J)/2, if NONNEG=T.

PR: **(1)** Set MINEQ (see Sec 3.5) to satisfy the above requirement. **(2)** If NONNEG=F and N_{ineq} is large, then try to redefine $s(g)$ in Eq (3.1–2) so that NONNEG=T can be used with a resultant reduction of N_{ineq} to N_{ineq} − NG. **(3)** Do without Peak-Constrained Analyses by setting NNSGN(K)=0, K=1,2.

ANALYZ 7 (F at Stage 1; NF at Stage 2) Same as ANALYZ 6.

ANPEAK 1 (NF) The Peak-Constrained Analysis still has not found a solution after MQPITR iterations. CONTIN will skip to the next analysis.

PR: **(1)** Give better starting values for the extrema positions in LSIGN. **(2)** Increase MQPITR.

ANPEAK 2 (NF) In a Peak-Constrained Analysis, the number of iterations calling the quadratic programming algorithm would exceed MDONE, and there would be no more storage for information on the iterations. CONTIN will skip to the next analysis as if a Flat Spot had been found for this analysis.

PR: **(1)** Give better starting values for the extrema positions in LSIGN. **(2)** Increase MDONE (see Sec 3.5). MDONE=MQPITR is sufficient.

BETAIN 1 (F) The arguments of BETAIN are out of range. This should only occur if NY $\geq$ 40000; otherwise it is an Illogical Stop.

PR: Reduce NY to be less than 40000.

BETAIN 2 (F) Illogical Stop. It can only occur if convergence to the incomplete beta function was still not attained after 20000 iterations.

CVNEQ 1 (F) NBIND, the number of active inequality constraints, is greater than or equal to NGLE, the original number of free parameters. Thus there are no free parameters left. NBIND and NGLE are printed below the diagnostic.

PR: **(1)** Check your modifications of USEREQ or USERNQ for redundant constraints. **(2)** Check the points in Sec 5.2, especially point (5).

CVNEQ 2 (NF) The singular value decomposition in CVNEQ failed to converge. DEGFRE (=N_{DF}, the effective number of degrees of freedom) is arbitrarily set to 0.0, and the run continues.

PR: None, except perhaps to check the points in Sec 5.2. (We have never received this diagnostic.)

FISHNI 1 (W) One of the degree-of-freedom arguments in Eq (4.1.6.5–1) or (4.1.6.5–3) is not positive. This can happen if **(1)** the number of degrees of freedom equals the number of data points or **(2)** ERROR CVNEQ 2 has occurred. In either case, PROB1 or PROB2 TO REJECT has no meaning and the value 1.0 is arbitrarily assigned to it. The run continues.

PR: **(1)** Reduce NG or NLINF so that NG + NLINF < NY. **(2)** See PR of CVNEQ 2.

INIT 1 (F) The accuracy of the REAL arithmetic is less than about 5 significant figures (if you are using Version 2SP) or the accuracy of DOUBLE PRECISION is less than about 9 figures (if you are using Version 2DP).

PR: In the likely case that your computer really has more accuracy, this is an Illogical Stop. Otherwise, try changing Versions.

INPUT 1 (F) The card (in your Card Set 2) printed above does not have a valid Control Variable name in columns 2–7. This is a fatal error, but CONTIN will continue checking the rest of the data set for errors until it has found MIOERR errors or until it has reached the end of the data set.

PR: (This is the most common error. See Sec 3.3 of [19].) **(1)** Correct the name of the Control Variable. **(2)** Did you erroneously use two cards for the preceding Control Variable when it requires one only? **(3)** Check that Cards 1 and 3 are correct.

INPUT 2 (F) At least one of the following requirements is not met:

$0.0 \le \text{PLEVEL(J,K)} \le 1.0, \quad \text{J,K} = 1,2$

$1 \le \text{ICRIT(K)} \le 2, \quad \text{K} = 1,2$

$\text{RSVMNX(J,1)} > 0, \quad \text{J} = 1,2, \text{ if } \text{NQPROG(1)} > 0$

$\text{NG} \ge 2$

$\text{NG} + \text{NLINF} > \text{NEQ}$

$\text{NLINF, NEQ} \ge 0$

$1 \le \text{IGRID, IQUAD} \le 3$

$1 \le \text{IWT} \le 5$

$\text{NORDER} \le 5$

$\text{MREG} \ge \text{NG} + \text{NLINF} + 1$

or Eqs (3.5–1)–(3.5.3), (3.5–5), or (3.5–7). You can directly check all of these relations because the final values of the Control Variables will have been output (as on Page 2 of the test output) and the values of the 7 relevant DIMENSION Parameters (Sec 3.5) are output following the diagnostic.

LDPETC 1 (F) Illogical Stop. [This can only occur if NGLE (= NG + NLINF − NEQ) is negative, and this is tested earlier in CONTIN.]

LDPETC 2 (NF) Inequality constraints are incompatible. If this error only occurs for one or two α values, then it may be possible to use the results anyway, provided that condition (1) of Sec 3.6.5 is satisfied. If it occurs for every α, then the following steps must be taken:

PR: **(1)** Correct your modification of USEREQ or USERNQ so that all constraints are compatible. **(2)** Roundoff error can sometimes destroy the compatibility of constraints. For example, with NONNEG=T, it may be necessary to change the constraint $G(1) = 0$ to $G(1) = \text{SMALL}$, where SMALL is a small positive number. Otherwise SMALL=0 can be effectively changed to a small negative number during the computation and therefore become incompatible with NONNEG=T. **(3)** Use Version 2DP instead of Version 2SP.

MOMENT 1 (F) Illogical stop. The argument STDDEV was negative in the calling program.

MOMENT 2 (F) Illogical stop. There are two g values that are zero.

MOMENT 3 (NF) A $g=0$ and a negative J would have caused division by zero in computing MOMENT(J).

PR: Correct the input of MOMNMX or the setup of the g-grid.

PLRES 1 (NF) The plot of the residuals is being skipped because LINEPG < 17 and this does not provide enough space per page for the plot.

PR: **(1)** Increase LINEPG. **(2)** Do without the plot by setting IPLRES=0.

PLRES 2 (W) The plot of the residuals is being interrupted after MPAGE (=30) pages of output from this single plot.

PR: **(1)** Do without the plot by setting IPLRES=0. **(2)** If NY is so large and you still want plots of the residuals, then increase MPAGE in the DATA statement in Line 3902.

READYT 1 (F) While reading Card Set 4 (Sec 3.3 of [19]), a card without the characters "NSTEND" in columns 2–7 was encountered. The card

is printed above the diagnostic.

PR: **(1)** Correct columns 2–7. **(2)** Is NINTT correct? **(3)** Are Card 3 and Card Set 4 in the proper positions in the input data deck?

READYT 2 (F) While reading Card Set 4 (Sec 3.3 of [19]), a card was encountered with NT < 2 or NT so large that NY would exceed MY and therefore violate Eq (3.5–3). The card is printed above the diagnostic.

PR: **(1)** Correct NT. (You can always use NT at least 2, regardless of the spacing of the T values.) **(2)** If the spacing of the T values is very irregular, it may be easier to use NINTT=0 and Card 5a and Card Set 5b. **(3)** If NT is correct, then increase MY (Sec 3.5).

READYT 3 (F) Card 5a (Sec 3.3 of [19]) does not contain the characters "NY" followed by 4 blanks in columns 2–7. The card is printed above the diagnostic.

PR: **(1)** Correct columns 2–7. **(2)** Is NINTT correct? **(3)** Are Card 3 and Card 5a in the proper positions in the deck?

READYT 4 (F) NY on Card 5a (Sec 3.3 of [19]) exceeds MY and therefore violates Eq (3.5–3). Card 5a is printed above the diagnostic.

PR: **(1)** Correct NY. **(2)** Increase MY (Sec 3.5).

READYT 5 (F) You have input a negative least squares weight. The W(K), K = 1,..., NY are printed after the diagnostic.

PR: **(1)** If you used DOUSIN=T and computed the weights in USERIN, then correct this. **(2)** Otherwise, you must have used IWT=4; correct IFORMW or Card Set 7 (Sec 3.3 of [19]).

SETGRD 1 (W) This is just a reminder that IGRID=2 and that GMNMX(1) or GMNMX(2) is not positive. When IGRID=2, some $H(g)$ (Sec 4.1.2.1) have no meaning and will cause an abort in USERTR if g is not positive. This is the case for $H(g) = \ln(g)$ in the version of USERTR in Files 1 and 2. Of course, another $H(g)$ may be perfectly reasonable with g not positive.

PR: If an abort occurs in USERTR, then correct GMNMX(1) or GMNMX(2).

SETGRD 2 (F) The grid points are not strictly monotonic; i.e., you have violated the requirement that either $g_1 > g_2 > \ldots > g_{NG}$ or $g_1 < g_2 < \ldots < g_{NG}$. The grid points are printed after the diagnostic.

PR: If IGRID=2, then correct your version of USERTR. If IGRID=3, then correct your version of USERGR or your input data for the g-array in USERGR.

SETGRD 3 (F) IQUAD is not 1, 2, or 3. However, this should have already led to ERROR INPUT 2. Therefore, you must have somehow overwritten IQUAD since then.

PR: Check points (1)–(7) in Sec 5.2.

SETNNG 1 (F) NINEQ > MINEQ.

PR: **(1)** Correct your version of USERNQ. **(2)** Increase MINEQ. **(3)** Input NONNEG=F or DOUSNQ=F.

SETREG 1 (F) You have not set NREG positive in USERRG.

PR: **(1)** Correct your version of USERRG so that NREG is properly set. **(2)** Use a nonnegative NORDER so that USERRG is not called.

SETREG 2 (F) NORDER > 5; but this should already have led to ERROR INPUT 2. Therefore, you must have somehow overwritten NORDER since then.

PR: Check points (1)–(7) in Sec 5.2.

SETREG 3 (NF) Eq (4.1.2.8–1) is violated. If NENDZ(J) < 0, then CONTIN sets NENDZ(J)=0. If NENDZ(J) > NORDER, then CONTIN sets NENDZ(J)=NORDER. The run then continues.

PR: Correct NENDZ or NORDER.

SETREG 4 (F) One of the following two requirements has been violated: NREG < MREG and NG + NLINF − NEQ < MREG. Since ERROR INPUT 2 did not occur, the only possibility is that you have set NREG ≥ MREG in USERRG.

PR: **(1)** Increase MREG (Sec 3.5). **(2)** Correct your version of USERRG so that NREG < MREG. **(3)** Do not use a negative NORDER, thereby avoiding the use of USERRG.

SETREG 5 (F) All elements of the scaled regularizor matrix, R, in Eq (3.2.2–1) are zero.

PR: **(1)** Correct your version of USERRG. **(2)** Rescale the variables g and t so that the elements of A and R are closer to unity.

SETSCA 1 (F) Illogical Stop. This could only occur if an internal scale factor had been somehow set zero or negative.

SETSCA 2 (F) All elements of the regularizor matrix R in Eq (3.2.2–1) are zero.

PR: Correct your version of USERRG so that at least some of the elements of R are non-zero.

SETSCA 3 (F) A column of the coefficient matrix, A, in Eq (3.1) has all zeroes.

PR: Correct USERK and USERLF or the scaling of g and t so that at least some elements in each column of A are nonzero.

SETSCA 4 (F) The scaled matrix of inequality constraints, D, in Eq (3.1–4) has a row of all zeroes.

PR: **(1)** Correct your version of USERNQ. **(2)** Rescale the variables g and t so that the elements of A, R, and D (Sec 8.1) are closer to unity.

SETSGN 1 (F) In the J^{th} Peak-Constrained Analysis, one of the following requirements was not satisfied:
$1 \leq$ NSGN(J) ≤ 4 and LSIGN(1,J) $= \pm 1$.

PR: Correct NSGN, LSIGN, or NNSGN.

SETSGN 2 (F) In the J^{th} Peak-Constrained Analysis, one of the following requirements of LSIGN(K,J), for K $= 1, \ldots,$ NSGN(J) is violated: **(1)** its absolute value must be strictly increasing with increasing K, **(2)** its signs must alternate with increasing K, and **(3)** |LSIGN(K,J)| $<$ NG.

PR: Correct LSGN, NSGN, or NNSGN.

SETWT 1 (F) Illogical Stop. (This can only occur if IWT is not 2, 3, or 5 in SETWT.)

SETWT 2 (F) IWT=2 or 3 and YSAFE$_k = 0.0$ (see Sec 4.1.2.7). Therefore the expression for w_k would be 1.0/0.0 and division by zero would be attempted. This is very unlikely, since all NERFIT residuals as well as YFIT$_k$ would have to be zero.

PR: Increase NERFIT or change IWT.

STORIN 1 (F) The input value in the E15.6 field (Sec 3.3 of [19]) for setting the value of a LOGICAL Control Variable is not 1. or -1.

PR: Correct card printed above this error message.

STORIN 2 (F) The input value in the I5 field (Sec 3.3 of [19]) for the subscript of a Control Variable is out of range.

PR: Correct card printed above this error message.

UPDLLS 1 (F) Illogical Stop. (A programming error of mine has caused a counter to be less than 1.)

UPDSGN 1 (F) Illogical Stop. In the J^{th} Peak-Constrained Analysis, the positions of two neighboring extrema have swapped positions and changed order.

UPDSGN 2 (F) Illogical Stop. (This can only occur if MINEQ is not large enough for the extra constraints in a Peak-Constrained Analysis, but this should have already led to ERROR ANALYZ 7.)

UPDSGN 3 (F) Illogical Stop. (The number of inequality constraints has increased during a Peak-Constrained Analysis.)

See Applications Packages (Sec 6) for more or different diagnostics in the USER subprograms.

USEREX 1 (F) One of the following requirements is not satisfied:
NLINF ≤ 10; IUSER(11) ≤ 7; max{NLINF, IUSER(11)} > 0.

PR: Correct input of NLINF or IUSER(11) or set SIMULA=F.

USEREX 2 (F) RUSER(14), the normalization constant for γ in Eq (6) of [19], is not positive.

PR: **(1)** Correct the input of the amplitudes RUSER(J), J=26,28,... **(2)** Correct your version of USERK. **(3)** Avoid the problem by setting SIMULA=F or IWT $\neq 5$.

USERIN 1 (F) IUSER(10) is not 1, 2, 3, or 4.

PR: Correct input of IUSER(10). See Sec 4.2 of [19].

USERIN 2 (F) RUSER(19), the viscosity in Eq (5) of [19], is not positive.

PR: Correct input of RUSER(19) or IUSER(10).

USERK 1 (F) Illogical Stop. (This can only occur if JT < 0, JG < 0, JT $>$ NY, or JG $>$ NG $+ 1$, where JT is the subscript of the data point and JG the subscript of the grid point in the program calling USERK.)

USERK 2 (F) NG > 50 and there is not enough storage for the form factors.
PR: Reduce NG or increase the DIMENSION specification of RUSER (Sec 4.1.1).

USERK 3 (F) A g value needed in USEREX is not between GMNMX(1) and GMNMX(2), and the form factor cannot be interpolated.
PR: **(1)** Correct the input of RUSER(J), J=25,27,... . **(2)** Correct your version of USEREX or USERK. **(3)** Avoid the problem by inputting LUSER(3)=F or SIMULA=F.

USERK 4 (F) Either RUSER(21) or RUSER(22) is zero. This eliminates all λ-dependence in the exponential in Eq (2) of [19].
PR: Correct the input of RUSER(21) or RUSER(22).

USERK 5 (F) Both a g value and RUSER(22) are nonpositive or g is negative. This would lead to an error in evaluating $g^{\text{RUSER(22)}}$ in Eq (2) of [19].
PR: **(1)** Correct the input of RUSER(22) or the set up of the g-grid (Sec 4.1.2.1). **(2)** Using IUSER(10)=4, instead of 1, will allow g to be negative if RUSER(22) is a whole number.

USERK 6 (F) A g value is negative, implying a negative radius.
PR: **(1)** Correct the set up of the g-grid (Sec 4.1.2.1). **(2)** Correct the input of IUSER(10).

USERK 7 (F) A g value is negative and either RUSER(22) or RUSER(23) is not a whole number. This would lead to the same error as for USERK 5.
PR: See PR(1) of USERK 5.

USERK 8 (F) A g value is zero and RUSER(23) is negative. This causes a singularity in Eq (2) of [19].
PR: Correct the input of RUSER(23) or the set up of the g-grid (Sec 4.1.2.1).

USERLF 1 (F) Illogical Stop. (This can only occur if JY $\leq$ 0 or JY > NY, where JY is the subscript of the data point in the program calling USERLF.)

USERLF 2 (F) This is specialized to the versions of USERLF in Files 1 and 2. The requirement $1 \leq$ JLINF ≤ 2 is violated, where JLINF is the subscript i in Eq (3.1-3). If you modify USERLF, you should modify the requirement to be $1 \leq$ JLINF $\leq N_L$.

PR: Correct USERLF or NLINF.

USERNG 1 (F) NINEQ > MINEQ.
PR: **(1)** Correct your version of USERNQ. **(2)** Increase MINEQ. **(3)** Input DOUSNQ=F.

USERSI 1 (F) EXACT(J), a noise-free simulated data point, is negative and IWT=2. This would cause the square root of a negative number in computing the noise level.
PR: **(1)** Correct computation of EXACT in USEREX or the input of the USER arrays for USEREX (Table 5 of [19]). **(2)** Change IWT, SIMULA, or USERSI.

USERSI 2 (F) SQRTW(J), the square root of a least-squares weight, is negative and IWT=4.
PR: **(1)** Correct the input of SQRTW (but this should have been caught with READYT 5). **(2)** Correct your version of USERIN if it computes SQRTW. **(3)** Change IWT or USERSI.

USERTR 1 (F) Illogical Stop. (This can only occur if the actual argument IFUNCT in the program calling USERTR is not 1, 2, or 3.)

USERTR 2 (F) Illogical Stop. (This can only occur if USERTR is called when IGRID is not 1 or 2.)

6. APPLICATIONS PACKAGES

6.1 Photon Correlation Spectroscopy (PCS)

6.1.1 File contents

There are no extra files. This Applications Package is built into the version of CONTIN in Files 1 and 2. The Test Data Set 1 in Files 1 and 2 are PCS data on NBS sample SRM 705 scattering at 60° [3].

6.1.2 Problem solved

The wide range of problems solved by the PCS package is described in Sec 4 of [19] and in [21]. The solution can be normalized by dividing each value under ORDINATE by MOMENT(0) (OF THE ENTIRE SOLUTION), which is just the integral over the solution.

6.1.3 Input data deck

General instructions for input are given in Sec 3.3 of [19] and Secs 3.3 and 3.4. With respect to PCS, note the following:

(1) You must change GMNMX to specify the lowest and highest values of the g-grid appropriate for your data set. See the discussion in Sec III A of [3] and in Sec 3.4.2.

(2) IWT and NERFIT specify least-squares weights equivalent to Eq (14) of [1]. Thus this weighting is appropriate in the common case that the second order correlation function approximately follows Poisson statistics. You can easily change USERWT if some other weighting is appropriate. If for some reason you do not want to do a weighted analysis, then you can set IWT=1.

(3) DOUSNQ=T and NONNEG=T are strongly recommended. DOUSNQ=T constrains $\Delta_2 \geq 0$ in Eq (6) of [3] and, most importantly, NONNEG=T (the default value) constrains the solution to be nonnegative.

(4) You must change the other Control Variables to values appropriate for your data set. (See Sec 3.3 of [19].)

(5) To be safest, you should make two analyses of each set of data: Run 1, with NLINF=1 as it is in the Test Data, and Run 2, with NLINF=0. With Run 2, there is no "dust" term; i.e., Δ_2, as well as Δ_1 are set to zero in Eqs (6) of [3]. If there is a significant amount of dust present, then Run 2 will often give a MWD skewed toward the high molecular weight or even an extra peak at the high molecular weight end of the grid, as

with Test Data Set 3 (see Sec 4.1.6.4). In this case the weight-average and Z-average molecular weights are severely overestimated (as they would be with all methods ignoring the dust). You will then be forced to use the results of Run 1. You should try to avoid dust contamination and other sources of error (like scattering from the walls of the cell) that require the dust term. If you are successful, then Runs 1 and 2 should agree quite well, and the absence of the extra term in Run 2 would ordinarily make Run 2 more reliable. If you are forced to use a dust term, make sure that the t_k are large enough so that the data decay well into the noise (see Sec 6.1.5.3).

6.1.4 Interpreting the output

The interpretation of the output from the test run with Test Data Set 1 is discussed in detail in Sec 3.6, and you should read this. Only a few points specific to PCS will be mentioned here.

The CHOSEN SOLUTION on *Page 31* of the test output has only 3.137 degrees of freedom. This does not mean that the data are only capable of determining 3 parameters or 3 independent pieces of information. It simply means that only about 3 were needed to describe the MWD at the resolution attainable. This is consistent with the fact that DISCRETE (Sec 3.6.8) could also fit the data with 3 parameters – a monodisperse MWD (with 2 parameters: molecular weight and amplitude) plus a constant dust term. However, results with broader MWDs [3] and singular value analysis [1] indicate that measurements with $O(10^6)$ counts per channel with, say, 96 channels could yield at least 5 independent pieces of information or degrees of freedom.

The dust term on *Page 31* is 0.078725. This is more than 10% of y_1. As discussed in Sec IV of [3], this is probably already beyond the safe limit. You should try to keep this below about 5%, and preferably below 1%, of y_1. This should be possible if care is taken to avoid dust and other high molecular weight contaminants. Little care was taken during the experiments reported in [3], mainly to show that the method was not unduly sensitive to such contaminants. However, it is still important to keep these contaminants at as low a level as possible because Δ_2 only approximately accounts for them [15, 3].

If the least squares weights were taken as the reciprocal of the variance as in Eq (14) of [1], then STD. DEV. on *Page 27* should be about 1.0. However, the weights used were Eq (14) divided by the constant B. This means that STD. DEV. should be about $B^{-1/2}$ and that (STD. DEV.)$^{-2}$ should be about B. For good resolution, you usually must have B at least 10^6 (preferably 10^7)

counts per channel; i.e., STD. DEV. should be no greater than 10^{-3}. [Do not expect (STD. DEV.)$^{-2}$ to closely agree with the B expected on the basis of your autocorrelation counts; approximations and idealizations have been made in computing the least squares weights and STD. DEV..]

6.1.5 Other remarks

6.1.5.1 Parsimony

The most parsimonious solutions are usually (1) monodisperse, (2) smooth and unimodal, (3) unimodal but perhaps with a shoulder, or (4) bimodal, etc. Normally, you should always test to see if (1) can fit your data. You can do this with DISCRETE. (Be sure and read Sec 3.6.8, which gives a good illustration of this.) The PCS Package, used as in Sec 6.1.3, will tend to automatically protect you against artifacts causing (3) or (4) if (2) can fit the data. However, if you do get a bimodal or more complex solution, it is always a good idea to see if a unimodal (or perhaps a bimodal) solution can be found to fit the data. As explained in Sec 4.1.6.4, you can force a unimodal solution by simply adding to Card Set 2 the specifications NNSGN(2)=1, NSGN(1)=2, and LSIGN(1,1)=-1 [with NFLAT(1,2)=2 or 3, if you wish].

To see the dangers of a method that does not attempt to find the most parsimonious solution, you could use CONTIN to implement a histogram method. You could set NQPROG(1)=1, RSVMNX(1,1)=1.0 (making the regularizing parameter effectively zero), and IQUAD=1, and evaluate the integral over each bin in USERK. The number of peaks in the resultant solution is often strongly dependent on the number of bins that you decide to use in the histogram – the more bins, the more peaks. This is because, without the regularizor, you are trying to estimate typically 15 or more parameters from the data and these estimates are very unstable to noise in the data. On the other hand, specifying too few bins can also lead to a very inaccurate solution. The dangers of such a subjective procedure are obvious.

6.1.5.2 Simulating PCS data

This is discussed in Sec 4.3 of [19] and in Sec 4.1.6.2. It is illustrated with Test Data Set 2. Note the following:

(1) If you want to simulate the noise level observed in a previous analysis of experimental data with IWT=5, then you simply set RUSER(3)=STD. DEV. of the CHOSEN SOLUTION.

(2) Some useful types of simulations are described in Sec 3.7 of [2]. These involve the computation of point spread functions, e.g., the simulation of monodisperse molecular weight distributions (MWDs). For example, if simulated data sets from a monodisperse MWD yield solutions with a certain average degree of polydispersity, then obviously it will be impossible to detect this degree of polydispersity with real data following the same statistics; such a MWD is effectively indistinguishable from a monodisperse one, since they both fit the data to within experimental error.

If the MWD in the simulation does not contain δ-functions, then you may find it useful to plot this exact MWD together with the solution. You can do this with ONLY1=F and USERSX (see Sec 4.1.6.3).

6.1.5.3 Optimizing the t_k

Generally, for best resolution, you should measure at large enough t_k so that the y_k have decayed well into the noise. This is especially important when you are using NLINF=1, since then high molecular weight components could otherwise be taken as dust and vice versa. However, you should also measure at small enough t_k to sample the initial rapidly decreasing y_k finely enough. With correlators with only a small number of channels equally spaced in time, it is not always possible to adequately satisfy both requirements. We had to make compromises with our 37-channel correlator that effectively covered 65 equally-spaced time channels [3]. With 24 or 48 equally-spaced channels you will often be sacrificing resolution, especially with broad or complex MWDs. You are much better off with 96 channels, especially if you can increase the spacing of the later channels. Simulations (Sec 6.1.5.2) can help you evaluate the resolving power and optimize any degrees of freedom you may have in the spacing of the channels in time.

6.2 Inverting Laplace Transforms

There is no special package for this. It is now incorporated into the version of CONTIN in Files 1 and 2, as described in Sec 4 of [19]. Note the following:

(1) Laplace transforms are especially difficult to invert and CONTIN has been particularly effective for this, especially when NONNEG=T can be used [1, 4].

(2) It is usually helpful to also analyze the data in terms of a discrete sum of exponentials using DISCRETE [6–8]; see Sec 1.1 and [1, 3, 4].

(3) It is very important that you measure at large enough t_k such that the data have decayed well into the noise. This is especially important if you are using NLINF=1 (i.e., a constant background term), since some slowly decaying components could otherwise be taken as part of the background and vice versa. Obviously, you should also measure at small enough t_k so that the rapidly decaying part of the curve is well sampled. The most efficient way of meeting both requirements is to measure with sampling intervals increasing with t, roughly in equal intervals of $\log(t)$.

6.3 Inverting Fourier-Bessel Transforms

6.3.1 File contents

File 3 (see Sec 2.1.2) contains the subprograms BLOCK DATA, USERK, USERNQ, and USERWT for Version 2SP of the Fourier-Bessel Package. File 4 contains these subprograms for Version 2DP. The subprograms are written in the same format as in Files 1 and 2 (see Sec 2.1.3) except that the first card of each subprogram has the characters "FOURIER-BESSEL PACKAGE" as well as "MAR 1984". File 5 contains the test data and is shown in Sec 6.3.3.

You must replace BLOCK DATA, USERK, USERNQ, and USERWT in File 1 or 2 with those in File 3 or 4, respectively. Version 2DP is recommended unless your computer has at least 13-significant-figure accuracy in its Fortran single-precision (REAL) arithmetic (see Sec 2.2).

6.3.2 Problem solved

The Fourier-Bessel Package solves Eq (3.1-2) with

$$K(g,t_k) = 2\pi g J_o(2\pi g t_k) \qquad (1)$$

where $J_o(\cdot)$ is the zero order Bessel function of the first kind.

6.3.3 Input data deck

File 5, the test input data, is listed below (in the same format as in Sec 3.3.1). It is a typical example of the analysis of equatorial fiber diffraction data, and the discussion will be in this context. However, this package is directly applicable to other problems involving the kernel in Eq (6.3.2-1). The indexes in Sec 8 tell you where the Control Variables and other terms mentioned below are fully explained.


```
TEST DATA FOR FOURIER-BESSEL PACKAGE (VERSION 2)
GMNMX     2          40.5
NG                   28.
NEQ                   1.
NENDZ     1           0.
DOUSNQ                1.
RUSER    12        -1.E-2
IWT                   5.
RUSER    11           2.
NERFIT                0.
IFORMY
(5F14.6)
END
NSTEND   61          .016          .076
     1.523315      1.190113      0.814006      0.535417     -0.402695
    -0.567089     -0.751270     -0.948299     -1.118157     -1.254776
    -1.378472     -1.460039     -1.478993     -1.465018     -1.434473
    -1.351100     -1.278322     -1.160452     -1.037279     -0.895879
    -0.722611     -0.603666     -0.435656     -0.283385     -0.193872
     0.077192      0.270718      0.381172      0.450668      0.437786
     0.515502      0.525926      0.536378      0.459074      0.410563
     0.402334      0.347775      0.242373      0.000000      0.189404
     0.000000      0.000000      0.000000      0.000000     -0.243996
    -0.393584     -0.412606     -0.466926     -0.475384     -0.468763
    -0.506462     -0.542777     -0.578173     -0.548281     -0.518167
    -0.575197     -0.485255     -0.402439     -0.471114     -0.411118
    -0.200000
```

Line 1 of File 5 is just the heading (Card 1 in Table 3 of [19]).

6.3.3.1 Specifying the solution grid

GMNMX(1)=0 by default. This is the lower limit, a, in the integral in Eq (3.1-2). It should be zero unless you want to try constraining a solution to satisfy $s(g) = 0$ for $g \leq a$, e.g., to constrain a particle to have a cylindrical hole along its cylindrical axis.

GMNMX(2)=40.5. This is the upper limit, b, in the integral in Eq (3.1-2). It therefore constrains $s(g) = 0$ for $g \geq b$, e.g., it represents the maximum extent of a particle perpendicular to its cylindrical axis. As mentioned in Sec 3.4.2, you should leave a safety margin of about two grid points to avoid edge effects due to the boundary conditions; i.e., the third-to-last grid point ($g_{26} = 37.5$) should already represent the maximum radial extent. In the case illustrated here, this is actually quite an overestimate. [GMNMX(2)=36 would have been better.] However, such an overestimate may be necessary when there is no

precise prior knowledge, and the test run illustrates how the minor problems that can occur with such an overestimate can be resolved.

NG=28. This is a reasonable size for NG. An NG < 20 is often too low a resolution, at least for easily viewing the plots of the solution. An NG > 40 is usually a waste of computer time.

6.3.3.2 Equality constraints

NEQ=1. This constrains the last solution point to be zero; i.e., $s(g_{N_g}) = s(40.5) = 0.0$.

NENDZ(1)=0. This removes the boundary condition at the low-g end of the solution; i.e., it does not attempt to have $s(0)$ smoothly approach zero.

6.3.3.3 Inequality constraints

DOUSNQ=T. This causes USERNQ to constrain $s(g_j) \geq$ RUSER(12) for $j = 1, \ldots,$ IUSER(12) and $s(g_j) \geq$ RUSER(13) for $j =$ IUSER(12)+1, ..., NG. If IUSER(12) does not satisfy $1 \leq$ IUSER(12) $\leq$ NG, then IUSER(12) is effectively set to NG, and all solution points are constrained so that $s(g) \geq$ RUSER(12); this is the case with this test data set, since IUSER(12) is not changed from its default value of zero.

RUSER(12)=-0.01. In fiber diffraction, you know that $s(g)$ must be greater than or equal to minus the electron density of the solvent, and you may have good reason to set RUSER(12) even higher. Usually, however, you do not know the absolute scaling of the intensities until after the analysis [when CONTIN has computed the first moment of $s(g)$, as discussed in Sec 6.3.3.6]. In this case you can either set RUSER(12) to a very negative value (that is almost certainly not above the true lower bound) or let DOUSNQ=F (and therefore not have any lower bound at all). This is usually not very critical (although you can only profit by putting in as high a lower bound as your prior knowledge permits). When this test data was run with DOUSNQ=F, the results were practically unchanged. Section 6.3.4 discusses other possible uses of these constraints.

In other applications, inequality constraints based on prior knowledge (especially NONNEG=T, i.e., nonnegativity) can greatly improve the resolution and accuracy of the solution.

6.3.3.4 Weighting the data

IWT=5. This causes a weighted analysis to be performed after the usual unweighted analysis. The least squares weights are computed using Eq (12), which is derived below.

RUSER(11)=2. You must set it to a value appropriate for your data, as described below.

NERFIT=0. You should always use this with the weighting described below. [It sets ERRFIT=0 in Eq (4.1.2.7-1).]

IWT=5 can be useful if the statistics of your noise are approximately described by the following simple model. (Otherwise, you should use IWT=1 or introduce the appropriate weighting as described in Sec 4.1.2.7.) Suppose

$$I(t) = I_0(t) + B(t) \tag{1}$$

where $I(t)$ is the observed intensity on the equator at a distance t from the center of the diffraction pattern,

$$I_0(t) = y^2(t) \tag{2}$$

is the intensity due to the particle, and $B(t)$ is the background due to, e.g., other scattering, film fog, etc. The input data are

$$y(t) = [I_0(t)]^{1/2} = [I(t) - B(t)]^{1/2} \tag{3}$$

From here on, we consider a given arbitrary t, and to simplify the notation we write I instead of $I(t)$, etc. You can skim over the following until Eq (12) if you are not interested in the derivation.

From Eq (13) of [1] we have

$$\mathrm{Var}(y) \approx (\partial y/\partial I_0)^2 \, \mathrm{Var}(I_0) \tag{4}$$

where $\mathrm{Var}(x)$ is the variance (i.e., the square of the standard deviation) of the random variable x. Substituting Eq (3) into Eq (4), we get

$$\mathrm{Var}(y) = \mathrm{Var}(I_0)/(4I_0) \tag{5}$$

We consider $\mathrm{Var}(I_0) = \mathrm{Var}(I-B)$ as being due to two sources, which we assume uncorrelated:

$$\mathrm{Var}(I_0) = \mathrm{Var}(I) + \mathrm{Var}(\mathrm{rest}) \tag{6}$$

where Var(I) is assumed to follow Poisson statistics:

$$\mathrm{Var}(I) = cI = cI_0 + cB \tag{7}$$

where, with film, c is an unknown proportionality constant. In Var(rest) we collect all the other errors, e.g., due to the disorientation correction, background subtraction, nonlinear film response, etc. Substituting Eqs (6) and (7) into (5), we get

$$\mathrm{Var}(y) = [cI_0 + cB + \mathrm{Var(rest)}]/(4I_0) \tag{8}$$

$$\mathrm{Var}(y) = (c/4)[I_0 + \texttt{RUSER(11)}]/I_0 \tag{9}$$

$$\texttt{RUSER(11)} \equiv B(1+r) \tag{10}$$

$$r \equiv \mathrm{Var(rest)}/(cB) \tag{11}$$

The least squares weights must be inversely proportional to Var(y). Dropping the proportionality constant ($c/4$) in Eq (9) and using Eq (2), we can write the least squares weights, w, as

$$w = y^2/[y^2 + \texttt{RUSER(11)}] \tag{12}$$

[When IWT=5, subprogram USERWT evaluates (the square root of) Eq (12) with YSAFE in place of y.] In Eq (11), cB is just the (Poisson) number-fluctuation variance that you would get if there were only background, and Var(rest) is the variance due to all other sources; r is just the ratio of these two. If you are not sure what to use for r, an r about 10 may be often reasonable.

A small RUSER(11) tends to give unit weighting, but you have that during the PRELIMINARY UNWEIGHTED ANALYSIS anyway. Therefore, it is probably better to overestimate rather than underestimate RUSER(11), especially since some of the errors are probably systematic. If you are not sure what to use for RUSER(11), then you might try one about equal to the largest y_k^2, as was done in File 5. You can even use a very large RUSER(11) (say, 10 times the largest y_k^2) to get weighting nearly proportional to the intensities, y_k^2 [5]. This gives you the opposite extreme to the unweighted analysis.

In the derivation of Eq (12), we have ignored the dependence of the quantities in Eqs (10) and (11) on t, but, in view of the many other approximations, a more elaborate weighting scheme usually does not seem warranted. However, you can easily modify USERWT, e.g., to account for very poor accuracy of part of the data because of strongly overlapping layer lines or strong background.

Fortunately, $s(g)$ is usually not sensitive to the weights; if it is, then you should collect more and better data anyway. The best way to assess the weighting is to see if the weighted residuals (Sec 6.3.4) have a more uniform distribution of magnitudes than the unweighted residuals, which are usually much larger for small y_k.

6.3.3.5 Other input in File 5

IFORMY=(5F14.6). This specifies the FORMAT for Card Set 6 (see Table 3 of [19]).

6.3.4 Interpreting the output

You should read Sec 3.6 first. Only those aspects peculiar to fiber diffraction will be discussed here.

Selected pages from the output of a run on the *VAX 11/780* (Sec 2.4) with the test data in File 5 are shown in Part 2 of this manual. This run took 113 CPU seconds (not counting compiling and loading). As explained in Sec 3.6, because the spacing of the α-grid is machine-dependent, you can only directly compare your output with *Pages 1-3* and approximately with *Page 18*.

The test data are some early (and relatively noisy) data on PAK pili [16] kindly provided by Waltraud Folkhard. The structure is that of a hollow cylinder with inner and outer radii of about 6 and 26 Å and a girdle of low electron density at a radius of about 15 Å.

The reference solution, with practically no regularizor, is on *Page 4*. As usual, it is meaningless, with 6 peaks (the last 5 negligible compared to the first) despite the lower bound of -0.01. As mentioned in Sec 3.6.5, because the solution can go negative, the interpretation of the MOMENTs and (STD. DEV.)/MEAN is not simple.

The solution on *Page 14* is the one with PROB1 TO REJECT closest to 0.5, and it is therefore used to compute the least squares weights. It would have been the CHOSEN SOLUTION if IWT=1. The corresponding plot of the residuals and fit to the data are on *Pages 16* and *17*. Note the PRUNS of only 0.0000, indicating systematic errors; this is unfortunately the usual case in fiber diffraction. Note also that the residuals for small y_k tend to be much larger than the rest. The least squares weights, shown on *Page 17* attempt to account for this.

The final weighted analysis starts on *Page 17*. The reference solution on *Page 18* is again meaningless. The solution on *Page 27* is the one with PROB1 TO REJECT closest to 0.5 and is therefore printed again on *Page 32* as the CHOSEN

SOLUTION. The oscillations for $g > 30$ are clearly within experimental error, as indicated by the "error bars".

You can test this more directly by inputting IUSER(12)=20 and leaving RUSER(13) at its default value of zero. As explained in Sec 6.3.3.4, this constrains $s(g) \geq 0$ for $g \geq 30$. When this was done, the CHOSEN SOLUTION was very similar to those on *Page 27*, except that $s(g)$ smoothly approached zero without any oscillations. You could even go further and constrain $s(g) \geq 0$ for all g by inputting NONNEG=T (and removing DOUSNQ). When this was done, the CHOSEN SOLUTION hardly changed except that it smoothly approached zero at about $g = 1.5$, as well as $g = 31.5$. If a Peak-Constrained Analysis is performed constraining $s(g)$ to have only one peak, no $s(g)$ consistent with the data is found. It can be useful to try such a series of analyses with a wide variety of constraints. It is more than an aesthetic exercise. It gives you information on what features of the solution are demanded by the data and accurately determined and which are not necessary to fit the data and poorly determined. In this case, the positions of the two peaks and their relative heights, the position of the minimum between the peaks and the inner and outer radii of the particle are quite stable to changes in the weighting, PROB1 TO REJECT, and the constraints. Therefore they are probably quite reliable. In contrast, the negative $s(0)$ and the oscillations when $g > 30$ are quite variable and poorly determined by the data. There is no strong evidence that they do (or do not) exist. The most parsimonious solution would generally be without the oscillations.

The magnitudes of the weighted residuals on *Page 30* are slightly more uniformly distributed than those on *Page 16*. However, PRUNS=0.0026 is still very poor. Even the reference solution on *Page 18* only has a PRUNS=0.0363, indicating systematic errors. All the necessary preprocessing of fiber data makes systematic errors difficult to avoid, but you should try to keep them as small as possible. The worst type of preprocessing is one that results in artificially smoothed y_k. This destroys the validity of the PROB1 TO REJECT criterion, which will tend to be fooled into overfitting the unrealistically smooth data. In this case, you should input IPLRES(2)=3 and IPLFIT(2)=3 to get plots of the residuals and the fit for every solution. You will then have to visually judge from these plots which is the most parsimonious solution that still fits the data to within the actual uncertainties in the data.

One important measure to improve the reliability of PROB1 TO REJECT and the solution is to use as many data points as practical. The only limit is that you should not sample so finely that the noise in neighboring data points becomes correlated. This depends on such things as the resolution of the film and film scanner, etc. It has nothing to do with the sampling theorem.

6.3.5 Other remarks

It is not necessary to read Sec 6.3.5 to use CONTIN, but, if you have ever been tempted to use the Fourier-Bessel inversion formula (or other analytic inversion formulas) on data, it might be worthwhile reading. This inversion formula,

$$s(g) = 2\pi \int_0^\infty y(t)\, t\, J_0(2\pi g t)\, dt \qquad (1)$$

is an exact solution to the problem stated in Sec 6.3.2 and is the most common basis for analyzing fiber data. Unfortunately it is only practical when $y(t)$ is exactly known. This formula and the other Fourier inversion formulas have been the major hindrance to the proper analysis of diffraction data. There are three reasons not to use this formula:

(1) incorrect weighting: The data, $y(t)$, are measured at a discrete set of points and therefore Eq (1) must be evaluated by numerical quadrature. The relative weight assigned to each data point in estimating $s(g)$ depends solely on such things as the quadrature formula, the spacings between the data points, and $J_0(2\pi g t)$. No account is taken of the relative accuracies of the data points.

(2) no *a priori* knowledge: No account is taken of such *a priori* knowledge as the minimum electron density or maximum extent of the particle.

(3) truncation errors: Equation (1) requires that the data be extrapolated to zero and infinity or that the integral be incorrectly truncated before these limits. Typically, the data are multiplied by a gaussian to force the y_k data points with the largest t_k smoothly to zero and therefore reduce the oscillations due to the truncation. It is often attempted to justify this taper as a low-pass filter that imposes the natural resolution limit of the data on the solution. However, the resolving power of the data depends on its accuracy as well as its extent, whereas the severity of the taper is dictated by the magnitudes of the y_k at the largest t_k. This is particularly serious in fiber diffraction, where relatively accurate data with high intensity can suddenly end because of strongly overlapping layer lines. The gaussian tapering introduces a needless distortion and loss of resolution. In addition, the statistical accuracies of the data points are again ignored. There is no reason for distorting the data with extrapolation or tapering.

CONTIN, on the other hand, expands the solution in a set of eigenvectors or eigenfunctions that are specific for each particular discrete set of data and

statistical weights. The dominant eigenvectors tend to concentrate their power (i.e., to have more oscillations and resolving power) in the regions where $s(g)$ is more accurately determined by the data, and they tend to be smoother at the extreme values of g where the solution is poorly determined [2]. This explains why CONTIN has less severe problems with oscillations at high g values than methods that use (suboptimal) low-pass filtered Fourier-Bessel (or other Fourier) representations.

6.4 Inverting Small-Angle Scattering Data

6.4.1 File contents

There are no extra files. You can simply modify USERK to evaluate the kernel appropriate to your application. Be sure to include in the kernel all collimation and polychromaticity corrections. You should *not* break the analysis into more than one step where, e.g., collimation effects are deconvoluted and the deconvoluted data are then inverted. If possible, the data should be analyzed in one step, since this permits a much more reliable statistical weighting of the data.

You may want to use USERWT and USERNQ from the Fourier-Bessel Package (Sec 6.3).

6.4.2 Problem solved

You can use CONTIN to analyze intensity data for distance distributions or particle size distributions. If you assume a centrosymmetric particle and guess the signs of the amplitudes, then you can analyze the (deconvoluted) amplitude data for the spherically averaged radial density distribution function.

6.4.3 Input data deck

Sections 6.3.3.1–6.3.3.4 and 6.3.3.6 probably apply directly to your problem as well. However, if you are solving for distance distributions, then you should probably use NENDZ(1)=2 and NEQ=2. This imposes the additional constraint, $s(0) = 0$.

If you are analyzing amplitudes, and the noise statistics were not seriously changed by a previous deconvolution, then Sec 6.3.3.5 may be applicable. If you are analyzing intensities, then in USERWT you might want to evaluate the square root of

$$w_k = 1/(y_k + B_k) \qquad (1)$$

where w_k is the least-squares weight for data point k, B_k is the background, and the total observed intensity, $y_k + B_k$, is assumed to follow Poisson statistics. You should use YSAFE_k (Sec 4.1.2.7) in place of y_k and leave NERFIT at its default value of 10.

6.4.4 Other remarks

The remarks of Sec 6.3.5 on the disadvantages of applying an analytic inversion formula or representing the solution in terms of a finite Fourier sum apply here as well.

6.5 Estimating Globular Protein Secondary Structure from Circular Dichroism

6.5.1 Introduction

This CD Package implements the method described in [10]. This is a very specialized and simple application of CONTIN, and the CD Package is especially easy to use. You need not read most of this Users Manual. You need only read Secs 6.5.1–6.5.3 and the few other subsections that you are referred to there. Section 6.5.4 describes further options, but is not necessary reading. Section 6.5.5 is only for when your run aborts with an error message. Before proceeding, you should read Sec 1 and [10].

6.5.2 Installing the CD Package on your computer

If the CD Package has already been installed, compiled, and tested on your computer, then you can skip to Sec 6.5.3. Otherwise, read Sec 2.1 before proceeding.

File 6 contains the following 7 subprograms for Version 2SP of the CD Package: main subprogram, BLOCK DATA, USERIN, USERK, USERNQ, USERRG, and USERWT. File 7 contains these same subprograms for Version 2DP. The subprograms are written in the same format as in Files 1 and 2 (see Sec 2.1.3) except that the first "card" of each subprogram has the characters "CD PACKAGE", as well as "MAR 1984". File 8 contains the test data and is shown in Sec 6.5.3.1.

You must replace the corresponding 7 subprograms in File 1 (or 2) with those in File 6 (or 7). See Sec 2.2 to decide which Version to use. Read Secs 2.3, 2.3.1–2.3.3, and 2.4 before proceeding.

6.5.3 Essential information on using the CD Package

6.5.3.1 Input data deck

Below is listed File 8 from the magnetic tape. These are the test input data for the CD Package. The line numbers have been added here for convenience; they are not part of the input data or File 8. Section 3.3 of [19] explains the input data format in more detail.

```
 1  TEST DATA SET 1 - FOR CD PACKAGE
 2  IFORMY
 3  (7F9.0)
 4  LAST                -1.
 5  END
 6  NSTEND   26          190.          240.
 7    46832.   44401.   35975.   20418.    2592.   -6482.  -19770.
 8   -26252.  -30465.  -28034.  -24955.  -23335.  -23011.  -22849.
 9   -23173.  -23173.  -22849.  -19932.  -19284.  -15719.  -11991.
10    -8750.   -5995.   -3565.   -2268.   -1458.
11  TEST DATA SET 2 - FOR CD PACKAGE
12  LAST                 1.
13  IWT                  5.
14  IUSER    14         31.
15  IUSER    15         -1.
16  RUSER    14          1.
17  RUSER    15         .03
18  RUSER    16        500.
19  END
20  NSTEND   51         240.          190.
21     -781.    -933.   -1107.   -1345.   -1617.   -1931.   -2300.
22    -2691.   -3114.   -3548.   -4003.   -4458.   -4903.   -5315.
23    -5706.   -6021.   -6270.   -6476.   -6584.   -6639.   -6661.
24    -6639.   -6585.   -6541.   -6465.   -6400.   -6357.   -6357.
25    -6368.   -6422.   -6465.   -6324.   -6053.   -5652.   -5077.
26    -4426.   -3709.   -2762.   -1677.    -263.    1216.    2894.
27     4505.    6050.    7365.    8220.    8910.    9206.    9239.
28     8913.    8417.
```

Only Test Data Set 1 (Lines 1–10 of File 8) will be discussed in Sec 6.5.3. This is normally all that you need in a Data Set to analyze CD data using the method in [10]. Test Data Set 2 (Lines 11–28) illustrate further options, which will be discussed in Sec 6.5.4.

Line 1 contains a heading in columns 1–80. It will be printed at key places in the output. Line 1 is always needed, but it can be blank.

Lines 2 and 3 specify the input FORMAT of the mean residue ellipticities in Lines 7–10. Line 2 is always the same. Columns 2–71 of Line 3 contain the

Fortran FORMAT specification enclosed in parentheses.

Line 4 is only needed if there is more than one Data Set. If you have more than one Data Set, then you must have Line 4 in the first Data Set and Line 12 in the last Data Set, but nothing in the middle ones.

Line 5 is always needed.

Line 6 specifies that there are 26 mean residue ellipticities in Lines 7–10 and that they correspond to equal intervals in wavelength with the first at 190 nm and the last at 240 nm. Read the description of Card Set 4 in Sec 3.3.2; NINTT has the default value of 1 in the CD package and the t_k are the wavelengths in nm. (If your spectra are not in equal intervals of wavelength, Sec 3.3.2 and Sec 3.3 of [19] explain how to input them with NINTT > 1 and Card Set 4 or with Card Set 5b).

Lines 7–10 are the 26 mean residue ellipticities (in deg cm^2/dmole) in FORMAT (7F9.0), as specified in Line 3. [You can use units other than deg cm^2/dmole provided that you set RUSER(14) to the conversion factor, as described in Sec 6.5.4.3.]

It is very important to determine the protein concentration (and all other quantities in the conversion factors) as accurately as possible (preferably to within 5%). If your concentrations have an uncertainty exceeding 5%, then make two extra runs with say RUSER(14)=0.9 (corresponding to a concentration decrease of about 10%) and RUSER(14)=1.1 (corresponding to a concentration increase of 10%). If the uncertainty were ±20%, then you would have to use 0.8 and 1.2. The sensitivity of the estimates to these various concentrations will give you an idea of the resultant uncertainties.

As discussed in [10], the analyses are based on 16 protein CD spectra available from J.T. Yang's laboratory [18]. They range from 190 to 240 nm, and all of your data must be within this wavelength range. (If you want to extend the range or the library of spectra, then you will have to use new spectra, as described in Sec 6.5.4.6.) However, the wavelengths can take on any values in the range 190 to 240 nm; they need not be integers.

It is suggested that you input data at least every nm, if possible (i.e., have at least 51 data points). Inputting only every 2 nm, as in Test Data Set 1, is probably the bare minimum, and is not recommended. Best of all is to have digital data acquisition, say with 251 points per spectrum (5 points per nm). If you have more than 251 points per spectrum, then you will have to increase MSTORE and MY, as described in Sec 6.5.4.5. However, in principle, the only requirement that limits the number of data points is that the *noise* in the data

be uncorrelated; except for this, the more data points, the better. To avoid large uncertainties in the estimates, try to go all the way down to 190 nm, even if the automatic weighting in Sec 6.5.4.2 has to be used.

6.5.3.2 Interpreting the output

At the end of Part 2 of this manual is listed the output from a run with Version 2DP and File 8 as input data on the *VAX 11/780*. The run took about 36 sec of CPU time (not counting compiling and loading). Because the spacing of the α's is machine-dependent, you can only exactly compare your output with that on *Pages 1*, *2*, and *11-13* (except for the smallest SINGULAR VALUES and the lines with PRECIS. However, the final results, i.e., the FRACTIONs in the CHOSEN SOLUTIONs (*Pages 10* and *27*) should be comparable to within a few percent, as should the FRACTIONs and ORDINATES at the top of *Pages 3* and *14*.

This section also uses the output to illustrate important points and guidelines. Only the output most important in CD is discussed in this section. Section 3, [2], and [19] contain a more precise discussion of the methods used and of all the output, but it is not necessary that you read this.

To make the notation of [10] consistent with that of this Users Manual, you can temporarily replace α by α^2 in Eq (4) of [10].

The first line of *Page 1* of the output contains the Version and date of CONTIN and the heading from Line 1 of File 8. Then Lines 2-6 are printed out as they are read in. Then come the FINAL VALUES OF CONTROL VARIABLES, which will be used throughout the analysis of the Data Set. For example, IFORMY=(7F9.0) verifies that Lines 2 and 3 have been properly read.

On *Page 2* the wavelengths and mean residue ellipticities are printed under the headings T and Y. (There is always an extra data point, T=0.0 and Y=1.0, added at the end; this simply imposes Eq (5) of [10].)

On *Pages 3-8* are a series of solutions with increasing α, as discussed in the next-to-last paragraph of the Appendix of [10]. Each solution starts with a heading. In terms of quantities in Eq (4) of [10], ALPHA $= \alpha^{1/2}$; OBJ. FCTN. is the left-hand side; VARIANCE is the first sum on the left-hand side; STD. DEV. $=$ [VARIANCE$/(N_y - N_{DF})$]$^{1/2}$ is the estimated standard deviation of the noise in the data (according to this solution), where N_{DF} = DEG FREEDOM is the effective number of degrees of freedom, which is analogous to the number of free parameters (Sec 3.5 of [2]).

The preferred solutions are chosen on the basis of PROB1 TO REJECT (abbreviated PROB1). (Section 3.2.3 and [2] discuss this in more detail, if you are interested.) PROB1 increases monotonically with increasing ALPHA. A PROB1 near zero means that ALPHA is probably too small and that the data are being overfitted with too large a DEG FREEDOM and with an unstable solution. A PROB1 near 1.0 means that ALPHA is probably too large and DEG FREEDOM too small to adequately fit the data. You should consider all solutions with PROB1 between about 0.01 and 0.99 as possibilities and give special preference to those between about 0.1 and 0.9. The solution with PROB1 closest to 0.5 is printed at the end of the analysis on Page 10 as the CHOSEN SOLUTION. PROB2 TO REJECT should be ignored.

Printed under the heading ORDINATE are the γ_j, under ABSCISSA the j in Eq (2) of [10], and under ERROR something analogous (Sec 3.7 of [2]) to standard error estimates for the γ_j. The γ_j are also plotted horizontally. Then under the headings HELIX, etc., are the three f_i, which are the quantities of interest. The class β-turn has been put in the "remainder" class, because it is not as reliably determined as helix and β-sheet. (SCALE FACTOR is only of interest if you are using the option discussed in Sec 6.5.4.3.)

The STANDARD ERRORs on the next line must be considered lower bounds. They assume that there are no systematic errors, that the set of library spectra is adequate to represent the data, and that the regularization is not biasing the solution.

Finally come summaries of statistical tests on the randomness of the residuals of the fit to the data. Read the next to last paragraph of Sec 5 of [19] and point (5a) of Sec 3.6.5.

The residuals for the CHOSEN SOLUTION are plotted on *Page 9*, and they do appear to be fairly randomly scattered. Next is a plot of the y and y_{obsd} for the CHOSEN SOLUTION. Read Secs 3.6.6 and 3.6.7, where these two plots are discussed. (In our case the $w_k = 1$ in Sec 3.6.6.)

In the rest of Sec 6.5.3.2, I will use the output from the test data to illustrate the most important guidelines for interpreting the output.

You should not go immediately to the last line of the last page (*Page 12*) and look at the HELIX and BETA-SHEET values and nothing else. You should check *Pages 1* and *2* to make sure that your data were properly input. The plots on *Page 9* are also useful in spotting gross input errors. The plot of the fit to the data are useful in judging the overall quality of the data from a natural perspective. For example, the region 198 to 206 nm on *Page 9* is clearly the

noisiest, with three cases where the "X" and "O" have a space between them. You should try for data approaching the quality of that on *Pages 17* and *18*.

If you are measuring the data manually from x y recorder paper, you should of course visually put a smooth curve through the very high frequency noise to get a good estimate. However, you should keep this smoothing very localized (typically over no more than the wavelength interval between data points) and make measurements at least every nm. (*Never* do any large-scale smoothing, e.g., with splines.) This will help avoid building systematic errors into the data and give a more realistic image of the actual noise levels in the data. For your first few spectra, you may want to input an extra card after Line 1, with the characters "IPLFIT" in columns 2-7, "2" in column 12 and "3." in columns 26 and 27. This will cause a plot of the fit to the data after every solution, not just the CHOSEN SOLUTION. You can then see if there are solutions with ALPHA greater than that of the CHOSEN SOLUTION that nevertheless seem to fit the data to within the actual noise level in the data. If so, then these are to be preferred. Of course, it is best to avoid these problems altogether by having spectra with so little noise that drastic visual smoothing is unnecessary.

The γ_j values themselves are usually not very informative. A large γ_j does not necessarily mean that there is a structural resemblance between that protein and the one being analyzed. (The names of the 16 proteins corresponding to the ABSCISSA values are listed in subprogram USERK.) It is interesting, however, how the γ_j can blow up when ALPHA is very small and effectively a conventional least squares analysis is performed, as on *Page 3*.

6.5.4 More infomation on using the CD Package

It is normally not necessary to read Sec 6.5.4 to use the CD Package. This section describes additional options and possible changes. The run with Test Data Set 2 will be used to illustrate these options.

6.5.4.1 Controlling the range of ALPHA

Version 2 does this quite efficiently automatically. Therefore it should not be necessary to use the Control Variables described in Sec 4.1.3 to improve upon this.

6.5.4.2 Specifying least-squares weights

The data in the range 190 to 210 nm are often especially noisy. The CD Package

has an option for automatically assigning two different least-squares weights to two parts of the spectrum. This option has been useful, but it usually has not resulted in dramatic improvements. If the results from the weighted and unweighted analyses differ greatly, then you should collect more and better data anyway. (Section 4.1.2.7 explains how you can implement your own weighting scheme).

IWT=5 in Line 13 of File 8. This specifies that the special CD weighting be implemented.

IUSER(14)=31 in Line 14. The CD spectrum is divided into two Regions. The first IUSER(14) data points make up Region 1 and the rest of the spectrum makes up Region 2. From, e.g., the first 31 ABSCISSA values on *Pages 17* and *18*, you can see that Region 1 covers the range 240 nm to 210 nm and Region 2 covers 209 nm to 190 nm.

IUSER(15)=-1 in Line 15. An IUSER(15) < 0 specifies that the least-squares weights for Region 1 cannot be less than those for Region 2. An IUSER(15) > 0 specifies the converse. An IUSER(15) $= 0$ puts no restrictions on the relative sizes of the weights. In the usual case that Region 1 is the (noisier) low-wavelength part, IUSER(15) > 0 is recommended. In Test Data Set 2, the spectrum is input in order of decreasing wavelength, and IUSER(15) < 0 is therefore used.

The weights are computed as follows. First a PRELIMINARY UNWEIGHTED ANALYSIS (*Pages 14-16*) is performed, simply to get a smooth fit through the data, which is plotted on *Pages 17* and *18*. The root-mean-square residuals in Regions 1 and 2 are computed and printed on *Page 18*, and they are used as estimates of the noise levels in the two Regions. Their reciprocals are taken as the SQUARE ROOTS OF THE LEAST SQUARES WEIGHTS, which are also printed on *Page 18*. (The last weight is for the added data point mentioned in Sec 6.5.3.2 and will be discussed in Sec 6.5.4.3.) The CD spectrum in Test Data Set 2 is of such uniform high precision that the weights are the same for the two Regions, but this is not usually the case.

The final weighted analysis then begins on *Page 18*. Note that STD. DEV. no longer is in units of deg cm^2/dmole. It is typically slightly larger than 1.0, and, for the CHOSEN SOLUTION, VARIANCE is normally about N_y, the number of data points.

6.5.4.3 Accounting for uncertainties in the concentration

Usually the main source of error in CD measurements is the determination of

the protein concentration. This means that your mean residue ellipticities may be in error by a constant factor. In principle, by removing Eq (5) and modifying Eq (4) of [10], the concentration could be left as another free parameter to be determined by CONTIN. Unfortunately, this leads to disastrously unstable and unreliable solutions, at least for the CD spectra we have tried. Below is described an option for an intermediate approach, where the concentration is effectively put in as an extra data point (which is actually the case) and given limited freedom to be adjusted by CONTIN (corresponding to a few percent uncertainty in the concentration). This option has only resulted in minor improvements to the results. As with weighting, if the results change drastically with this option, then this is a warning that you should collect better data anyway. RUSER(14) has a second use in that it can allow you to input your spectra in units other than deg cm^2/dmole.

RUSER(14)=1.0 in Line 16 of File 8. RUSER(14) is the factor that you would *divide* your data by to put them in deg cm^2/dmole. If your data are already in deg cm^2/dmole, then you should use 1.0, as here. [Actually RUSER(14)=1.0 is the default value; so in this case you really do not need Line 16.] If you use an RUSER(14) not equal to 1.0, then you must adjust RUSER(15) and RUSER(16) accordingly, as described below.

RUSER(15)=0.03 in Line 21. RUSER(15) is the estimated uncertainty (the standard deviation) that you are assigning to RUSER(14), e.g., due to uncertainties in the protein concentration determination. It is strongly recommended that you always make RUSER(15) satisfy

$$(0.001)\text{RUSER(14)} \leq \text{RUSER(15)} \leq (0.03)\text{RUSER(14)} \qquad (1)$$

The lower limit implies that your concentration is accurate to about 0.1%. It is therefore practically the method of [10], where Eq (5) allowed no uncertainty to the concentration or conversion factor. Using an RUSER(15) below this lower limit would not significantly change the results, except that it could adversely affect the accuracy of the numerical computations.

The upper limit implies that your concentration is accurate to about 3%. Even if your uncertainty is greater than this (a situation you should try to avoid), you should still adhere to this upper limit. This is because of the instability mentioned above. If you want account for a larger uncertainty in the concentration, a better way would be to do several analyses with RUSER(15) at the *lower* limit and with different values of RUSER(14) covering the range of uncertainty. The variation of the results over this series of analyses will give you an idea of the uncertainties in the results.

RUSER(16)=500.0 in Line 22. This is the estimated average uncertainty (standard deviation) that you are assigning to your CD data. An extra data point is always added to replace Eq (5) of [10]:

$$\sum_{i=1}^{N_f} f_i = \text{RUSER(14)} \qquad (2)$$

In the unweighted analysis, this extra data point is given the least-squares weight of [RUSER(16)/RUSER(15)]^2, whereas the weights for all the other data points are 1.0. If you are using the lower limit of Eq (1) for RUSER(15), then the weight for the extra data point will be large anyway, as long as RUSER(16) is accurate to within a factor of 5 or 10. Therefore, in this case, RUSER(16) = (500)RUSER(14) is usually close enough. If you are using the upper limit of Eq (1) for RUSER(15), then a bad guess for RUSER(16) can distort the weighting of the extra data point in the unweighted analysis. This is why the output is reduced for the PRELIMINARY UNWEIGHTED ANALYSIS on *Pages 14–16*. In the final weighted analysis on *Pages 18–27*, there is no problem because RUSER(16) is not used to compute the weights.

The run with Test Data Set 2 on *Pages 11–27* illustrates a case where RUSER(15) at the upper limit in Eq (1) proved useful at detecting a problem in the data. These data are the spectrum of subtilisin BPN′ of Chang et al. [18], which are likely to be significantly in error [10].

After each solution, under SCALE FACTOR, is printed the ratio of the left-hand side of Eq (2) to the right-hand side. (The f_i printed under HELIX, etc. have been later normalized to sum to 1.0.). Thus the 0.898 for the CHOSEN SOLUTION on *Page 27* means that the input data values were considered 10% too small by CONTIN. This 10% change in RUSER(14) is 3.4 times the standard deviation of 3% specified by RUSER(15). This very large deviation of the last data point can be seen in the PLOT OF WEIGHTED RESIDUALS. The last point is the maximum residual and the second line on *Page 25* says that this MAX = 3.4(standard deviations). Such an unusually large deviation indicates that the data cannot be well fit with the given RUSER(14), perhaps due to a large error in measuring the concentration. This seems to be the case [10]. Unfortunately, such a clear warning usually does not occur, and even when it does it is usually not possible for CONTIN to automatically correct for such large errors in the concentration.

6.5.4.4 Changing the smoothing polynomial

It is recommended that you *not* change the Control Variables described in Sec 6.5.4.4. They are included here only for completeness.

In order to estimate the mean residue ellipticity of one of the 16 protein spectra in subprogram USERK for any given wavelength between 190 and 240 nm, a least-squares polynomial of degree [IUSER(16)-1] through the IUSER(17) points whose wavelengths are closest to the given wavelength is computed. With IUSER(17) > IUSER(16), this provides a smoothing, as well as an interpolation, polynomial. The default values are IUSER(16)=4 and IUSER(17)=7, and there should be no reason to change them. In any case they must satisfy

$$2 \leq \text{IUSER(16)} \leq \text{MIDEG1} \quad (1)$$

$$\text{IUSER(16)} \leq \text{IUSER(17)} \leq \text{MIPTS} \quad (2)$$

where MIDEG1=6 and MIPTS=15 in the versions in Files 6 and 7. (If for some reason you even wanted to change MIDEG1 or MIPTS, Sec 6.5.4.6 explains how.)

Two other Control Variables, LUSER(2) and LUSER(3) are used internally by the CD Package and therefore should not be changed by you.

6.5.4.5 Changing the DIMENSION parameter MSTORE

You only need to change this if NY, the number of data points for your CD spectrum, exceeds 251. You must always have

$$\text{MSTORE} \geq \text{NY} \quad (1)$$

Furthermore, for the CD Package, Eq (3.5-3) should be modified to

$$\text{MY} \geq \max(\text{NSPECT}, \text{NY}+1) \quad (2)$$

where NSPECT, the number of reference protein spectra, is 16 in the versions in Files 6 and 7.

To change MSTORE, you must make the following changes: (The line numbers below all refer to Files 6 and 7, not Files 1 and 2.)

(1) In Line 253 set MSTORE to the desired value.

(2) In Line 100, set MY to satisfy Eq (2); i.e., set MY = MSTORE + 1. Then in Line 54, adjust the DIMENSION specifications as shown in Line 67.

(3) In Lines 92, 201, and 343 adjust the DIMENSION specification to satisfy AINTRP(MSTORE,NSPECT).

6.5.4.6 Changing the reference protein spectra

You might want to do this if you had more spectra to add or a more accurate set of spectra or F_{ji} in Eq (3) of [10]. Furthermore, this method is by no means restricted to CD spectra or to proteins. You could modify the CD Package for similar problems involving the analysis of a spectrum in terms of a linear combination of library spectra, e.g., with ORD, IR, or Raman spectra.

There are two types of changes: changes to DATA and to DIMENSION specifications. In the DATA specifications you set, for example, the protein reference spectra and DIMENSION parameters that specify how many spectra and wavelengths there are. You must then readjust the DIMENSION specifications to agree with the DIMENSION parameters. Below are defined in the notation of [10] all the variables that you may wish to change in the DATA specifications. Their values in the versions in Files 1, 2, 6, and 7 are also given.

NSPECT = 16 = N_γ in [10]

NWAVEL = 51 = the number of wavelengths in each protein reference spectrum.

AREF(51,16) = the NWAVEL-by-NSPECT array of protein reference spectra, i.e., the R_j for $j = 1, \ldots,$ NSPECT in Eq (2) of [10].

WLSTRT = 240.0 = the first wavelength of each R_j.

WLEND = 190.0 = the last wavelength of each R_j.

NCLASS = 3 = N_f in Eq (3).

FCAP(3,16) = the NCLASS-by-NSPECT array with F(I,J) = F_{ji} in Eq (3).

The following three DIMENSION parameters are not directly related to changing the protein reference spectra, but are included here for completeness.

MSTORE = 251 is defined in Sec 6.5.4.5.

MIDEG1 = 6 = the upper limit for IUSER(16), as discussed in Sec 6.5.4.4.

MIPTS = 15 = the upper limit for IUSER(17), as discussed in Sec 6.5.4.4.

Below are listed all the changes that you may have to make if you change the above DATA specifications. Line numbers below refer to File 6 or 7, except when preceded by the characters AAO, in which case they refer to File 1 or 2.

(1) In Lines 100 and 101, set MA and MG to NSPECT + 2, MY = max(MG,MSTORE + 1), MINEQ = NCLASS, and MWORK according to Eq (3.5-7). Then adjust the DIMENSION specifications in Lines 54-62 as specified in Lines 67-75.

(2) In Lines 92 and 201, adjust the DIMENSION specifications to AINTRP(MSTORE,NSPECT) and PSINV(MIDEG1,MIPTS).

(3) In Line 218, adjust GMNMX(2) to NSPECT.

(4) In Line 228, adjust NG to NSPECT.

(5) In Lines 252 and 253, put in your DATA specifications.

(6) In Line 317, adjust the DIMENSION specifications to XD(MIPTS,MIDEG1), ED(MIPTS,MIPTS), SD(MIDEG1,3), AREF(NWAVEL,NSPECT).

(7) In Lines 318-321, DIMENSION the NSPECT arrays as SPEC01(NWAVEL),..., SPEC*nn*(NWAVEL), where *nn* = NSPECT.

(8) In Lines 322-329, add or remove EQUIVALENCE specifications so that there are NSPECT of them.

(9) In Line 343, make the same change as in change (2).

(10) In Lines 346-505, put in your NSPECT reference protein spectra.

(11) In Line 616, adjust the DIMENSION specification to FCAP(NCLASS,NSPECT).

(12) In Lines 632-638, put in your DATA specifications for NCLASS and FCAP.

(13) In Line AAO 1001, adjust the DIMENSION specifications to FCAP(NCLASS,NSPECT), F(NCLASS) and ERROR(NCLASS).

(14) In Lines AAO 1015-AAO 1021, put in your DATA specifications for NSPECT, NCLASS, and FCAP.

(15) In Lines AAO 1068-AAO 1071, change the output FORMAT for the f_i, if you have changed the classes.

6.5.5 Diagnostics

If your run aborts with an error message or your results seem incorrect, then read Sec 5. Section 6.5.5 lists some additional diagnostics that are only in the CD Package. (NY below is the number of mean residue ellipticity data points.)

"NY OR RUSER(J), J=14, 15, OR 16 IS OUT OF RANGE IN USERIN" occurs when one of the following requirements is violated: NY $\leq$ MY $-$ 1, RUSER(14) $\neq$ 0,

RUSER(15) > 0, RUSER(16) > 0, where the three RUSERs are explained in Sec 6.5.4.3.

"IUSER(14) OUT OF RANGE IN USERWT" occurs when the following condition is violated: 1 $\leq$ IUSER(14) $\leq$ NY. (See Sec 6.5.4.2).

"ALL RESIDUALS SOMEHOW ZERO. CANNOT ESTIMATE WEIGHTS." This is extremely unlikely to occur (in USERWT) unless you have made gross errors in the input, such as inputting all zeroes. In any case, you can avoid the problem by using IWT=1.

The CD Package also has the following additional standard diagnostics (See Sec 5.3).

USERK 2 (F) Equation (6.5.4.4-1) is violated.
PR: Change IUSER(16).

USERK 3 (F) Equation (6.5.4.4-2) is violated.
PR: Change IUSER(16) or IUSER(17).

USERK 4 (F) Equation (6.5.4.5-1) is violated.
PR: **(1)** Correct input of NY or NT. **(2)** Increase MSTORE.

USERK 5 (F) NG > NSPECT.
PR: Input NG=NSPECT. (NSPECT=16 in the versions in Files 6 and 7. Otherwise, see Sec 6.5.4.6.)

USERK 6 (F) The computation of the pseudoinverse for the least-squares smoothing failed to converge.
PR: Use the original version in File 6 or 7 and the default values of IUSER(16)=4 and IUSER(17)=7. If the problem remains, then this is an Illogical Stop.

USERK 7 (F) The least-squares matrix for smoothing is singular.
PR: Same as for USERK 6.

USERK 8 (F) One of your input wavelengths, T(J), does not lie between WLSTRT and WLEND. (In the version in Files 6 and 7, WLSTRT=240.0 and WLEND=190.0.)
PR: Correct the input of your Ts in Card Set 4 or 5b (e.g., Line 6 of File 8).

CONTIN Update 2

A. Address Correction Request

If necessary, please correct the address label used for this update, and send it to

Stephen W. Provencher
Max-Planck-Institut für biophysikalische Chemie
Postfach 2841
D-3400 Göttingen
Federal Republic of Germany.

B. Summary

Users of the CD (circular dichroism) Package do not need to read this update. This update should be inserted in the Users Manual.

This update describes the new Splice Applications Package. The main application area is in the simultaneous global analysis of several data sets that differ from each other only in (unknown) scale factors and in the values of T at which they have been measured, e.g., in the "splicing" together of a series of data sets that have been measured over different (possibly overlapping) T ranges. This can greatly improve the reliability and resolution of the solution, especially if the solution extends over a range too wide to be covered by any one of the data sets.

In addition, more powerful global analyses can be performed by making modifications to USERK. For example, high-resolution multi-angle analyses of photon correlation (PCS) data sets would be possible by including form factors in USERK. Detailed theoretical models describing the behavior of a system over a wide range of conditions could be tested by including these in USERK and analyzing the corresponding wide range of data sets with the Splice Package. With further modifications to USERK and SETGRD, you can split the distribution into two or more parts, with each part having a different dependence on the conditions (e.g., scattering vector) being varied between the data sets. These are especially rigorous and demanding tests of a model if the range of conditions in the data sets is wide enough.

C. Method

We have N_d experimental data sets described by a generalized eq. (1.2) of [1],

$$(1+\delta_j)y_{kj} = \int_a^b F_{kj}(\lambda)s(\lambda)\mathrm{d}\lambda + \sum_{i=1}^{N_L} L_{kji}\beta_i + \epsilon_{kj}, \qquad j = 1,\ldots,N_d, \tag{1}$$

where δ_j, the optimal scaling correction for the y_{kj} in data set j, is to be determined for $j = 1,\ldots,N_d-1$, and $\delta_{N_d} \equiv 0$. Thus the scale factor for the last data set is 1, and the rest of the data sets are scaled to match it. Equation (1) can be restored to eq. (1.2) of [1] for CONTIN by simply subtracting the correction terms from both sides and putting them in the sum on the right, with $\beta_{j+N_L} = \delta_j$ for $j = 1,\ldots,N_d-1$ and $L_{kji} = -y_{kj}$ when $i = j+N_L$ and $L_{kji} = 0$ otherwise. The sum on the right of eq. (1.2) now has NLINF $= N_L + N_d - 1$ terms.

The main assumption is, of course, that $s(\lambda)$ is the same for all (properly scaled) data sets. In the default version of the Splice package, the kernel, F_{kj}, is also assumed to be independent of j, which is the usual case. However, as mentioned at the end of Section B, you can modify USERK to include an F_{kj} that also depends on j, and thus test much more complicated models.

There are restrictions on the constraints that you can use in the present version when you are splicing (i.e., when $N_d \geq 2$). You must have NONNEG=T [nonnegativity of $s(\lambda)$] and NEQ=0 (no equality constraints). Additional restrictions on the allowed values of N_L are described in Sec. E. However, these restrictions still probably allow the vast majority of applications.

Furthermore, the Splice Package is completely compatible with the standard CONTIN; i.e., your old input files will yield identical results. The only requirement is that you have left IUSER(19) and IUSER(20) at their default values of 0. In this case, then, there are no restrictions on constraints, and you can do everything that you can do with the original version of CONTIN. The only price is that about MY1$\times$(MG1+2) extra working-precision words of memory are needed, where MY1 is the maximum number of data points in *one* data set and MG1 the maximum number of grid points plus one.

D. Implementing the Splice Applications Package

There are three files for the Splice Package, a file with test data, one file for updating your Version 2SP, and one for Version 2DP.

The Splice Package has 7 subprograms – the main, BLOCK DATA, USERK, USERLF, USERNQ, MULTIW, and FIT1Y subprograms. You must delete the main, BLOCK DATA, USERK, USERLF, and USERNQ subprograms from your March 1984 or August 1982 version of CONTIN and insert the Splice Applications Package. (MULTIW and FIT1Y were not in the original CONTIN.) Some of the code is now in Fortran 77, which most current compilers prefer.

If you have made modifications to your BLOCK DATA, it is easier not to delete it and to make the only change yourself, changing the characters "PCS-1" to "Splice" in line 0244. Similarly, the only change in USERK is to line 0676, which now permits more than 50 grid points if you are not using form factors.

The dimensioning requirements in this paragraph are automatically checked for you, and fatal error messages occur if they are not satisfied. The arrays Y, T, etc. will now contain all the data points from all the data sets. You may therefore have to increase MY to be at least the *total* number of data points in *all* data sets, as described in Sec. 3.5 of the Users Manual. In addition, you may have to increase MY1 in the PARAMETER statement (set off by asterisks) in MULTIW to be at least the largest number of data points in any *one* data set. In the same PARAMETER statement, you must have MG1 $\geq$ MG, where MG is defined in Sec. 3.5. In the PARAMETER statement (also set off by asterisks) in USERLF, you may have to reset MYPAR to equal MY in line 0100 in the main subprogram.

E. Input

The data sets are input in the usual way, but concatenated as a single Y array and a single T array. The scale factors are automatically determined, but they should not vary from 1 by an order of magnitude or more. If necessary, you should rescale the input data sets so that the scale factors agree to within about a factor of 2. This accuracy is usually easily achieved by eye.

Normally, the only extra input that you need for splicing are the Control Variables NLINF and IUSER(20), and possibly IUSER(20+J), J=1,..., $N_d - 1$. The normal usage of these input variables is given below. Other possible options are discussed in Sec. F. You must also input DOUSNQ=T and leave NONNEG=T.

IUSER(20) = N_d, the number of data sets.

NLINF = $N_L + N_d - 1$, and N_L has only three allowed values. (Note, however, that it is NLINF, not N_L, that you input.):

$N_L = N_d$ for 1 different constant nonnegative baseline ("dust term") for each of the N_d data sets. This is probably the most common case.

= 1 for the same baseline (after proper rescaling) for all data sets. This would only be reasonable if the baseline were the same *and* reproducible over all data sets.

= 0 for no baseline at all.

IUSER(20+J), J= 1,..., N_d−1, = the subscript of the last point of data set J in the concatenated Y array. This specifies the boundaries of the data sets in the concatenated arrays.

IUSER(21) = 0 specifies that the boundaries IUSER(21),... are to be found automatically by CONTIN. This is only allowed if the T array in each data set is monotonically increasing and if the T ranges of successive data sets overlap. Otherwise a fatal error message occurs, and you must input IUSER(21),... yourself.

F. Further possibilities and details

For standard applications of the Splice Package, you do not have to read this section. IUSER(20)=1 can be useful if you want to speed up the standard CONTIN analysis by about a factor of 2. "Adaptive Weighting," using IUSER(19), can be useful if there are difficulties in specifying the noise statistics of your data. This can be seen in strong systematic variations in the magnitudes of the (otherwise reasonably well-scattered) weighted residuals. On the other hand, systematic oscillations of the residuals, with large regions only positive and others only negative, indicate problems with the model or data (i.e., systematic errors), and Adaptive Weighting cannot help here. Adaptive Weighting can also be used if you only have a single data set.

With Adaptive Weighting, the weighted residuals from the standard analysis are used

to compute refined least-squares weights for a further analysis. For a given data point, the reciprocal of the mean-square weighted residuals in a window 2*IUSER(19)+1 points wide and centered at the given point are used to compute "adaptive" weights, for the final analysis. A similar approach has been used in the CD package since 1980. However, you should only resort to this if the standard weighted analysis shows strong outliers or clear systematic modulations in the magnitudes of the weighted residuals.

IUSER(19) > 0 causes the window for Adaptive Weighting to be 2*IUSER(19)+1 points wide. IUSER(19)>3 is recommended to avoid excessive bias. A reasonable choice would probably often be IUSER(19)=6.

≤ 0 (default) suppresses Adaptive Weighting.

IUSER(20) = 1 will cause a faster Preliminary Unweighted Analysis to be performed, cutting the standard computation time almost in half.

≤ 0 will cause the standard CONTIN Preliminary Unweighted Analysis to be performed. This is default, since it does not restrict constraints, etc. An IUSER(19)>0 effectively changes an IUSER(20)≤0 to IUSER(20)=1.

The method is exact as far as the rescaling of the data, y_{kj}, is concerned, but the least-squares weights are not simultaneously rescaled. (This is the only reason for recommending in Sec. E that you rescale your input data by eye if the scale factors are very different.) As a final refinement, the least-squares weights are rescaled by the estimated scale factors, and the analysis repeated. LUSER(10)=.TRUE. will suppress this repeated analysis and cut the computation time almost in half. This is only recommended if the scale factors for your input data sets agree to within about a factor of 2 and if computation time is important. You can try a few test runs with both values of LUSER(10) and see if the results significantly change.

LUSER(10) = T to suppress the final analysis with the rescaled least-squares weights.

= F (default) for a final analysis with rescaled weights.

G. Interpreting the output

One solution from an analysis on a VAX/VMS system of the test data supplied with the Splice Package is shown on the next page. The CHOSEN SOLUTION had two peaks, and the solution shown is the one with the smallest ALPHA that still has only two peaks. This is slightly preferable to the CHOSEN, because it has less bias, but is still just as parsimonious.

Two data sets following PCS statistics were synthesized with two δ–functions at $\lambda = 0.02$ and $\lambda = 0.1$. The amplitudes in the last data set were both 0.2 plus a constant "dust term" with an amplitude of 0.004. The first data set had all three amplitudes a factor of two greater; therefore $\delta_1 = -0.5$, ideally. Since the dust terms were identical (after proper rescaling), $N_L = 1$ could have been used. However, an independent dust term for each data set was allowed by using $N_L = 2$ (i.e., inputting NLINF = 3).

MOMENT(0) agrees well with the amplitudes for each peak, and the means, MOMENT(1)/MOMENT(0), agree well with the positions of the δ–functions. The dust terms for the two data sets are the first two LINEAR COEFFICIENTS and the last one is δ_1; the agreement is very good in all cases. As noted in Sec. 3.6, you should not expect exact agreement with your results, but the noise level was set extremely low so that you should also have good agreement with the parameters just mentioned.

The Splice Package outputs a table at the end of the analysis to help you judge whether a global analysis was appropriate. This table has been attached to the bottom of the sample solution shown here.

SD(single) is STD. DEV. from a preliminary conventional CONTIN anlaysis of the single data set; i.e., it is the estimated standard deviation of the weighted fit to the data.

SD(global) is as SD(single), except that the fit is using the solution from the final global analysis of all the data sets simultaneously (e.g., splicing all the sets together).

"Incompatability ratio" = SD(global)/SD(single). It is a very sensitive measure of the deterioration in the global fit, when all data sets are forced to have the same distribution, $s(\lambda)$. The only individual degrees of freedom are one scale factor for each data set and possibly a dust term (if $N_L = N_d$). Therefore, you shouldn't be too disappointed if these ratios climb above 1.0, and very confident if they don't. Furthermore, SD(global) cannot go below a lower limit determined by inconstencies in the model (e.g., slight changes in conditions from one data set to the next), whereas SD(single) is limited only by noise in a single data set. So, once the inconsistency limit has been reached by SD(global), the Incompatibility ratio will increase with increasingly accurate data. Therefore, SD(global) is a better absolute criterion of how well the global model fits the data. On the other hand, if one data set has both Incompatibility ratio *and* SD(global) significantly larger than the others, then this is strong evidence that this data set is inconsistent with the rest.

The least-squares weights are also scaled to yield a STD. DEV. of about 1.0 in the final global analysis. (In the example shown here, it happens to be 1.000, but you should not be concerned if your test run yields 0.95 or 1.05.) This is an overall Incompatibility ratio, and, for the reasons in the preceding paragraph, is almost always above 1.0 with global analyses of real data.

H. Diagnostics

This section contains diagnostic messages in the Splice Package. See Sec. 5 of the Users Manual for general hints, for an explanation of abbreviations below, and for diagnostics that you do not find here.

FIT1Y 1 (F) The present version of the Splice Package uses an unregularized analysis in some parts, and this failed to converge.

PR: (1) Check for gross errors or inconsistencies in the input data. (2) A properly stabilized version will be available later.

Test data for Splice Applications Package (April 1991)

ALPHA 4.50E-07 ALPHA/S(1) 1.83E-13 OBJ. FCTN. 2.04033E+02 VARIANCE 1.92205E+02 STD. DEV. 1.000E+00 DEG FREEDOM 7.944 PROB1 TO REJECT 0.001 PROB2 TO REJECT 1.000

ORDINATE	ERROR	ABSCISSA
0.000E+00	4.3D-18	1.00E-03X
0.000E+00	1.6D-17	1.26E-03X
0.000E+00	9.4D-18	1.58E-03X
0.000E+00	2.0D-18	2.00E-03X
0.000E+00	1.6D-17	2.51E-03X
0.000E+00	2.9D-18	3.16E-03X
0.000E+00	1.7D-17	3.98E-03X
0.000E+00	1.7D-18	5.01E-03X
0.000E+00	2.0D-17	6.31E-03X
0.000E+00	2.5D-17	7.94E-03X
0.000E+00	3.4D-17	1.00E-02X
0.000E+00	2.4D-17	1.26E-02X
9.294E-01	1.1D+00	1.58E-02X....
3.157E+01	8.7D-01	2.00E-02X....
1.182E+02	7.5D-01	2.51E-02X....
0.000E+00	1.8D-16	3.16E-02X
0.000E+00	2.1D-16	3.98E-02X
0.000E+00	7.7D-17	5.01E-02X
0.000E+00	3.0D-16	6.31E-02X
0.000E+00	1.9D-16	7.94E-02X
1.301E+01	3.5D-02	1.00E-01
4.323E-03	1.0D-02	1.26E-01X
0.000E+00	1.3D-16	1.58E-01X
0.000E+00	2.9D-16	2.00E-01X
0.000E+00	1.0D-16	2.51E-01X
0.000E+00	1.6D-17	3.16E-01X
0.000E+00	7.5D-17	3.98E-01X
0.000E+00	2.9D-17	5.01E-01X
0.000E+00	7.4D-17	6.31E-01X
0.000E+00	5.7D-17	7.94E-01X
0.000E+00	1.9D-17	1.00E+00X

LINEAR COEFFICIENTS = 3.9472E-03 +- 4.5D-05 3.9371E-03 +- 4.6D-05 -4.9997E-01 +- 2.5D-05

PEAK 1 GOES FROM 1.000E-03 TO 7.943E-02

J	MOMENT(J)		PERCENT ERROR	M(J)/M(J-1)	PERCENT ERROR	J
-1	1.0017 X (10**)	1)	1.3E-01			
0	2.0021 X (10**)	-1)	8.8E-02	1.9988E-02	2.1E-01	0
1	4.0090 X (10**)	-3)	2.1E-01	2.0024E-02	2.9E-01	1
2	8.0434 X (10**)	-5)	4.7E-01	2.0064E-02	6.8E-01	2
3	1.6174 X (10**)	-6)	8.9E-01	2.0108E-02	1.4E+00	3

(STD. DEV.)/MEAN = 4.4E-02

PEAK 2 GOES FROM 1.000E-01 TO 1.000E+00

J	MOMENT(J)		PERCENT ERROR	M(J)/M(J-1)	PERCENT ERROR	J
-1	1.9981 X (10**)	0)	1.1E-01			
0	1.9985 X (10**)	-1)	7.1E-02	1.0002E-01	1.8E-01	0
1	1.9989 X (10**)	-2)	3.0E-02	1.0002E-01	1.0E-01	1
2	1.9994 X (10**)	-3)	5.7E-02	1.0003E-01	8.7E-02	2
3	2.0001 X (10**)	-4)	1.4E-01	1.0003E-01	1.9E-01	3

(STD. DEV.)/MEAN = 7.5E-03

MOMENTS OF ENTIRE SOLUTION

J	MOMENT(J)		PERCENT ERROR	M(J)/M(J-1)	PERCENT ERROR	J
-1	1.2015 X (10**)	1)	1.1E-01			
0	4.0006 X (10**)	-1)	5.7E-02	3.3297E-02	1.6E-01	0
1	2.3998 X (10**)	-2)	4.3E-02	5.9987E-02	9.9E-02	1
2	2.0799 X (10**)	-3)	5.8E-02	8.6669E-02	1.0E-01	2
3	2.0163 X (10**)	-4)	1.3E-01	9.6943E-02	1.9E-01	3

(STD. DEV.)/MEAN = 6.7E-01

(FOR ALPHA/S(1) = 1.83E-13) PRUNS = 0.0132 PUNCOR = 0.0195 0.1204 0.0127 0.2185 0.1485

Data set	SD(global)	SD(single)	Incompatibility Ratio
1	1.059E-05	9.94E-01	
2	1.015E-05	1.031E-05	9.85E-01

FIT1Y 2 (W) More degrees of freedom than data points were obtained in the preliminary analysis of a single data set. SD(single) will be set to -1.0 and the analysis will continue. See FIT1Y 1 for reasons and PRs.

MULTIW 1 (F) N_d >MDSET.
PR: (1) Check your input value for IUSER(20). (2) Increase MDSET in the PARAMETER statement in the main subprogram, but note that MDSET+20 cannot exceed the DIMENSION specification for IUSER (see Sec. 4.1.1).

MULTIW 2 (F) The automatic determination of the data set boundaries has produced an excessive number of boundaries.
PR: See the instructions for IUSER(21)=0 in Sec. E. If your T arrays do not satisfy the monotonicity and overlap restrictions, input the boundaries yourself with IUSER(21),...

MULTIW 3 (F) The automatic determination of the data set boundaries has produced a number of boundaries that disagrees with your IUSER(20).
PR: (1) Correct your input of IUSER(20). (2) See PR of MULTIW 2.

MULTIW 4 (F) Either NONNEG = F or NEQ > 0, and these are prohibited when you are using the non-standard CONTIN.
PR: (1) Correct your input of NONNEG and NEQ. (2) Do the standard analysis by inputting IUSER(20)=IUSER(19)=0.

MULTIW 5 (F) N_L does not have one of the three permitted values given in Sec. E.
PR: (1) Correct your input of NLINF. (2) Do the standard analysis by inputting IUSER(20)=IUSER(19)=0.

MULTIW 6 (F) $N_L > 0$ and DOUSNQ=F.
PR: (1) Input DOUSNQ=T. (2) Correct your input of NLINF to correspond to $N_L = 0$. (3) Do the standard analysis by inputting IUSER(20) = IUSER(19) = 0.

MULTIW 7 (F) The number of data points in one data set is either negative or greater than MY1, or MG1 is less than NG+1.
PR: (1) Correct your input of NG or IUSER(21),..., as specified in Sec. E. (2) Correct MY1 or MG1, as specified in Sec. D.

MULTIW 8 (F) Same as SETWT 2 in Sec. 5.3 of the Users Manual.

MULTIW 9 (W) IUSER(19) is either 1 or greater than half the number of data points in one of the data sets. This can cause excessive bias or very insensitive adaptive weighting, respectively.
PR: Check your input of IUSER(19).

MULTIW 10 (F) The weighted deviations of the fit to the data happen to be exactly zero for all points in a window for adaptive weighting. This is very unlikely.
PR: (1) Check for gross errors in your input (e.g., all zeroes for your data or least-squares weights). (2) Increase IUSER(19).

MULTIW 11 (F) Either there are as many free parameters as data points in one of the data sets, or the variance of the fit to the data is exactly zero for one of the data sets. This is very unlikely.
PR: (1) See FIT1Y 2 or PR (1) for MULTIW 10. (2) Decrease NG.

USERLF 1 (F) A subscript in the Y array exceeds MYPAR.
PR: (1) Correct the PARAMETER specification of MYPAR, as specified in Sec. D. (2) Otherwise, this is an Illogical Stop.

USERLF 2 (F) The index i in the sum in eq. (1) is either less than 1 or exceeds NLINF.
PR: (1) Correct your input of NLINF as specified in Sec. E. (2) Otherwise, this is an Illogical Stop.

USERLF 3 (F) Same as MULTIW 5.

USERNQ 1 (F) Same as USERNQ 1 of Sec. 5.3 (misprinted as USERNG 1).

USERNQ 2 (F) Same as MULTIW 5.

Reference

[1] S.W. Provencher, *Comput. Phys. Commun.* **27**, 213-227 (1982).

7. REFERENCES

[1] S.W. Provencher, Inverse problems in polymer characterization: Direct analysis of polydispersity with photon correlation spectroscopy, Makromol. Chem. **180**: 201–209 (1979).

[2] S.W. Provencher, A constrained regularization method for inverting data represented by linear algebraic or integral equations, Comput. Phys. Commun. **27**: 213–227 (1982).

[3] S.W. Provencher, J. Hendrix, L. De Maeyer, and N. Paulussen, Direct determination of molecular weight distributions of polystyrene in cyclohexane with photon correlation spectroscopy, J. Chem. Phys. **69**: 4273–4276 (1978).

[4] S.W. Provencher and V.G. Dovi, Direct analysis of continuous relaxation spectra, J. Biochem. Biophys. Meth. **1**: 313–318 (1979).

[5] C. Nave, R.S. Brown, A.G. Fowler, J.E. Ladner, D.A. Marvin, S.W. Provencher, A. Tsugita, J. Armstrong, and R.N. Perham, Pf1 filamentous bacterial virus: X-ray fibre diffraction analysis of two heavy atom derivatives, J. Mol. Biol. **149**: 675–707 (1981).

[6] S.W. Provencher and R.H. Vogel, Regularization techniques for inverse problems in molelular biology, in *Numerical Treatment of Inverse Problems in Differential and Integral Equations*, P. Deuflhard and E. Hairer, eds. (Birkhäuser, Boston, 1983);
Information loss with transform methods in system identification: A new set of transforms with high information content, Math. Biosci. **50**: 251–262 (1980).

[7] S.W. Provencher, An eigenfunction expansion method for the analysis of exponential decay curves, J. Chem. Phys. **64**: 2772–2777 (1976);
R.W. Wijnaendts van Resandt, R.H. Vogel, and S.W. Provencher, Double beam fluorescence lifetime spectrometer with subnanosecond resolution; Application to aqueous tryptophan, Rev. Sci. Instrum. **53**: 1392–1397 (1982).

[8] S.W. Provencher, A Fourier method for the analysis of exponential decay curves, Biophys. J. **16**: 27–41 (1976).

[9] C.L. Lawson and R.J. Hanson, *Solving Least Squares Problems*, (Prentice-Hall, Englewood Cliffs, NJ, 1974).

[10] S.W. Provencher and J. Glöckner, Estimation of globular protein secondary structure from circular dichroism, Biochemistry **20**: 33–37 (1981).

[11] N.R. Draper and H. Smith, *Applied Regression Analysis*, (Wiley, New York, 1966).

[12] C. de Boor, *A Practical Guide to Splines*, (Springer, New York, 1978) Chap. 11.

[13] P.F. Price, A comparison of least-squares and maximum-likelihood estimators for counts of radiation quanta which follow a Poisson distribution, Acta Cryst. A **35**: 57–60 (1979).

[14] W.J. Hemmerle and T.F. Brantle, Explicit and constrained generalized ridge estimation, Technometrics **20**: 109–120 (1978).

[15] H.Z. Cummins and P.N. Pusey, Dynamics of macromolecular motion, in *Photon Correlation Spectroscopy and Velocimetry*, H.Z. Cummins and E.R. Pike, eds. (Plenum, New York, 1977), pp. 181–184.

[16] W. Folkhard, D.A. Marvin, T.H. Watts, and W. Paranchych, Structure of polar pili from *Pseudomonas aeruginosa* strains K and O, J. Mol. Biol. **149**: 79–93 (1981).

[17] H.S. Hou and H.C. Andrews, Least squares image restoration using spline basis functions, IEEE Trans. Computers **C-26**: 856–873 (1977).

[18] C.T. Chang, C.S.C. Wu, and J.T. Yang, Circular dichroic analysis of protein conformation: Inclusion of the β-turns, Anal. Biochem. **91**: 13–31 (1978).

[19] S.W. Provencher, CONTIN: A general purpose constrained regularization program for inverting noisy linear algebraic and integral equations, Comput. Phys. Commun. **27**: 229–242 (1982).

[20] F.B. Hildebrand, *Introduction to Numerical Analysis* (McGraw-Hill, New York, 1956).

[21] S.W. Provencher, A general-purpose constrained regularization method for inverting photon correlation data, in *Photon Correlation Techniques*, E.O. Schulz-Du Bois, ed. (Springer-Verlag, Berlin, 1983), pp. 322–328.

8. INDEXES

8.1 Important Equations

$$y_k \approx \sum_{j=1}^{N_x} A_{kj} x_j\,, \quad k = 1, \ldots, N_y \qquad (3.1-1)$$

$$y_k \approx \int_a^b K(g, t_k)\, s(g)\, dg + \sum_{i=1}^{N_L} \beta_i L_i(t_k)\,, \quad k = 1, \ldots, N_y \qquad (3.1-2)$$

$$y_k \approx \sum_{m=1}^{N_g} c_m K(g_m, t_k)\, s(g_m) + \sum_{i=1}^{N_L} \beta_i L_i(t_k)\,, \quad k = 1, \ldots, N_y \qquad (3.1-3)$$

$$\sum_{j=1}^{N_x} D_{ij} x_j \geq d_i\,, \quad i = 1, \ldots, N_{ineq} \qquad (3.1-4)$$

$$\sum_{j=1}^{N_x} E_{ij} x_j = e_i\,, \quad i = 1, \ldots, N_{eq} \qquad (3.1-5)$$

$$\mathrm{VAR} \equiv \sum_{k=1}^{N_y} w_k \left(y_k - \sum_{j=1}^{N_x} A_{kj} x_j \right)^2 = \text{minimum} \qquad (3.2.1-1)$$

$$\mathrm{VAR} + \alpha^2 \sum_{i=1}^{N_{reg}} \left(r_i - \sum_{j=1}^{N_x} R_{ij} x_j \right)^2 = \text{minimum} \qquad (3.2.2-1)$$

8.2 Control Variables

8.3 Other Terms