(12) LEVEL III

B 5

AD

TECHNICAL REPORT ARBRL-TR-02290

FINLIE: A FORTRAN PROGRAM FOR FITTING ORDINARY DIFFERENTIAL EQUATIONS WITH NONLINEAR PARAMETERS TO DATA

James W. Bradley

February 1981

US ARMY ARMAMENT RESEARCH AND DEVELOPMENT COMMAND
BALLISTIC RESEARCH LABORATORY
ABERDEEN PROVING GROUND, MARYLAND

Destroy this report when it is no longer needed.
Do not return it to the originator.

Secondary distribution of this report by originating or sponsoring activity is prohibited.

Additional copies of this report may be obtained from the National Technical Information Service, U.S. Department of Commerce, Springfield, Virginia 22161.

The findings in this report are not to be construed as an official Department of the Army position, unless so designated by other authorized documents.

The use of trade names or manufacturers' names in this report does not constitute indorsement of any commercial product.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER TECHNICAL REPORT ARBRL-TR-02290	2. GOVT ACCESSION NO. AD-A098 238	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) FINLIE: A FORTRAN Program for Fitting Ordinary Differential Equations With Nonlinear Parameters to Data		5. TYPE OF REPORT & PERIOD COVERED Final 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) James W. Bradley		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS U.S. Army Ballistic Research Laboratory ATTN: DRDAR-BLL Aberdeen Proving Ground, MD 21005		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 1L161102AH43
11. CONTROLLING OFFICE NAME AND ADDRESS U.S. Army Armament Research & Development Command U.S. Army Ballistic Research Laboratory ATTN: DRDAR-BL Aberdeen Proving Ground, MD 21005		12. REPORT DATE FEBRUARY 1981 13. NUMBER OF PAGES 95
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release, distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Nonlinear Least Squares	Parameter Optimization
Chapman-Kirk Fitting Technique	FORTRAN Least Squares Program
Marquardt Algorithm	Fitting Differential Equations
Curve Fitting	Differential Corrections

20. ABSTRACT (Continue on reverse side if necessary and identify by block number) (lcb)

This paper presents and documents a FORTRAN program FINLIE for fitting a system of ordinary differential equations (or a system of algebraic or transcendental equations) to observed data. FINLIE determines those values of the possibly nonlinear system parameters and initial conditions that yield a best fit--in the least squares sense--of the solution curves to measurements of one or more of the dependent variables. The basic fitting technique is Chapman-Kirk, with the Marquardt algorithm aiding convergence. The data from more than (Continued)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. ABSTRACT (Continued):

one experiment can be handled simultaneously to obtain one common set of parameters and a set of initial conditions for each experiment. For each computer run, the value of any parameter or initial condition can be held fixed or adjusted by FINLIE.

TABLE OF CONTENTS

LIST OF TABLES

FIGURE

I. INTRODUCTION

This report presents and discusses a general-purpose FORTRAN equation-fitting program called FINLIE.

Assume that the behavior of some physical system can be adequately described by a set of equations involving one independent variable x and N2 dependent variables (N2 $\geq$ 1). FINLIE requires that these equations be reducible to one of two forms:

(a) a system of N2 first-order ordinary differential equations of the form

$$dy_j/dx = f_j\ (x, Y, C) \quad [j=1, 2, \ldots N2] \qquad (1)$$

where Y is the vector of N2 dependent variables:

$$Y \equiv (y_1, y_2, \cdots y_{N2})$$

and where C is a vector of N3 linearly independent parameters (N3 $\geq$ 0):

$$C \equiv (c_1, c_2, \cdots c_{N3})$$

(b) a system of N2 algebraic and/or transcendental equations of the form

$$y_j = g_j\ (x, Y_0, C) \qquad [j = 1, 2, \ldots N2] \qquad (2)$$

where Y_0 is the initial condition vector:

$$Y_0 = (y_{10}, y_{20}, \cdots y_{N2,0})$$

The user writes his system (1) or (2) as a FORTRAN subroutine whose name is submitted to FINLIE. FINLIE's task is to adjust the parameters and initial conditions of (1) or (2) so as to fit the solution curves to measurements taken on one or more of the dependent variables. For system (1), no knowledge of the form of the solution is necessary. Indeed, we may in general assume that system (1) possesses no closed-form solution of the form (2). Otherwise, we would fit the solution equations rather than the differential equations.

System (1) can be linear or nonlinear in the parameters; system (2) can be linear or nonlinear in the parameters and in the initial conditions. However, linear parameters and initial conditions are not much of a challenge to FINLIE. Indeed, the word FINLIE can be viewed as an acronym for "FItting NonLInear Equations"; the program was created to handle nonlinear situations. (System (1) may also be nonlinear in the more common sense of "nonlinear in the dependent variables"; for our purposes, this is irrelevant.)

As a rather elementary example of system (1), consider:

$$\left.\begin{aligned} dy_1/dx &= 1/y_2 \\ dy_2/dx &= -c_1\ (1+c_2y_2)y_2,\ c_1 \neq 0 \end{aligned}\right\} \quad (3)$$

Here N2 = N3 = 2. If x is interpreted as distance, y_1 as time and y_2 as the magnitude of a missile's velocity, then (3) is essentially the drag equation for a horizontal flight in which the drag coefficient varies linearly with Mach number.

One of the reasons we chose this particular example is that it does possess a closed-form solution:

$$\left.\begin{aligned} y_1 &= y_{10} - c_2\ (x - x_0) + (b/c_1)\ (u - 1) \\ y_2 &= (bu - c_2)^{-1} \\ \text{where} \quad u &= \exp\ [c_1\ (x - x_0)] \\ b &= c_2 + (y_{20})^{-1} \end{aligned}\right\} \quad (4)$$

In "real life" we would always fit (4)--which is of the form (2)--and forget about (3). In this report, however, we will use both (3) and (4) to illustrate our remarks.

FINLIE is given measurements on the first N1 of the N2 dependent variables; that is, on y_1, y_2 ... y_{N1} where $1 \leq N1 \leq N2$. The m-th data point R_m thus consists of N1 measurements at the independent variable value x_m:

$$R_m = (x_m, \bar{y}_{1m}, \bar{y}_{2m}, \ldots \bar{y}_{N1,m})$$

where $\bar{y}_{jm}$ denotes the measured value of y_j at x_m.

Assume that the measurements have been obtained from one or more distinct experiments, each experiment having its own initial condition vector. Because our first practical application of FINLIE was to rounds fired in an enclosed range, we will call each distinct experiment a round. By "multi-round" data, then, we mean NR sets of measurements ($NR \geq 1$), all applicable to the same system of equations and hence helping to determine the single parameter vector C, but each measurement set determining its own initial condition vector.

Thus there are NR × N2 initial conditions to be determined:

$$IC = \{(Y_0)_1, (Y_0)_2, \ldots (Y_0)_{NR}\}$$

FINLIE requires that these initial conditions refer to the same independent variable value x_0 for every round. However, x_0 need not coincide with any value x_m at which measurements were taken and x_0 need not even fall within the interval bounded by the smallest and largest of the x_m values. (Of course, the farther x_0 lies from that interval, the more unreliable is the extrapolation to that point.)

We assume that within each round, the x_m values increase with increasing m. For example, if we have two rounds with four and five data points, respectively, then

$$x_1 < x_2 < x_3 < x_4$$

and

$$x_5 < x_6 < x_7 < x_8 < x_9$$

but no demands are made on the combined ordering of the nine values. A member of the first string of inequalities above can be less than, equal to or greater than some member of the second string.

For convenience we coin the word "paramic" to mean "parameter or initial condition." Of course, the initial conditions are parameters of a sort: parameters whose values can change with x_0 and with the round. Thus, for example, system (4) could have been written in terms of four "parameters"; say, in the form

$$y_1 = c_3 - c_2 x + (c_4/c_1)\, z$$

$$y_2 = (c_4 z - c_2)^{-1}$$

where $z = \exp(c_1 x)$. This form conceals the fact that the values of two of the four c_j's will change with the initial conditions.

By our definition, a parameter is independent of the choice of x_0 and applies to (and is influenced by the measurements from) all the rounds. This is the essential condition we impose on the NR rounds to be fitted simultaneously: that the same parameter vector C applies to each round. The measured data for any one round may be incapable of determining C adequately; the combined rounds have a much better chance.

FINLIE's task is to find the set of paramics

$$P = \{IC, C\} \tag{5}$$

that best fits the solution curves to the multi-round measurements.

Note that P consists of NR × N2 initial conditions and N3 parameters, a total of

$$N \equiv (NR \times N2) + N3 \tag{6}$$

paramics. By a "best fit", we mean a least squares fit. That is, FINLIE seeks a particular set P--call it $\hat{P}$--that minimizes ε, the sum of the weighted squares of the residuals of the fit:

$$\varepsilon(P) = \sum_{m=1}^{N4} \sum_{j=1}^{N1} w_{jm} \left[\bar{y}_{jm} - y_j\ (x_m,\ P)\right]^2 \tag{7}$$

where

N4 = the total number of data points R_m for all the rounds;

w_{jm} = a non-negative weighting factor associated with $\bar{y}_{jm}$;

$y_j(x_m,P) = y_j$ evaluated at x_m, using the current value of P.

Other convenient measures of the goodness of fit include:

(a) the estimated variance of the fit = $s^2 \equiv \frac{\varepsilon(P)}{N4-N}$

(b) the estimated standard deviation of the fit = s

(c) the estimated probable error of the fit = 0.67449 s.

Note that for a least squares fit we must have N4 > N; that is, there must be more data points than paramics. (We also assume that the number of data points in each round exceeds N2, the number of initial conditions for each round.)

The function ε is nondimensional. Hence, if we let

$$[\ ]_d \equiv \text{dimensions of } [\ ],$$

Eq. (7) implies that

$$[w_{jm}]_d = [y_j^{-2}]_d \tag{8}$$

If the user fails to specify the values of the weights, FINLIE will set all weights to unity. This may or may not be adequate. Usually the weights are chosen so that each term in (7) is of the same order of magnitude. This can be done by making w_{jm} inversely proportional to the square of the uncertainty in measurement $\bar{y}_{jm}$:

$$w_{jm} = K/(\sigma_{jm})^2 \qquad (9)$$

where K is a nondimensional, positive--but otherwise arbitrary--number. That is, in general only relative uncertainties are needed.* Suppose, for example, that there are two measured variables:

y_1 (furlongs), for which the uncertainty in each measurement is about ten furlongs;

y_2 (fortnights), for which each uncertainty is about 0.1 fortnight.

If we choose K equal to, say, $(\sigma_{1m})^2$ in (9), we have

$$w_{1m} = 100/100 = 1 \text{ (furlong)}^{-2}$$

$$w_{2m} = 100/0.01 = 10^4 \text{ (fortnight)}^{-2}$$

Any other weights for which $w_{2m}/w_{1m} = 10^4$ would work as well. In fact, any weights for which the ratio is "close" to 10^4--say, within a factor of two larger or smaller--would probably work as well. Letting FINLIE set all weights at unity, on the other hand, would not work well at all in this situation. The y_1 measurements would then have much too great an influence on the fit; their noise would drown out the y_2 measurements.

If measurements are taken on more than one dependent variable (that is, if N1>1), it may happen that for some data point R_m, one or more (but not all) of the measurements is missing or is clearly very wrong. There is no need to discard the entire data point; it suffices to set the weights of any missing or outlier measurements at zero.

If we are fitting the solution system (2) to the data, FINLIE computes the values $y_j(x_m,P)$ in (7) directly from the given expressions. If we are fitting the differential equation system (1), however, then FINLIE must obtain $y_j(x_m,P)$ by numerical integration. When we have a choice, we pick (2) over (1) to avoid this integration: 'tis a summation devoutly to be missed.

Each time FINLIE is called by the user, it performs one iteration of its search procedure. That is, the user gives FINLIE the paramic set P_0 and FINLIE returns a set P_1. P_1 is almost certainly not the desired solution, but it should be an improvement over P_0 in the sense that $\varepsilon(P_1) < \varepsilon(P_0)$. The user then gives FINLIE the set P_1 and gets back P_2, and so on. The process stops when a specified convergence criterion is satisfied or some computational disaster arises.

*However, for an absolute interpretation of ε (and any error measure based on ε) K should be 1.

To illustrate some of the above generalities, we return to our sample systems (3) and (4). Suppose that from three enclosed-range firings we obtain the data points $(x_m, \bar{y}_{1m})$ listed in Table I. Assume that the x_m values in the table are exact but that each of the sixteen y_1 measurements has an associated uncertainty σ_{1m} (seconds).

Table I. Sample Data Points for System (3) or (4)

m	x_m (metres)	$\bar{y}_{1m}$ (seconds)	
1	0.0	2.0000000	Round E1
2	1.0	2.0100507	
3	2.0	2.0202034	
4	3.0	2.0304591	
5	4.0	2.0408189	
6	-3.0	-0.0147728	Round E2
7	-2.0	-0.0098987	
8	-1.0	-0.0049746	
9	0.5	0.0025064	
10	1.5	0.0075577	
11	2.0	0.0101027	
12	0.0	3.0000000	Round E3
13	1.0	3.0033506	
14	2.0	3.0067358	
15	3.0	3.0101561	
16	5.0	3.0171031	

Here we have NR = 3 rounds (the three firings), N4 = 16 measurements and N = 8 paramics. The paramics are the six initial conditions and two parameters:

$$P = \{(y_{10}, y_{20})_{E1}, (y_{10}, y_{20})_{E2}, (y_{10}, y_{20})_{E3}, c_1, c_2\} \tag{10}$$

where we arbitrarily let x_o--the x value at which all six initial conditions apply--be zero. The values of the eight paramics are to be adjusted so as to minimize

$$\varepsilon(P) = \sum_{m=1}^{16} w_{1m} [\bar{y}_{1m} - y_1(x_m, P)]^2$$

Whenever only one dependent variable has been measured (N1 = 1), the user--unless he has information to the contrary--can assume that all the uncertainties σ_{1m} are equal. This simplifies matters by allowing the user to set $w_{1m} = 1$ for all m. Thus, for Table I, we set

$$w_{1m} = 1 \text{ (seconds)}^{-2} \quad [m=1,2,\ldots 16]$$

The "measured" $\bar{y}_{1m}$ values in Table I were actually obtained by rounding to seven decimal places the values computed from the solution system (4), using $x_0 = 0$ and

$$\hat{P} = \{(2,100)_{E1},\ (0,200)_{E2},\ (3,300)_{E3},\ 0.01,\ 0.0001\}$$

The $\bar{y}_{1m}$ values in the table are thus equal to $y_1(x_m,\hat{P})$ to the number of decimal places shown. FINLIE's task--given system (3) or (4) and the Table I data--would be to find $\hat{P}$.

FINLIE must be given another bit of information before it can begin its search for $\hat{P}$: a starting point P_0. For systems (3) and (4) and the Table I data, we gave FINLIE the relatively poor first estimate

$$P_0 = \{(1.5,50)_{E1},\ (-0.5,250)_{E2},\ (2.5,250)_{E3},\ 0.02,\ 0\}$$

FINLIE then proceeded from P_0 to P_1 to P_2 and so on to P_7, an acceptable approximation to $\hat{P}$ (see Table II). Within the idiosyncracies of machine computation, this path from P_0 to P_7 is the same whether we fit system (3) or system (4). As one might expect in a convergent situation, the last two points (P_6 and P_7) are practically coincident. The slight discrepancy between P_7 and $\hat{P}$ is due almost entirely to the round-off error in the $\bar{y}_{1m}$ data of Table I.

Unfortunately, a poor choice of P_0 can sometimes prevent FINLIE's ever finding $\hat{P}$. Hence a reasonable amount of labor expended in determining P_0 may pay dividends. For frequently recurring applications, it may be worthwhile for the user to write his own FORTRAN subroutine for extracting a first estimate P_0 from the data points. Usually only a few of the paramic estimates are critical for obtaining convergence to $\hat{P}$; the remaining paramics can have surprisingly poor first estimates with impunity. And for some systems of equations, the choice of P_0 is very nearly immaterial: all roads lead to $\hat{P}$.

A useful feature of FINLIE is its ability--at the user's request--to hold fixed the input values of any specified paramics, rather than allow those input values to be adjusted by the fitting process. Thus, for example, the effect of a given parameter--say, c_2 in system (3) or (4)--can be <u>suppressed</u> during a computer run by giving that parameter an initial value of zero and specifying that this

Table II. Path from P_0 to P_7 for System (3) or (4) and the Data of Table I.

	Round E1		Round E2	
	y_{10}(s)	y_{20} (m/s)	y_{10}(s)	y_{20}(m/s)
P_0	1.5	50	-.5	250
P_1	1.998	69.75	-.000555497	189.54
P_2	1.9999830	90.83	-.000005638	198.90
P_3	1.9999996	99.18	-.000000546	199.931
P_4	2.0000001	99.993	.000000391	200.009
P_5	2	99.999727	.000000010	199.999576
P_6	2	99.999727	.000000012	199.999612
P_7	2	99.999727	.000000012	199.999612

	Round E3			
	y_{10}(s)	y_{20}(m/s)	c_1(1/m)	c_2(s/m)
P_0	2.5	250	.02	.0
P_1	2.9987	270.35	-.0338	.0096
P_2	2.9999974	298.31	-.0059	.0058
P_3	2.9999997	300.13	.0082	.0122
P_4	3.0000003	300.02	.00997	-.0024
P_5	3	299.999242	.009998427	.000106786
P_6	3	299.999324	.009998405	.000100384
P_7	3	299.999324	.009998405	.000100384

	$10^6\varepsilon(P_n)$	$\varepsilon(P_n)/\varepsilon(P_{n-1})$	std. dev.
P_0	3888205.		.6972
P_1	198.66	.00005	.0050
P_2	15.27	.07688	.0014
P_3	3.33	.21791	.000645
P_4	.20	.05936	.000157
P_5	.0000013	.00001	.000000398
P_6	.0000000023	.00185	.000000017
P_7	.0000000023	.9999995	.000000017

value is to be retained. Since it is the user's task to program his particular version of equation set (1) or (2), we see that the above feature can save the user from programming many versions of the same equations, the versions differing only in the nature of the parameters involved. If the version programmed contains every parameter a reasonable (or only slightly unreasonable) person might ever want to consider, the programmer need never alter his program; he can always suppress unwanted parameters at will.

Of course, the user can also fix any paramic at a nonzero value. Consider, for example, the situation where some of the input paramic estimates are known to be respectable, ball-park values, while the remaining estimates are little more than wild guesses. There is no provision in FINLIE for weighting the paramic estimates. Thus when the data are especially noisy, FINLIE--in its single-minded effort to decrease ε--might very well downgrade an excellent estimate. One way to avoid (or at least to try to avoid) this difficulty is to make two computer runs. On the first run, all highly regarded paramic estimates are held fixed, so that the other paramics will be determined for these fixed values. The fixed and determined paramic values from this first run then serve as the estimates for a second run in which none of the paramics is held fixed.

The mechanics of informing FINLIE as to which, if any, of the paramics is to be held constant will be covered later.

In Section II, we discuss in more detail what FINLIE does for the user; in Section III, we discuss what the user must do for FINLIE.

II. INSIDE FINLIE: WHAT FINLIE DOES FOR THE USER

We rewrite the paramic set P of Eq. (5) in the form

$$P = (p_1, p_2, \cdots p_N) \tag{11}$$

where the first NR × N2 elements of P are the initial conditions and the remaining N3 elements are the parameters.

We can regard P as a point in an N-dimensional paramic space S. Then $\varepsilon(P)$, as defined by Eq. (7), is the value of the continuous scalar point function ε at point P. For each point P in the paramic space S, there corresponds a single value $\varepsilon(P)$. FINLIE's task, given a starting point P_0, is to search S for a point $\hat{P}$ that yields a minimum value $\varepsilon(\hat{P})$. (When more than one minimum exists, our choice of starting point P_0 usually determines whether or not $\varepsilon(\hat{P})$ is the desired absolute minimum.)

The fitting process carried out by FINLIE can best be explained in terms of a single-round situation. Once the single-round procedure has been established, it will then be relatively easy to see how the process can be extended to any number of rounds.

Hence we introduce a single-round paramic set Q:

$$Q = \{q_1, q_2, \cdots q_{N23}\} \tag{12}$$

where

$$N23 = N2 + N3, \tag{13}$$

the number of paramics for a single round. The first N2 elements of Q are the initial conditions and the remaining N3 elements are the parameters. For our sample system (3) or (4), we have (for any one round)

$$Q = (y_{10}, y_{20}, c_1, c_2).$$

Similarly, we introduce a single-round version of $\varepsilon(P)$:

$$\gamma(Q) = \sum_m \sum_{j=1}^{N1} w_{jm} [\bar{y}_{jm} - y_j(x_m, Q)]^2 \tag{14}$$

where the summation on m is over the measured data for the single round. (For Round E2 of Table I, for example, m would range from 6 to 11.) Note that the ε for a multi-round situation, Eq. (7), is the sum of the γ's for the individual rounds:

$$\varepsilon = \sum_{n=1}^{NR} (\gamma)_{En} \tag{15}$$

For the moment, then--a rather long moment, lasting until Section II (G)--we will assume that FINLIE is handling a single-round situation: only one set of initial conditions is being determined.

A. Condition for a Minimum γ

We can regard Q as a point in an N23-dimensional space S_1. A necessary (though insufficient) condition for point $\hat{Q}$ to yield a minimum value of γ is that the gradient of γ at that point be the zero vector:

$$\text{grad } \gamma(\hat{Q}) \equiv \left(\frac{\partial \gamma(\hat{Q})}{\partial q_1}, \frac{\partial \gamma(\hat{Q})}{\partial q_2}, \ldots \frac{\partial \gamma(\hat{Q})}{\partial q_{N23}} \right)_{S_1} = \vec{0} \tag{16}$$

Thus FINLIE must seek a point that satisfies all N23 components of (16) simultaneously. From Eq. (14), we see that at any point Q

$$\frac{\partial \gamma(Q)}{\partial q_k} = -2\ \beta_k(Q) \quad [k=1,2,\ldots N23] \tag{17}$$

where

$$\beta_k(Q) \equiv \sum_m \sum_{j=1}^{N1} w_{jm}\ [\bar{y}_{jm} - y_j(x_m,Q)]\ .\ D_{jk}(x_m,Q) \tag{18}$$

$$D_{jk}\ (x_m,Q) \equiv \partial y_j(x_m,Q)/\partial q_k \tag{19}$$

and where, in our dimensional notation,

$$[\beta_k]_d = [q_k^{-1}]_d \tag{20}$$

$$[D_{jk}]_d = [y_j q_k^{-1}]_d \tag{21}$$

Thus condition (16) can be written in the form

$$\boxed{\beta_k\ (\hat{Q}) = 0} \quad [k=1,2,\ldots N23] \tag{22}$$

The N23 components β_k define a vector:

$$\vec{\beta}\ (Q) \equiv \left(\beta_1(Q),\ \beta_2(Q),\ \ldots\ \beta_{N23}(Q)\right)_{S_1} \tag{23}$$

which, from (16-17), has the direction of the negative gradient of γ at point Q; that is, the direction in which the rate of decrease of γ is greatest:

$$\vec{\beta} = -(1/2)\ \text{grad}\ \gamma. \tag{24}$$

$\vec{\beta}$ is a vector point function of Q. For each point Q in the paramic space S_1, there corresponds a unique vector $\vec{\beta}$. Thus FINLIE's search for a point $\hat{Q}$ that yields a minimum value $\gamma(\hat{Q})$ has become a search for a point $\hat{Q}$ at which $\vec{\beta}$ is zero.

B. Influence Coefficients

The partial derivatives D_{jk} in (18) are sometimes called "influence" (or "sensitivity") coefficients because they reflect the influence of the paramics on the solution curves.

To satisfy (22), FINLIE must be able to evaluate the influence coefficients at any point Q for each independent variable value x_m.

The manner in which FINLIE evaluates $D_{jk}(x_m,Q)$ depends on which equation set, (1) or (2), we are fitting to the data.

C. Influence Equations for System (1)

If we give FINLIE the differential equation system (1), then we must also give FINLIE a system of differential equations for the influence coefficients. Taking the partial derivative of each side of (1) with respect to paramic q_k, we have

$$\frac{\partial}{\partial q_k}\left(\frac{d\,y_j}{dx}\right) = \frac{\partial f_j}{\partial q_k}$$

or, assuming that the order of differentiation can be reversed,

$$\frac{d\,D_{jk}}{dx} = \frac{\partial f_j}{\partial q_k} \qquad (25)$$

$$\begin{bmatrix} j=1,2,\ldots N2 \\ k=1,2,\ldots N23 \end{bmatrix}$$

The system (25) is subject to the initial conditions:

$$D_{jk}\,(x_0,Q) = \begin{cases} 1 \text{ if } j=k \\ 0 \text{ otherwise} \end{cases} \qquad (26)$$

(These initial conditions merely reflect the fact that the influence coefficient D_{jj} is, by our definition, $\partial y_j/\partial y_{j0}$ and hence equals one at x_0.)

The paramics affect $f_j\,(x,Y,C)$ in two ways: indirectly through their effect on the dependent variable vector Y and directly through the parameter vector C. Hence (25) can be rewritten in the more cumbersome but (possibly) more revealing form:

$$\frac{d\,D_{jk}}{dx} = \sum_{i=1}^{N2}\left(\frac{\partial f_j}{\partial y_i}\right)_C D_{ik} + \begin{cases} 0 \text{ if } k \leq N2 \\ \left(\dfrac{\partial f_j}{\partial c_{k-N2}}\right)_Y \text{ if } k > N2 \end{cases} \qquad (27)$$

$$\begin{bmatrix} j=1,2,\ldots N2 \\ k=1,2,\ldots N23 \end{bmatrix}$$

where

subscript C indicates that x and vector C are considered constant in taking the partial derivatives of $f_j(x,Y,C)$;

subscript Y indicates that x and vector Y are considered constant in taking the partial derivatives of $f_j(x,Y,C)$.

Thus, by "paramic differentiation" we obtain an auxiliary system of differential equations (27) whose solutions are the influence coefficients needed to fit equation set (1). Note from (27) that these influence equations are always linear in the influence coefficients D_{jk}. The number of influence equations is

$$NA \equiv N2 \times N23 \tag{28}$$

The user must include his version of system (27) in the FORTRAN subroutine containing his version of system (1).

For our by-now-familiar example, system (3), we have NA = 2 × 4 = 8. The eight influence equations for system (3) are shown in the upper portion of Table III, where $(\ )' \equiv d(\ )/dx$.

Recall that our only purpose in obtaining the influence coefficients is to be able to evaluate $\beta_k(Q)$, Eq. (18), in our effort to satisfy condition (22). From (18) we see that β_k involves D_{jk} only for j=1 to N1; that is, only for the measured variables. Yet Eqs. (25-27) show j running from 1 to N2; that is, over all the dependent variables. Do we have more influence equations here than we need? The answer is: no. We have implicity assumed that there are no extraneous dependent variables in system (1): all of the unmeasured dependent variables are needed to solve the differential equations for the measured variables. Hence the D_{jk} for $N1 < j \leq N2$ are also needed.

For our example, system (3) with N1=1, y_2 is clearly needed to solve the differential equation for y_1. Thus each D_{2k} is also needed, as we see in Table III (A). (On the other hand, if y_2 had been the only measured variable in system (3), then y_1 would be an extraneous variable and should be thrown out.)

The mechanics of writing and submitting the influence equations will be discussed later. FINLIE will automatically assign the proper initial conditions (26) and integrate the influence equations simultaneously with the original system (1) to obtain $y_j(x_m,Q)$ and $D_{jk}(x_m,Q)$ at each x_m.

Table III. Influence Equations for System (3) and for System (4)

(A) For System (3):

$$(D_{11})' \equiv \partial y_1'/\partial y_{10} = -(y_2^{-2})D_{21}$$

$$(D_{12})' \equiv \partial y_1'/\partial y_{20} = -(y_2^{-2})D_{22}$$

$$(D_{13})' \equiv \partial y_1'/\partial c_1 = -(y_2^{-2})D_{23}$$

$$(D_{14})' \equiv \partial y_1'/\partial c_2 = -(y_2^{-2})D_{24}$$

$$(D_{21})' \equiv \partial y_2'/\partial y_{10} = -c_1(1+2c_2y_2)D_{21}$$

$$(D_{22})' \equiv \partial y_2'/\partial y_{20} = -c_1(1+2c_2y_2)D_{22}$$

$$(D_{23})' \equiv \partial y_2'/\partial c_1 = -c_1(1+2c_2y_2)D_{23} - (1+c_2y_2)y_2$$

$$(D_{24})' \equiv \partial y_2'/\partial c_2 = -c_1(1+2c_2y_2)D_{24} - c_1y_2^2$$

where $(D_{11})_0 = (D_{22})_0 = 1$; $(D_{jk})_0 = 0$ for $j \neq k$

(B) For System (4):

$$D_{11} \equiv \partial y_1/\partial y_{10} = 1$$

$$D_{12} \equiv \partial y_1/\partial y_{20} = -(u-1)/(c_1y_{20}^2)$$

$$D_{13} \equiv \partial y_1/\partial c_1 = (b/c_1^2)\ [1-u+c_1(x-x_0)u]$$

$$D_{14} \equiv \partial y_1/\partial c_2 = (u-1)/c_1 - (x-x_0)$$

(C) Unneeded Influence Equations for System (4):

$$D_{21} \equiv \partial y_2/\partial y_{10} = 0$$

$$D_{22} \equiv \partial y_2/\partial y_{20} = (y_2/y_{20})^2\ u$$

$$D_{23} \equiv \partial y_2/\partial c_1 = -by_2^2\ (x-x_0)\ u$$

$$D_{24} \equiv \partial y_2/\partial c_2 = -(u-1)\ y_2^2$$

One final remark. For large systems with many paramics, the exact influence equations (27) can be rather cumbersome. In many cases, certain liberties can be taken with the influence equations: expressions can be approximated by simpler ones, the effect of certain paramics on certain terms in the original equations can be ignored, etc. If done with care and judgment, such simplifications will have no effect on the final answer: the same point Q will be reached with or without the simplifications. Note, however, that discretion is called for. If the user has any doubts as to the merits of some modification to the exact influence equations (and even when he hasn't any doubts), his safest course is to avoid such a modification.

D. Influence Equations for System (2)

If we give FINLIE the solution set (2), then we must also give FINLIE the influence equations obtained by differentiating (2):

$$D_{jk} = \frac{\partial g_j}{\partial q_k} \qquad \begin{bmatrix} j=1,2, \ \ldots \ N1 \\ k=k,2, \ \ldots \ N23 \end{bmatrix} \tag{29}$$

We assume--as with system (1)--that there are no extraneous variables in system (2). (For (2), this means that the initial conditions for all of the unmeasured dependent variables are needed to evaluate the expressions for the measured variables.) However, the D_{jk} for $N1 < j \leq N2$ are superfluous and should be ignored. Thus the number of influence equations required to fit system (2) is

$$NB \equiv N1 \times N23 \tag{30}$$

To fit system (4), for example, (where N1 = 1 and N2 = 2), the D_{2k} values are not required and we need submit only four influence equations to FINLIE. These equations are shown in Table III (B). FINLIE will automatically set all undefined D_{jk}'s to zero. For the sake of completeness, expressions for the unneeded D_{jk} are given in Table III (C), but we emphasize that these latter equations should not be given to FINLIE. Note that the eight expressions for D_{jk} in Table III (B and C) do indeed satisfy the initial conditions indicated in part A of the table.

The remarks in the previous section on the possibility of simplifying the influence equations apply to system (29), although here the urge to simplify may be less compelling.

E. An Overview

To summarize thus far: FINLIE determines the values of $y_j\ (x_m, Q)$ and $D_{jk}\ (x_m, Q)$ either

(i) by numerically integrating a system of N2 plus NA first-order differential equations or

(ii) by evaluating a system of N2 plus NB algebraic or transcendental expressions.

Except for this difference--but what a difference it can be in terms of machine time!--the fitting process used by FINLIE is the same for the two equation sets (1) and (2).

This fitting process consists of adjusting Q until it satisfies condition (22). Of course, it would be pleasant if FINLIE could solve (22) for $\hat{Q}$ in some direct, one-step fashion. No fooling around with Q_0, Q_1, etc; just leap in and solve the N23 equations of (22) for the N23 components of $\hat{Q}$. Unfortunately, when system (22) is nonlinear in one or more of the paramics, no such general one-step scheme exists. Hence FINLIE, expecting the worst, sets out to solve (22) by an iterative process.

Two of the standard iterative techniques are:

(i) differential corrections (alias Taylor-series linearization, alias Gauss method, alias Gauss-Newton method);

(ii) steepest descent (alias gradient search).

FINLIE uses a third method, due to Marquardt*, which is a blend of the first two methods, retaining the best features of each and avoiding their disadvantages. We will discuss enough of the differential corrections and steepest descent techniques to see what is involved in combining the two.

F. Differential Corrections in Space S_1

For each point Q in S_1 there corresponds a position vector $\vec{Q}$. Let $\Delta\vec{Q}$ be the vector from point Q to point $\hat{Q}$:

$$\left.\begin{aligned} \hat{\vec{Q}} &= \vec{Q} + \Delta\vec{Q} \\ &= \vec{Q} + (\Delta q_1, \Delta q_2, \ldots \Delta q_{N23})_{S_1} \end{aligned}\right\} \quad (31)$$

* *See the Bibliography, Part A.*

In the differential corrections technique, we approximate the basic condition (22) by a system of equations (to be derived in the next paragraph) that is <u>linear</u> in the increments Δq_k. We can't solve (22) for $\hat{Q}$, but given a point, say Q_0, we <u>can</u> solve the approximate conditions for an approximate increment vector $\Delta\vec{Q}_0$. This increment is then added to $\vec{Q}_0$ to reach the next way-station on our trek to $\hat{Q}$:

$$\vec{Q}_1 = \vec{Q}_0 + \Delta\vec{Q}_0 \tag{32}$$

Point Q_1 is an improvement over point Q_0 if $\gamma(Q_1)$ is less than $\gamma(Q_0)$. But improvement or not, the differential corrections method plows ahead, using Q_1 to re-solve the approximate equations for a new increment $\Delta\vec{Q}_1$. The process continues in this manner through a series of points until a specified convergence criterion has been met or a specified number of iterations have been performed or some numerical catastrophe occurs.

The desired approximation to condition (22) can be obtained by expanding y_j and D_{jk} in Taylor series about point Q. We have

$$\left.\begin{aligned} y_j\ (x_m,\hat{Q}) = y_j\ (x_m,Q) + \sum_{n=1}^{N23} D_{jn}\ (x_m,Q) \cdot \Delta q_n \\ + \text{ (higher-order terms)} \end{aligned}\right\} \tag{33}$$

$$D_{jk}\ (x_m,\hat{Q}) = D_{jk}\ (x_m,Q) + \text{(higher-order terms)} \tag{34}$$

We assume--an assumption that is not always valid--that Q is close enough to $\hat{Q}$ to permit us to ignore the higher-order terms in (33) and (34). Then from definition (18), we have

$$\beta_k(\hat{Q}) = \sum_m \sum_{j=1}^{N1} w_{jm}\left[\bar{y}_{jm} - y_j\ (x_m,\hat{Q})\right] \cdot D_{jk}\ (x_m,\hat{Q})$$

$$\cong \sum_m \sum_{j=1}^{N1} w_{jm}\left[\bar{y}_{jm} - y_j\ (x_m,Q) - \sum_{n=1}^{N23} D_{jn}(x_m,Q) \cdot \Delta q_n\right] \cdot D_{jk}(x_m,Q)$$

$$\cong \beta_k(Q) - \sum_m \sum_{j=1}^{N1} w_{jm}\, D_{jk}\ (x_m,Q) \cdot \left[\sum_{n=1}^{N23} D_{jn}\ (x_m,Q) \cdot \Delta q_n\right]$$

By rearranging the sums, we obtain

$$\beta_k(\hat{Q}) \cong \beta_k(Q) - \sum_{n=1}^{N23} \alpha_{kn}(Q) \cdot \Delta q_n \tag{35}$$

where

$$\alpha_{kn}(Q) \equiv \sum_{m} \sum_{j=1}^{N1} w_{jm} \cdot D_{jk}(x_m,Q) \cdot D_{jn}(x_m,Q) \tag{36}$$

Thus the conditions $\beta_k(\hat{Q}) = 0$, which hold at a point $\hat{Q}$ where γ is at a minimum, are replaced by the conditions

$$\boxed{\beta_k(Q) = \sum_{n=1}^{N23} \alpha_{kn}(Q) \cdot \Delta q_n} \tag{37}$$

$$[k = 1, 2, \ldots N23]$$

which are applicable to points in the vicinity of $\hat{Q}$.

The quantities $\alpha_{kn}(Q)$ have at least four interesting properties:

$$\left.\begin{aligned} [\alpha_{kn}]_d &= [(q_k q_n)^{-1}]_d \\ \alpha_{nk} &= \alpha_{kn} \\ \alpha_{kk} &> 0 \end{aligned}\right\} \tag{38}$$

$$\alpha_{nn}\,\alpha_{kk} \geq \alpha_{nk}^2$$

The first three properties follow at once from definition (36); the fourth is a consequence of Hölder's Inequality (alias Cauchy's, alias Schwarz's, alias Buniakovski's Inequality). In general, we regard α_{kn} as the (k,n)-th element of an N23 by N23 symmetric matrix α.

In matrix form, (37) becomes

$$\boxed{[\alpha(Q) \cdot \Delta\vec{Q}^T = \vec{\beta}^T(Q)]_{S_1}} \tag{39}$$

where the superscript T (for Transpose) denotes a column vector and the subscript S_1 indicates that all components are in the N23-dimensional space S_1. For either of our examples, system (3) or (4), (39) becomes

$$\begin{pmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} & \alpha_{14} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} & \alpha_{24} \\ \alpha_{31} & \alpha_{32} & \alpha_{33} & \alpha_{34} \\ \alpha_{41} & \alpha_{42} & \alpha_{43} & \alpha_{44} \end{pmatrix} \begin{pmatrix} \Delta y_{10} \\ \Delta y_{20} \\ \Delta c_1 \\ \Delta c_2 \end{pmatrix} = \begin{pmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \beta_4 \end{pmatrix} \tag{40}$$

System (39) is linear in the increments Δq_k; hence the process of solving for these increments is routine work for the computer. (We assume that a solution does exist; this amounts to assuming that the determinant of matrix α is not zero.)

The differential corrections process, then, consists of substituting Q_0 in (39), solving for $\vec{\Delta Q}_0$, substituting in (39) the point Q_1 obtained by the vector addition $\vec{Q}_1 = \vec{Q}_0 + \vec{\Delta Q}_0$, solving for $\vec{\Delta Q}_1$, etc.

Unfortunately, even when this process converges to some point, there is no guarantee that this point will yield the absolute minimum γ. Condition (22)--which is approximated by the matrix equation (39)--guarantees only that its solution point $\hat{Q}$ will yield some relative extremum value of γ. Space S_1 could be teeming with points of local extremum. Each of these extremum points, including the one we seek, is a sort of black hole in space S_1, capable of drawing a nearby search party into its core. The particular black hole into which we are drawn depends mainly on where we start in space S_1.

G. Differential Corrections in Space S

So far in Section II, we have assumed single-round data, NR = 1. For this situation, the differential corrections technique led to matrix equation (39).

Consider now the three-round situation of Table I. For each round Ei (i = 1, 2, 3), FINLIE forms a vector $\vec{\beta}_{Ei}$ and a matrix α_{Ei} by Eqs. (18) and (36) respectively, using the Q and m indicated below:

Round	Point Q	Range of Subscript m in Eqs. (18) and (36)
E1	$(y_{10})_{E1}$, $(y_{20})_{E1}$, c_1, c_2	1 to 5
E2	$(y_{10})_{E2}$, $(y_{20})_{E2}$, c_1, c_2	6 to 11
E3	$(y_{10})_{E3}$, $(y_{20})_{E3}$, c_1, c_2	12 to 16

In the 8-dimensional space S associated with the eight paramics p_k of Eq. (10), the $\vec{\beta}_{Ei}$ vectors take the form:

$$\left.\begin{aligned} \vec{\beta}_{E1} &= [(\beta_1)_{E1},\ (\beta_2)_{E1},\ 0,\ 0,\ 0,\ 0,\ (\beta_3)_{E1},\ (\beta_4)_{E1}]_S \\ \vec{\beta}_{E2} &= [0,\ 0,\ (\beta_1)_{E2},\ (\beta_2)_{E2},\ 0,\ 0,\ (\beta_3)_{E2},\ (\beta_4)_{E2}]_S \\ \vec{\beta}_{E3} &= [0,\ 0,\ 0,\ 0,\ (\beta_1)_{E3},\ (\beta_2)_{E3},\ (\beta_3)_{E3},\ (\beta_4)_{E3}]_S \end{aligned}\right\} \quad (41)$$

Similarly, in space S the matrix α for round E1 expands to:

$$\alpha_{E1} = \begin{pmatrix} (\alpha_{11})_{E1} & (\alpha_{12})_{E1} & 0 & 0 & 0 & 0 & (\alpha_{13})_{E1} & (\alpha_{14})_{E1} \\ (\alpha_{21})_{E1} & (\alpha_{22})_{E1} & 0 & 0 & 0 & 0 & (\alpha_{23})_{E1} & (\alpha_{24})_{E1} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ (\alpha_{31})_{E1} & (\alpha_{32})_{E1} & 0 & 0 & 0 & 0 & (\alpha_{33})_{E1} & (\alpha_{34})_{E1} \\ (\alpha_{41})_{E1} & (\alpha_{42})_{E1} & 0 & 0 & 0 & 0 & (\alpha_{43})_{E1} & (\alpha_{44})_{E1} \end{pmatrix} \quad (42)$$

with similar expressions for α_{E2} and α_{E3}.

Since the multi-round ε to be minimized is the sum of the single-round γ's, FINLIE obtains the multi-round version of Eq. (40) by summing --in space S--the three single-round $\vec{\beta}_{Ei}$ vectors of Eq. (41):

$$\vec{B} \equiv [\vec{\beta}_{E1} + \vec{\beta}_{E2} + \vec{\beta}_{E3}]_S \tag{43}$$

and the three single-round α_{Ei} matrices:

$$A \equiv [\alpha_{E1} + \alpha_{E2} + \alpha_{E3}]_S \tag{44}$$

The desired multi-round matrix equation is then

$$\boxed{[A(P) \cdot \vec{\Delta}P^T = \vec{B}^T(P)]_S} \tag{45}$$

A detailed form of this equation for our three-round sample system is given in Table IV; the generalization to any number of rounds can be easily visualized.

The N by N symmetric matrix A will always contain

$$NR \times (NR-1) \times N2 \times N2$$

zeroes distributed among the off-diagonal elements of all but the last N3 rows and columns. Let a_{kn} be the (k,n)-th element of matrix A. As in Eqs. (38), we have

$$\left.\begin{aligned} [a_{kn}]_d &= [(p_k p_n)^{-1}]_d \\ a_{nk} &= a_{kn} \\ a_{kk} &> 0 \\ a_{nn}a_{kk} &\geq a_{nk}^2 \end{aligned}\right\} \tag{46}$$

Similarly, if b_k denotes the k-th component of vector $\vec{B}$, then

$$[b_k]_d = [p_k^{-1}]_d \tag{47}$$

We have taken some pains to distinguish between the multi-round paramics p_k and the single-round paramics q_k, which for our three-round sample systems take the form

$$P = ((y_{10}, y_{20})_{E1}, (y_{10}y_{20})_{E2}, (y_{10}y_{20})_{E3}, c_1, c_2)$$

$$Q = (y_{10}, y_{20}, c_1, c_2)$$

Table IV. Matrix Equation for System (3) or (4), Given Data from Three Rounds

$$\begin{pmatrix} (\alpha_{11})_{E1} & (\alpha_{12})_{E1} & 0 & 0 & 0 & 0 & (\alpha_{13})_{E1} & (\alpha_{14})_{E1} \\ (\alpha_{21})_{E1} & (\alpha_{22})_{E1} & 0 & 0 & 0 & 0 & (\alpha_{23})_{E1} & (\alpha_{24})_{E1} \\ 0 & 0 & (\alpha_{11})_{E2} & (\alpha_{12})_{E2} & 0 & 0 & (\alpha_{13})_{E2} & (\alpha_{14})_{E2} \\ 0 & 0 & (\alpha_{21})_{E2} & (\alpha_{22})_{E2} & 0 & 0 & (\alpha_{23})_{E2} & (\alpha_{24})_{E2} \\ 0 & 0 & 0 & 0 & (\alpha_{11})_{E3} & (\alpha_{12})_{E3} & (\alpha_{13})_{E3} & (\alpha_{14})_{E3} \\ 0 & 0 & 0 & 0 & (\alpha_{21})_{E3} & (\alpha_{22})_{E3} & (\alpha_{23})_{E3} & (\alpha_{24})_{E3} \\ (\alpha_{31})_{E1} & (\alpha_{32})_{E1} & (\alpha_{31})_{E2} & (\alpha_{32})_{E2} & (\alpha_{31})_{E3} & (\alpha_{32})_{E3} & S_{33} & S_{34} \\ (\alpha_{41})_{E1} & (\alpha_{42})_{E1} & (\alpha_{41})_{E2} & (\alpha_{42})_{E2} & (\alpha_{41})_{E3} & (\alpha_{42})_{E3} & S_{43} & S_{44} \end{pmatrix} \begin{pmatrix} (\Delta y_{10})_{E1} \\ (\Delta y_{20})_{E1} \\ (\Delta y_{10})_{E2} \\ (\Delta y_{20})_{E2} \\ (\Delta y_{10})_{E3} \\ (\Delta y_{20})_{E3} \\ \Delta c_1 \\ \Delta c_2 \end{pmatrix} = \begin{pmatrix} (\beta_1)_{E1} \\ (\beta_2)_{E1} \\ (\beta_1)_{E2} \\ (\beta_2)_{E2} \\ (\beta_1)_{E3} \\ (\beta_2)_{E3} \\ T_3 \\ T_4 \end{pmatrix}$$

where $(\ )_{Ei}$ denotes the value of $(\ )$ based solely on the measured data from round Ei (i=1,2,3).

$$S_{kn} = (\alpha_{kn})_{E1} + (\alpha_{kn})_{E2} + (\alpha_{kn})_{E3}$$

$$T_k = (\beta_k)_{E1} + (\beta_k)_{E2} + (\beta_k)_{E3} \qquad \begin{bmatrix} k=3,4 \\ n=3,4 \end{bmatrix}$$

The chief reason for taking these pains is that the FINLIE user must himself make this distinction in a multi-round situation. The FINLIE input arguments (to be discussed later) are defined in terms of the N paramics p_k, but the influence equations submitted to FINLIE must always be written in terms of the N23 (= N2 + N3) paramics q_k. The values of the initial conditions may change with the round, but the influence equations themselves, like the original equations (1) or (2) on which they are based, remain the same. Thus, regardless of the number of rounds, there will always be N23 influence coefficients, defined in terms of the N23 paramics q_k, and there will always be NA (= N2 × N23) or NB (= N1 × N23) influence equations (depending on whether the user is working with system (1) or system (2)).

H. Differential Corrections in Space $\tilde{S}$

If the paramics p_k are not all of the same dimension, our paramic space S is a hodgepodge: a salmagundi, a gallimaufry, an olla-podrida of units. Certain computational advantages can be obtained by working in a space $\tilde{S}$ in which the paramics--and hence the components of grad ε--are nondimensional. (The advantages of $\tilde{S}$ are especially compelling in the steepest descent technique, some of whose properties are not scale-invariant.)

To achieve the desired paramic transformation from S to $\tilde{S}$, we note from Eqs. (46) that

$$[a_{kk}]_d = [p_k^{-2}]_d$$

or

$$[(a_{kk})^{\frac{1}{2}} p_k] = 1 \tag{48}$$

That is, the bracketed quantity in (48) is nondimensional. Thus the paramic transformation

$$\boxed{\tilde{p}_k \equiv (a_{kk})^{\frac{1}{2}} p_k} \tag{49}$$

creates the desired* paramic space $\tilde{S}$. The elements of A and $\vec{B}$ in $\tilde{S}$ are

**From Eq. (47), we see that the product $b_k p_k$ is also nondimensional. Thus, the transformation*

$$\tilde{p}_k = b_k p_k$$

seems appealing; it would lead to a space in which all components of $\vec{B}$ are unity. The appeal, however, is illusory. It would not be very wise to use as scale factors the very quantities b_k that we are trying to drive to zero. The scale factors $(a_{kk})^{\frac{1}{2}}$, on the other hand, are never zero (see Eq. (46)).

$$\tilde{a}_{jk} = (a_{jj}\, a_{kk})^{-\frac{1}{2}}\, a_{jk} \tag{50}$$

$$\tilde{b}_{k} = (a_{kk})^{-\frac{1}{2}}\, b_{k} \tag{51}$$

These space $\tilde{S}$ components have the following admirable features:

(i) $\tilde{p}_k$, $\tilde{a}_{jk}$ and $\tilde{b}_k$ are nondimensional;

(ii) the diagonal elements of matrix A are unity:

$$\tilde{a}_{kk} = 1 \tag{52}$$

(iii) the off-diagonal elements of A satisfy the inequality:

$$-1 \leq \tilde{a}_{jk} \leq 1 \tag{53}$$

Finally, the form of matrix equation (45) is unchanged:

$$\boxed{\left[\, A(P) \cdot \vec{\Delta}P^{T} = \vec{B}^{\,T}(P) \,\right]_{\tilde{S}}} \tag{54}$$

the subscript $\tilde{S}$ serving to remind us that all components are now in the scaled paramic space. Ninety-nine percent of the labor in solving (54) for $\Delta\vec{P}$ is usually expended in inverting matrix A. Use of the scaled components $\tilde{a}_{jk}$ tends to increase the accuracy of the matrix inversion process.

Note that each scale factor $(a_{kk})^{\frac{1}{2}}$ in (49) is a function of point P, the current set of paramic values. Hence each time the paramics are up-dated, a new transformation must be made: a new $\tilde{S}$ space created. This is no big problem for a computer. FINLIE handles the scaling to space $\tilde{S}$ and back again to the user's space S; the process is automatic and invisible (in computer jargon, "transparent") to the user.

I. Steepest Descent

Consider a given point P_o and the corresponding vector $\vec{B}(P_o)$ proceeding from that point. Recall that $\vec{B}$ at any point is a vector in the direction of the negative gradient of ε at that point. Hence, provided that the magnitude of $\vec{B}(P_o)$ is not zero (if it were, P_o would be the desired solution $\hat{P}$), $\vec{B}(P_o)$ is the steepest descent vector for point P_o: a vector in whose direction $\varepsilon(P)$ will decrease most rapidly (at least at first) as we move away from P_o. Let P_1 be any other point

in this steepest descent direction:

$$\vec{P}_1 = \vec{P}_o + h \cdot \vec{B}(P_o) \tag{55}$$

where h is a nondimensional positive constant.

There always exists a range of h values, $0 < h \leq h_{max}$, for which the point P_1 obtained by (55) is an improvement: $\varepsilon(P_1) < \varepsilon(P_0)$. The steepest descent method determines the optimum h in this range: the value of h for which ε is a local minimum along the vector $\vec{B}(P_0)$. This can be done by evaluating P_1 and $\varepsilon(P_1)$ for a series of h values: $h_0 < h_1 < h_2 < \ldots$ Presumably, for a while ε will decrease with increasing h. As soon as an h is found for which the ε has increased, the (approximately) optimum h for point P_0 can be determined by interpolation.

Given the new point P_1 based on this optimum h, the next point P_2 will lie in the direction of steepest descent from P_1; that is, along the new vector $\vec{B}(P_1)$. Another optimum h must be determined to obtain P_2. And so on to $\hat{P}$.

The difficulty with this approach is that in the neighborhood of the solution point $\hat{P}$, where $|\vec{B}|$ is nearly zero and yet we are not quite close enough to $\hat{P}$ to be able to quit with honor, further progress is painfully slow. Often the sampling size on h, the Δh intervals, must be shortened beyond all endurance in an effort to find a P_1 for which $\varepsilon(P_1) < \varepsilon(P_0)$. Ingenious variations on the basic steepest descent theme have lessened but not removed this difficulty.

J. Marquardt Interpolation in Space $\tilde{S}$

The two fitting techniques we have discussed so far are:

(i) differential corrections, which in space $\tilde{S}$ is based on matrix equation (54); this equation has the component form

$$\sum_{n=1}^{N} \tilde{a}_{kn}(P) \cdot \Delta\tilde{P}_n = \tilde{b}_k(P) \tag{56}$$

$$[k=1,2,..N]$$

(ii) steepest descent, based on the vector equation (55), which in space $\tilde{S}$ has the component form

$$\Delta\tilde{p}_k = h\tilde{b}_k(P) \qquad (57)$$

$$[k=1,2,\ldots N]$$

Comparing these two methods, we note that:

(a) Far from the solution point, the steepest descent technique is superior. It must proceed so as to decrease ε, whereas the differential corrections method is under no such compulsion and is likely to lead us into strange pastures.

(b) Close to the solution point, the differential corrections method is superior. It converges rapidly in the very region where the steepest descent technique languishes.

Marquardt* has proposed an interpolation between the two methods: a technique that behaves like the steepest descent when we are far from the solution and like the differential corrections method when we enter a neighborhood in which the higher-order terms in Eqs. (33) and (34) are negligible.

To achieve this interpolation, a positive nondimensional constant λ is added to each diagonal element of the scaled matrix A. That is, the system (56) is replaced by

$$\boxed{\sum_{n=1}^{N} \tilde{\tilde{a}}_{kn}(P) \cdot \Delta\tilde{p}_n = \tilde{b}_k(P)} \qquad (58)$$

where

$$\left.\begin{aligned} \tilde{\tilde{a}}_{kn} &= 1 + \lambda \text{ when } k=n \\ &= \tilde{a}_{kn} \text{ when } k\neq n \end{aligned}\right\} \qquad (59)$$

System (58) is the bedrock upon which the FINLIE fitting process rests. Note the behavior of this system as a function of λ:

(a) As $\lambda \to 0$, system (58) clearly reverts to the differential corrections system (56).

(b) As $\lambda \to \infty$, the diagonal terms of system (58) dominate and the system degenerates into N uncoupled equations of the form

$$(1 + \lambda)\,\Delta\tilde{p}_k = \tilde{b}_k$$

*See the Bibliography, part A.

or, since by assumption $\lambda >> 1$,

$$\Delta\tilde{p}_k = \lambda^{-1} \tilde{b}_k \tag{60}$$

Comparing (60) with (57), we see that for large λ_1 values, system (58) simulates the steepest descent approach with $h=\lambda^{-1}$. That is, for $\lambda >> 1$, (58) will take us to a new point a rather short distance from the current point P in the direction of the negative gradient.

Marquardt has suggested an algorithm for determining a suitable value of λ for each iteration; that is, for each step P_0 to P_1, P_1 to P_2, etc., on the path to the desired solution point $\hat{P}$. This algorithm (with a few very minor "refinements") has been incorporated into FINLIE. The basic scheme is as follows.

For the first iteration, P_0 to P_1, FINLIE assigns a tentative value to λ:

$$(\text{starting } \lambda)_{P_0 \text{ to } P_1} \equiv \lambda_{1A} = 0.001 \tag{61}$$

Let P_{1A} denote a candidate for point P_1, obtained by solving (58) with $P=P_0$ and $\lambda=\lambda_{1A}$:

$$\vec{P}_{1A} = \vec{P}_0 + \vec{\Delta P}(P_0, \lambda_{1A}) \tag{62}$$

The basic test that any point P should pass is that it be an improvement over the current point:

$$\epsilon(P) < \epsilon(P_0) \tag{63}$$

If P_{1A} satisfies test (63), then FINLIE returns that point to the user as the updated point P_1 and is ready to start the next iteration, P_1 to P_2.

If P_{1A} fails test (63), then FINLIE must take a smaller step in a more propitious direction. This can be accomplished by increasing λ. That is, FINLIE re-solves system (58) with $P=P_0$ as before, but with λ increased to, say,

$$\lambda_{1B} = 10\ \lambda_{1A} \tag{64}$$

(Note that in re-solving the system (58), the elements $\tilde{\alpha}_{kn}$ ($k \neq n$) and $\tilde{\beta}_k$ do not have to be re-evaluated. They depend only on the current point and thus are evaluated only once each iteration.) The new increment vector for λ_{1B} yields the new candidate point:

$$\vec{P}_{1B} = \vec{P}_0 + \vec{\Delta P}\ (P_0, \lambda_{1B}) \tag{65}$$

If P_{1B} satisfies test (63), then FINLIE returns this point to the user; if P_{1B} fails test (63), then FINLIE increases λ again by a factor of ten, and so on. Sooner or later, an acceptable candidate will be found:

$$\vec{P}_1 = \vec{P}_0 + \vec{\Delta P}\ (P_0, 10^n \lambda_{1A}) \tag{66}$$

where n is zero or a positive integer.

The cost-conscious reader may ask: if P_{1A} fails test (63), why not skip over a possibly long line of rejected candidates by increasing λ by some factor much larger than ten? This should get us to an acceptable candidate point at once or at least in fewer trials. True, but the general principle is this: the larger the λ, the smaller the progress we are making. Hence we don't want FINLIE to use a λ "very much" larger than needed to satisfy test (63). It is not worth the effort to find the optimum λ for each iteration, but by testing after each ten-fold increase in λ, FINLIE will not grossly exceed that optimum. (Indeed, a case could be made out for merely doubling λ each time an increase is required.)

The only way in which the second and subsequent iterations differ from the first is in the formula FINLIE uses for determining the starting λ value for the iteration:

$$(\text{starting } \lambda)_{P_{n-1} \text{ to } P_n} = 0.1 \times (\text{final } \lambda \text{ value used to produce point } P_{n-1} \text{ in the previous iteration}) \tag{67}$$

That is, FINLIE always decreases the current value of λ by a factor of ten at the start of each new iteration. This decrease is an essential part of the λ manipulation. When all is going well, FINLIE will have no need to increase λ; thus rule (67) will insure that λ goes to zero - and hence that the process approaches the differential corrections technique - as FINLIE approaches the solution point $\hat{P}$.

A typical set of λ values encountered in the course of some hypothetical fit (not our familiar examples, (3) and (4)) is shown in Table V. The reader can infer from these λ values the fleeting existence of rejected candidate points. Thus, to get from P_2 to P_3, FINLIE clearly had to solve system (58) six times: for $\lambda=10^{-4}$ (that is, one-tenth the previous λ), 10^{-3}, 10^{-2}, 10^{-1}, 10^{0} and 10^{1} (the λ value that produced a successful candidate). Similarly, to get from P_5 to P_6,

TABLE V. Typical λ Values During a Fit

Iteration	λ value returned by FINLIE at the end of the iteration	No. of times system (58) must have been solved by FINLIE
P_0 to P_1	10^{-2}	2
P_1 to P_2	10^{-3}	1
P_2 to P_3	10^{1}	6 (for $\lambda=10^{-4}, 10^{-3},\ldots 10^{1}$)
P_3 to P_4	10^{0}	1
P_4 to P_5	10^{-1}	1
P_5 to P_6	10^{-1}	2 (for $\lambda=10^{-2}, 10^{-1}$)
P_6 to P_7	10^{-2}	1
P_7 to P_8	10^{-3}	1
P_8 to P_9	10^{-4}	1
P_9 to P_{10}	10^{-5}	1
P_{10} to P_{11}	10^{-6}	1

FINLIE must have solved (58) twice: for $\lambda=10^{-2}$ and 10^{-1}. Thereafter, the fitting process seemed to get back on the track and λ decreased steadily. Without Marquardt's λ in the system, it is likely that the search represented by Table V would have gone astray after point P_2 and come to some abrupt and ignoble conclusion.

K. Convergence Criterion

The question arises: when can the user accept a point returned by FINLIE as being "close enough" to the desired solution? One possible answer is: when FINLIE tells him he can. At the end of each iteration, FINLIE returns to the user a flag whose value indicates whether or not the returned point has satisfied a built-in convergence criterion. (This flag will be discussed in section III(C).)

The convergence criterion installed in FINLIE is as follows. Let P_{n-1} and P_n be any two consecutive points returned by FINLIE: the end points of two consecutive iterations. Then FINLIE will signal convergence at point P_n if and only if

$$0.99999 \leq \varepsilon(P_n)/\varepsilon(P_{n-1}) \leq 1 \qquad (68)$$

The right-hand portion of this double inequality is essentially inequality (63) and hence is always satisfied, thanks to the Marquardt λ feature. The left-hand inequality in (68), however, constitutes an arbitrary definition of convergence: namely, that the percent change in ε has dropped below 0.001.

As an example of criterion (68) in action, consider the search summarized by Table II. The values of $CR \equiv \varepsilon(P_n)/\varepsilon(P_{n-1})$ listed in the next to last column of that table jump about erratically (always between 0 and 1, of course) before the criterion is satisfied at point P_7. The sudden transition from the value of CR at P_6 to its value at P_7 is not typical. In searches based on more realistically inaccurate measured data, CR will often be close to - and monotonically approach - the value 1 over the final few iterations.

Note that (68) is only a measure of convergence to a <u>local</u> minimum. We have said it before, but it bears repeating: there is no guarantee that the point P_n satisfying (68) will yield the desired absolute minimum ε.

Of course, the user need not accept definition (68); he can ignore the FINLIE convergence flag and impose his own convergence test on the data returned by FINLIE after each iteration.

L. Estimated Errors

In addition to computing the estimated standard deviation of the fit:

$$s \equiv \left[\frac{\epsilon(\hat{P})}{N4-N} \right]^{\frac{1}{2}} \tag{69}$$

FINLIE computes s_k, the estimated standard deviation of paramic p_k, k=1,2,...N.

For linear least-squares, the conventional formula is

$$s_k = [\Delta_{kk}]^{\frac{1}{2}} s \tag{70}$$

where

Δ_{kk} = the k-th diagonal element of the inverse of the unscaled matrix A.

Note that while s is nondimensional, s_k has the same dimensions as p_k.

For nonlinear least-square fits, Eq. (70) should be viewed with a healthy suspicion. Indeed, Celmins (Ref. 33 in the Bibliography) points out that even in the linear case, the equation should be applied only in "very limited special cases." Unfortunately, the alternative formula that he develops for s_k is a rather complicated one involving second-order derivative terms - terms that so far we have managed to avoid. The inclusion of these terms would mean more work not only for FINLIE - which would be acceptable - but for the user, who would have to derive and program some possibly horrendous expressions. The labor here seems out of proportion to its reward, since the "crude" error estimates provided by (70) are usually not all that crude when the search has converged to the proper point. Hence FINLIE returns these estimates to the user and the user is expected to provide his own grain of salt.

(Note that (70) uses only the diagonal elements of the inverse matrix. In some situations, all of the elements of A^{-1} are useful for error analysis. In these special situations, $A^{-1}s^2$ can be regarded as the variance-covariance matrix. However, for nonlinear least squares, we are pushing our luck in making use of the diagonal elements; to try to assign any significance to the off-diagonal elements would really be folly.)

Recall that FINLIE transforms the elements of matrix A to the scaled space $\tilde{S}$, Eq. (50), and then replaces the diagonal elements by $1+\lambda$. Hence FINLIE actually obtains the paramic error estimates by the relation

$$s_k = \left[\frac{(1+\lambda)\tilde{\Delta}_{kk}}{a_{kk}} \right]^{\frac{1}{2}} s \qquad (71)$$

where

$\tilde{\Delta}_{kk}$ = the k-th diagonal element of the inverse of matrix $\{\tilde{a}_{kn}\}$, Eq. (58)

(I felt there should be some compensation in the error estimate formula for the presence of Marquardt's λ in the equations. By a chain of nonrigorous reasoning, I was thus led to insert the $(1+\lambda)$ factor in (71). Since $\lambda << 1$ for a good fit, $(1+\lambda)$ seems relatively harmless sitting there.)

M. The Composition of FINLIE

So far, the word FINLIE has denoted an apparently monolithic program. Actually, for reasons that seemed persuasive at the time, FINLIE was written as an assemblage of six linked FORTRAN subroutines:

DUBLIN, LONDON, PARIS, BONN, MATINV, MERSO

only one of which - DUBLIN - is called by the user. "FINLIE", then, is merely a convenient name for an ensemble of six subroutines.

[FINLIE is also the name of a permanent file (in Update format) stored on the front end of BRL's Control Data Corporation computer system. (At BRL, this system consists of two linked mainframes: the CYBER 170/Model 173 and the CYBER 70/Model 76.) File FINLIE contains five of the six subroutines: all but MATINV, which is already available from a system library.]

The relationship between

(i) the user's program that calls FINLIE,

(ii) FINLIE

and (iii) the user's subroutine defining his equations,

and the inter-relationship of the six subroutines that constitute FINLIE are all indicated in Figure 1. A vertical bar between two subroutines in the figure indicates that the upper subroutine calls the lower one.

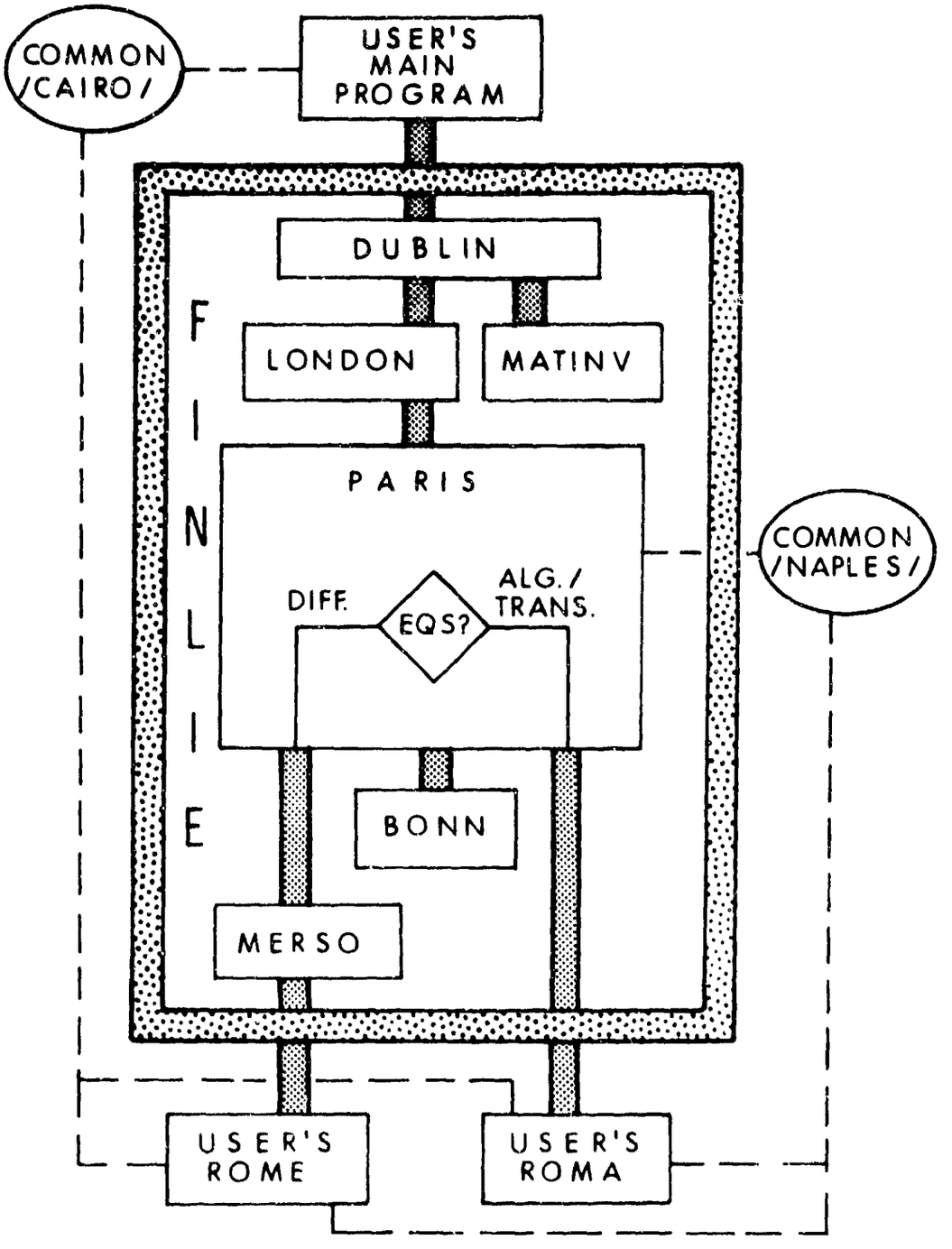

Fig. 1. FINLIE and the User: A Schematic

All six subroutines of FINLIE are listed in the Appendix. Only four of the six - the four "cities" - were written by the author; the other two (namely, MATINV and MERSO) are general-purpose subroutines to be discussed shortly.* With some minor exceptions in subroutine MERSO (these will be spelled out), the FORTRAN used in FINLIE is a "more or less standard" version of FORTRAN IV (alias FORTRAN 4, alias FORTRAN 66).

Converting FINLIE to a later model FORTRAN - say, FORTRAN 77 - should be relatively uneventful. One possible difficulty is as follows. FINLIE was written for a compiler that automatically retains the values of entities defined within a subroutine but not linked to the calling program. For such a compiler, subsequent calls to the subroutine will find the previous values waiting. However, in FORTRAN 77 the SAVE statement is available for specifying what if anything is to be retained; hence some FORTRAN 77 compilers may not automatically retain local values. In that case, it may be necessary to SAVE the arrays ALPHA and BETA in subroutine DUBLIN.

DUBLIN is the interface between the user and FINLIE. The user must write the FORTRAN program that calls subroutine DUBLIN with the required input data. Hereafter, we will refer to (and think of) the user's calling program as a main program, although it could itself be a subprogram. Each time that DUBLIN is called by this main program, DUBLIN activates the other subroutines of FINLIE, causing one iteration of the search procedure to be carried out. That is, if the main program submits point P_{n-1} to DUBLIN, DUBLIN will return to the main program the next point P_n. Information and advice on writing the main program and in particular on calling DUBLIN will be given in overwhelming detail in Part III.

Subroutines LONDON, PARIS and BONN are buried within FINLIE, so that their individual purposes should be of little significance to the user. However, the following features of PARIS can be noted from Fig. 1. If the user is fitting a set of differential equations, PARIS calls a numerical integration subroutine MERSO (of which more will be said shortly) and MERSO in turn calls the subroutine - written by the user and arbitrarily labelled ROME in the figure - that defines the differential equations to be fitted. On the other hand, if the user is fitting algebraic or transcendental equations, PARIS calls the user's equation-defining subroutine ROMA directly. Both ROME and ROMA must get additional information from PARIS through the labelled COMMON

A reviewer of this paper questioned the implication that DUBLIN, LONDON, PARIS, and BONN are the only "cities" in the sextuplet of subroutines. He went so far as to consult an atlas to see if there is a town, a village, a hamlet or a crossroads by the name of MATINV or MERSO somewhere in the world. Apparently there isn't.

block NAPLES. ROME and ROMA may require additional information from the user's main program, this can be passed through the labelled COMMON block CAIRO. Abundant details on writing ROME and ROMA and on the COMMON blocks will be given in Part III.

Subroutine MATINV is a general-purpose matrix inversion subroutine borrowed intact from the computer library here at BRL. Upon return from MATINV, the input matrix has been replaced by its inverse.

Subroutine MERSO is a general-purpose numerical integration subroutine based on a method proposed by R.H. Merson of Australia. The method is a fourth-order member of the Runge-Kutta family, requiring five function evaluations at each integration step. The subroutine adjusts the integration step size automatically to obtain a predefined accuracy. (All of this is transparent to the FINLIE user.)

The computer library at BRL contains a subroutine MERSON (see References 11 and 32 in the Bibliography) for performing Merson integration. Subroutine MERSO is identical to MERSON with the exception of two statements. Firstly, where MERSON has

```
DIMENSION T(100), G(100), S(100),
```

MERSO has increased the three dimensions to 400 each. Secondly, where MERSON has

```
IF (NT.LE.100) GOTO 100,
```

MERSO compares NT with 400. The reason for these changes is as follows. The size of the three arrays T, G and S above must equal or exceed

$$N5 \equiv N2 + NA \tag{72}$$

that is, the number of differential equations (N2) plus the number of influence equations (NA). MERSON requires $N5 \leq 100$. In my largest application of FINLIE so far, N5 exceeded 100 (was, in fact, 368). Hence the minor surgery that altered MERSON into MERSO; the maximum permitted value of N5 is now 400. This value of 400 appears not only in MERSO but in PARIS, where it is the declared dimensions of arrays U and DU (see the Appendix). Hence the user can change the upper limit on N5 by

(i) changing the dimensions of U and DU in PARIS;

and (ii) changing the 400 in the PARIS statement that currently reads:

```
IF(M5.LE.400) GOTO 24
```

and (iii) changing the two previously mentioned statements in

MERSO:

DIMENSION T(400), G(400), S(400)

and IF(NT.LE.400) GOTO 100

Note that a "nonstandard" FORTRAN function appears on the line above statement 410 in MERSO:

H = SIGN1(H) * HMI

Here SIGN1 is the signum function; if it is not recognized by the user's FORTRAN compiler, the above statement can be replaced by

IF(H.NE.0.) H=SIGN(1.,H)*HMI

The use of multiple arithmetic and logical assignment statements in MERSO may also be unacceptable to some compilers. In a multiple statement of the form

VN = ··· = V2 = V1 = expression,

the assignments are carried out from right to left:

V1 = expression
V2 = V1, etc.

It should be pointed out after all this exposition on MERSO that when the user is fitting algebraic or transcendental equations rather than differential equations, MERSO is not needed and may be removed from FINLIE.

FINLIE was written for BRL's CDC computer system for which the single precision of real numbers is approximately 14 decimal digits. So far, this has proven adequate for all our FINLIE applications. If the user is working with a machine whose single precision is significantly less than 14 decimal digits, he may have to add some double-precision declarations to the subroutines of FINLIE. One source of trouble is the possibly erratic behavior of ε near a minimum, due mainly to round-off noise. Hence a likely candidate for double precision is array GAMMA in subroutine DUBLIN (and its dummy version A in subroutine MATINV). A more complete list of variables that may require double precision includes:

in DUBLIN: EA, EB, EPS, GAMMA
in LONDON: EP, EPS, RSQ
in PARIS: RM, RSQ
in MATINV: A, T1

In extreme cases, the user can simply double-precision everything in

sight; this may be inefficient in terms of storage, but it could save wear and tear on the user.

III. OUTSIDE FINLIE: WHAT THE USER MUST DO FOR FINLIE

Assuming that the FINLIE user is given a set of equations of the form (1) or (2) - or equations that can be put into one of those forms - the first task the user must perform is to derive the associated influence equations, as indicated in parts C and D of Section II. The next task is to write a FORTRAN subroutine defining all these equations - the original set and the influence equations - in a manner acceptable to FINLIE. The rules for constructing this subroutine are slightly different for sets (1) and (2). (We assume in what follows that the reader has some familiarity with - though he need not be an expert in - some version of FORTRAN equivalent to or newer than FORTRAN IV.)

A. ROME: The User's Subroutine for Fitting Differential Equations

The first three statements of ROME have the form:

```
SUBROUTINE ROME(N5,XE,U,DU)
DIMENSION U(N5), DU(N5)
COMMON/NAPLES/PAR(40), FLAG(60)
```

It should be noted that the only name in the above three statements that the user is not allowed to change is NAPLES. All other names, including ROME, may be replaced by other legal FORTRAN names of the user's choice. (Of course, the distinction between integer and real names should be maintained.)

ROME is called by MERSO (see Figure 1) and hence the nature of the four arguments of the SUBROUTINE ROME statement has been decreed by MERSO. The first three arguments are input to ROME (from MERSO):

N5 = the number of equations to be defined in ROME: N2 (first-order differential equations) plus NA (influence equations). Thus for sample set (3), the value of N5 is 2 + 8 = 10. Note, however, that this argument is an integer name, not an integer constant. As certain arrays are currently dimensioned, N5 cannot exceed 400 (see the pertinent remarks in section II(M)).

XE = x, the independent variable value at which the N5 equations are to be evaluated. The argument, of course, must be a real name, not a real constant. If the independent variable does not appear in any of the equations, then argument XE will not be used in the body of subroutine ROME.

U = the vector of N2 dependent variables and NA influence coefficients, where

$$\left.\begin{aligned} y_j &= U(J), \\ D_{jk} &= U(J + K*N2) \end{aligned}\right\} \quad (73)$$

$$[J=j=1,2,..N2 \quad K=k=1,2,..N23]$$

Thus for sample set (3), where N2=2 and N23=4, we have

$$\begin{aligned} U(1) &= y_1 \\ U(2) &= y_2 \\ U(3) &= D_{11} = \partial y_1/\partial y_{10} \\ U(4) &= D_{21} = \partial y_2/\partial y_{10} \\ U(5) &= D_{12} = \partial y_1/\partial y_{20} \\ U(6) &= D_{22} = \partial y_2/\partial y_{20} \\ U(7) &= D_{13} = \partial y_1/\partial c_1 \\ U(8) &= D_{23} = \partial y_2/\partial c_1 \\ U(9) &= D_{14} = \partial y_1/\partial c_2 \\ U(10) &= D_{24} = \partial y_2/\partial c_2 \end{aligned}$$

The final argument of ROME is an output (to MERSO):

DU = the derivative vector at the current value XE of the independent variable, where

$$DU(J) = dU(J)/dx \quad (74)$$

$$[J=1,2,..N5]$$

Additional input to ROME comes from PARIS via the labelled COMMON block NAPLES. The one hundred elements of the NAPLES block are as follows:

PAR = a vector of the current values of the N3 parameters (not paramics), where N3 $\leq$ 40. For sample set (3),

PAR(1) = c_1

PAR(2) = c_2

and the remaining 38 elements of PAR are undefined.

FLAG = a vector of N23 flags (N23 ≤ 60) associated with the N23 single-round paramic set Q, Eq. (12). That is, the first N2 elements of FLAG are associated with the N2 initial conditions and the remaining N3 elements of FLAG are associated with the N3 parameters. The value of FLAG(J) is

(i) zero if the value of the corresponding paramic q_j is fixed;

or (ii) 1.0 if the value of q_j is to be adjusted by the fitting process.

For sample set (3), we have

FLAG(1) = flag for y_{10}

FLAG(2) = flag for y_{20}

FLAC(3) = flag for c_1

FLAG(4) = flag for c_2

and the remaining 56 flags are undefined.

Note that PAR and FLAG are inputs to ROME from FINLIE; when writing ROME, the user assumes that the two arrays already contain their proper values. In the case of the initial condition flags, these values may change from round to round. For example, in our tri-round situation, we might decide to make a computer run with y_{10} for round E1 and y_{20} for round E3 fixed at specified values. Then

FLAG(1) = 0.0, 1.0, 1.0

FLAG(2) = 1.0, 1.0, 0.0

for rounds E1, E2 and E3, respectively. FINLIE will automatically change the values of FLAG(1) and FLAG(2) to match the round whose measured data is currently being fitted. Of course, FINLIE can't guess what the user wants to do; it must be told. FINLIE can only define PAR and FLAG on the basis of certain inputs given to it by the user's main program. These inputs will be discussed in section III(C).

The dimensions of PAR and FLAG are arbitrary to this extent: they can be changed in ROME if the user is willing to make all the associated changes in FINLIE. To save space, I leave the nature of such changes as an exercise for the interested reader. The simplest

course is to make no changes if N2 $\leq$ 40 and N23 $\leq$ 60.

After writing down the first three statements of ROME, the user is ready to encode the body of the subroutine: the statements defining the N5 elements of output array DU. Consider, for example, system (3). For convenience, we repeat here the original equations:

$$y_1' = 1/y_2$$

$$y_2' = -c_1(1+c_2y_2)y_2$$

and the associated influence equations (Table III-A):

$$(D_{11})' = -D_{21}/y_2^2$$

$$(D_{21})' = -c_1(1+2c_2y_2)D_{21}$$

$$(D_{12})' = -D_{22}/y_2^2$$

$$(D_{22})' = -c_1(1+2c_2y_2)D_{22}$$

$$(D_{13})' = -D_{23}/y_2^2$$

$$(D_{23})' = -c_1(1+2c_2y_2)D_{23} - (1+c_2y_2)y_2$$

$$(D_{14})' = -D_{24}/y_2^2$$

$$(D_{24})' = -c_1(1+2c_2y_2)D_{24} - c_1(y_2^2)$$

For these equations, a likely version of subroutine ROME is given in Table VI.

Table VI. Subroutine ROME for System (3)

```
      SUBROUTINE ROME (N5,XE,U,DU)
      DIMENSION U(N5),DU(N5)
      COMMON /NAPLES/ C1,C2,BLANK(38),FLAG(60)
            V = U(2)
         DU(1) = 1./V
            A1 = C2*V
         DU(2) = -C1*(1. + A1)*V
            A2 = DU(1)**2
            A3 = C1*(1. + A1 + A1)
      IF (FLAG(1) .EQ. 0.) GOTO 10
         DU(3) = -A2*U(4)
         DU(4) = -A3*U(4)
   10 IF (FLAG(2) .EQ. 0.) GOTO 20
         DU(5) = -A2*U(6)
         DU(6) = -A3*U(6)
   20 IF (FLAG(3) .EQ. 0.) GOTO 30
         DU(7) = -A2*U(8)
         DU(8) = -A3*U(8) - (1. + A1)*V
   30 IF (FLAG(4) .EQ. 0.) GOTO 40
         DU(9) = -A2*U(10)
         DU(10)= -A3*U(10) - C1*V*V
   40 RETURN
      END
```

Note that in COMMON/NAPLES/ I opted to write the forty-element parameter set in the form

C1, C2, BLANK(38)

since only the first two of the forty elements have any meaning. I could just as well have written PAR(40) in the COMMON statement and used PAR(1) and PAR(2) instead of C1 and C2 in the body of the subroutine. Note also that I dimensioned FLAG as 60 even though the last 56 elements are meaningless. This was a courtesy to our CDC

FORTRAN compiler, which likes all COMMON blocks of the same name (NAPLES in this case) to have the same length. The compiler doesn't insist when you break this rule, but it comments on your bad form.

Because ROME will be called many times by MERSO during the course of the numerical integration, the user should take the time to make ROME as efficient as practicable. For large and labyrinthian systems of equations, a worthy ROME isn't built in a day.

One of the aids to efficiency in ROME is the FLAG vector. Note that if any paramic value is fixed during a computer run (that is, if the associated flag value is zero), the influence equations for that paramic need not be calculated. Hence the FLAG vector can - and in my opinion should - be used as indicated in Table VI to avoid these unnecessary calculations. The general rule is that if FLAG(J) is zero, then ROME need not evaluate DU(LA) through DU(LB), where

$$LA = (J \times N2) + 1$$
$$LB = (J \times N2) + N2 = LA + (N2-1).$$

Of course, if the user is convinced that he will never, ever want to hold fixed the value of some paramic, he can omit the corresponding IF-statement from ROME.

Some systems of equations may involve constants whose values are always fixed (that is, never adjusted by FINLIE) and yet these values may change from run to run. It would be possible - but not too bright - to handle such a constant as a fixed parameter: a parameter whose associated flag is always zero. A better approach is to pass any such constant directly from the user's main program to ROME through a new labelled COMMON block (see block CAIRO in Figure 1). Of course, if a constant will never change from run to run, it need only be defined within ROME.

A final, rather minor comment: Sample set (3) is one of those cases where the input argument XE is not used in the body of subroutine ROME, simply because the independent variable does not appear explicitly in the N5 equations of this example.

B. ROMA: the User's Subroutine for Fitting Algebraic or Transcendental Equations

Many of the comments in the previous section concerning ROME apply to ROMA as well. Hence, if the reader has skipped over that section because his interest in fitting differential equations is minimal, he may have missed something noteworthy. Or possibly not.

The first three statements of ROMA have the form:

```
SUBROUTINE ROMA (COND,XO,XE,U)
DIMENSION COND(n2), U(n5)
COMMON/NAPLES/PAR(40), FLAG(60)
```

The first three arguments in the SUBROUTINE statement are inputs (from PARIS):

COND = a vector of N2 current initial condition values. For sample set (4),

$$\text{COND}(1) = y_{10}$$

$$\text{COND}(2) = y_{20}$$

For multi-round situations, the initial conditions change with the round as well as with the current state of the fitting process. FINLIE supplies the proper COND vector to ROMA automatically.

XO = x_0, the independent variable value at which the initial conditions apply. (This one value must apply to all rounds.)

XE = x, the independent variable value at which the equations are to be evaluated.

The final argument, U, is an output vector defined exactly as in the previous section for subroutine ROME.

In the DIMENSION statement, n2 and n5 denote the values of N2 and N5, respectively. (Actually, on the CDC system and on most other computers, a one-dimensional argument array in a subroutine need not be declared at its maximum size; the value 1 is adequate.)

The labelled COMMON block NAPLES brings to ROMA the arrays PAR and FLAG, defined in the previous section.

The body of subroutine ROMA consists of the statements defining the needed elements of array U. Consider, for example, system (4). For convenience we repeat here the original equations (4) and the needed influence equations (Table III(B)):

$$y_1 = y_{10} - c_2(x-x_0) + (b/c_1)(u-1)$$

$$y_2 = (bu - c_2)^{-1}$$

$$D_{11} = 1.$$

$$D_{12} = -(u-1)/(c_1 y_{20}^2)$$

$$D_{13} = (b/c_1^2)[1-u+c_1(x-x_0)u]$$

$$D_{14} = (u-1)/c_1 - (x-x_0)$$

where

$$u = \exp[c_1(x-x_0)]$$

$$b = (y_{20})^{-1} + c_2$$

For these equations, a likely version of subroutine ROMA is given in Table VII.

Table VII. Subroutine ROMA for System (4)

```
SUBROUTINE ROMA (COND,XO,XE,U)
DIMENSION COND(2),U(10)
COMMON /NAPLES/ C1,C2,BLANK(38),FLAG(60)
T0 = COND(1)
V0 = COND(2)
XA = XE - X0
 Z = EXP(C1*XA)
 B = C2 + 1./V0
                    U(1) = T0 - C2*XA + B*(Z - 1.)/C1
                    U(2) = 1./(B*Z - C2)
IF (FLAG(1) .NE. 0.) U(3) = 1.
IF (FLAG(2) .NE. 0.) U(5) = (1. - Z)/(C1*V0**2)
IF (FLAG(3) .NE. 0.) U(7) = B*(1. - Z + C1*XA*Z)/(C1**2)
IF (FLAG(4) .NE. 0.) U(9) = (Z - 1.)/C1 - XA
RETURN
END
```

As discussed in section II(D), FINLIE does not require expressions for the influence coefficients D_{jk} when j is greater than N1. Hence in this sample ROMA, where N1=1, the D_{2k} equations (namely, the equations for U(4), U(6), U(8) and U(10)) are simply omitted from the subroutine.

As with ROME in the previous section, the FLAG array in ROMA is used to avoid calculating D_{jk} when the value of paramic q_k is fixed. Also as with ROME, any needed "changeable constants" can be passed directly from the user's main program to ROMA through, say, the labelled COMMON block CAIRO.

C. Calling Subroutine DUBLIN

After the user has written his subroutine defining the equations to be fitted, his next task is to write a program unit - we assume a main program - that utilizes FINLIE. Before discussing this main program as a whole, we will concentrate on one statement within that main program: the CALL DUBLIN statement.

This statement is the link between the user and FINLIE. It can be written in the form

```
CALL DUBLIN (ROME,NF,N1,N2,N3,N7,N8,NR,NM,XO,
             X,Y,F,NW,W,P,RL,NC,YC,R,RS,EPS,
             SIG,EK,NS)
```

where all integer names happen to start with the letter N. The first fourteen of the twenty-five arguments are inputs.

[1] ROME is the name of the subroutine (written by the user) that defines the equations to be fitted (See sections III(A-B)).

The values of the remaining thirteen input arguments must be established in the user's main program before DUPLIN is called. These values will not be changed by FINLIE; hence actual values rather than names may be used for arguments [2] through [8], [10] and [14] below.

[2] NF is a flag that indicates the nature of the equations to be fitted:

NF=0 if the fitting equations are algebraic or transcendental (System (2));

NF=1 if the fitting equations are differential equations (System (1)).

[3] N1 is the number of measured dependent variables in the system, where

$$1 \leq N1 \leq 10 \qquad (75)$$

(The upper bound on N1 - and the upper bounds indicated for some of the other arguments defined below - can be increased only by delving into FINLIE.) N1 must have the same value for each round; FINLIE insists that the same dependent variables be measured for each round used in the fitting process.

[4] N2 is the total number of dependent variables in the system, where

$$N1 \leq N2 \leq 20 \qquad (76)$$

[5] N3 is the maximum number of parameters (not paramics) whose values can be determined from the fit, where

$$0 \leq N3 \leq 40 \qquad (77)$$

I use the word "maximum" above because the actual number of parameters to be determined in the course of a computer run may be less than N3. The user specifies (by argument F, to be discussed below) which, if any, of the parameters and initial conditions are to be held fixed at their input values and which are to be adjusted by FINLIE during the run. Input N3 is the total number of parameters: those to be adjusted plus those held fixed. (If input N3 is zero - the lower limit in inequality (77) - then presumably there is at least one initial condition to be determined; otherwise there would be no reason for running the program.)

[6] N7 is the number of rows declared in the user's main program for the two-dimensional arrays Y, W and R defined below as arguments [12], [15] and [20], respectively. As we will see, these three arrays serve as N1 by N4 matrices. At first glance, then, it might seem that N7=N1. However, the user may not want to restrict his main program DIMENSION statement to the current values of N1 and N4. It is often more convenient to dimension arrays at their largest anticipated sizes. For example, in our recurring case where N1=1 and N4=16, the user might want to dimension arrays Y, W and R as, say (2,50) rather than (1,16). FINLIE will go along with this sort of thing, but it wants to be told about it. Thus if the user dimensions Y, W and R as (2,50), he must set N7 equal to 2. In general, then, $N7 \geq N1$. (The declared column dimension for the three arrays - say, 50 - is not needed by FINLIE. The declared row dimension is sufficient - assuming the computer stores matrices in the usual way, that is, by columns - to maintain notational row-column agreement between calling program and subroutine. Neither is FINLIE interested in the declared dimensions of its vector arguments.*)

In FINLIE, I have declared 1 as the last (right-most) dimension of subroutine dummy argument arrays. This is fairly common FORTRAN 4 practice, but FORTRAN 77 prefers an asterisk: Y(N7,) instead of Y(N7,1).

[7] N8 is the number of rows declared in the user's main program for the two-dimensional array YC defined below as argument [19]. Array YC serves as an N2 by N4 matrix; hence N8 ≥ N2. (See the comments for argument [6] above.)

[8] NR is the number of data rounds to be considered simultaneously, where

$$1 \leq NR \leq (60-N3)/N2 \tag{78}$$

The right-half of this double inequality may seem a rather strange condition to spring upon the reader. Until now, no limit has been implied on the number of rounds. The basic condition (somewhat concealed in (78)) is

$$N \leq 60 \tag{79}$$

where N is the total number of paramics, (NR x N2) + N3. Condition (79), like the limits on N1, N2 and N3, is a result of arbitrary DIMENSION decisions that had to be made when constructing FINLIE. Since N is not itself an input to DUBLIN, I have simply converted (79) to the equivalent form (78). By satisfying (78), the user can be sure that (79) is also satisfied. For sample set (3) or (4) we must have NR ≤ (60-2)/2=29. For the associated data of Table I, we have NR=3, well below the maximum permitted. Recall that the data for an individual round solely determine the initial conditions for that round, but combine with the data from all the other rounds to determine the parameters.

[9] NM is a vector of NR elements, where

NM(J) = the number of data points R_m (that is, the number of independent variable values x_m at which measurements were taken) for the J-th round.

Thus for the sample data of Table I, the user's main program must set

```
NM(1) = 5
NM(2) = 6
NM(3) = 5
```

FINLIE determines N4, the total number of data points, by summing the NM components:

$$N4 = \sum_{J=1}^{NR} NM(J) \qquad (80)$$

The user must insure that N4 satisfies the inequalities

$$\left.\begin{array}{l} N < N4 \leq 1000 \\ N4 \times N5 \leq 10000 \end{array}\right\} \qquad (81)$$

and that

$$\left.\begin{array}{l} N1 \times (\text{MAX.ELEMENT OF NM}) \leq 200 \\ N2 \times (\text{MAX.ELEMENT OF NM}) \leq 400 \end{array}\right\} \qquad (82)$$

Again, these restrictions are the result of arbitrary DIMENSION statements in FINLIE.

[10] XO is the independent variable point x_0 at which all initial conditions apply. The same x_0 must apply to all rounds.

[11] X is a vector of the N4 independent variable values x_m at which measurements were taken. The first NM(1) values in X are the first-round values, in increasing order:

$$X(M-1) < X(M), \quad M = 2,3,\ldots NM(1)$$

The next NM(2) values of X are the second-round values, also in increasing order among themselves:

$$X(M-1) < X(M), \; M - NM(1) = 2,3,\ldots NM(2)$$

and so on. For the Table I data, we have

$$X(M) = x_m, \quad m = 1,2,\ldots 16.$$

[12] Y is an N1 by N4 matrix of measured dependent variable values, where

$$Y(I,M) = \text{the measured value of } y_i \text{ at } X(M)$$

For the Table I data, we have $Y(1,M) = \bar{y}_{1m}$, $m=1,2,\ldots 16$.

[13] F is a vector of N flags associated with the paramic point P (argument [16] below), where

F(J) = 0.0 if the input value of P(J) is to be held fixed;

= 1.0 if the input value of P(J) is to be adjusted by the fitting process.

[14] NW is a weight flag associated with matrix W (argument [15] below).

NW = 0 if the user's weights (already stored in matrix W) are to be used by FINLIE;

= 1 if all weights are unity (in which case, the user need not store 1.0's in matrix W before calling DUBLIN).

Recall the comments in the vicinity of Eqs. (8) and (9) regarding weights. The important point is that the "easy" way out - assigning unit weights, merely by setting NW=1 - will often lead to a poor fit. Give some minimum consideration to the possibility of unequal uncertainties in the measurements, particularly when more than one variable has been measured (N1 > 1).

The fifteenth argument of DUBLIN may or may not be defined by the user before DUBLIN is called:

[15] W is the N1 by N4 matrix of weights associated with input Y (argument [12] above). The user has a choice to make. If each of the N1 by N4 measurements in matrix Y can be assigned unit weight; that is, if

$$W(I,J) = 1.0$$

then the user need not define the W array. Simply set NW (argument [14] above) to 1. If, on the other hand, the user decides that one or more of the weights must differ from 1, then the user must define the entire array, subject to the conditions that each weight be nonnegative and that

$$[W(I,J)]_d = [1./Y(I,J)^2]_d$$

See the comments near Eqs. (8) and (9).

The next three arguments of DUBLIN are input/output. That is, the user must define them before the first CALL DUBLIN statement, but FINLIE will change their values.

[16] P is the current N-dimensional paramic point P, Eq. (5), where

P(1), ...	P(N2)	= first round initial conditions,
P(1+N2),.	P(2*N2)	= second round initial conditions,
.		
.		
.		
P(1+(NR-1)*N2) . . .	P(NR*N2)	= last round initial conditions,
P(1+NR*N2)	P(N)	= the parameters.

Thus for sample system (3) or (4) and for tri-round data, the eight elements of P are given by Eq. (10). Clearly, P is the essential argument in the CALL statement; the other arguments play a necessary but supportive role. As indicated in Part I of this report, an effort should be made to find suitable starting values for the elements of P. Not all first estimates will lead to the right answer. Each time the program returns from DUBLIN, array P will contain an updated point. More precisely, the first call to DUBLIN is a special situation and P is unchanged upon return. Thereafter, each call serves to update P. For more on this first call, see argument [18] below. In general, then, each DUBLIN call after the first advances P one step on the road to the solution. DUBLIN should be called repeatedly (say, in a DO-loop) until convergence is achieved. Not all elements of P will necessarily change with the iteration. If input F(J) is zero (see argument [13] above), then the original, user-assigned value of P(J) will be maintained no matter how many times DUBLIN is called.

[17] RL is a Marquardt argument. Before DUBLIN is called the first time,

(i) Set RL = 0.0 if the Marquardt algorithm is to be omitted from the fitting process (that is, if the user wants FINLIE to fit by differential corrections, Eq. (56), rather than by Marquardt interpolation, Eq. (58)). In this case, RL will remain at zero.

(ii) Set RL = 1.0 if the Marquardt algorithm is to be used. Upon the first return from

DUBLIN, RL will have the "starting" λ value of 0.01. (On subsequent calls to DUBLIN, the input value of RL is immediately divided by ten; hence the true starting value of λ is 0.001, as indicated in Eq. (61)). Upon the second and subsequent returns from DUBLIN, RL will have the value of λ used to obtain the point returned in array P. Note that since FINLIE changes RL, a name (not the value 1.0) must be used in the CALL list.

[18] NC is a "first call" flag. The user must set NC=0 initially. This value alerts FINLIE to the fact that it is being called for the first time. FINLIE behaves differently on this first call than it does on all subsequent calls. In particular, on the first call, FINLIE

(i) sets all elements of argument W to unity if NW=1;
(ii) sets argument RL to 0.01 if the input RL is 1.0 (that is, if Marquardt's algorithm is to be used);
(iii) determines the number of paramics to be adjusted (the total number minus the number of paramics held fixed) and stores this value back in argument NC (hence use a name, not the integer zero, for the "first call" flag in the CALL list);
(iv) evaluates the next five arguments in the CALL list (YC,R,RS,EPS and SIG, all described below) at the input point P_0.

Note that FINLIE does not update the input point P_0 on this first call: P_0 goes in and P_0 comes back. The paramics are updated only on the second and subsequent calls.

The remaining seven arguments of DUBLIN are outputs, evaluated at the current value of P.

[19] YC is an N2 by N4 matrix of computed dependent variable values, based on the current point P, where

YC(J,M) = the computed value of y_j at X(M)

Thus for the Table I data in our examples,

$$YC(1,M) = y_1 (x_m, P)$$

$$YC(2,M) = y_2 (x_m, P)$$

$$[m = 1,2,\ldots 16]$$

When fitting differential equations, FINLIE obtains the YC values by numerical integration of system (1); when fitting algebraic or transcendental equations, FINLIE obtains YC directly from the equation set (2).

[20] R is an N1 by N4 matrix of residuals of the fit, where

$$R(I,M) = Y(I,M) - YC(I,M) \tag{83}$$

[21] RS is a vector of N1 nondimensional error measures associated with the N1 measured dependent variables, where

RS(I) = that part of ε (see Eq. (7) and argument [22] below) that can be attributed to the fit on y_i

$$= \sum_{M=1}^{N4} W(I,M) \, [R(I,M)]^2 \tag{84}$$

[22] EPS is $\varepsilon(P)$, the nondimensional sum of the weighted squares of the residuals of the fit (Eq. (7)), where

$$EPS = \sum_{I=1}^{N1} RS(I) \tag{85}$$

If the Marquardt feature is being used (see argument [17]), then after the first call, DUBLIN should return an EPS no greater than the input EPS.

[23] SIG is the estimated standard deviation of the fit (Eq. (69)), where

$$SIG = \left[\frac{EPS}{N4 - N} \right]^{\frac{1}{2}} \tag{86}$$

[24] EK is a vector of crude estimates of the errors in the N paramics of point P, where

EK(K) = the estimated standard deviation in paramic P(K)

$= s_k$ as defined in Eq. (71)

[25] NS is a convergence flag. Before returning to the user's main program, FINLIE will set

NS=0 if the process has not yet converged by criterion (68), but there is still hope. FINLIE is saying in effect, "Nothing obvious has gone wrong yet so give DUBLIN another call."

NS=1 if all output arguments (except this one) are invalid. Usually this happens when some input argument is invalid. (FINLIE performs a few simple checks to spot invalid inputs.) If DUBLIN returns an NS value of 1, the main program should take some special action (e.g., STOP).

NS=2 if the latest iteration has satisfied convergence criterion (68). If the user is willing to accept this criterion, his main program should stop calling DUBLIN when NS=2. If the user is imposing some more stringent convergence criterion of his own, he should regard NS=2 as having the same meaning as NS=0.

To illustrate the use of these twenty-five arguments, consider our sample systems (3) and (4) with three-round data. Assume that in the calling program, arrays Y, W, R, and YC have been dimensioned as (2,50). Then for system (3) and subroutine ROME, we can write

```
CALL DUBLIN(ROME,1,1,2,2,2,2,3,NM,0.0,X,Y,F,1,W,P,RL,NC,
     YC,R,RS,EPS,SIG,EK,NS)
```

For system (4) and subroutine ROMA, only the first two arguments above are changed:

```
CALL DUBLIN(ROMA,0,...)
```

D. Writing the Program that Calls DUBLIN

In this final section, a typical main program for utilizing FINLIE is broken down into six steps. Some of these steps are essential, others are optional.

Step (1). Dimension all ten arrays appearing in the CALL DUBLIN statement:

```
DIMENSION NM(nr), X(n4), Y(n1,n4), F(n), W(n1,n4),
  P(n), YC(n2,n4), R(n1,n4), RS(n1), EK(n)
```

where small-letter dimensions above denote constants no less than the values of the corresponding capital-letter names. That is, nr ≥ NR, etc. As we have mentioned, it is often useful to dimension arrays larger than their current working sizes. For example, in our tri-round test cases (3) and (4), we might write:

```
DIMENSION NM(5), X(50), Y(2,50), F(8), W(2,50), P(8), YC(2,50),
     R(2,50), RS(2), EK(8)
```

This would allow for up to five rounds (nr=5), fifty measurement points (n4=50) and two measured variables (n1=2). Note that the values given to the row sizes n1 and n2 in this DIMENSION statement become the values of arguments N7 and N8 when DUBLIN in called. On the other hand, the dimensions allotted above to the vector arguments and to the columns of the matrix arguments are of no interest to FINLIE.

Step (2). Declare in an EXTERNAL statement the user subroutine whose name will be passed to DUBLIN. Thus for sample set (3) and the corresponding ROME (Table VI), we would write

```
EXTERNAL ROME
```

and similarly for set (4) and ROMA.

Step (3). Establish initial values for seven DUBLIN arguments:

NM, X,Y,F,P,RL,NC

and if necessary, for an eighth argument: W. There is no standard coding for obtaining the values of these arguments; the technique will vary with the situation. For example, initial estimates for array P might be read in at this stage, or they might be obtained by calling some subroutine whose sole purpose is to derive adequate estimates from the data. For simplicity, let's assume that in our main program for sample set (3) or (4),

(a) the arrays NM,X,Y,F and P are read in;

(b) RL and NC are defined explicitly:

```
RL = 1.0
NC = 0
```

(c) array W is not defined (since argument NW will be 1 in the CALL statement).

Note that the values of the remaining nine input arguments:

NF, N1, N2, N3, N7, N8, NR, XO, NW

can be established in the CALL DUBLIN statement itself.

Step (4). Write column headings for everything of interest that will be determined at the end of each iteration. Of course, "interest" is subjective. One user may want a detailed print-out of the progress from P_0 to $\hat{P}$; a less inquisitive user may care only for what pertains to the final, converged point. Personally, for each iteration, I like to see:

(a) the iteration number i (i=0,1,2,...)

(b) the N elements of point P_i

(c) the value of Marquardt's λ required to produce P_i

(d) the residual function $\varepsilon(P_i)$ and/or the standard deviation of the fit $s(P_i)$.

These desiderata, then, determine my column headings. (Of course, if what I want to see can not be conveniently spread across a single output page, then some of the results of each iteration have to be saved - by storing them in additional arrays - so that they can be printed later on a second page.)

Step (5). Program the DO-loop that calls DUBLIN. For our sample set (3), we might write:

```
      DO 60 K=1,26
         CALL DUBLIN (ROME,1,1,2,2,2,2,3,NM,0.0,X,Y,F,
     1          1,W,P,RL,NC,YC,R,RS,EPS,SIG,EK,NS)
         NPOINT=K-1
         WRITE(6,100)NPOINT,P,RL,SIG
         IF (NS-1) 60,70,80
   60 CONTINUE
      WRITE (6,101)
C --------The above is a warning that the process has
C --------failed to converge in 25 iterations.
      GOTO 80
   70 WRITE (6,102)
C --------The above is a warning that something is wrong.
      STOP
   80 CONTINUE
```

In the above code, DUBLIN will be called until output argument NS equals 1 or 2, or until the DO-loop variable K exceeds 26, whichever occurs first. (The limit 26 - that is, 25 iterations - is arbitrary; 1 is not enough, 10^{10} is too many.) After each iteration, we obtain a print-out - presumably under the proper column headings - of NPOINT (the number of iterations), the N elements of the current point P, and

finally the λ and s values at the current point. The first line of this print-out, where NPOINT=0, gives the initial estimated values of the paramics. If NS=1 at the end of any iteration, the program stops; otherwise the program moves eventually to statement 80. Note: the principal results of the fitting process are the final printed values of the N paramics. All else is in a sense window-dressing.

Step (6). Write anything else of interest. My usual scheme is as follows:

(a) aligned under the final paramic values (but with one line skipped for clarity), I write the corresponding crude error estimates contained in array EK. (If flag array F is of interest, the elements of F can be written on the next line, again aligned under the corresponding paramic values.)

(b) on a new output page, I write the values stored in arrays

X, Y, YC, R and (possibly) W,

one line for each X value. (In multi-round fits, I skip a line for clarity at the end of the data for each round.)

(c) wherever convenient, I write the suitably labelled values of some or all of the following:

N1,N2,N3,N4,N,NC,NM,NR,NW,RS,XO

IV. SUMMARY

The recent patter of tiny details has very likely blurred the big picture. To review, then, assume that the reader has a problem reducible to fitting a set of equations of the form (1) or (2) to measured data. Further assume that this reader--an adventurous spirit--decides to use FINLIE to solve the problem. Then this invoker of FINLIE must:

(a) derive the related set of influence equations (Section II, C or D);

(b) write a FORTRAN subroutine that lists the original equations and the related influence equations (Section III, A or B);

(c) write a FORTRAN main program (Section III, D) that will:

(i) furnish adequate initial estimates of the parameters and initial conditions;

(ii) specify which, if any, of these estimates are to be adjusted by FINLIE;

(iii) assign weights to the measurements (if the weights are not all equal);

(iv) call subroutine DUBLIN (Section III, C) in a DO-loop;

(d) submit the entire program (main, FINLIE and equation-defining subroutine) to the computer and ponder the ensuing output.

This output will take one of four forms, listed in decreasing order of desirability:

(1) convergence to the right answer;

(2) failure to converge in the specified number of iterations (sometimes achieving an apparent oscillation about an answer);

(3) divergence (the program crashes);

(4) convergence to the wrong answer.

Result (1) above--convergence to the right answer--should prevail when all of the following hold:

(a) the measured data are a representative sample of the total behavior they are meant to define. (An elementary violation would be measurements taken every τ seconds on a periodic variable of period τ.)

(b) the measured data are free of gross errors.

(c) "least squares" is a suitable fitting criterion. (This implies that the measurement errors possess certain statistical traits; however, the degree to which the errors must possess these traits in order to be considered amenable to least squares is a matter of judgment.)

(d) the fitting equations with their associated parameters are appropriate for describing the measured events.

(e) the initial paramic estimates are not too far from the right answer. (What constitutes "too far" varies with the nature of the fitting equations and the measured data.)

V. ACKNOWLEDGEMENTS

We have already acknowledged our debt to Marquardt, whose algorithm [see References 1 to 7 in the Bibliography] has been incorporated into FINLIE. This algorithm is applicable whether we are fitting differential equations or algebraic/transcendental equations. FINLIE is also indebted--especially in fitting differential equations--to the following sources:

(a) Theodore R. Goodman of Oceanics, Inc., Plainview, New York, who first called to our attention (in a private communication in 1967) the feasibility of fitting ordinary differential equations--rather than their pseudo-solutions--to observed data. Goodman's technique [see References 8 to 11 in the Bibliography] differs from FINLIE's mainly in the manner in which the influence coefficients are obtained.

(b) Gary T. Chapman and Donn B. Kirk of NASA Ames Research Center, Moffett Field, California, who developed what is now commonly referred to as the "Chapman-Kirk" technique for fitting the aerodynamic equations of motion to free-flight data [see References 12 to 16 in the Bibliography]. When applied to differential equations, FINLIE is essentially a general-purpose Chapman-Kirk program with frills.

(c) Robert H. Whyte, of General Electric, Burlington, Vermont, who for a number of years was apparently indefatigable in applying the Chapman-Kirk technique to a variety of problems. References 17 to 31 in the Bibliography are a sampling of Whyte's reports on his labors in this field.* Many of the handy features of Whyte's programs (for example, the ability to handle multi-round data and to consider any paramic value as fixed or adjustable) have found their way into FINLIE (where they apply to algebraic/transcendental equations as well). It was through my efforts to adapt one of Whyte's specialized programs to our needs that I decided that what was needed was a more general-purpose Chapman-Kirk program. Thus, the idea for FINLIE was conceived. (Unfortunately, the gestation period exceeded that of an elephant.)

** It should be noted that in applying Chapman-Kirk to the 6D equations of motion, Whyte used an unweighted least squares criterion. Since the angular and translational residuals of the fit are not of equal magnitude, Whyte was forced to decouple the angular equations from the translational equations. Dissatisfaction with this enforced and often unrealistic decoupling led Whyte and Hathaway to abandon an unweighted least squares in favor of a weighted maximum likelihood criterion. Since this criterion was derived on the assumption of a normal error distribution, their Maximum Likelihood Method [see References 35-39 in the Bibliography] should yield the same final fit (albeit by a difference path) as a comparably weighted least squares approach.*

BIBLIOGRAPHY

A. Marquardt's Algorithm

(1) Donald W. Marquardt, "An Algorithm for Least-Squares Estimation of Nonlinear Parameters," J. Soc. Indust. Appl. Math. 11, No. 2, June 1963, pp. 431-441.

(2) Duane A. Meeter, "On a Theorem Used in Nonlinear Least Squares," J. Soc. Indust. Appl. Math. 14, No. 5, September 1966, pp. 1176-1179. [Offers a stronger version of one of the theorems in (1) above.]

(3) E.M.L. Beale, "Numerical Methods," an article in the book Nonlinear Programming edited by J. Abadie, John Wiley & Sons, New York, 1967.

(4) Philip R. Bevington, Data Reduction and Error Analysis for the Physical Sciences, McGraw-Hill, New York, 1969. [Lists and discusses a FORTRAN subroutine CURFIT for fitting a single nonlinear equation of the type y=g(x,P) with the aid of Marquardt's algorithm.]

(5) Cuthbert Daniel and Fred S. Wood, Fitting Equations to Data: Computer Analysis of Multifactor Data for Scientists and Engineers, Wiley-Interscience, New York, 1971. [Gives a User's Manual (and addresses at which the reader can obtain a program listing) for Wood's FORTRAN program for fitting a single nonlinear equation of the type y=g(x,P) with the aid of Marquardt's algorithm. Offers a chapter-long discussion of an involved application.]

(6) James W. Bradley, "Application of Marquardt's Nonlinear Least Squares Algorithm to Free-Flight Yaw Data Analysis," Ballistic Research Laboratories Memorandum Report No. 2526, September 1975. AD A016906. [Lists and discusses a FORTRAN subroutine MARQ, an offspring of the subroutine CURFIT given in (4) above. The minor refinements in Marquardt's algorithm that have been written into FINLIE are discussed in detail here.]

(7) Keyboard 1978/3 (a publication of Hewlett-Packard Desktop Computer Division). This issue mentions the availability of a "9845 Nonlinear Regression Software" package (09845-15040) for use with the HP System 45. I'm not familiar with the program, but to quote, "This software pack contains programs using Marquardt's Method to fit nonlinear models using up to ten parameters."

B. Goodman's Method

(8) Theodore R. Goodman, "System Identification and Prediction - An Algorithm Using a Newtonian Iteration Procedure," Quarterly of Applied Mathematics, XXIV, No. 3, October 1966.

(9) Theodore R. Goodman and Theodore P. Sargent, "A General Method for Identifying Nonlinear Dynamical Systems," Oceanics, Inc. Technical Report No. 68-53, September 1968. [Oceanics, Inc. is located at Technical Industrial Park, Plainview, New York 11803.]

(10) Theodore R. Goodman and Theodore P. Sargent, "A Method for Identifying Nonlinear Systems with Applications to Vehicle Dynamics and Chemical Kinetics," Oceanics, Inc. Technical Report No. 71-83, August 1971. [Presented at the 1971 Joint Automatic Control Conference, Washington University, St. Louis, Missouri.]

(11) James W. Bradley, "CHLOE: a FORTRAN Subroutine for Fitting Ordinary Differential Equations to Observed Data," Ballistic Research Laboratories Memorandum Report No. 2184, February 1972. AD 743878.

C. The Chapman-Kirk Technique

(12) Gary T. Chapman and Donn B. Kirk, "A Method for Extracting Aerodynamic Coefficients from Free-Flight Data," AIAA Journal 8, No. 4, April 1970, pp. 753-758. [Originally presented in slightly different form as AIAA Paper No. 69-134 at the AIAA 7th Aerospace Sciences Meeting, New York City, January 1969.]

(13) Charles H. Murphy, "Comment on 'A Method for Extracting Aerodynamic Coefficients from Free-Flight Data'", AIAA Journal 8, No. 11, November 1970, pp. 2109-2111.

(14) Gary T. Chapman and Donn B. Kirk, "Reply by Authors to C.H. Murphy," AIAA Journal 8, No. 11, November 1970, p. 2111.

(15) Gary T. Chapman, "Aerodynamic Parameter Identification in Ballistic Range Tests," a paper presented at the 1972 Army Numerical Analysis Conference, Edgewood Arsenal, Maryland.

(16) Donald C. Daniel, "An Analysis of Methods for Extracting Aerodynamic Coefficients from Test Data," Air Force Armament Laboratory Technical Report AFATL-TR-73-32, Eglin Air Force Base, Florida, February 1973. [Compares the deterministic Chapman-Kirk technique with the stochastic extended-Kalman-filter technique.]

D. Whyte's Applications of the Chapman-Kirk Technique

("RHW" in the following denotes Robert H. Whyte; "GE" denotes General Electric Company, Armament Systems Department, Burlington, VT 05401.)

(17) RHW and Jean-Guy Beliveau, "An Investigation of the Chapman-Kirk Free Flight Data Reduction Technique," GE Advance Munitions Report, August 1969.

(18) RHW and Jean-Guy Beliveau, "Non-Linear Free Flight Aerodynamic Reduction Using Angle of Attack or Angular Rate Data," GE Class 2 Report No. R69APB6, December 1969. [A revision of a report with the same report number but dated September 1969.]

(19) RHW and Jean-Guy Beliveau, "Recent Applications of a Numerical Integration Scheme for Analyzing Free Flight Motion Data," GE Advance Munitions Report, July 1970.

(20) RHW and Jean-Guy Beliveau, "Reduction of 4.2 Inch Mortar Free Flight Spark Range Data by Numerical Integration," GE Advance Munitions Report, August 1970.

(21) RHW and Angela Jeung, "Aerodynamic Reduction of Free Flight Transonic Range Data Utilizing Numerical Integration," GE Report No. 71APB514, April 1971.

(22) Wayne H. Hathaway and RHW, "Interior Ballistic Data Reduction Technique," GE Report No. 71APB561, December 1971.

(23) RHW and Wayne H. Hathaway, "Reduction of Yaw Sonde and Position Time Radar Data by Numerical Integration," GE Report No. 72APB506, January 1972.

(24) RHW and Ray C. Houghton, "Reduction of Range and Yaw Sonde Data Using Numerical Integration Techniques," GE Report No. 72APB539, September 1972.

(25) RHW and Ray C. Houghton, "User Manual for AEDC Range G Free Flight Reduction Computer Programs," GE Report No. 72APB550, October 1972. [AEDC is the Arnold Engineering Development Center, Tullahoma, Tennessee.]

(26) RHW, Angela Jeung and James W. Bradley, "Chapman-Kirk Reduction of Free-Flight Range Data to Obtain Nonlinear Aerodynamic Coefficients," Ballistic Research Laboratories Memorandum Report No. 2298, May 1973. AD 762148. [Covers much the same material as (21) above, but from a different viewpoint.]

(27) RHW, Ray C. Houghton and Wayne H. Hathaway, "Description of Yaw Sonde Numerical Integration Data Reduction Computer Programs," GE Report No. 73APB514, May 1973.

(28) RHW and William H. Mermagen, "A Method for Obtaining Aerodynamic Coefficients from Yawsonde and Radar Data," Journal of Spacecraft and Rockets 10, No. 6, July 1973, pp. 384-388. [Originally presented as AIAA Paper No. 72-978 at the AIAA 2nd Atmospheric Flight Mechanics Conference, Palo Alto, California, September 1972. See also Ballistic Research Laboratories Memorandum Report No. 2280, March 1973. AD 759482.]

(29) RHW and Wayne H. Hathaway, "Aeroballistic Range Data Reduction Technique Utilizing Numerical Integration," Air Force Armament Laboratory Technical Report AFATL-TR-74-41, Eglin Air Force Base, Florida, February 1974.

(30) RHW, Ray C. Houghton and Wayne H. Hathaway, "Description of Yaw Sonde Numerical Integration Data Reduction Computer Programs," Ballistic Research Laboratories Contract Report No. 280, December 1975. [Documents work done by GE under a U.S. Government Contract, November 1972 to April 1973.]

(31) RHW and Ray C. Houghton, "Reduction of Range and Yaw Sonde Data Using Numerical Integration Techniques," U.S. Army Armament Research and Development Command, Ballistic Research Laboratory Control Report ARBRL-CR-00400, May 1979. [Documents work done by GE under a U.S. Government Contract, April to August 1972.]

E. Miscellaneous

(32) Monte W. Coleman, "MERSON Integration Routine," SPB-10-70, August 1970. [One of a series of informal bulletins issued by what was then the Computer Support Division and is now the Management Information Systems Support Division (MISSD) of BRL.]

(33) Aivars Celmins, "Least Squares Adjustment with Finite Residuals for Non-Linear Constraints and Partially Correlated Data," Ballistic Research Laboratories Report No. 1658, July 1973. AD 766283. [Derives a better variance-covariance matrix associated with a least-squares fit.]

(34) Martin Becker, "Yaw Sonde and Radar Data Reduction to Obtain Aerodynamic Coefficients," U.S. Naval Weapons Laboratory Technical Report No. TR-3073, September 1974. [Describes two RHW/GE computer programs: HEEVE for reducing radar data and ANGLES for obtaining aerodynamic coefficients from yaw sonde data.]

(35) R.D. Grove, R.L. Bowles and S.C. Mayhew, "A Procedure for Estimating Stability and Control Parameters from Flight Test Data by Using Maximum Likelihood Methods Employing a Real-Time Digital System," Langley Research Center, NASA TN D-6735, May 1972. [Apparently the report that led Whyte and Hathaway to apply the Maximum Likelihood Method to free flight data analysis. See (36)-(39) below.]

F. Whyte and Hathaway's Maximum Likelihood Method

("RHW" and "WHH" in the following denote Robert H. Whyte and Wayne H. Hathaway.)

(36) Kenneth O. West and RHW, "Free Flight and Wind Tunnel Test of a Missile Configuration at Subsonic and Transonic Mach Numbers with Angles of Attack up to 30 Degrees," Vol. II, Paper 39 of the Proceedings of the 11th Navy Symposium on Aeroballistics, Trevose, PA, August 1978.

(37) RHW, WHH and E.M. Friedman, "Analysis of Free Flight Transonic Range Data of the 155mm, M483A1 and XM795 Projectiles," ARLCD-CR-79016, August 1979.

(38) WHH and RHW, "Aeroballistic Research Facility Free Flight Data Analysis Using the Maximum Likelihood Method," Air Force Armament Laboratory Technical Report AFATL-TR-79-98, December 1979.

(39) RHW, J.R. Burnett and WHH, "Analysis of Free Flight Transonic Range Data of the 155mm M549 Projectile," Unnumbered GE Report, April 1980.

LIST OF SYMBOLS

A(P)	$\sum_{n=1}^{NR} \alpha_{En}$, an N by N multi-round matrix.
a_{jk}	the (j,k)-th element of matrix A. [j,k=1,2,...N]
$\tilde{a}_{jk}$	$(a_{jj}a_{kk})^{-\frac{1}{2}}\, a_{jk}$ [nondimensional]
$\approx{a}_{jk}$	$1 + \lambda$ when j=k; $\tilde{a}_{jk}$ when j≠k
$\vec{B}$	$\sum_{n=1}^{NR} \vec{\beta}_{En}$, an N-dimensional multi-round vector.
b	$c_2 + (y_{20})^{-1}$ in system (4)
b_k	the k-th component of $\vec{B}$.
$\tilde{b}_k$	$(a_{kk})^{-\frac{1}{2}}\, b_k$ [nondimensional]
C	the vector of N3 parameters.
COND	a ROMA input argument vector of N2 single-round initial conditions.
CR	$\varepsilon(P_n)/\varepsilon(P_{n-1})$, FINLIE's measure of convergence.
c_j	the j-th parameter. [j=1,2,...N3]
$D_{jk}(x_m,Q)$	$\partial y_j(x_m,Q)/\partial q_k$, the (j,k)-th influence coefficient evaluated at x_m, using the current point Q. [j=1,2,...N2; k=1,2,...N23]
DU	a ROME output argument vector, Eq. (74).
EK	a DUBLIN output argument vector of crude estimates s_k. [k=1,2,...N]
En	the n-th round identifier. [n=1,2,...NR]
EPS	a DUBLIN output argument: $\varepsilon(P)$, the value of ε at the point currently stored in argument P.
F	a DUBLIN input argument vector of N adjust-or-hold-fixed flags associated with the N paramics p_k.
FLAG	a COMMON/NAPLES/ input vector to ROME (and ROMA) containing the N23 single-round adjust-or-hold-fixed flags associated with the N23 paramics q_k.

f_j	y_j', system (1)
g_j	y_j, system (2)
h	a nondimensional positive constant in the steepest descent technique, Eq. (55).
IC	the set of multi-round initial conditions.
N	(NR × N2) + N3, the number of paramics in the system.
NA	N2 × N23, the number of influence equations for system (1).
NB	N1 × N23, the number of needed influence equations for system (2).
NC	a DUBLIN I/O argument: originally zero (the "first-call" flag), it becomes the number of paramics adjusted.
NF	a DUBLIN input argument: 1 to fit system (1); 0 to fit system (2).
NM	a DUBLIN input argument vector, where NM(J) is the number of data points R_m in the j-th round.
NR	a DUBLIN input argument: the number of rounds (hence the number of distinct sets of initial conditions to be determined).
NS	a DUBLIN output argument: 0 means "CALL again"; 1 means "a disaster has occurred"; 2 means "convergence by FINLIE's criterion".
NW	a DUBLIN input argument: 0 to use the user's weights; 1 to set all weights at unity.
N1	a DUBLIN input argument: the number of measured dependent variables.
N2	a DUBLIN input argument: the number of dependent variables in the system.
N3	a DUBLIN input argument: the number of parameters in the system.
N4	the number of data points R_m for all the rounds.

N5	a ROME input argument: the number of equations (N2 differential equations plus NA influence equations) to be defined in ROME.
N23	N2 + N3, the number of paramics in a single round.
P	(a) a set of N multi-round paramics; (b) a point in the N-dimensional paramic space; (c) a DUBLIN I/O argument vector containing the N paramic values.
$\vec{P}$	the vector from the origin to point P in the N-dimensional paramic space.
$\hat{P}$	the value of point P that minimizes ε.
PAR	a COMMON/NAPLES/ input vector to ROME (and ROMA) containing the N3 parameters
P_n	the value of point P at the end of the n-th iteration. [n=0,1,...]
P_{nA}	a candidate for point P_n, obtained by setting $\lambda=\lambda_{nA}$ [n=1,2,...]
paramic	parameter or initial condition.
p_k	the k-th paramic (k=1,2,...N) where the order is: first-round initial conditions, second-round initial conditions,... and finally the N3 parameters.
$\tilde{p}_k$	$(a_{kk})^{\frac{1}{2}} p_k$, the k-th nondimensional paramic.
Q	the single-round equivalent of P.
$\hat{Q}$	the value of point Q that minimizes γ.
q_k	the k-th single-round paramic, where the order is: the N2 initial conditions, then the N3 parameters.
R	a DUBLIN output argument: the N1 by N4 matrix of residuals, Eq. (83), at the current point P.
RL	a DUBLIN I/O argument: initially set at 0.0 to avoid Marquardt's algorithm; 1.0 to invoke it.
R_m	the m-th data point, consisting of x_m and the N1 dependent variable measurements $\bar{y}_{jm}$.

ROMA	an arbitrary name for the user's subroutine defining system (2).
ROME	an arbitrary name for the user's subroutine defining system (1).
RS	a DUBLIN output argument vector of N1 error measures, Eq. (84).
round	an experiment at which measurements were taken and with which a distinct set of initial condition values can be associated.
S	the N-dimensional space in which point P has coordinates $(p_1, p_2, \cdots p_N)$.
$\tilde{S}$	the N-dimensional scaled space in which point P has coordinates $(\tilde{p}_1, \tilde{p}_2, \cdots \tilde{p}_N)$
S_1	the N23-dimensional single-round space in which point Q has coordinates $(q_1, q_2, \cdots q_{N23})$.
SIG	a DUBLIN output argument: the value of s at the current point P.
s	estimated standard deviation of the fit, Eq. (69).
s_k	the crude estimated standard deviation in paramic p_k, Eqs. (70-71).
U	a ROME input argument vector, Eq. (73); a ROMA output argument vector.
u	$\exp[c_1(x-x_0)]$ in system (4).
W	a DUBLIN I/O argument: the N1 by N4 matrix of weights w_{jm}.
w_{jm}	the non-negative weighting factor associated with $\bar{y}_{jm}$.
X	a DUBLIN input argument vector of the N4 values x_m.
XE	a ROME and ROMA input argument: the value of x.
XO	a DUBLIN and ROMA input argument: x_0.
x	the independent variable of the system

x_m	the m-th value of x at which measurements were taken.
x_0	the value of the independent variable at which all initial conditions apply.
Y	(a) a vector of N2 dependent variables; (b) a DUBLIN input argument: the N1 by N4 matrix of dependent variable measurements $\bar{y}_{jm}$.
YC	a DUBLIN output argument: the N2 by N4 matrix of computed dependent variable values $y_j(x_m, P)$.
Y_0	a vector of N2 single-round initial conditions.
y_j	the j-th dependent variable. [j=1,2,...N2]
$\bar{y}_{jm}$	the measured value of y_j at x_m [j=1,2,...N1; m=1,2,...N4]
$y_j(x_m, P)$	the calculated value of y_j at x_m, using the current point P.
$\alpha(Q)$	the N23 by N23 single-round matrix at point Q
α_{En}	the N by N expansion of the matrix α associated with round En.
$\alpha_{kn}(Q)$	the (k,n)-th element of $\alpha(Q)$, Eqs. (36-38).
$\vec{\beta}(Q)$	a vector point function of Q: the vector in space S_1 in the direction of the negative gradient of γ at point Q.
$\vec{\beta}_{En}$	the N-dimensional expansion of the vector $\vec{\beta}$ associated with round En.
β_k	the k-th component of $\vec{\beta}$. [k=1,2,...N23]
$\vec{\Delta}P_n$	the increment vector in S from point P_n to the nearby point P_{n+1}.
$\vec{\Delta}Q_n$	the increment vector in S_1 from point Q_n to the nearby point Q_{n+1}.
$\gamma(Q)$	the single-round equivalent of $\varepsilon(P)$, Eq. (14).

$\varepsilon(P)$	the nondimensional sum of the weighted squares of the residuals, Eq. (7); the value of the scalar point function ε at point P.
λ	a nonnegative constant added to each diagonal element of the scaled form of matrix A and adjusted by Marquardt's algorithm so that $\varepsilon(P_{n+1}) < \varepsilon(P_n)$.
λ_{nA}, λ_{nB}	two consecutive trial values assigned to λ in an effort to move from point P_n, where $\lambda_{nB}=10\lambda_{nA}$.

Superscripts

$(\vec{\ })$	a row vector
$(\vec{\ })^T$	the transpose of a row vector: that is, a column vector
$(\tilde{\ })$	the scaled (hence nondimensional) form of ( ).
$(\ )'$	d( )/dx

Subscripts

$[\ ]_d$	denotes the dimensions of [ ]
$(\ )_S$	the components of ( ) are in space S
$(\ )_{\tilde{S}}$	the components of ( ) are in space $\tilde{S}$
$(\ )_{S_1}$	the components of ( ) are in space S_1.

APPENDIX

```
C
C                                  **** FINLIE ****                                    FNLE    1
C                                                                                      FNLE    2
      SUBROUTINE DUBLIN (ROME,NF,N1,N2,N3,N7,N8,NR,NM,X0,X,Y,F,NW,W,                   FNLE    3
     1                   P,RL,NC,YC,R,RS,EPS,SIG,EK,NS)                               FNLE    4
C                                                                                      FNLE    5
C    * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *   FNLE    6
C    *                                                                               *  FNLE    7
C    *     INPUT ARGUMENTS ...                                                       *  FNLE    8
C    *      ROME = THE DUMMY NAME OF THE SUBROUTINE (WRITTEN BY THE                  *  FNLE    9
C    *             USER) THAT DEFINES EITHER                                         *  FNLE   10
C    *                 (A) THE DEPENDENT VARIABLES (NF = 0)                          *  FNLE   11
C    *              OR (B) THE DERIVATIVES OF THE DEPENDENT VARIABLES                *  FNLE   12
C    *                 (NF .NE. 0)                                                   *  FNLE   13
C    *        NF = THE FLAG THAT INDICATES THE NATURE OF SUBROUTINE                  *  FNLE   14
C    *             ROME (SEE PREVIOUS ARGUMENT)                                      *  FNLE   15
C    *        N1 = NUMBER OF MEASURED DEPENDENT VARIABLES                            *  FNLE   16
C    *             (1 .LE. N1 .LE. 10)                                               *  FNLE   17
C    *        N2 = TOTAL NUMBER OF DEPENDENT VARIABLES                               *  FNLE   18
C    *             (N1 .LE. N2 .LE. 20)                                              *  FNLE   19
C    *        N3 = THE MAXIMUM NUMBER OF PARAMETERS (NOT COUNTING                    *  FNLE   20
C    *             INITIAL CONDITIONS) WHOSE VALUES CAN BE DETERMINED                *  FNLE   21
C    *             FROM THE USER'S SUBROUTINE ROME (0 .LE. N3 .LE. 40)               *  FNLE   22
C    *        N7 = THE NUMBER OF ROWS IN ARRAYS Y,W AND R BELOW, AS                  *  FNLE   23
C    *             DIMENSIONED IN THE CALLING PROGRAM (N7 .GE. N1)                   *  FNLE   24
C    *        N8 = THE NUMBER OF ROWS IN ARRAY YC BELOW, AS                          *  FNLE   25
C    *             DIMENSIONED IN THE CALLING PROGRAM (N8 .GE. N2)                   *  FNLE   26
C    *        NR = THE NUMBER OF ROUNDS (INDIVIDUAL CASES) TO BE REDUCED             *  FNLE   27
C    *             SIMULTANEOUSLY ( 1 .LE. NR .LE. (60 - N3)/N2 )                    *  FNLE   28
C    *        NM = A VECTOR OF NR ELEMENTS, WHERE                                    *  FNLE   29
C    *             NM(J) = THE NUMBER OF INDEPENDENT VARIABLE VALUES AT              *  FNLE   30
C    *                     WHICH MEASUREMENTS WERE TAKEN FOR THE J-TH                *  FNLE   31
C    *                     ROUND                                                     *  FNLE   32
C    *             NOTE ... WE DEFINE                                                *  FNLE   33
C    *                N4 = NM(1) + NM(2) + ... + NM(NR)                              *  FNLE   34
C    *                   = THE TOTAL NUMBER OF INDEPENDENT VARIABLE                  *  FNLE   35
C    *                     VALUES FOR ALL THE ROUNDS                                 *  FNLE   36
C    *                 N = NR*N2 + N3                                                *  FNLE   37
C    *                   = THE NO. OF ELEMENTS IN P BELOW                            *  FNLE   38
C    *              NMAX = THE MAXIMUM ELEMENT OF ARRAY NM                           *  FNLE   39
C    *             THEN WE MUST HAVE                                                 *  FNLE   40
C    *                 N .LT. N4 .LE. 1000                                           *  FNLE   41
C    *                 N1*NMAX .LE. 200                                              *  FNLE   42
C    *                 N2*NMAX .LE. 400                                              *  FNLE   43
C    *        X0 = THE REFERENCE INDEPENDENT VARIABLE VALUE AT WHICH                 *  FNLE   44
C    *             ALL INITIAL CONDITIONS APPLY.  NOTE ... INPUTS X0,                *  FNLE   45
C    *             N1,N2 AND N3 ARE ASSUMED TO HAVE THE SAME VALUE FOR               *  FNLE   46
C    *             EACH ROUND.                                                       *  FNLE   47
C    *         X = A VECTOR OF THE N4 INDEPENDENT VARIABLE VALUES AT                 *  FNLE   48
C    *             WHICH MEASUREMENTS WERE TAKEN, WHERE                              *  FNLE   49
C    *             X(1), .........,X(NM(1))        FOR THE FIRST ROUND               *  FNLE   50
C    *             X(NM(1)+1), ...,X(NM(1)+NM(2)) FOR THE SECOND ROUND               *  FNLE   51
C    *             ETC.                                                              *  FNLE   52
C    *         Y = THE N1 BY N4 MATRIX OF MEASURED DEPENDENT VARIABLE                *  FNLE   53
C    *             VALUES, WHERE                                                     *  FNLE   54
C    *             Y(I,J) = THE MEASURED VALUE OF THE I-TH DEPENDENT                 *  FNLE   55
C    *                      VARIABLE AT X(J)                                         *  FNLE   56
C    *         F = THE VECTOR OF N FLAGS FOR ARGUMENT P BELOW, WHERE                 *  FNLE   57
C    *             F(J) = 0.0 IF THE VALUE OF P(J) IS FIXED                          *  FNLE   58
```

```
C     *          F(J) = 1.0 IF THE CURRENT VALUE OF P(J) IS TO BE      * FNLE  59
C     *                     ADJUSTED BY THE FITTING PROCESS            * FNLE  60
C     *       NW = THE WEIGHT FLAG ASSOCIATED WITH W BELOW, WHERE      * FNLE  61
C     *          NW = 0 IF THE USER'S WEIGHTS IN ARGUMENT W ARE TO     * FNLE  62
C     *                 BE USED                                        * FNLE  63
C     *          NW = 1 IF ALL WEIGHTS ARE UNITY.  IN THIS EVENT, THE  * FNLE  64
C     *                 THE FIRST TIME THIS SUBROUTINE IS CALLED, ALL  * FNLE  65
C     *                 ELEMENTS OF MATRIX W ARE SET TO 1.0.  (HENCE,  * FNLE  66
C     *                 THE USER NEED NOT ESTABLISH W WHEN ALL ITS     * FNLE  67
C     *                 ELEMENTS ARE 1.0.)                             * FNLE  68
C     *        W = THE N1 BY N4 MATRIX OF WEIGHTS ASSOCIATED WITH INPUT * FNLE  69
C     *          Y ABOVE.  SEE ARGUMENT NW ABOVE.                      * FNLE  70
C     *    INPUT/OUTPUT ARGUMENTS ...                                  * FNLE  71
C     *        P = THE CURRENT POINT AT WHICH OTHER ARGUMENTS ARE       * FNLE  72
C     *          EVALUATED, WHERE                                      * FNLE  73
C     *          P(1),.....P(N2) = I.C. FOR FIRST ROUND                * FNLE  74
C     *          P(N2+1),.....P(2*N2) = I.C. FOR SECOND ROUND          * FNLE  75
C     *          ....                                                  * FNLE  76
C     *          P(NR*N2-N2+1),...P(NR*N2) = I.C. FOR LAST ROUND       * FNLE  77
C     *              P(NR*N2+1),.....P(N) = PARAMETERS                 * FNLE  78
C     *       RL = A MARQUARDT ARGUMENT.  BEFORE DUBLIN IS CALLED THE  * FNLE  79
C     *          FIRST TIME,                                           * FNLE  80
C     *          SET RL = 0.0 IF THE MARQUARDT ALGORITHM IS TO BE      * FNLE  81
C     *                       OMITTED.  THEREAFTER, RL WILL REMAIN     * FNLE  82
C     *                       AT 0.0.                                  * FNLE  83
C     *          SET RL = 1.0 IF THE MARQUARDT ALGORITHM IS TO BE      * FNLE  84
C     *                       USED.  THEREAFTER, RL UPON RETURN WILL   * FNLE  85
C     *                       BE MARQUARDT'S LAMBDA.  (HENCE, USE A    * FNLE  86
C     *                       NAME, NOT 1.0, IN THE CALL LIST.)        * FNLE  87
C     *       NC = THE 'FIRST CALL' FLAG'.  BEFORE THIS SUBROUTINE IS  * FNLE  88
C     *          CALLED FOR THE FIRST TIME, NC MUST BE SET TO 0.       * FNLE  89
C     *          THIS SUBROUTINE THEN ESTABLISHES NC AS THE ACTUAL     * FNLE  90
C     *          NUMBER OF INITIAL CONDITIONS AND PARAMETERS BEING     * FNLE  91
C     *          DETERMINED  (1 .LE. NC .LE. N)                        * FNLE  92
C     *    OUTPUT ARGUMENTS ...                                        * FNLE  93
C     *       YC = THE N2 BY N4 MATRIX OF COMPUTED DEPENDENT VARIABLE  * FNLE  94
C     *          VALUES AT THE POINT CONCURRENTLY STORED IN ARRAY P,    * FNLE  95
C     *          WHERE YC(J,K) = COMPUTED VALUE OF THE J-TH DEPENDENT  * FNLE  96
C     *                          VARIABLE AT X(K)                      * FNLE  97
C     *        R = THE N1 BY N4 MATRIX OF RESIDUALS, WHERE             * FNLE  98
C     *          R(I,J) = Y(I,J) - YC(I,J)                             * FNLE  99
C     *       RS = THE VECTOR OF N1 ERROR MEASURES, WHERE              * FNLE 100
C     *          RS(I) = WEIGHTED SUM OF THE SQUARES OF THE            * FNLE 101
C     *                  RESIDUALS IN THE I-TH MEASURED DEPENDENT      * FNLE 102
C     *                  VARIABLE                                      * FNLE 103
C     *      EPS = THE ERROR MEASURE OF THE FIT AT THE POINT           * FNLE 104
C     *          CONCURRENTLY STORED IN ARRAY P.  EPS IS THE WEIGHTED  * FNLE 105
C     *          SUM OF THE SQUARES OF THE RESIDUALS OVER ALL THE      * FNLE 106
C     *          POINTS, OVER ALL THE MEASURED DEPENDENT VARIABLES     * FNLE 107
C     *          AND OVER ALL THE ROUNDS.                              * FNLE 108
C     *      SIG = THE ESTIMATED STANDARD DEVIATION OF THE FIT         * FNLE 109
C     *       EK = THE VECTOR OF CRUDELY ESTIMATED STANDARD DEVIATIONS * FNLE 110
C     *          IN THE ELEMENTS OF POINT P.                           * FNLE 111
C     *       NS = OUTPUT FLAG, WHERE                                  * FNLE 112
C     *          NS = 0 IF THE PROCESS HAS NOT YET CONVERGED BY        * FNLE 113
C     *                 THE CRITERION BUILT INTO SUBROUTINE DUBLIN     * FNLE 114
C     *             = 1 IF ALL OUTPUT ARGUMENTS ARE INVALID (PROBABLY  * FNLE 115
C     *                 BECAUSE SOME INPUT ARGUMENTS ARE TOO LARGE     * FNLE 116
C     *                 FOR CERTAIN DIMENSIONED ARRAYS).  THE CALLING  * FNLE 117
C     *                 PROGRAM SHOULD TAKE SPECIAL ACTION (E.G.,      * FNLE 118
```

```
C     *                    STOP) IN THIS EVENT.                                  * FNLE 119
C     *                = 2 IF THE PROCESS HAS SATISFIED THE BUILT-IN             * FNLE 120
C     *                    CONVERGENCE CRITERION                                 * FNLE 121
C     *                                                                          * FNLE 122
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * FNLE 123
C                                                                                 FNLE 124
      DIMENSION NM(1),X(1),Y(N7,1),F(1),W(N7,1),P(1),YC(N8,1),R(N7,1),            FNLE 125
     1          RS(1),EK(1)                                                       FNLE 126
      DIMENSION ALFA(3600),ALF(60,60),GAMMA(60,60),BATA(60),S(60),                FNLE 127
     1          PA(60),PB(60),PC(60),ALPHA(60,60),BETA(60)                        FNLE 128
      EXTERNAL ROME                                                               FNLE 129
C                                                                                 FNLE 130
C     ***  PART 1.   PRELIMINARIES                                                FNLE 131
C                                                                                 FNLE 132
      M1 = N1                                                                     FNLE 133
      M2 = N2                                                                     FNLE 134
      M3 = N3                                                                     FNLE 135
      M7 = N7                                                                     FNLE 136
      M8 = N8                                                                     FNLE 137
      MC = NC                                                                     FNLE 138
      MR = NR                                                                     FNLE 139
      QL = RL                                                                     FNLE 140
      EA = EPS                                                                    FNLE 141
      IF (MC .GT. 0) GOTO 40                                                      FNLE 142
C                                                                                 FNLE 143
C                   THE FIRST TIME DUBLIN IS CALLED (INPUT NC = 0),               FNLE 144
C                   SET ALL WEIGHTS TO 1.0 IF NW = 1.  THEN EVALUATE              FNLE 145
C                   ALPHA, BETA, YC, R, RS, EPS AND SIG AT INPUT                  FNLE 146
C                   POINT P.  OUTPUT NC IS THE NUMBER OF PARAMETERS AND           FNLE 147
C                   I.C. TO BE DETERMINED.  IF THE MARQUARDT ALGORITHM            FNLE 148
C                   IS TO BE USED, SET RL = .01.  RETURN.                         FNLE 149
C                                                                                 FNLE 150
      M4 = 0                                                                      FNLE 152
      DO 5 J = 1,MR                                                               FNLE 153
          M4 = M4 + NM(J)                                                         FNLE 154
    5 CONTINUE                                                                    FNLE 155
      IF (NW .NE. 1) GOTO 30                                                      FNLE 156
      DO 20 J = 1,M4                                                              FNLE 157
          DO 10 I = 1,M1                                                          FNLE 158
              W(I,J) = 1.0                                                        FNLE 159
   10     CONTINUE                                                               FNLE 160
   20 CONTINUE                                                                    FNLE 161
   30 CALL LONDON (ROME,NF,M1,M2,M3,M7,M8,60,MR,NM,X0,X,Y,W,F,P,                  FNLE 162
     1             EA,MC,ALPHA,BETA,YC,R,RS,NS)                                 FNLE 163
      EPS = EA                                                                    FNLE 164
      EM = M4 - MC                                                                FNLE 165
      IF (EM .GT. 0.) SIG = SQRT(EPS/EM)                                          FNLE 166
      IF (EM .LE. 0.) SIG = 0.                                                    FNLE 167
      NC = MC                                                                     FNLE 168
      IF (QL .NE. 0.) RL = 0.01                                                   FNLE 169
      RETURN                                                                      FNLE 170
C                                                                                 FNLE 171
C     ***  PART 2.   ON SUBSEQUENT DUBLIN CALLS, DECREASE THE INPUT RL.           FNLE 172
C                   SHRINK INPUT ALPHA, BETA AND P TO ALF, BATA AND PA            FNLE 173
C                   BY ELIMINATING ALL 'FIXED' (F(K)=0.0) COMPONENTS.             FNLE 174
   40 CONTINUE                                                                    FNLE 175
      IF (QL .GT. 0.5E-16) QL = 0.1*QL                                            FNLE 176
      LD = MR*M2 + M3                                                             FNLE 177
      JA = 0                                                                      FNLE 178
      JB = 0
```

```
      DO 60 J = 1,LD                                                    FNLE 179
         IF (F(J) .EQ. 0.) GOTO 60                                      FNLE 180
         DO 50 K = 1,LD                                                 FNLE 181
            IF (F(K) .EQ. 0.) GOTO 50                                   FNLE 182
            JB = JB + 1                                                 FNLE 183
            ALFA(JB) = ALPHA(K,J)                                       FNLE 184
   50       CONTINUE                                                    FNLE 185
         JA = JA + 1                                                    FNLE 186
         BATA(JA) = BETA(J)                                             FNLE 187
         PA(JA) = P(J)                                                  FNLE 188
   60 CONTINUE                                                          FNLE 189
      JB = 0                                                            FNLE 190
      DO 80 J = 1,MC                                                    FNLE 191
         DO 70 K = 1,MC                                                 FNLE 192
            JB = JB + 1                                                 FNLE 193
            ALF(K,J) = ALFA(JB)                                         FNLE 194
   70       CONTINUE                                                    FNLE 195
   80 CONTINUE                                                          FNLE 196
C                                                                       FNLE 197
C                  FORM SCALE FACTORS S(J).                             FNLE 198
C                  REPLACE BATA WITH SCALED BATA.                       FNLE 199
C                  FORM SCALED ALF(J,K) AND STORE ABOVE THE PRINCIPAL   FNLE 200
C                           DIAGONAL AS ALF(K,J).                       FNLE 201
C                  FORM BM = THE SQUARE OF THE MAGNITUDE OF THE         FNLE 202
C                           SCALED BATA VECTOR.                         FNLE 203
C                                                                       FNLE 204
      BM = 0.                                                           FNLE 205
      DO 100 J = 1,MC                                                   FNLE 206
         S(J) = 1./SQRT(ALF(J,J))                                       FNLE 207
         BATA(J) = S(J)*BATA(J)                                         FNLE 208
         BM = BM + BATA(J)**2                                           FNLE 209
         K = J - 1                                                      FNLE 210
   90    IF (K .EQ. 0) GOTO 100                                         FNLE 211
         ALF(K,J) = S(J)*S(K)*ALF(J,K)                                  FNLE 212
         K = K - 1                                                      FNLE 213
         GOTO 90                                                        FNLE 214
  100 CONTINUE                                                          FNLE 215
C                                                                       FNLE 216
C   ***  PART 3.   FORM MATRIX GAMMA BASED ON THE CURRENT VALUE OF      FNLE 217
C                  MARQUARDT'S LAMBDA.                                  FNLE 218
  110 CONTINUE                                                          FNLE 219
      DIAG = 1. + QL                                                    FNLE 220
      DO 130 J = 1,MC                                                   FNLE 221
         GAMMA(J,J) = DIAG                                              FNLE 222
         K = J - 1                                                      FNLE 223
  120    IF (K .EQ. 0) GOTO 130                                         FNLE 224
         GAMMA(J,K) = ALF(K,J)                                          FNLE 225
         GAMMA(K,J) = ALF(K,J)                                          FNLE 226
         K = K - 1                                                      FNLE 227
         GOTO 120                                                       FNLE 228
  130 CONTINUE                                                          FNLE 229
C                                                                       FNLE 230
C                  REPLACE GAMMA BY ITS INVERSE.                        FNLE 231
C                                                                       FNLE 232
      CALL MATINV (GAMMA,MC,PB,60,0,DOT)                                FNLE 233
C                                                                       FNLE 234
C                  FORM THE COMPONENTS OF DP OF THE SCALED DELTA P VECTOR. FNLE 235
C                  FORM THE NEW POINT PC.                               FNLE 236
C                  FORM DOT = THE DOT PRODUCT OF THE SCALED BATA AND    FNLE 237
C                           THE SCALED DELTA P VECTORS.                 FNLE 238
```

```
C                   FORM DPM = THE SQUARE OF THE MAGNITUDE OF THE          FNLE 239
C                              SCALED DELTA P VECTOR.                      FNLE 240
C                   FORM TR  = THE SQUARE OF THE COSINE OF THE ANGLE       FNLE 241
C                              BETWEEN BATA AND DELTA P.                   FNLE 242
C                                                                          FNLE 243
      DOT = 0.                                                             FNLE 244
      DPM = 0.                                                             FNLE 245
      DO 150 J = 1,MC                                                      FNLE 246
          DP = 0.                                                          FNLE 247
          DO 140 K = 1,MC                                                  FNLE 248
              DP = DP + BATA(K)*GAMMA(J,K)                                 FNLE 249
  140     CONTINUE                                                         FNLE 250
          PC(J) = PA(J) + DP*S(J)                                          FNLE 251
          DOT = DOT + DP*BATA(J)                                           FNLE 252
          DPM = DPM + DP*DP                                                FNLE 253
  150 CONTINUE                                                             FNLE 254
      TR = DOT**2/(DPM*BM)                                                 FNLE 255
C                                                                          FNLE 256
C                   EXPAND PC TO FULL SIZE AS PB, THE CANDIDATE            FNLE 257
C                   REPLACEMENT FOR INPUT POINT P.                         FNLE 258
C                                                                          FNLE 259
      K = 1                                                                FNLE 260
      DO 170 J = 1,LD                                                      FNLE 261
          IF (F(J) .EQ. 0.) GOTO 160                                       FNLE 262
          PB(J) = PC(K)                                                    FNLE 263
          K = K + 1                                                        FNLE 264
          GOTO 170                                                         FNLE 265
  160     PB(J) = P(J)                                                     FNLE 266
  170 CONTINUE                                                             FNLE 267
C                                                                          FNLE 268
C   ***  PART 4.    FOR THE CANDIDATE POINT P, OBTAIN THE ERROR MEASURE    FNLE 269
C                   EB (AND ASSOCIATED ARRAYS YC, R, RS, ALPHA AND BETA).FNLE 270
C                                                                          FNLE 271
  180 CONTINUE                                                             FNLE 272
      CALL LONDON (ROME,NF,M1,M2,M3,M7,M8,60,MR,NM,X0,X,Y,W,F,PB,          FNLE 273
     1             EB,MD,ALPHA,BETA,YC,R,RS,NS)                            FNLE 274
      IF (NS .EQ. 1) RETURN                                                FNLE 275
C                                                                          FNLE 276
C                   COMPARE NEW ERROR EB AT POINT PB WITH INPUT ERROR EA   FNLE 277
C                   AT POINT P.  IF EB IS NO LARGER (OR IF THE MARQUARDT   FNLE 278
C                   ALGORITHM IS NOT BEING USED) PROCEED TO PART 5.        FNLE 279
C                                                                          FNLE 280
      IF (EB .LE. EA) GOTO 210                                             FNLE 281
      IF (QL .EQ. 0.) GOTO 210                                             FNLE 282
C                                                                          FNLE 283
C                   IF THE ANGLE BETWEEN BATA AND DELTA P IS LESS THAN     FNLE 284
C                   45 DEGREES, OBTAIN A NEW POINT PB BY DECREASING THE    FNLE 285
C                   LENGTH OF DELTA P AND GO BACK TO PART 4.               FNLE 286
C                                                                          FNLE 287
      IF (TR .GE. .5) GOTO 200                                             FNLE 288
      DO 190 J = 1,LD                                                      FNLE 289
          PB(J) = P(J) + 0.1*(PB(J)-P(J))                                  FNLE 290
  190 CONTINUE                                                             FNLE 291
      GOTO 180                                                             FNLE 292
C                                                                          FNLE 293
C                   INCREASE MARQUARDT'S LAMBDA AND GO BACK TO PART 3.     FNLE 294
C                                                                          FNLE 295
  200 CONTINUE                                                             FNLE 296
      QL = 10.*QL                                                          FNLE 297
      GOTO 110                                                             FNLE 298
```

```
C                                                                       FNLE 299
C   ***  PART 5.    THE MARQUARDT ITERATIVE PROCESS HAS BEEN COMPLETED  FNLE 300
C                   SATISFACTORILY.  UPDATE ERROR MEASURES EPS AND SIG.  FNLE 301
C                   TEST FOR CONVERGENCE.  UPDATE POINT P AND COMPUTE    FNLE 302
C                   ERROR ESTIMATES EK.                                 FNLE 303
  210 CONTINUE                                                          FNLE 304
      RL = QL                                                           FNLE 305
      EPS = EB                                                          FNLE 306
      SIG = SQRT(EB/EM)                                                 FNLE 307
      CR = 1.0 - EB/EA                                                  FNLE 308
      IF (CR .GE. 0. .AND. CR .LT. 0.000010) NS = 2                     FNLE 309
      K = 1                                                             FNLE 310
      DO 220 J = 1,LD                                                   FNLE 311
          P(J) = PB(J)                                                  FNLE 312
          IF (F(J) .EQ. 0.) GOTO 215                                    FNLE 313
              EK(J) = SIG*S(K)*SQRT(GAMMA(K,K)*DIAG)                    FNLE 314
              K = K + 1                                                 FNLE 315
              GOTO 220                                                  FNLE 316
  215     EK(J) = 0.                                                    FNLE 317
  220 CONTINUE                                                          FNLE 318
      RETURN                                                            FNLE 319
      END                                                               FNLE 320
C                                                                       FNLE 321
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - 322
C                                                                       FNLE 323
      SUBROUTINE LONDON (ROME,NA,N1,N2,N3,N7,N8,NL,NR,NM,XA,X,Y,W,F,P,  FNLE 324
     1                   EPS,NC,ALPHA,BETA,YC,R,RS,IS)                  FNLE 325
C                                                                       FNLE 326
C   * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * FNLE 327
C   *                                                                   * FNLE 328
C   *    FOR A GIVEN SET OF PARAMETER AND I.C. VALUES AND A GIVEN       * FNLE 329
C   *    MULTI-ROUND SET OF MEASUREMENTS, THIS SUBROUTINE PRODUCES      * FNLE 330
C   *    THE ERROR MEASURE EPS, THE COMPUTED DEPENDENT VARIABLE         * FNLE 331
C   *    VALUES, THE RESIDUALS AND THE ALPHA AND BETA ARRAYS FOR THE    * FNLE 332
C   *    MULTI-ROUND DATA.  ALL ARGUMENTS ARE DEFINED IN THE COMMENTS   * FNLE 333
C   *    WITHIN SUBROUTINE DUBLIN.                                      * FNLE 334
C   *                                                                   * FNLE 335
C   * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * FNLE 336
C                                                                       FNLE 337
      DIMENSION NM(1),P(1),F(1),X(1),Y(N7,1),W(N7,1),                   FNLE 338
     1          YC(N8,1),R(N7,1),RS(1),ALPHA(NL,1),BETA(1)              FNLE 339
      DIMENSION RSQ(10)                                                 FNLE 340
C * THE ABOVE DIMENSION ASSUMES THAT                    N1 .LE. 10      FNLE 341
      DIMENSION C(20),CF(20)                                            FNLE 342
C * THE ABOVE DIMENSIONS ASSUME THAT                    N2 .LE. 20      FNLE 343
      DIMENSION CP(40),FP(40)                                           FNLE 344
C * THE ABOVE DIMENSIONS ASSUME THAT                    N3 .LE. 40      FNLE 345
      DIMENSION ALFA(3600),BATA(60)                                     FNLE 346
C * THE ABOVE DIMENSIONS ASSUME THAT              N2 + N3 .LE. 60      FNLE 347
      DIMENSION RE(200),WR(200),YM(200)                                 FNLE 348
C * THE ABOVE DIMENSIONS ASSUME THAT N1*(MAX. ELEMENT OF NM) .LE. 200  FNLE 349
      DIMENSION YCOMP(400)                                              FNLE 350
C * THE ABOVE DIMENSION ASSUMES THAT N2*(MAX. ELEMENT OF NM) .LE. 400  FNLE 351
      DIMENSION XX(1000)                                                FNLE 352
C * THE ABOVE DIMENSION ASSUMES THAT                                   FNLE 353
C                              NM(1) + NM(2) + ... + NM(NR) .LE. 1000   FNLE 354
      EXTERNAL ROME                                                     FNLE 355
C                                                                       FNLE 356
C  *** PART 1. *** PRELIMINARIES                                        FNLE 357
C                                                                       FNLE 358
```

```
      IS = 0                                                          FNLE 359
      MR = NR                                                         FNLE 360
      M1 = N1                                                         FNLE 361
      M2 = N2                                                         FNLE 362
      M3 = N3                                                         FNLE 363
      M23 = M2 + M3                                                   FNLE 364
      M6 = M2*MR                                                      FNLE 365
      LC = 1 + M6                                                     FNLE 366
      LD = M3 + M5                                                    FNLE 367
      JX = 1-(1+M23)*(LC-M2)                                          FNLE 368
      DO 220 N = 1,LD                                                 FNLE 369
        ALPHA(N,N) = 0.                                               FNLE 370
        BETA(N) = 0.                                                  FNLE 371
        IF (N .EQ. LD) GOTO 220                                       FNLE 372
        MA = N + 1                                                    FNLE 373
        DO 210 K = MA,LD                                              FNLE 374
          ALPHA(K,N) = 0.                                             FNLE 375
          ALPHA(N,K) = 0.                                             FNLE 376
  210   CONTINUE                                                      FNLE 377
  220 CONTINUE                                                        FNLE 378
      DO 230 I = 1,M1                                                 FNLE 379
        RS(I) = 0.                                                    FNLE 380
  230 CONTINUE                                                        FNLE 381
      MC = 0                                                          FNLE 382
      DO 240 K = 1,M3                                                 FNLE 383
        KA = K + M6                                                   FNLE 384
        CP(K) = P(KA)                                                 FNLE 385
        FP(K) = F(KA)                                                 FNLE 386
        IF (FP(K) .NE. 0.) MC = MC + 1                                FNLE 387
  240 CONTINUE                                                        FNLE 388
      JA = 0                                                          FNLE 389
      JB = 0                                                          FNLE 390
      EP = 0.                                                         FNLE 391
C                                                                     FNLE 392
C  *** PART 2. *** THE DO-LOOP FOR HANDLING MULTIPLE ROUNDS           FNLE 393
C                                                                     FNLE 394
      DO 370 JR = 1,MR                                                FNLE 395
        M4 = NM(JR)                                                   FNLE 396
        DO 260 M = 1,M4                                               FNLE 397
          IA = (M-1)*M1                                               FNLE 398
          JA = JA + 1                                                 FNLE 399
          XX(M) = X(JA)                                               FNLE 400
          DO 250 I = 1,M1                                             FNLE 401
            IM = IA + I                                               FNLE 402
            YM(IM) = Y(I,JA)                                          FNLE 403
            WR(IM) = W(I,JA)                                          FNLE 404
  250     CONTINUE                                                    FNLE 405
  260   CONTINUE                                                      FNLE 406
        LA = (JR-1)*M2                                                FNLE 407
        DO 270 K = 1,M2                                               FNLE 408
          LA = LA + 1                                                 FNLE 409
          C(K) = P(LA)                                                FNLE 410
          CF(K) = F(LA)                                               FNLE 411
          IF (CF(K) .NE. 0.) MC = MC + 1                              FNLE 412
  270   CONTINUE                                                      FNLE 413
        CALL PARIS (ROME,NA,1,M1,M2,M23,M4,C,CF,CP,FP,WR,XA,XX,YM,    FNLE 414
     1              YCOMP,RE,RSQ,ALFA,BATA,IR)                        FNLE 415
        IF (IR .EQ. 0) GOTO 280                                       FNLE 416
          IS = 1                                                      FNLE 417
          PRINT 275                                                   FNLE 418
```

```
275         FORMAT(1H ,10X,'UNSUCCESSFUL RETURN FROM SUBROUTINE PARIS.FNLE 419
   1'/1H ,10X,'ALL SUBROUTINE DUBLIN OUTPUTS INVALID.')              FNLE 420
            RETURN                                                   FNLE 421
280      CONTINUE                                                    FNLE 422
         DO 290 I = 1,M1                                             FNLE 423
            RS(I) = RS(I) + RSQ(I)                                   FNLE 424
            EP = EP + RSQ(I)                                         FNLE 425
290      CONTINUE                                                    FNLE 426
         DO 310 M = 1,M4                                             FNLE 427
            NI = (M-1)*M1                                            FNLE 428
            NJ = (M-1)*M2                                            FNLE 429
            JB = JB + 1                                              FNLE 430
            DO 300 J = 1,M2                                          FNLE 431
               IM = NI + J                                           FNLE 432
               JM = NJ + J                                           FNLE 433
               YC(J,JB) = YCOMP(JM)                                  FNLE 434
               IF (J .LE. M1) R(J,JB) = RE(IM)                       FNLE 435
300         CONTINUE                                                 FNLE 436
310      CONTINUE                                                    FNLE 437
         LA = 1 + (JR-1)*M2                                          FNLE 438
         LB = LA + M2 - 1                                            FNLE 439
         J = 1                                                       FNLE 440
         JJ = 1                                                      FNLE 441
         DO 340 N = LA,LB                                            FNLE 442
            K = N                                                    FNLE 443
            J = J + K - LA                                           FNLE 444
320         ALPHA(N,K) = ALFA(J)                                     FNLE 445
            IF (K .GT. N) ALPHA(K,N) = ALFA(J)                       FNLE 446
            J = J + 1                                                FNLE 447
            K = K + 1                                                FNLE 448
            IF (K .LE. LB) GOTO 320                                  FNLE 449
            K = LC                                                   FNLE 450
330         ALPHA(N,K) = ALFA(J)                                     FNLE 451
            ALPHA(K,N) = ALFA(J)                                     FNLE 452
            J = J + 1                                                FNLE 453
            K = K + 1                                                FNLE 454
            IF (K .LE. LD) GOTO 330                                  FNLE 455
            BETA(N) = BATA(JJ)                                       FNLE 456
            JJ = JJ + 1                                              FNLE 457
340      CONTINUE                                                    FNLE 458
         JJ = 1 + M2                                                 FNLE 459
         DO 360 N = LC,LD                                            FNLE 460
            K = N                                                    FNLE 461
            J = JX + (1 + M23)*N                                     FNLE 462
350         ALPHA(N,K) = ALPHA(N,K) + ALFA(J)                        FNLE 463
            IF (K .GT. N .AND. JR .EQ. MR) ALPHA(K,N) = ALPHA(N,K)   FNLE 464
            J = J + 1                                                FNLE 465
            K = K + 1                                                FNLE 466
            IF (K .LE. LD)  GOTO 350                                 FNLE 467
            BETA(N) = BETA(N) + BATA(JJ)                             FNLE 468
            JJ = JJ + 1                                              FNLE 469
360      CONTINUE                                                    FNLE 470
370   CONTINUE                                                       FNLE 471
      EPS = EP                                                       FNLE 472
      NC = MC                                                        FNLE 473
      RETURN                                                         FNLE 474
      END                                                            FNLE 475
C                                                                    FNLE 476
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - 477
C                                                                    FNLE 478
```

```
      SUBROUTINE PARIS(ROME,NA,NB,N1,N2,N23,N4,C,CF,P,PF,W,XA,X,YM,      FNLE 479
     1                 YC,R,RSQ,ALFA,BATA,IR)                           FNLE 480
C                                                                       FNLE 481
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *   FNLE 482
C     *                                                               *  FNLE 483
C     *    FOR A GIVEN SET OF PARAMETER AND IC ESTIMATES AND THE GIVEN*  FNLE 484
C     *    MEASUREMENTS FOR A SINGLE ROUND, THIS SUBROUTINE PRODUCES  *  FNLE 485
C     *    THE COMPUTED DEPENDENT VARIABLE VALUES, THE RESIDUALS AND  *  FNLE 486
C     *    (IF NB .NE. 0) THE ALPHA AND BETA ARRAYS FOR THE GIVEN ROUND.* FNLE 487
C     *                                                               *  FNLE 488
C     *    INPUT ARGUMENTS ...                                        *  FNLE 489
C     *     ROME = THE DUMMY NAME OF THE SUBROUTINE (WRITTEN BY THE   *  FNLE 490
C     *            USER) THAT DEFINES EITHER                          *  FNLE 491
C     *               (A) THE DEPENDENT VARIABLES (NA = 0)            *  FNLE 492
C     *             OR (B) THE DERIVATIVES OF THE DEPENDENT VARIABLES *  FNLE 493
C     *                    (NA .NE. 0)                                *  FNLE 494
C     *       NA = THE FLAG THAT INDICATES THE NATURE OF SUBROUTINE   *  FNLE 495
C     *            ROME (SEE PREVIOUS ARGUMENT)                       *  FNLE 496
C     *       NB = 0  IF ARGUMENTS ALFA AND BATA BELOW ARE NOT TO BE  *  FNLE 497
C     *               COMPUTED                                        *  FNLE 498
C     *          = 1  IF ALFA AND BATA ARE TO BE COMPUTED             *  FNLE 499
C     *       N1 = NUMBER OF MEASURED DEPENDENT VARIABLES (N1.GE.1)   *  FNLE 500
C     *       N2 = TOTAL NUMBER OF DEPENDENT VARIABLES (N2.GE.N1)     *  FNLE 501
C     *      N23 = TOTAL NUMBER OF PARAMETERS AND INITIAL CONDITIONS  *  FNLE 502
C     *            (N23.GE.N2)                                        *  FNLE 503
C     *       N4 = NUMBER OF MEASUREMENTS TAKEN ON EACH OF THE N1     *  FNLE 504
C     *            MEASURED VARIABLES                                 *  FNLE 505
C     *        C = VECTOR OF THE N2 INITIAL CONDITION ESTIMATES       *  FNLE 506
C     *       CF = VECTOR OF THE N2 INITIAL CONDITION FLAGS           *  FNLE 507
C     *            (0. IF THE INITIAL CONDITION VALUE IS FIXED)       *  FNLE 508
C     *        P = VECTOR OF THE N3 PARAMETER VALUES                  *  FNLE 509
C     *       PF = VECTOR OF THE N3 PARAMETER FLAGS                   *  FNLE 510
C     *            (0. IF THE PARAMETER VALUE IS FIXED)               *  FNLE 511
C     *        W = VECTOR OF THE N1 BY N4 WEIGHTS ASSOCIATED WITH INPUT*  FNLE 512
C     *            YM DEFINED BELOW                                   *  FNLE 513
C     *       XA = THE REFERENCE X VALUE AT WHICH INITIAL CONDITIONS   *  FNLE 514
C     *            APPLY                                              *  FNLE 515
C     *        X = VECTOR OF THE N4 VALUES OF THE INDEPENDENT VARIABLE*  FNLE 516
C     *            AT WHICH MEASUREMENTS WERE TAKEN                   *  FNLE 517
C     *       YM = VECTOR OF THE N1 BY N4 MEASURED Y VALUES FOR ONE   *  FNLE 518
C     *            ROUND, WHERE                                       *  FNLE 519
C     *               YM(IM) = MEASURED VALUE OF Y(I) AT X(M)         *  FNLE 520
C     *                        IM = I + (M-1)*N1                      *  FNLE 521
C     *                         I = 1,2, ... N1                       *  FNLE 522
C     *                         M = 1,2, ... N4                       *  FNLE 523
C     *                                                               *  FNLE 524
C     *    OUTPUT ARGUMENTS ...                                       *  FNLE 525
C     *       YC = VECTOR OF N2 BY N4 COMPUTED Y VALUES FOR ONE ROUND,*  FNLE 526
C     *            WHERE                                              *  FNLE 527
C     *               YC(JM) = COMPUTED VALUE OF Y(J) AT X(M)         *  FNLE 528
C     *                        JM = J + (M-1)*N2                      *  FNLE 529
C     *                         J = 1,2, ... N2                       *  FNLE 530
C     *        R = VECTOR OF N1 BY N4 RESIDUALS, WHERE                *  FNLE 531
C     *               R(IM) = YM(IM) - YC(IM2),                       *  FNLE 532
C     *                       IM2 = I + (M-1)*N2                      *  FNLE 533
C     *      RSQ = VECTOR OF N1 ERROR MEASURES, WHERE                 *  FNLE 534
C     *            RSQ(I) = WEIGHTED SUM OF THE SQUARES OF THE RESIDUALS* FNLE 535
C     *                     IN THE I-TH DEPENDENT VARIABLE            *  FNLE 536
C     *     ALFA = VECTOR OF N23 BY N23 ALPHA VALUES, WHERE           *  FNLE 537
C     *             ALFA(NK) = ALPHA(N,K)                             *  FNLE 538
```

```
C     *                    NK = N + (K-1)*N23                                 * FNLE 539
C     *      BATA = VECTOR OF N23 BETA VALUES                                 * FNLE 540
C     *        IR = 1 IF A WARNING WAS PRINTED AND A RETURN EXECUTED          * FNLE 541
C     *             BEFORE OBTAINING THE ABOVE OUTPUT                         * FNLE 542
C     *           = 0 OTHERWISE                                               * FNLE 543
C     *                                                                       * FNLE 544
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * FNLE 545
C                                                                             FNLE 546
      DIMENSION C(1),CF(1),P(1),PF(1),X(1),YM(1),YC(1),R(1),RSQ(1),           FNLE 547
     1   ALFA(1),BATA(1),W(1)                                                 FNLE 548
      DIMENSION U(400),DU(400),S(10000)                                       FNLE 549
C                                                                             FNLE 550
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * FNLE 551
C     *                                                                       * FNLE 552
C     *    WARNING ... THE DIMENSIONS OF U AND DU ABOVE MUST EQUAL            * FNLE 553
C     *                OR EXCEED N5 = N2*(1+N23). THE NUMBER OF EQUATIONS     * FNLE 554
C     *                IN SUBROUTINE ROME.  THE DIMENSION OF ARRAY S          * FNLE 555
C     *                ABOVE MUST EQUAL OR EXCEED N4*N5.   FOR EACH VALUE     * FNLE 556
C     *                OF THE INDEPENDENT VARIABLE X.                         * FNLE 557
C     *                                                                       * FNLE 558
C     *          U SUBSCRIPTS...       DENOTE THE COMPUTED VALUES OF ...    * FNLE 559
C     *                                                                       * FNLE 560
C     *         1,2,... N2           THE N2 DEPENDENT VARIABLES              * FNLE 561
C     *        N2+1,... N5           THE N2*N23 PARTIAL DERIVATIVES, WHERE   * FNLE 562
C     *                               PARTIAL(J,K) = U(J+K*N2)                * FNLE 563
C     *                                                                       * FNLE 564
C     *                ARRAY DU DENOTES DU/DX.  ARRAY S IS A COMPILATION      * FNLE 565
C     *                OF ALL THE U VALUES.  THAT IS,                         * FNLE 566
C     *                    S(1),...S(N5)  = U(1),...U(N5) AT X(1)             * FNLE 567
C     *                S(N5+1),...S(2*N5) = U(1),...U(N5) AT X(2), ETC.       * FNLE 568
C     *                                                                       * FNLE 569
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * FNLE 570
C                                                                             FNLE 571
      COMMON /NAPLES/ PAR(40),FLAG(60)                                        FNLE 572
C                                                                             FNLE 573
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * FNLE 574
C     *                                                                       * FNLE 575
C     *    THE ABOVE LABELLED COMMON MUST BE LINKED TO AND USED IN THE        * FNLE 576
C     *    USER'S SUBROUTINE ROME. THE VALUES ARE ESTABLISHED HERE IN         * FNLE 577
C     *    SUBROUTINE PARIS.                                                  * FNLE 578
C     *      PAR = THE ARRAY OF N3 PARAMETER VALUES                           * FNLE 579
C     *     FLAG = THE ARRAY OF N23 FLAGS FOR A GIVEN ROUND, WHERE            * FNLE 580
C     *            FLAG(K) = K-TH INITIAL CONDITION FLAG, K = 1,2,..N2        * FNLE 581
C     *            FLAG(K+N2) = K-TH PARAMETER FLAG, K = 1,2,..N3             * FNLE 582
C     *                                                                       * FNLE 583
C     * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * FNLE 584
C                                                                             FNLE 585
      EXTERNAL ROME                                                           FNLE 586
      DATA HMIN,Q,DELX/-1.0, .00010, .125/                                    FNLE 587
C                                                                             FNLE 588
C *** PART (1)   PRELIMINARIES                                                FNLE 589
C                                                                             FNLE 590
      IR = 0                                                                  FNLE 591
      M23 = N23                                                               FNLE 592
      M1 = N1                                                                 FNLE 593
      M2 = N2                                                                 FNLE 594
      M4 = N4                                                                 FNLE 595
      M5 = M2*(1+M23)                                                         FNLE 596
      IF (M5 .LE. 400) GOTO 24                                                FNLE 597
          IR = 1                                                              FNLE 598
```

```
      PRINT 10                                                          FNLE 599
10    FORMAT(1H0,' SUBROUTINE PARIS WARNING ... ')                      FNLE 600
      PRINT 22,M5                                                       FNLE 601
22    FORMAT(1H ,10X,'INCREASE DIMENSIONS OF U AND DU (AND IN SUBROUFNLE 602
     1TINE MERSO,  INCREASE DIMENSIONS OF T, G AND S) TO ',I5)          FNLE 603
24    M6 = M4*M5                                                        FNLE 604
      IF (M6 .LE. 10000) GOTO 28                                        FNLE 605
      IR = 1                                                            FNLE 606
      PRINT 10                                                          FNLE 607
      PRINT 26,M6                                                       FNLE 608
26    FORMAT(1H ,10X,'INCREASE DIMENSION OF S TO ',I5)                  FNLE 609
28 CONTINUE                                                             FNLE 610
      IF (IR .EQ. 1) RETURN                                             FNLE 611
      IM = 0                                                            FNLE 612
      H = DELX*(X(M4)-X(1))/FLOAT(M4-1)                                 FNLE 613
      JA = M2 + 1                                                       FNLE 614
      DO 30 JB = JA,M5                                                  FNLE 615
      U(JB) = 0.0                                                       FNLE 616
      DU(JB) = 0.0                                                      FNLE 617
30 CONTINUE                                                             FNLE 618
      DO 32 K = 1,M2                                                    FNLE 619
      FLAG(K) = CF(K)                                                   FNLE 620
32 CONTINUE                                                             FNLE 621
      M3 = M23 - M2                                                     FNLE 622
      DO 34 K = 1,M3                                                    FNLE 623
      PAR(K) = P(K)                                                     FNLE 624
      KA = K + M2                                                       FNLE 625
      FLAG(KA) = PF(K)                                                  FNLE 626
34 CONTINUE                                                             FNLE 627
C                                                                       FNLE 628
C *** PART (2)   DETERMINE KL = NO. OF POINTS IN X LESS THAN XA         FNLE 629
C                      AND KR = NO. OF POINTS IN X GREATER THAN XA.     FNLE 630
C                IF XA COINCIDES WITH A POINT IN X, THEN THE 'COMPUTED' FNLE 631
C                Y VALUES AT THAT POINT ARE THE INITIAL CONDITIONS FROM FNLE 632
C                SUBROUTINE BONN.                                       FNLE 633
C                                                                       FNLE 634
      DO 40 M = 1,M4                                                    FNLE 635
      IF (XA-X(M)) 70,50,40                                             FNLE 636
40 CONTINUE                                                             FNLE 637
      KL = M4                                                           FNLE 638
      KR = 0                                                            FNLE 639
      GOTO 80                                                           FNLE 640
50    KL = M - 1                                                        FNLE 641
      KR = M4 - M                                                       FNLE 642
      IM = 1                                                            FNLE 643
      CALL BONN(M2,M5,C,U)                                              FNLE 644
      LB = KL*M5                                                        FNLE 645
      DO 60 L = 1,M5                                                    FNLE 646
      LB = LB + 1                                                       FNLE 647
      S(LB) = U(L)                                                      FNLE 648
60 CONTINUE                                                             FNLE 649
      GOTO 80                                                           FNLE 650
70    KL = M - 1                                                        FNLE 651
      KR = M4 - KL                                                      FNLE 652
C                                                                       FNLE 653
C *** PART (3)   FOR EACH POINT IN X, SOLVE THE SYSTEM OF EQUATIONS     FNLE 654
C                DEFINED IN SUBROUTINE ROME TO OBTAIN COMPUTED Y VALUES.FNLE 655
C                THESE Y VALUES ARE STORED IN S.                        FNLE 656
80 CONTINUE                                                             FNLE 657
      IF (KL .EQ. 0) GOTO 90                                            FNLE 658
```

```
      IA = -1                                                          FNLE 659
      IB = 0                                                           FNLE 660
      LK = KL                                                          FNLE 661
      GOTO 100                                                         FNLE 662
   90 IA = 1                                                           FNLE 663
      IB = M4 + 1                                                      FNLE 664
      LK = IB - KR                                                     FNLE 665
  100 X1 = XA                                                          FNLE 666
      IF (IM .EQ. 0) CALL BONN(M2,M5,C,U)                              FNLE 667
      IM = 0                                                           FNLE 668
  110   = X(LK)                                                        FNLE 669
      IF (NA .NE. 0) GOTO 114                                          FNLE 670
        CALL ROME(C,X1,X2,U)                                           FNLE 671
        GOTO 116                                                       FNLE 672
  114 CONTINUE                                                         FNLE 673
      HA = DELX*ABS(X2-X1)                                             FNLE 674
      HB = ABS(H)                                                      FNLE 675
      H = AMIN1(HA,HB)                                                 FNLE 676
      CALL MERSO  (ROME,M5,X1,X2,U,DU,H,HMIN,Q)                        FNLE 677
  116 CONTINUE                                                         FNLE 678
      LB = (LK-1)*M5                                                   FNLE 679
      DO 120 L = 1,M5                                                  FNLE 680
        LB = LB + 1                                                    FNLE 681
        S(LB) = U(L)                                                   FNLE 682
  120 CONTINUE                                                         FNLE 683
      LK = LK + IA                                                     FNLE 684
      IF (LK .NE. IB) GOTO 110                                         FNLE 685
      IF (IA .EQ.  1) GOTO 130                                         FNLE 686
      IF (KR .NE.  0) GOTO 90                                          FNLE 687
C                                                                      FNLE 688
C *** PART (4)   CONVERT VECTOR S TO VECTOR YC AND FORM THE RESIDUAL    FNLE 689
C                VECTOR R.                                             FNLE 690
  130 CONTINUE                                                         FNLE 691
      DO 150 M = 1,M4                                                  FNLE 692
        NR = (M-1)*M1                                                  FNLE 693
        NY = (M-1)*M2                                                  FNLE 694
        LA = (M-1)*M5                                                  FNLE 695
        DO 140 J = 1,M2                                                FNLE 696
          IM = NR + J                                                  FNLE 697
          JM = NY + J                                                  FNLE 698
          LB = LA + J                                                  FNLE 699
          YC(JM) = S(LB)                                               FNLE 700
          IF (J .LE. M1 .AND. W(IM) .NE. 0.) R(IM) = YM(IM) - YC(JM)   FNLE 701
          IF (J .LE. M1 .AND. W(IM) .EQ. 0.) R(IM) = 0.                FNLE 702
  140   CONTINUE                                                       FNLE 703
  150 CONTINUE                                                         FNLE 704
C                                                                      FNLE 705
C *** PART (5)   COMPUTE VECTOR RSQ                                    FNLE 706
C                                                                      FNLE 707
      DO 170 I = 1,M1                                                  FNLE 708
        RM = 0.0                                                       FNLE 709
        DO 160 M = 1,M4                                                FNLE 710
          IM = I + (M-1)*M1                                            FNLE 711
          RM = RM + W(IM)*R(IM)**2                                     FNLE 712
  160   CONTINUE                                                       FNLE 713
        RSQ(I) = RM                                                    FNLE 714
  170 CONTINUE                                                         FNLE 715
      IF (NB .EQ. 0) RETURN                                            FNLE 716
C                                                                      FNLE 717
C *** PART (6)   COMPUTE VECTOR ALFA                                   FNLE 718
```

```
C                                                                       FNLE 719
      DO 210 K = 1,M23                                                  FNLE 720
          LA = K*M2                                                     FNLE 721
          NL = (K-1)*M23                                                FNLE 722
          DO 200 N = K,M23                                              FNLE 723
              LB = N*M2                                                 FNLE 724
              NK = NL + N                                               FNLE 725
              ALFA(NK) = 0.0                                            FNLE 726
              DO 190 I = 1,M1                                           FNLE 727
                  ALF = 0.0                                             FNLE 728
                  LC = LA + I                                           FNLE 729
                  LD = LB + I                                           FNLE 730
                  DO 180 M = 1,M4                                       FNLE 731
                      IM = I + (M-1)*M1                                 FNLE 732
                      ALF = ALF + W(IM)*S(LC)*S(LD)                     FNLE 733
                      LC = LC + M5                                      FNLE 734
                      LD = LD + M5                                      FNLE 735
  180             CONTINUE                                              FNLE 736
            ALFA(NK) = ALFA(NK) + ALF                                   FNLE 737
  190         CONTINUE                                                  FNLE 738
              IF (N .EQ. K) GOTO 200                                    FNLE 739
                  KN = K + (N-1)*M23                                    FNLE 740
                  ALFA(KN) = ALFA(NK)                                   FNLE 741
  200     CONTINUE                                                      FNLE 742
  210 CONTINUE                                                          FNLE 743
C                                                                       FNLE 744
C *** PART (7)   COMPUTE VECTOR BATA                                    FNLE 745
C                                                                       FNLE 746
      DO 240 N = 1,M23                                                  FNLE 747
          LA = N*M2                                                     FNLE 748
          BATA(N) = 0.0                                                 FNLE 749
          DO 230 I = 1,M1                                               FNLE 750
              BAT = 0.0                                                 FNLE 751
              LB = LA + I                                               FNLE 752
              DO 220 M = 1,M4                                           FNLE 753
                  IM = I + (M-1)*M1                                     FNLE 754
                  BAT = BAT + W(IM)*R(IM)*S(LB)                         FNLE 755
                  LB = LB + M5                                          FNLE 756
  220         CONTINUE                                                  FNLE 757
          BATA(N) = BATA(N) + BAT                                       FNLE 758
  230     CONTINUE                                                      FNLE 759
  240 CONTINUE                                                          FNLE 760
      RETURN                                                            FNLE 761
      END                                                               FNLE 762
C                                                                       FNLE 763
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -  - - 764
C                                                                       FNLE 765
      SUBROUTINE RONN(N2,N5,C,U)                                        FNLE 766
C                                                                       FNLE 767
C   * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * FNLE 768
C   *                                                                   * FNLE 769
C   *    THIS SUBROUTINE (CALLED BY SUBROUTINE PARIS) ASSIGNS INITIAL   * FNLE 770
C   *    CONDITIONS (THE VALUES AT X = XA) TO VECTOR U.  FOR THE        * FNLE 771
C   *    DEFINITIONS OF THE ARGUMENTS, SEE THE COMMENTS IN SUBROUTINE   * FNLE 772
C   *    PARIS.                                                         * FNLE 773
C   *                                                                   * FNLE 774
C   * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * FNLE 775
C                                                                       FNLE 776
      DIMENSION C(N2),U(N5)                                             FNLE 777
      M2 = N2                                                           FNLE 778
```

```
      M5 = N5                                                                   FNLE 779
      DO 10 J = 1,M2                                                            FNLE 780
          U(J) = C(J)                                                           FNLE 781
   10 CONTINUE                                                                  FNLE 782
      LA = 0                                                                    FNLE 783
      DO 30 JA = 1,M2                                                           FNLE 784
          LA = LA + M2                                                          FNLE 785
          DO 20 JB = 1,M2                                                       FNLE 786
              LB = LA + JB                                                      FNLE 787
              IF (JA .EQ. JB) U(LB) = 1.0                                       FNLE 788
              IF (JA .NE. JB) U(LB) = 0.0                                       FNLE 789
   20     CONTINUE                                                             FNLE 790
   30 CONTINUE                                                                  FNLE 791
      IF (LB .GE. M5) GOTO 50                                                   FNLE 792
      LA = LB + 1                                                               FNLE 793
      DO 40 LB = LA,M5                                                          FNLE 794
          U(LB) = 0.0                                                           FNLE 795
   40 CONTINUE                                                                  FNLE 796
   50 CONTINUE                                                                  FNLE 797
      RETURN                                                                    FNLE 798
      END                                                                       FNLE 799
C                                                                               FNLE 800
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -   801
C                                                                               FNLE 802
      SUBROUTINE MERSO (FUNC,N,X,Z,Y,F,H,HMIN,E)                                FNLE 803
      DIMENSION Y(1),F(1) $ DIMENSION T(400),G(400),S(400)                      FNLE 804
      LOGICAL BC,BE,BH,BR,BX $ NT=N$ ZT=Z$ HMI=HMIN$ ET=ABS(E)                  FNLE 805
      IF(HMI.LT.0.0)HMI=0.01*ABS(H) $ BH=BR=BX=.TRUE.                           FNLE 806
      BC=0.0.LT.ET.AND.ET.LT.1.0 $ E5=ET*5.0                                    FNLE 807
      IF((ZT.GT.X.AND.H.LT.0.0).OR.(ZT.LT.X.AND.H.GT.0.0))H=-H                  FNLE 808
      IF(NT.LE.400)GOTO100$ PRINT 1,NT$ STOP                                    FNLE 809
    1 FORMAT(22H RUN ERROR, MERSON, N=,I10)                                     FNLE 810
  100 XS=X $ DO 110 J=1,NT $ G(J)=Y(J)                                          FNLE 811
  110 CONTINUE                                                                  FNLE 812
  200 HS=H$ Q=X+H-ZT$ BE=.TRUE.                                                 FNLE 813
      IF((Q.LT.0.0.AND.H.GE.0.0).OR.(Q.GT.0.0.AND.H.LE.0.0)) GO TO 210          FNLE 814
      H=ZT-X$ BR=.FALSE.                                                        FNLE 815
  210 H3=H/3.0 $ DO 510 ISW=1,5 $ CALL FUNC(NT,X,Y,F) $ DO 450 I=1,NT           FNLE 816
      Q=H3*F(I)$ GOTO(301,302,303,304,305),ISW                                  FNLE 817
  301 T(I)=R=Q$ GOTO 400                                                        FNLE 818
  302 R=0.5*(Q+T(I))$ GOTO 400                                                  FNLE 819
  303 S(I)=R=3.0*Q$ R=0.375*(R+T(I))$ GO TO 400                                 FNLE 820
  304 T(I)=R=T(I)+4.0*Q$ R=1.5*(R-S(I))$ GO TO 400                              FNLE 821
  305 R=0.5*(Q+T(I))$ Q=ABS(2.0*R-1.5*(Q+S(I)))                                 FNLE 822
  400 Y(I)=G(I)+R $ IF(ISW.NE.5) GO TO 450 $ IF(.NOT.BC)GOTO 450 $ R=E5         FNLE 823
      IF(ABS(Y(I)).GE.0.001)R=R*ABS(Y(I))                                       FNLE 824
      IF((Q.LT.R).OR..NOT.BX)GOTO 420 $ BR=.TRUE.$ BH=.FALSE.$ H=0.5*H          FNLE 825
      IF(ABS(H).GE.HMI)GOTO 410 $ H=SIGN1(H)*HMI$ BX=.FALSE.                    FNLE 826
  410 DO 411 J=1,NT $ Y(J)=G(J)                                                 FNLE 827
  411 CONTINUE $ X=XS$ GOTO 200                                                 FNLE 828
  420 IF(Q.GE.0.03125*R)BE=.FALSE.                                              FNLE 829
  450 CONTINUE $ GOTO(501,510,503,504,510),ISW                                  FNLE 830
  501 X=X+H3$ GOTO 510                                                          FNLE 831
  503 X=X+0.5*H3$ GOTO 510                                                      FNLE 832
  504 X=X+0.5*H                                                                 FNLE 833
  510 CONTINUE $ IF(.NOT.BC) GO TO 521                                          FNLE 834
      IF(.NOT.(BE.AND.BH.AND.BR)) GO TO 520 $ H=2.0*H $ BX=.TRUE.               FNLE 835
  520 BH=.TRUE.                                                                 FNLE 836
  521 IF(BR) GO TO 100 $ H=HS$ RETURN $ END                                     FNLE 837
```

SUBROUTINE MATINV

OBTAINED FROM COMPUTER SUPPORT DIVISION
ABERDEEN RESEARCH AND DEVELOPMENT CENTER

```
      SUBROUTINE MATINV(A,N,C,NMAX,K,DET)                          *****  1
      DIMENSION A(NMAX,1),C(1)                                     MATINV 2
      NN = N                                                       MATINV 3
      KK = K                                                       MATINV 4
      IF (1-KK) 3,1,1                                              MATINV 5
    1 N3 = NN                                                      MATINV 6
      IF (KK) 2,4,2                                                MATINV 7
    2 ASSIGN  9 TO N5                                              MATINV 8
      ASSIGN 13 TO N7                                              MATINV 9
      GOTO 5                                                       MATINV10
    3 N3 = KK + NN - 1                                             MATINV11
    4 ASSIGN 10 TO N5                                              MATINV12
      ASSIGN 14 TO N7                                              MATINV13
    5 DET = 1.0                                                    MATINV14
      DO 15 I = 1,NN                                               MATINV15
         IF (A(I,I)) 7,6,7                                         MATINV16
    6    WRITE(6,17)                                               MATINV17
         DET = 0.0                                                 MATINV18
         GOTO 16                                                   MATINV19
    7    T1 = 1.0/A(I,I)                                           MATINV20
         DET = DET*A(I,I)                                          MATINV21
         A(I,I) = 1.0                                              MATINV22
         DO 8 J = 1,N3                                             MATINV23
            A(I,J) = A(I,J)*T1                                     MATINV24
    8    CONTINUE                                                  MATINV25
         GOTO N5, (9,10)                                           MATINV26
    9    C(I) = C(I)*T1                                            MATINV27
   10    DO 14 J = 1,NN                                            MATINV28
            IF (I-J) 11,14,11                                      MATINV29
   11       T1 = A(J,I)                                           MATINV30
            A(J,I) = 0.0                                           MATINV31
            DO 12 L = 1,N3                                         MATINV32
               A(J,L) = A(J,L) - T1*A(I,L)                         MATINV33
   12       CONTINUE                                              MATINV34
            GOTO N7, (13,14)                                       MATINV35
   13       C(J) = C(J) - T1*C(I)                                 MATINV36
   14    CONTINUE                                                  MATINV37
   15 CONTINUE                                                     MATINV38
   16 RETURN                                                       MATINV39
   17 FORMAT (16H SINGULAR MATRIX)                                 MATINV40
      END                                                          MATINV41
```

DISTRIBUTION LIST

No. of Copies	Organization
12	Commander Defense Technical Info Center ATTN: DDC-DDA Cameron Station Alexandria, VA 22314
1	Commander US Army Materiel Development and Readiness Command ATTN: DRCDMD-ST 5001 Eisenhower Avenue Alexandria, VA 22333
2	Commander US Army Armament Research and Development Command ATTN: DRDAR-TSS (2 cys) Dover, NJ 07801
4	Commander US Army Armament Research and Development Command ATTN: DRDAR-LCA, Mr. W. R. Benson DRDAR-LCA-F, Mr. A. Loeb DRDAR-LCA-FA, Mr. E. Friedman Mr. D. Mertz Dover, NJ 07801
2	Commander US Army Armament Research and Development Command ATTN: DRDAR-LCN, Mr. F. Saxe Mr. F. Scerbo Dover, NJ 07801
1	Commander US Army Armament Research and Development Command ATTN: DRDAR-LCU, Mr. A. Moss Dover, NJ 07801
1	Commander US Army Armament Materiel Readiness Command ATTN: DRSAR-LEP-L, Tech Lib Rock Island, IL 61299
1	Director US Army Armament Research and Development Command Benet Weapons Laboratory ATTN: DRDAR-LCB-TL Watervliet, NY 12189
1	Commander US Army Aviation Research and Development Command ATTN: DRSAV-E P.O. Box 209 St. Louis, MO 61366
1	Director US Army Air Mobility Research and Development Laboratory Ames Research Center Moffett Field, CA 94035
1	Commander US Army Communications Research and Development Command ATTN: DRDCO-PPA-SA Fort Monmouth, NJ 07703
1	Commander US Army Electronics Research and Development Command Technical Support Activity ATTN: DELSD-L Fort Monmouth, NJ 07703
1	Commander US Army Missile Command ATTN: DRSMI-R Redstone Arsenal, AL 35809

DISTRIBUTION LIST (continued)

No. of Copies	Organization
1	Commander US Army Missile Command ATTN: DRSMI-YDL Redstone Arsenal, AL 35809
1	Commander US Army Tank Automotive Research and Development Command ATTN: DRDTA-UL Warren, MI 48090
1	Director US Army TRADOC Systems Analysis Activity ATTN: ATAA-SL, Tech Lib White Sands Missile Range NM 88002
1	Commander US Army Yuma Proving Ground ATTN: STEYP-TMW, Mr. W. T. Vomocil Yuma, AZ 85364
1	Commander US Army Research Office ATTN: CRD-AA-EH P.O. Box 12211 Research Triangle Park NC 27709
1	AFATL (Tech Lib) Eglin AFB, FL 32542
1	AFFDL Wright-Patterson AFB, OH 45433
4	Director National Aeronautics and Space Administration Ames Research Center ATTN: Dr. Gary Chapman Mr. A. Seiff Mr. Murray Tobak Tech Lib Moffett Field, CA 94035
2	Director Jet Propulsion Laboratory ATTN: Tech Lib Mr. Peter Joffe 4800 Oak Grove Drive Pasadena, CA 91103
1	Commander David W. Taylor Naval Ship Research and Development Center ATTN: Tech Lib Bethesda, MD 20084
1	Commander Naval Research Laboratory ATTN: Tech Info Div Washington, D.C. 20375
5	Commander Naval Surface Weapons Center ATTN: Dr. Thomas Clare Dr. W.R. Chadwick Dr. W.G. Soper Dr. F. Moore Dr. T.R. Pepitone Dahlgren, VA 22448
1	Commander Naval Surface Weapons Center ATTN: Code 730, Tech Lib Silver Spring, MD 20910
1	Commander Naval Weapons Center ATTN: Code 233 China Lake, CA 93555
1	Arnold Research Organization, Inc. Project Support and Special Studies Section Aerodynamics Division Projects Branch ATTN: Dr. John C. Adams, Jr. Arnold AFS, TN 37389

DISTRIBUTION LIST (continued)

No. of Copies	Organization
2	Director Sandia Laboratories ATTN: Division 1331, Mr. H.R. Vaughn Mr. A.E. Hodapp, Jr. Albuquerque, NM 87115
1	Director National Aeronautics and Space Administration George C. Marshall Space Flight Center ATTN: MS-I, Library Huntsville, AL 35812
2	Director National Aeronautics and Space Administration Langley Research Center ATTN: MS-185, Tech Lib Dr. Clarence Young Langley Station Hampton, VA 23365
1	The Analytic Sciences Corporation (TASC) ATTN: Mr. James E. Kain 6 Jacob Way Reading, MA 01867
1	E.I. du Pont de Nemours and Company Engineering Department ATTN: Dr. Donald W. Marquardt Wilmington, DE 19898
1	Exxon Research Center ATTN: Mr. John Stevens Building 28 P.O. Box 45 Linden, NJ 07036
1	General Electric Company Armament Systems Department ATTN: Mr. Robert H. Whyte Lakeside Avenue Burlington, VT 05401
1	Oceanics, Inc. ATTN: Dr. Theodore R. Goodman Technical Industrial Park Plainview, NY 11803
1	Schering Corporation ATTN: M. Miller 60 Orange Street Bloomfield, NJ 07003
1	Systems Technology, Inc. ATTN: Arlene Muise, Librarian 13766 South Hawthorne Blvd. Hawthorne, CA 90250
1	Union Research Center ATTN: Dr. M. A. Selim P.O. Box 76 Brea, CA 92621
1	Case Western Reserve University ATTN: Professor Philip R Bevington Cleveland, OH 44106
1	Oklahoma State University Computing and Information Sciences ATTN: J. P. Chandler Stillwater, OK 74074
1	University of Virginia Department of Engineering Science ATTN: Professor Ira Jacobson Thornton Hall Charlottesville, VA 22904
	Aberdeen Proving Ground Dir, USAMSAA ATTN: DRXSY-D DRXSY-MP, H. Cohen Cdr, USATECOM ATTN: DRSTE-TO-F Dir, USACSL, Bldg. E3516, EA ATTN: DRDAR-CLB-PA

USER EVALUATION OF REPORT

Please take a few minutes to answer the questions below; tear out this sheet, fold as indicated, staple or tape closed, and place in the mail. Your comments will provide us with information for improving future reports.

1. BRL Report Number _____

2. Does this report satisfy a need? (Comment on purpose, related project, or other area of interest for which report will be used.)

_____

_____

_____

3. How, specifically, is the report being used? (Information source, design data or procedure, management procedure, source of ideas, etc.) _____

_____

_____

4. Has the information in this report led to any quantitative savings as far as man-hours/contract dollars saved, operating costs avoided, efficiencies achieved, etc.? If so, please elaborate.

_____

_____

5. General Comments (Indicate what you think should be changed to make this report and future reports of this type more responsive to your needs, more usable, improve readability, etc.) _____

_____

_____

_____

6. If you would like to be contacted by the personnel who prepared this report to raise specific questions or discuss the topic, please fill in the following information.

Name: _____

Telephone Number: _____

Organization Address: _____

_____

_____