BRL MR 2184

BRL

MEMORANDUM REPORT NO. 2184

CHLOE: A FORTRAN SUBROUTINE FOR FITTING ORDINARY DIFFERENTIAL EQUATIONS TO OBSERVED DATA

by

James W. Bradley

April 1972

U.S. ARMY ABERDEEN RESEARCH AND DEVELOPMENT CENTER
BALLISTIC RESEARCH LABORATORIES
ABERDEEN PROVING GROUND, MARYLAND

43

Security Classification

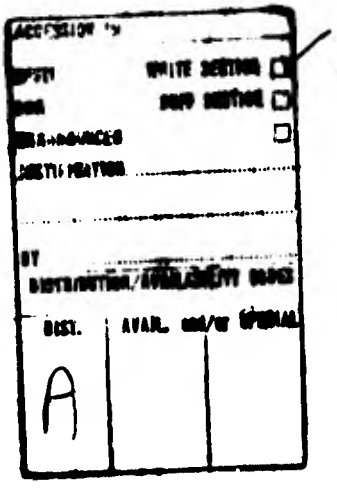
14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
FORTRAN Least Squares Ordinary Differential Equations Curve Fitting						

Security Classification

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 purchased from the U.S. Department of Commerce, National Technical Information Service, Springfield, Virginia 22151



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.

UNCLASSIFIED
Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) U. S. Army Aberdeen Research and Development Center Ballistic Research Laboratories Aberdeen Proving Ground, Maryland 21005	2a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED 2b. GROUP

3. REPORT TITLE

CHLOE: A FORTRAN Subroutine for Fitting Ordinary Differential Equations to Observed Data.

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)

5. AUTHOR(S) (First name, middle initial, last name)

James W. Bradley

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
April 1972	49	5

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
b. PROJECT NO. RDT&E 1T06110A2A33D	Memorandum Report No. 2184
c.	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)
d.	

10. DISTRIBUTION STATEMENT

Approved for public release; distribution unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
	U. S. Army Materiel Command Washington, D. C. 20315

13. ABSTRACT

This paper presents and documents a FORTRAN subroutine for applying Goodman's method to any given system of ordinary differential equations. The method determines those values of the parameters and initial conditions of the system that best fit the solution curves to observed data. No a priori knowledge or assumptions regarding the form of the solution are required.

DD FORM 1 NOV 68 1473 REPLACES DD FORM 1473, 1 JAN 64, WHICH IS OBSOLETE FOR ARMY USE.

UNCLASSIFIED
Security Classification

B A L L I S T I C R E S E A R C H L A B O R A T O R I E S

MEMORANDUM REPORT NO. 2184

APRIL 1972

CHLOE: A FORTRAN SUBROUTINE FOR FITTING ORDINARY DIFFERENTIAL EQUATIONS TO OBSERVED DATA

James W. Bradley

Exterior Ballistics Laboratory

RDT&E Project No. 1T061102A33D

A B E R D E E N P R O V I N G G R O U N D, M A R Y L A N D

B A L L I S T I C R E S E A R C H L A B O R A T O R I E S

MEMORANDUM REPORT NO. 2184

JWBradley/esb
Aberdeen Proving Ground, Md.
April 1972

CHLOE: A FORTRAN SUBROUTINE FOR FITTING ORDINARY DIFFERENTIAL EQUATIONS TO OBSERVED DATA

ABSTRACT

This paper presents and documents a FORTRAN subroutine for applying Goodman's method to any given system of ordinary differential equations. The method determines those values of the parameters and initial conditions of the system that best fit the solution curves to observed data. No a priori knowledge or assumptions regarding the form of the solution are required.

TABLE OF CONTENTS

TABLE OF SYMBOLS

$A = (A_{im})$, the measurement matrix; a CHLOE input argument

A_{im} = the measured value of y_i at x_m

ale = arithmetic or logical expression

B = a known constant in the drag equation, Eq. (9)

$C = (c_1, \ldots c_{N23})$; the vector of constants (initial conditions and parameters) to be determined; a CHLOE I/O argument

$C_D = C_{DO} + C_{D2}\,\delta^2$ = drag coefficient, Eq. (9)

C_{DO} = zero-yaw drag coefficient

C_{D2} = yaw drag coefficient

CHLOE = a subroutine for performing one iteration of Goodman's process

$c_1, \ldots c_{N2} = y_1(x_o), \ldots y_{N2}(x_o)$ = the initial conditions of S

$c_{N2+1}, \ldots c_{N23} = p_1, \ldots p_{N3}$ = the unknown parameters of S

$D = (D_\ell)$, a vector of N23 elements, obtained in Step 2 of Goodman's process

DE = the name of a subroutine defining the system of equations S2, Eqs (3) and (8); an input argument of CHLOE and MERSON

DER = a vector of N22 elements containing the derivatives defined by Eqs (3) and (8); an output argument of DE and MERSON

D_ℓ = an element of vector D, defined by Eq. (13)

$E = (E_{\ell n})$, an N23 x N23 matrix obtained in Step 3 of Goodman's process

$EI = E^{-1}$, the inverse of matrix E; a CHLOE output argument

$E_{\ell n}$ = an element of matrix E, defined by Eq. (14)

$f_j(\ ) = dy_j/dx$, a given function of x and of the dependent variables and parameters

$G = E^{-1}D$, a vector of N23 elements obtained in Step 4 of Goodman's process

G_ℓ = an element of vector G; the change in c_ℓ produced by one iteration of Goodman's process

H = a Merson I/O argument: the estimated step size at input; the adjusted step size upon return

HMIN = a MERSON input argument: the minimum step size allowed (.01 times the estimated step size if HMIN is negative)

IC = a CHLOE output argument indicating convergence (0), non-convergence but hope remains (1), or all is lost (2)

MATINV = a matrix inversion subroutine called by CHLOE

MERSON = an integration subroutine called by CHLOE

N1 = the number of measured dependent variables in S; a CHLOE input argument

N2 = the number of dependent variables in S; a CHLOE input argument

N3 = the number of parameters in S to be determined

N4 = the number of measurements taken on each of the N1 measured variables; a CHLOE input argument

N22 = N2+N2; a DE and MERSON input argument

N23 = N2+N3; the total number of constants in S (initial conditions plus parameters) to be determined; a CHLOE input argument (≤ 50)

$P_1, \cdots P_{N3}$ = the unknown parameters of S

Q = a relative error criterion for the integration; a MERSON input argument set on line CHLOE 7

R = (R_{im}), the N1xN4 matrix of residuals of the fit; a CHLOE output argument

RMS = a vector of N1 elements containing the root-mean-square error of the fit to each measured variable; a CHLOE output argument

R_{im} = the residual of the fit to the i-th variable at x_m (measured minus computed value)

S = the given system of N2 first order ordinary differential equations; Eq. (3)

SIGN1 ( ) = the signum function of a real or integer argument

S2 = a system of twice N2 first order ordinary differential equations consisting of a system S and the corresponding incremental equations, Eqs. (3) and (8)

t = time, a dependent variable in the drag equation; Eq. (9)

t_m = the measured value of t at z_m

t_o = the value of t at z_o

U = $(U_{\ell im})$, an N23xN1xN4 array obtained in Step 1 of Goodman's process; a CHLOE output argument

$U_{\ell im}$ = an element of array U; the computed value of u_i at x_m, based on the ℓ-th integration of S2

$u_\ell = \Delta y_\ell$ = the change in y_ℓ for a given change in the constant vector C; the ℓ-th incremental variable

V = a vector of twice N23 elements containing the dependent variables, the constants to be determined and the incremental variables of S2; a DE input argument

V1, ... VN = arithmetic or logical variables

v = the magnitude of the velocity vector; Eq. (9)

v_o = the value of v at z_o

W_{im} = a weight associated with A_{im}

X = $(x_1, \ldots x_{N4})$; a CHLOE input argument

XA = x_o; a CHLOE input argument

XB = the value of the independent variable x when DE is called by MERSON; a DE input argument

X1 = a MERSON I/O argument; at input, X1 contains the value of x at the start of the integration interval; upon return, X1=X2

X2 = the value of x at the end of the MERSON integration interval; a MERSON input argument

x = the independent variable of S2

x_o = the value of x at which initial conditions are to be determined; x_o can coincide with a point of X, lie between any two points of X or fall outside the interval x_1 to x_{N4}

$x_1, \cdots x_{N4}$ = the values of x at which measurements were taken; $x_1 < x_2 \cdots < x_{N4}$

Y = the N2xN4 matrix of computed values of the dependent variables; Y(J,M) = the computed value of y_j at x_m; a CHLOE output argument

$y_1, \cdots y_{N1}$ = the measured dependent variables of S

$y_{N1+1}, \cdots y_{N2}$ = the remaining dependent variables of S

$y_{N2+1}, \cdots y_{N23} = p_1, \cdots p_{N3}$ = the unknown parameters of S

z = distance; the independent variable in the drag equation, Eq. (9)

z_m = a value of z at which a time measurement was taken

z_o = the value of z at which initial time and velocity apply in the drag equation

γ = a relative error bound for determining CHLOE convergence, Eq. (17); set to 0.001 in line CHLOE 7

$\Delta C = (\Delta c_1, \cdots \Delta c_{N23})$

$= (1,0,0, \cdots 0), (0,1,0, \cdots 0), \cdots (0, \cdots 0,1)$

= the initial conditions for the incremental equations, Eq. (8)

Δc_ℓ = an arbitrary change in the current value of c_ℓ; assigned the value 0 or 1; Eq. (12)

$\Delta f_j = f_j(x, y_1 + \Delta y_1, \cdots, y_{N23} + \Delta y_{N23}) - f_j(x, y_1, \cdots, y_{N23})$

δ = magnitude of the complex yaw, Eq. (9)

ε = the sum of the squares of the residuals of the fit, Eq. (5)

INDICES

$i = 1, 2, \ldots N1$

$j = 1, 2, \ldots N2$

$k = 1, 2, \ldots N3$

$\ell = 1, 2, \ldots N23$

$m = 1, 2, \ldots N4$

$n = 1, 2, \ldots N23$

I. INTRODUCTION

This paper discusses a relatively new approach to an old problem. We are given the form of a system of ordinary differential equations (linear or non-linear) involving unknown constant parameters and unknown initial conditions. We are further given a set of measurements on one or more of the dependent variables. The problem is to determine those values of the parameters and initial conditions for which the solution curves are the least squares fit to the measured data.

The traditional approach assumes that the measured variables can be adequately represented over the interval of interest by conveniently chosen closed-form expressions containing the desired parameters and initial conditions. The values of the unknowns are then determined by a straightforward least squares fit of the expressions to the measured data (or by an iterative differential corrections process when the expressions are non-linear in the unknowns).

Ideally, these closed-form expressions satisfy the given system of differential equations. In practice, however, the exact solution is usually unknown or too complicated to be of any use and the expressions we work with satisfy a simplified system of differential equations in which all the troublesome terms have been linearized or discarded.

Clearly, there are situations where it would be desirable to bypass all closed-form pseudo-solutions and work directly with the given differential equations. That is, we need a mathematical technique that could determine – directly from the differential equations and without any knowledge or assumptions concerning the form of the solution – those values of the parameters and initial conditions that fit the solution curve to the measured data in a least squares sense.

In Reference 1*, Goodman presents an iterative technique to do very nearly what we want. (A similar technique has been described by Chapman and Kirk[2,3] for a system of equations describing

*References are listed on page 39.

the yawing motion of a missile in free flight.)

The purpose of this report is to present, describe and illustrate the use of the FORTRAN subroutine CHLOE for carrying out Goodman's method on any given system of ordinary differential equations. Goodman considers discrete or continuous measurements on a single dependent variable. We will limit our attention to discrete measurements only and generalize his results to handle measurements on more than one variable.

For the reader who is indifferent to the mechanics of FORTRAN but who might have use for the technique, it should be pointed out that the next three sections are FORTRAN-free. These sections comprise a relatively informal presentation of Goodman's method, subject to our slight generalization. The responsibility for this particular form of presentation and for the accompanying remarks is, of course, the present author's.

The remaining sections, V through X, are written for a potential user who knows — but is not necessarily an expert in — FORTRAN. Most FORTRAN programmers are not computer specialists; they are workers in other fields who have learned FORTRAN on the side. Accordingly, we have not hesitated to offer explanations and advice for the non-professional programmer.

II. STATEMENT OF THE PROBLEM

Assume that our given ordinary differential equations have been reduced to a system S of first order equations. We introduce the following notation:

N1 = the number of measured dependent variables in S.

N2 = the total number of dependent variables in S.
= the number of initial conditions to be determined.
(We will not consider the situation where one or more of the initial conditions is fixed.)

N3 = the number of parameters to be determined.

N4 = the number of measurements taken on each of the N1 measured variables. (It is assumed that the same number of measurements, at the same values of the independent variable, have been made for each measured variable.)

We further assume that

$$(1) \qquad 1 \leq N1 \leq N2 \leq N23 < N4$$

where N23 = N2 + N3

= the total number of unknowns (initial conditions plus parameters) to be determined.

Subscripts and indices used throughout this paper will consistently have the following ranges:

$$i = 1, 2, \ldots N1$$
$$j = 1, 2, \ldots N2$$
$$k = 1, 2, \ldots N3$$
$$\ell = 1, 2, \ldots N23$$
$$m = 1, 2, \ldots N4$$
$$n = 1, 2, \ldots N23$$

However, for the convenience of the reader these ranges will usually be repeated in the text wherever the subscripts appear.

We could write our system S in the form

$$(2) \qquad \begin{cases} \dfrac{dy_j}{dx} = f_j(x, y_1, y_2, \cdots y_{N2}, p_1, p_2, \cdots p_{N3}) \\ y_j(x_o) = c_j \qquad j = 1, 2, \ldots N2 \end{cases}$$

where

$y_1, y_2, \cdots y_{N1}$ are the measured dependent variables

$y_{N1+1}, y_{N1+2}, \cdots y_{N2}$ are the remaining dependent variables

$p_1, p_2, \cdots p_{N3}$ are the unknown parameters

$c_1, c_2, \cdots c_{N2}$ are the unknown initial conditions

However, the notation of Eqs. (2) can be simplified if we regard the parameters p_k as additional dependent variables, subject to the condition that $dp_k/dx = 0$. Thus we introduce two additional notations for the parameters:

$$y_{N2+k} = p_k$$

$$c_{N2+k} = p_k$$

$$k = 1, 2, \ldots N3$$

The system S then assumes the form

$$(3) \qquad \begin{cases} \dfrac{dy_j}{dx} = f_j(x, y_1, y_2, \cdots y_{N23}) \\ y_j(x_o) = c_j & j = 1, 2, \ldots N2 \\ y_{N2+k} = c_{N2+k} & k = 1, 2, \ldots N3 \end{cases}$$

where the initial conditions and parameters are represented by the constant vector

$$C = (c_1, c_2, \cdots c_{N23})$$

We will not go into any rigorous discussion of the mathematical properties the system S must possess. It is sufficient for our purposes to assume that the functions f_j are analytic over the interval of interest.

We are given a set of measurements, which we represent by the N1 x N4 matrix $A = (A_{im})$, and a vector $X = (x_m)$ of the corresponding N4 values of the independent variable. That is,

A_{im} = the measured value of y_i at x_m

$i = 1, 2, \ldots N1$

$m = 1, 2, \ldots N4$

where we assume

$$(4) \qquad x_1 < x_2 < \ldots < x_{N4}$$

The points x_m need not be evenly spaced. It is permissible but not necessary that any one of the points x_m coincide with the point x_o at which initial conditions are to be determined; x_o could fall outside the interval (x_1, x_{N4}) or between any two points.

We define the N1 x N4 matrix $R = (R_{im})$ as follows:

R_{im} = residual of the fit to the i-th variable at x_m

= A_{im} - (the computed value of y_i at x_m)

For any vector C, we can obtain a matrix R by numerical integration of our system S.

The problem is to determine the vector C producing an R that minimizes the quantity

$$\varepsilon = \sum_{m=1}^{N4} \sum_{i=1}^{N1} R_{im}^2 \tag{5}$$

Equation (5) involves two assumptions. First, we have assumed that each measurement A_{im} is equally important (or equally unworthy); otherwise the quantity R_{im}^2 would be multiplied by a weighting function W_{im}. This assumption is not essential; Eq. (5), the equations that follow and subroutine CHLOE can be modified (there is a temptation to say "easily") to include weights. Of course, we hope — and may even take some preliminary action to insure — that the measurements contain no "outliers" (maverick points whose values are so far wrong that they would completely invalidate the results). The second assumption is inherent in the least squares approach. The quantity ε to be minimized does not involve the vector X. Thus, we imply that all errors of measurement are contained in the matrix A; the vector X is assumed to be error-free.

III. THE INCREMENTAL EQUATIONS

For any given vector C there corresponds a set of solutions y_ℓ to Eqs. (3). If we change the initial conditions and parameters, that is, if we replace

$$c_\ell \text{ by } c_\ell + \Delta c_\ell, \quad \ell = 1, 2, \ldots N23$$

a new set of solutions $y_\ell + \Delta y_\ell$ will apply. The differences Δy_ℓ in the two sets of solutions will satisfy the incremental equations

$$(6) \qquad \begin{cases} \dfrac{d\,(\Delta y_j)}{dx} = \Delta f_j & \\ \Delta y_j\,(x_o) = \Delta c_j & j = 1, 2, \ldots N2 \\ \Delta y_{N2+k} = \Delta c_{N2+k} & k = 1, 2, \ldots N3 \end{cases}$$

where $\Delta f_j = f_j\,(x, y_1 + \Delta y_1, \ldots y_{N23} + \Delta y_{N23})$

$$-f_j\,(x, y_1, \ldots y_{N23})$$

We approximate the increments Δf_j by the differentials df_j:

$$(7) \qquad \Delta f_j \simeq df_j = \sum_{\ell=1}^{N23} \left(\frac{\partial f_j}{\partial y_\ell}\right) \Delta y_\ell, \quad j = 1, 2, \ldots N2$$

and introduce the simplifying notation

$$u_\ell = \Delta y_\ell, \quad \ell = 1, 2, \ldots N23$$

The incremental equations (6) then assume the form

$$(8) \qquad \begin{cases} \dfrac{du_j}{dx} = \displaystyle\sum_{\ell=1}^{N23} \left(\frac{\partial f_j}{\partial y_\ell}\right) u_\ell & \\ u_j(x_o) = \Delta c_j & j = 1, 2, \ldots N2 \\ u_{N2+k} = \Delta c_{N2+k} & k = 1, 2, \ldots N3 \end{cases}$$

What we might call the zero-th step in Goodman's technique is to write down these N2 incremental equations. In general, they will involve some or all of the variables y_j (j = 1, 2, ... N2) and hence if

numerical solutions of Eqs. (8) are required — and in Goodman's method they will be — the equations must be integrated simultaneously with the original system, Eqs. (3).

To help fix ideas, consider a relatively simple example: the drag equation

$$\frac{dv}{dz} = - B C_D v \tag{9}$$

where z = distance

$v = \frac{dz}{dt}$ = magnitude of the velocity vector

B = a given constant (the ratio of a dimensionless relative density factor to the diameter).

$C_D = C_{DO} + C_{D2} \delta^2$ = drag coefficient

δ = magnitude of the complex yaw (a given function of z involving <u>known</u> parameters).

We are given a set of measurements (z_m, t_m) and are asked to determine the values of the two parameters C_{DO} and C_{D2}.

In this example, we are free to choose z or t as the independent variable; since δ is given as a function of z, it is convenient to let z be the independent variable. Then our dependent variables are

$$y_1 = t$$

$$y_2 = v$$

$$y_3 = C_{DO}$$

$$y_4 = C_{D2}$$

Here N1 = 1, N2 = 2, N3 = 2 and N23 = 4. Equation (9), rewritten in the form of Equation (3), becomes

$$\begin{cases} \frac{dy_1}{dz} = \frac{1}{y_2} \\ \\ \frac{dy_2}{dz} = - B(y_3 + y_4 \delta^2) y_2 \end{cases} \tag{10}$$

which can be solved numerically for any given constant vector

$$C = (y_1(z_0),\ y_2(z_0),\ y_3,\ y_4) = (t_0,\ v_0,\ C_{D0},\ C_{D2})$$

Finally, our incremental equations for this example are

$$(11) \quad \begin{cases} \dfrac{du_1}{dz} = -\dfrac{u_2}{(y_2)^2} \\ \dfrac{du_2}{dz} = -B[(y_3 + y_4\,\delta^2)\,u_2 + (u_3 + u_4\,\delta^2)\,y_2] \end{cases}$$

which can be solved (simultaneously with Eqs. (10)) for any given incremental constant vector

$$\Delta C = (u_1\ (z_0),\ u_2\ (z_0),\ u_3,\ u_4)$$

IV. ALGORITHM

In this section we list the steps constituting Goodman's method; the motivation for these steps may be found in Reference 1. In the CHLOE subroutine listed in Appendix A, the corresponding steps are clearly indicated.

Assume that we have obtained by some rational means a first estimate for the values of the constant vector C. Further assume that we have written down (correctly) the incremental equations. Let S2 denote the system of twice N2 differential equations consisting of the original system S and the incremental equations, that is, Eqs. (3) and (8).

Step 1. Using the constant vector C, integrate the system S2 a total of N23 times. For the ℓ-th integration, $\ell = 1, 2, \ldots N23$, the incremental constants are taken to be

$$(12) \quad \begin{cases} \Delta c_\ell = 1 \\ \Delta c_n = 0, \quad n \neq \ell \end{cases}$$

Thus for the simple example of the previous section, Eqs. (10) and (11) would be numerically integrated four times, once each for

$$\Delta C = (1,0,0,0),\ (0,1,0,0),\ (0,0,1,0),\ (0,0,0,1)$$

What we are after are the resulting values of y_i and u_i, $i = 1, 2, \ldots N1$, at the measured points x_m. These computed values of y_i yield the matrix of residuals, $R = (R_{im})$, introduced in Section II, where we defined

$$R_{im} = A_{im} - (\text{the computed value of } y_i \text{ at } t_m)$$

We denote the computed values of u_i by the three-dimensional array

$$U = (U_{\ell im})$$

where

$U_{\ell im}$ = the computed value of u_i at x_m, based on the ℓ-th integration of S2

$\ell = 1, 2, \ldots N23$

$i = 1, 2, \ldots N1$

$m = 1, 2, \ldots N4$

Step 2. Evaluate the components of a column vector $D = (D_\ell)$, $\ell = 1, 2, \ldots N23$, where

$$D_\ell = \sum_{m=1}^{N4} \sum_{i=1}^{N1} R_{im}\, U_{\ell im} \tag{13}$$

Step 3. Evaluate the elements of an N23 x N23 symmetric matrix $E = (E_{\ell n})$, where

$$E_{\ell n} = \sum_{m=1}^{N4} \sum_{i=1}^{N1} U_{\ell im}\, U_{nim} \tag{14}$$

$\ell = 1, 2, \ldots N23$

$n = 1, 2, \ldots N23$

Step 4. Evaluate the components of a column vector $G = (G_\ell)$, $\ell = 1, 2, \ldots N23$, where

$$G = E^{-1}D \tag{15}$$

and E^{-1} is the inverse of matrix E.

Step 5. Obtain new values for the components of the constant vector C:

$$(c_\ell)_{new} = (c_\ell)_{old} + G_\ell \qquad \ell = 1, 2, \ldots N23 \tag{16}$$

Step 6. Test for convergence of the process. There are several criteria that could be used. One of the simplest, and the one used in subroutine CHLOE, is the requirement that

$$|G_\ell| \leq \gamma\, |c_\ell|_{new}, \qquad \ell = 1, 2, \ldots N23 \tag{17}$$

where γ is some pre-selected positive relative error bound. An alternative criterion is given in Section X. If the criterion fails, return to Step (1) and repeat the entire process.

We can go around in this loop until convergence is achieved, until an obvious divergence has arisen or until a specified number of iterations have been performed.

V. THE ARGUMENTS OF SUBROUTINE CHLOE

Subroutine CHLOE (listed in Appendix A) is called by the statement:

```
CALL CHLOE (DE,N1,N2,N23,N4,A,X,XA,C,EI,U,R,Y,RMS,IC)
```

Here and throughout the rest of this paper, the usual convention applies to FORTRAN real and integer variable names: integer names and only integer names start with I, J, K, L, M or N. The first eight arguments of CHLOE are inputs:

DE = the name of the subroutine that defines the set of twice N2 differential equations indicated in Eqs. (3) and (8). This name must be declared in an EXTERNAL statement in the program

that calls CHLOE. Details of subroutine DE will be discussed in Section VIII.

N1 = the number of measured dependent variables; N1≥1.

N2 = the total number of dependent variables; N2≥N1.

N23 = the number of unknown constants (initial conditions plus parameters) to be determined; N23≥N2. Note: N23 (and hence N2 and N1) should be no greater than 50. This restriction can be relaxed only by making appropriate coding changes.

N4 = the number of measurements taken on each of the N1 measured variables; N4>N23.

A = the N1 x N4 matrix of measured values of the dependent variables, where A(I,M) is the measured value of y_i at x_m.

X = the vector of the corresponding N4 values of the independent variable x, where

X(1)<X(2)<...<X(N4)

XA = the value of the independent variable at which initial conditions are to be determined. XA may (but need not) equal any element of vector X.

The next argument – in a sense, the crucial argument – serves as both input and output.

C = a vector of N23 elements. Upon entry, C contains a given set of estimates for the N23 constants to be determined, where

C(1), ..., C(N1)	are estimates of the initial conditions on the measured variables
C(N1+1), ..., C(N2)	are estimates of the initial conditions on the remaining dependent variables
C(N2+1), .., C(N23)	are estimates of the unknown parameters.

Upon return from CHLOE, the vector C contains the next (and hopefully improved) set of estimates as determined by one iteration of Goodman's procedure. (The final argument

of CHLOE, IC, indicates whether or not these returned estimates are adequate.)

The remaining arguments are outputs. The output arrays EI, U, R and RMS described below are obtained by CHLOE during the course of an iteration, whereas new values of C are not obtained until the end of that iteration. Thus we might anticipate that these output arrays are based on the input C values rather than on the C values returned. This is indeed the situation so long as the convergence test has failed; the output arrays in effect lag one iteration behind C However, if the convergence criterion is satisfied in the last step of a given iteration CHLOE will not return immediately to the calling program; first it will perform enough steps of an additional iteration to update the output arrays so that they correspond to the final C values.

The desirability of this up-dating feature could be debated. If the final change in C is small enough, the corresponding changes in the output arrays will presumably be small too. It's a question of whether or not these final small changes in the output arrays are worth the machine time taken up by an extra, "catch-up" iteration. At any rate, the feature is accomplished by the statement GOTO 24 on line CHLOE 51 and can be eliminated by deleting that statement.

The CHLOE output arguments are as follows:

EI = an N23 x N23 matrix that is the inverse of the matrix E defined by Eq. (14). The matrix EI was made an argument of CHLOE mainly for the convenience of using adjustable dimensions for E in the subroutine. However, EI is not entirely without interest. In the usual least squares process, an estimate of the RMS error in the ℓ-th calculated constant is obtained by multiplying the estimated RMS error of the fit by the square root of the ℓ-th diagonal element of the inverse of the matrix formed by the coefficients of the normal equations. We assert that the diagonal elements of the inverse matrix EI play a similar role in Goodman's

technique. (Goodman doesn't discuss error analysis in Reference 1 and we have seen no proof of this assertion.) In particular, if there is only one measured variable (N1=1), then we assert that an estimate of the RMS error in the ℓ-th determined constant is afforded by the product

RMS(1)*SQRT(EI(L,L))

where RMS(1) is defined below in the discussion of the next to last argument of CHLOE. It has been our somewhat limited experience that while these estimates are not always significant in themselves, the values obtained by two or more similar runs (runs using the same differential equations but different sets of measurements) are useful for comparison .

U = an N23 x N1 x N4 array defined in Step 1 of the algorithm. Array U was made an argument of CHLOE solely for the convenience of using adjustable dimensions for U in the subroutine.

R = the N1 x N4 matrix of residuals of the fit:

R(I,M) = A(I,M) - Y(I,M)

where matrix Y is defined next.

Y = the N2 x N4 matrix of computed values of the dependent variables, where

Y(1,M), ..., Y(N1,M)	are the computed values of the measured dependent variables at X(M).
Y(N1+1,M), ..., Y(N2,M)	are the computed values of the remaining dependent variables at X(M).

RMS = a vector of N1 elements containing the root-mean-square error of the fit to each measured variable:

$$RMS(I) = \left\{ \left[\sum_{M=1}^{N4} R(I,M)^2 \right] \Big/ (N4 - N23) \right\}^{1/2}$$

IC = a convergence indicator:

IC = 0 if the process has converged; that is, if

$$|C(L)_{return} - C(L)_{entry}| \leq \gamma |C(L)_{return}|$$

$$L = 1, 2, \ldots N23$$

The constant γ could have been made an input argument to CHLOE; instead, it was arbitrarily decided to assign γ the value 0.001 within the subroutine. This is done on line CHLOE 7 by the statement EPS = .001. The user, of course, can substitute any other positive value if he chooses. When IC=0, the output arrays EI,U,R,Y and RMS were obtained from the final C values.

IC=1 if the process has not yet converged, but more iterations are possible. The output arrays EI,U,R,Y and RMS were obtained from the input C values rather than from the C values returned.

IC=2 if the process has somehow gone astray to such an extent that matrix E of Step 3 is singular. A subroutine MATINV (see Appendix C) will then cause the phrase SINGULAR MATRIX to be printed. In this event, the matrix inversion of Step 4 is impossible and the process has broken down. No further iterations should be made. The input C values are returned in C and the output arrays U, R, Y and RMS are based on these input C values.

VI. COMMENTS ON THE VERSION OF FORTRAN USED

In addition to subroutines CHLOE and DE, two subroutines called by CHLOE must be included in the over-all program:

(a) MERSON, an integration subroutine listed in Appendix B and discussed in Section VII.

(b) MATINV, a matrix inversion subroutine listed in Appendix C.

Subroutines MERSON and MATINV were obtained from the Computer Support Division of the Aberdeen Research and Development Center (ARDC).

In the listing of the subroutines in the appendices, we have conserved space where feasible by writing two or more statements on a line, separating the statements by $ signs. This convention is wide-spread but not universal.

Subroutines CHLOE, MERSON and MATINV were written for use on ARDC's BRLESC I and BRLESC II computers (see Reference 4). Although the compilers in these two computers implement a FORTRAN which is not quite "standard" and not quite FORTRAN IV (and indeed not quite the same on the two computers), it is believed that the above-mentioned subroutines have been restricted for the most part to standard statements and features. However, there are at least four exceptions:

(a) the use on lines CHLOE 19, ... 20, ... 21 and ... 28 of subscript expressions which are the sums of two integer variables. In general, the BRLESC I/II compilers will accept as a subscript any integer expression not involving subscripted variables. Some compilers may impose more restrictions on the subscript expression.

(b) the use on lines CHLOE 5 and MERSON 24 of the predefined signum function SIGN1 ( ). This nonstandard function accepts a single argument, real or integer, and assumes the real value -1.0, 0.0, or 1.0 for negative, zero or positive argument, respectively.

(c) the use on lines MERSON 4, ... 16, ... 18 and ... 19 of

multiple arithmetic or logical assignment statements linked by equal signs. In a statement of the form

VN = ... = V2 = V1 = ale

where "ale" denotes an arithmetic or logical expression and the other quantities are the names of arithmetic or logical variables, respectively, the result of the expression is stored in V1, V2, ... VN in that order.

(d) the use on line MERSON 7 of a PRINT statement (which will be used only if N2 exceeds 50). On most computers, but not on BRLESC I/II, this nonstandard statement will print data on an on-line printer.

The only other output statement in the three subroutines – and the only one likely to be used – is the SINGULAR MATRIX reprimand on lines MATINV 9 and ... 10. Here the integer 6 in the statement WRITE (6,4) may have to be changed to specify the proper output unit at the user's installation.

VII. COMMENTS ON THE MERSON SUBROUTINE

The MERSON subroutine (listed in Appendix B and described in Reference 5) uses a method proposed by R. H. Merson of Australia to integrate a system of first order ordinary differential equations. The method is a fourth-order member of the Runge-Kutta family, requiring five function evaluations over each integration step.

MERSON was chosen from the three differential equation subroutines available at Computer Support Division because it seemed the easier to use of the two that allow the integration step size to be adjusted automatically to obtain a predefined accuracy.

MERSON is called in line CHLOE 23 by the statement

CALL MERSON (DE,N22,X1,X2,V,DER,H,HMIN,Q)

where N22 is twice N2. To call MERSON to integrate the set of N22 differential equations defined in subroutine DE over an interval X1 to X2. When MERSON is called, argument V must contain the values of the

dependent variables at X1. Upon return from MERSON, V contains the values of the dependent variables at X2 and argument X1 contains the value of X2. Arguments V and DER, which are passed by MERSON to subroutine DE, will be discussed in Section VIII.

The scheme used by CHLOE for calling MERSON can perhaps best be described by an example. Suppose array X consists of six points: X(1), ... X(6) and that XA, the point at which initial conditions are to be determined, lies between X(3) and X(4). (CHLOE, of course, determines where XA falls among the elements of X.) Then CHLOE will call MERSON six times, as follows:

	X1	X2
1	XA	X(3)
2	X(3)	X(2)
3	X(2)	X(1)
4	XA	X(4)
5	X(4)	X(5)
6	X(5)	X(6)

Before the first call, CHLOE will set V to the estimated values of the dependent variables at XA. For the second and third calls above, V will automatically have the right input values; they are the values returned by the previous call. For the fourth call, CHLOE will reset V to the estimated values at XA.

We can see from the above description that in general MERSON will be called N4 times (or N4-1 times if XA coincides with a point in X). All of this, of course, is for a given set of initial conditions. The reader will recall that the system of equations must be integrated N23 times, once for each set of initial conditions specified by Eq. (12). Hence, for each iteration of the CHLOE routine (that is, for each time CHLOE is called), MERSON is called N23 x N4 times.* This approach may be prohibitively time-consuming if the user's equations are laborious and/or his data points are numerous. CHLOE's selection of integration intervals might then be modified by the user to suit his needs.

Other (and simpler) changes the user may wish to make in CHLOE involve the last three arguments of MERSON: H, HMIN and Q. In this

---

* *In the final iteration, MERSON is called twice N23 x N4 times if convergence is achieved.*

paragraph, we discuss the general use of these three arguments; in the next paragraph, we discuss what CHLOE does with them. When MERSON is called, H must contain an estimated step size to be taken over the interval X1 to X2. The sign of H will be changed, if necessary, to proceed in the proper direction from X1 to reach X2. If the integration reaches a point x such that x+H exceeds the endpoint, then a temporary step size is used so that the endpoint is reached exactly. If input argument Q has a value equal to or greater than 1.0, the estimated step size is retained throughout the interval. If Q is positive but less than 1.0, relative error estimates are obtained by MERSON at each step along the interval and the step size H will be changed if necessary; the value of H upon return from MERSON will be the adjusted step size. If all the relative error estimates are less than Q/32.0, then the step size H is doubled before return. If any relative error is greater than Q, then the step size H is halved and computation begins again at the start of the interval. Step size H will be halved as many times as necessary, unless its value becomes less than a specified minimum step size: the input argument HMIN. In that event HMIN will be the adjusted step size. If input HMIN is negative, then MERSON uses one-hundredth the input step size as the minimum step size.

On line CHLOE 7, the relative error criterion Q is set at the arbitrary value of 0.0001. On line CHLOE 6, HMIN is set negative, so that the minimum step size will be .01 times the input step size. As presently coded, CHLOE makes no use of the adjusted step size returned in H from one interval to estimate the best input step size for the next interval. On line CHLOE 22, the input step size H for each interval X1 to X2 is arbitrarily set at one-eighth the interval length. The manner in which CHLOE determines H, HMIN and Q can be easily changed by the user to expedite his particular problem.

VIII. COMMENTS ON WRITING SUBROUTINE DE

Subroutine DE must be written by the user for his particular version of Eqs. (3) and (8). DE is called by MERSON and thus the first

statement of DE must have the form prescribed by MERSON:

SUBROUTINE DE (N22,XB,V,DER)

The first three arguments are inputs and must not be altered by DE:

N22 = twice N2 = the dimension of the argument DER

XB = the value of the independent variable x when DE is called by MERSON

V = a vector of twice N23 elements containing the dependent variables, the constants to be determined and the increments of system S2 in the following order:

$V(1),\ldots,V(N2) = y_1, \ldots y_{N2}$

$V(N2+1),\ldots,V(N22) = u_1, \ldots u_{N2}$

$V(N22+1),\ldots,V(N22+N3) = y_{N2+1},\cdots y_{N23} = p_1,\cdots p_{N3}$

$V(N22+N3+1),\ldots,V(N23+N23) = u_{N2+1},\cdots u_{N23}$

The final argument is the output:

DER = a vector of N22 elements containing the derivatives of the N2 dependent variables and their increments as defined by Eqs. (3) and (8):

$$DER(1),\ldots,DER(N2) = \frac{dy_1}{dx},\ldots,\frac{dy_{N2}}{dx}$$

$$DER(N2+1),.,DER(N22) = \frac{du_1}{dx},\ldots,\frac{du_{N2}}{dx}$$

The vectors V and DER must be declared in a DIMENSION statement in DE. As mentioned in Section V, we must have $N2 \leq N23 \leq 50$. Hence the dimensions of V and DER (twice N23 and twice N2, respectively) must not exceed 100 without making appropriate changes.

The body of subroutine DE consists of the statements defining the elements of vector DER in terms of the elements of vector V. It may happen that certain known constants appearing in the equations will take on different values for different cases. These constants should be assigned FORTRAN names in the program calling CHLOE and passed to DE from that calling program by a COMMON statement.

To illustrate these general remarks, consider Eq. (9), the drag equation. We have shown that the original and incremental equations

reduce to:

$$(10)\quad \begin{cases} \dfrac{dy_1}{dz} = 1/y_2 \\ \dfrac{dy_2}{dz} = -B(y_3 + y_4\,\delta^2)y_2 \end{cases}$$

$$(11)\quad \begin{cases} \dfrac{du_1}{dz} = -u_2/(y_2)^2 \\ \dfrac{du_2}{dz} = -B[(y_3 + y_4\,\delta^2)u_2 + (u_3 + u_4\,\delta^2)y_2] \end{cases}$$

where N1 = 1, N2 = 2, N23 = 4, y_1 and y_2 are the dependent variables and y_3 and y_4 are the constants to be determined.

Suppose that the function δ^2 is given in the form

$$\delta^2 = R_1^2 + R_2^2 + 2R_1R_2 \cos(R_3 + R_4 z)$$

where the R_i are known constants whose values may be different for each set of measured data. Since DE is entered many times in each computer run, it is usually a good idea to look for step-saving devices when writing DE. Here, for example, we would calculate

$$R_5 = R_1^2 + R_2^2$$

$$R_6 = 2R_1R_2$$

in the program that calls CHLOE and pass R_5 and R_6 (rather than R_1 and R_2) to DE by a COMMON statement.

In the same step-saving vein, it is possible to omit B from subroutine DE by proper scaling of the two constants to be determined. Thus we let

$V(1) = y_1$ $\qquad$ $V(5) = B.y_3 = B.C_{DO}$

$V(2) = y_2$ $\qquad$ $V(6) = B.y_4 = B.C_{D2}$

$V(3) = u_1$ $\qquad$ $V(7) = B.u_3$

$V(4) = u_2$ $\qquad$ $V(8) = B.u_4$

The complete DE subroutine might look like this:

```
SUBROUTINE  DE(N22,XB,V,DER)
DIMENSION   V(8),DER(4)
COMMON/CC/R3,R4,R5,R6
DELSQ = R5 + R6*COS(R3 + R4*XB)
CD = V(5) + V(6)*DELSQ
DER(1) = 1.0/V(2)
DER(2) = -CD*V(2)
DER(3) = -V(4)*DER(1)**2
DER(4) = -CD*V(4)-(V(7) + V(8)*DELSQ)*V(2)
RETURN
END
```

Each time the program returns from CHLOE, the argument C will consist of the latest values for

$C(1) = y_{10}$

$C(2) = y_{20}$

$C(3) = B \cdot C_{DO}$

$C(4) = B \cdot C_{D2}$

To obtain the desired values C_{DO} and C_{D2}, we must divide C(3) and C(4) by B in the program that calls CHLOE, as shown in the next section.

IX. COMMENTS ON WRITING THE PROGRAM THAT CALLS CHLOE

In the program that calls CHLOE:

(a) the subroutine DE must be declared in an EXTERNAL statement

(b) the eight array arguments of CHLOE

A(N1,N4), X(N4), C(N23), EI(N23,N23),

U(N23,N1,N4), R(N1,N4), Y(N2,N4), RMS(N1)

must be declared in a DIMENSION statement. Usually for a given problem the values of the dimensions N1,N2 and N23 are fixed and known, whereas the value of N4 varies from case to case within that problem. In that event, some number equal to or larger than the largest anticipated number of measurements should be used for N4 in the DIMENSION statement.

(c) any known constants of the differential equations whose values may change from case to case are assigned FORTRAN names

and passed to the DE subroutine by a COMMON statement. This process was illustrated in the previous section.

To illustrate how CHLOE might be used within the calling program, consider our by-now familiar drag equation problem as written up in the sample DE subroutine of the previous section. We assume that we have reached a stage within the calling program where the values of XA, B and N4 have been determined and where

array A contains the measurements,

vector X contains the values of the independent variable,

vector C contains initial estimates of the four constants y_{10}, y_{20}, $B \cdot C_{DO}$ and $B \cdot C_{D2}$.

Then the portion of the calling program that invokes CHLOE might look like this:

```
      DO 4 NA = 1,20
      CALL CHLOE (DE,1,2,4,N4,A,X,XA,C,EI,U,R,Y,RMS,IC)
      C3 = C(3)/B
      C4 = C(4)/B
      WRITE (6,100)C(1),C(2),C3,C4
      IB = IC +1
      GOTO (5,4,3),IB
    4 CONTINUE
```

When CHLOE is first called in the above DO-loop, the vector C transmits the initial estimates to CHLOE and returns with the values obtained by the first iteration. Upon each subsequent call, C transmits to CHLOE the values obtained by the previous iteration and returns with the latest values. If the process has converged (IC=0), the program goes to statement 5 where presumably the elements of output arrays R,Y and RMS will be printed. If the process has broken down (IC=2), the program goes to statement 3, which could cause the program to stop, to consider a new case or to take other appropriate action. If the process has failed to converge in 20 iterations, the program goes to the statement following statement 4, where again some appropriate action is taken. The maximum number of iterations allowed (the terminal parameter of the DO-loop) is, of course, arbitrary.

Probably 20 is a practical upper limit; if the process hasn't converged by then, it isn't likely to.

X. COMMENTS ON CONVERGENCE

One difficulty with the CHLOE convergence criterion given by Eq. (17) is that if the value of any constant to be determined is "small" (for example, if some initial condition is zero), we may achieve a perfectly satisfactory fit several wasted iterations before the criterion is satisfied — if, indeed, it ever is. In this situation, some more appropriate criterion is needed.

For example, an input vector DMAX of N23 elements could be added to the arguments of CHLOE, where

DMAX(L) = maximum absolute change in C(L) from one iteration to the next that satisfies the criterion

The value of each DMAX(L) would be defined in the program that calls CHLOE. Lines CHLOE 49 and ... 50 would have the form

```
IC = 1$ DO 22 L = 1,M23                                  CHLOE 49
IF(ABS(D(L)).GT.DMAX(L))GOTO 23                          CHLOE 50
```

More sophisticated tests could be devised for special situations.

Assume now that CHLOE contains an adequate and efficient convergence test. Troubles with convergence are not necessarily over. The CHLOE user will encounter one of several outcomes:

1. The routine will fail to converge
   a. by returning values in C that jump around from iteration to iteration in apparently random fashion;
   b. by returning progressively wilder values in array C;
   c. by oscillating between two (and occasionally more) sets of values in array C. That is, after a number of iterations we achieve a set of constants C_A which, when sent to CHLOE, returns as set C_B and when C_B is sent to CHLOE, we get back C_A again. In some cases, it might be worth while to code into the program that calls CHLOE a routine

to recognize this oscillation and break out of the cycle.

2. The routine will converge to the wrong answer. Wrong answers and divergence usually occur for any one or a combination of three reasons.

   a. The set of differential equations used may be inadequate to describe the observed variables. Any values obtained for the constants then have little or no physical meaning or at best require careful interpretation. This condition, of course, is not the fault of the procedure; it is part of the job of the user to apply the proper equations to the measured data.

   b. The noise in the measured data may be so great that one or more and possibly all of the constants simply can not be well-determined.

   c. The initial estimates of the constants to be determined may be too far from the "correct" values. We can regard the quantity ε of Eq. (5) as a scalar point function of the point C, where we seek the point that gives ε its absolute minimum value. If we start too far from this desired point, the process may settle for a closer point that gives ε a relative (but not the absolute) minimum, or the process may wander off helplessly in the wrong direction and never find its way back. The user will often find that some initial estimates in his problem are much more critical than others. Convergence seems to depend almost solely on these critical estimates; the remaining constants can have relatively poor first guesses with impunity.

In general, divergence is to be preferred to wrong answers because we can clearly recognize divergence. There are no signposts — except possibly the size of the errors of the fit and the strange values obtained for the constants (provided we know what constitutes a strange value) — to warn us that we have obtained a spurious set of results.

3. The routine will converge to the right answer. The reader may have grown a little disenchanted with the process during the reading of 1. and 2. above. He shouldn't; all this emphasis on divergence and wrong answers has painted far too bleak a picture. Given the right equations, good data and good initial guesses, the process should yield consistently good results. Moreover, in some situations, the process (or one very like it) is the only approach available; it is the "only game in town."

REFERENCES

1. Theodore R. Goodman, "System Identification and Prediction – An Algorithm Using a Newtonian Iteration Procedure," Quarterly of Applied Mathematics, 24, No. 3, 249-255 (1966).

2. Gary T. Chapman and Donn B. Kirk, "A Method for Extracting Aerodynamic Coefficients From Free-Flight Data," AIAA Journal, 8, No. 4, 753-758 (1970).

3. Charles H. Murphy, "Comments on 'A Method for Extracting Aerodynamic Coefficients From Free-Flight Data'", AIAA Journal, 8, No. 11, 2109-2111 (1970).

4. Lloyd W. Campbell and Glenn A. Beck, BRLESC I/II FORTRAN, Aberdeen Research and Development Center Technical Report No. 5, AD No. 704343 (1970).

5. Monte W. Coleman, MERSON Integration Routine, SPB-10-70, August 1970 (one of a series of informal notes issued by the Computer Support Division, Aberdeen Research and Development Center).

APPENDIX A

```
      SUBROUTINE CHLOE (DE,N1,N2,N23,N4,A,X,XA,C,E,U,R,Y,RMS,IC)          *****  1
      EXTERNAL DE                                                        CHLOE  2
      DIMENSION A(N1,N4),X(N4),C(N23),E(N23,N23),U(N23,N1,N4),           CHLOE  3
     D          R(N1,N4),Y(N2,N4),RMS(N1),V(100),DER(100),D(100)         CHLOE  4
C
C  STATEMENT FUNCTION
      F(J) = 1.0 - ABS(SIGN1(J))                                         CHLOE  5
C     F(J) = 1.0 (J ZERO), 0.0(J NONZERO)
C
      M1=N1$ M2=N2$ M23=N23$ M4=N4$ N22=M2+M2$ M7=M2+1$ HMIN=-1.$ IC=2    CHLOE  6
      Q=.0001$ EPS=.001                                                  CHLOE  7
C       Q= DESIRED BOUND FOR RELATIVE ACCURACY IN MERSON SUBROUTINE
C     EPS= RELATIVE ERROR BOUND FOR CONVERGENCE TEST
C
C  DETERMINE
C       KL= NO. OF POINTS IN ARRAY X LESS THAN XA
C       KR= NO. OF POINTS    ARRAY X GREATER THAN XA
C  IF XA COINCIDES WITH  INT IN X, THEN THE ELEMENTS OF
C  U, R AND Y FOR THAT      ARE OBTAINED AT ONCE.
C
     24 DO 1 M=1,M4$ IF(X      )5,2,1                                    CHLOE  8
      1 CONTINUE$ KL=M4$ KR=0$ GOTO 6                                    CHLOE  9
      2 KL=M-1$ KR=M4-M$ DO 4 J=1,M2$ IF(J.GT.M1)GOTO 4                  CHLOE 10
        R(J,M)=A(J,M)-C(J)$ DO 3 L=1,M23                                 CHLOE 11
                                                                         CHLOE 12
      3 U(L,J,M)=F(L-J)                                                  CHLOE 13
      4 Y(J,M)=C(J)$ GOTO 6                                              CHLOE 14
      5 KL=M-1$ KR=M4-KL
C
C  STEP 1.  FOR EACH L (L=1,2,...N23), DO AS FOLLOWS.  FROM INITIAL
C  POINT XA, INTEGRATE LEFT TO POINT X(KL) AND STORE THE REQUIRED VALUES
C  IN ARRAYS U, R AND Y.   THEN INTEGRATE FROM X(KL) TO X(KL-1),
C  STORE THE RESULTS, AND SO ON TO X(1).  RESETTING INITIAL CONDITIONS,
C  INTEGRATE FROM XA TO THE FIRST POINT ON THE RIGHT, FROM THERE TO
C  THE SECOND POINT AND SO ON TO X(N4).
C
      6 DO 15 L=1,M23                                                    CHLOE 15
        IF(KL.EQ.0)GOTO 7$ IA=-1$ IB=0$ LK=KL$ GOTO8                     CHLOE 16
      7 IA=1$ IB=M4+1$ LK=IB-KR                                          CHLOE 17
      8 X1=XA$ DO 9 J=1,M2$ V(J)=C(J)                                    CHLOE 18
      9 V(M2+J)=F(L-J)$ IF(M23.LT.M7)GOTO 11                             CHLOE 19
        DO 10 K=M7,M23$ V(M2+K)=C(K)                                     CHLOE 20
     10 V(M23+K)=F(L-K)                                                  CHLOE 21
     11 X2=X(LK)$ H=0.125*(X2-X1)                                        CHLOE 22
        CALL MERSON(DE,N22,X1,X2,V,DER,H,HMIN,Q)                         CHLOE 23
        IF(L.GT.1)GOTO 13$ DO 12 J=1,M2                                  CHLOE 24
        IF(J.GT.M1)GOTO 12$ R(J,LK)=A(J,LK)-V(J)                         CHLOE 25
     12 Y(J,LK)=V(J)                                                     CHLOE 26
     13 DO 14 I=1,M1                                                     CHLOE 27
     14 U(L,I,LK)=V(M2+I)$ LK=LK+IA$ IF(LK.NE.IB)GOTO 11                 CHLOE 28
        IF(IA.EQ.1)GOTO 15$ IF(KR.NE.0)GOTO 7                            CHLOE 29
     15 CONTINUE                                                         CHLOE 30
```

```
C
C  EVALUATE THE RMS ERROR OF THE FIT IN EACH MEASURED VARIABLE
C
      RA=FLOAT(M4-M23)$ DO 17 I=1,M1$ RM=0.0$ DO 16 M=I,M4                CHLOE 31
   16 RM=RM+R(I,M)**2                                                    CHLOE 32
   17 RMS(I)=SQRT(RM/RA)                                                 CHLOE 33
C
C  STEP 2.  COMPUTE THE VECTOR D.
C
      DO 18 L=1,M23$ D(L)=0.0                                            CHLOE 34
      DO 18 M=1,M4                                                       CHLOE 35
      DO 18 I=1,M1                                                       CHLOE 36
   18 D(L)=D(L)+R(I,M)*U(L,I,M)                                          CHLOE 37
C
C  STEP 3.  COMPUTE THE SYMMETRIC MATRIX E.
C
      DO 20 N=1,M23                                                      CHLOE 38
      DO 20 L=N,M23$ E(L,N)=0.0                                          CHLOE 39
      DO 19 M=1,M4                                                       CHLOE 40
      DO 19 I=1,M1                                                       CHLOE 41
   19 E(L,N)=E(L,N)+U(L,I,M)*U(N,I,M)                                    CHLOE 42
      IF(L.EQ.N)GOTO 20$ E(N,L)=E(L,N)                                   CHLOE 43
   20 CONTINUE                                                           CHLOE 44
C
C  STEP 4.  COMPUTE THE VECTOR G = (E INVERSE)*D.
C
      CALL MATINV(E,M23,D,M23,1,DET)                                     CHLOE 45
C     UPON RETURN, D IS REPLACED BY G, E IS REPLACED BY E INVERSE.
      IF(IC.EQ.0.OR.DET.EQ.0.)GOTO 23                                    CHLOE 46
C
C  STEP 5.  UPDATE THE VECTOR C.
C
      DO 21 L=1,M23                                                      CHLOE 47
   21 C(L)=C(L)+D(L)                                                     CHLOE 48
C
C  STEP 6.  TEST FOR CONVERGENCE.
C
      IC=1$ DO 22 L=1,M23$ RA=EPS*ABS(C(L))                              CHLOE 49
      IF(ABS(D(L)).GT.RA)GOTO 23                                         CHLOE 50
   22 CONTINUE$ IC=0$ GOTO 24                                            CHLOE 51
C     IF CONVERGENCE HAS BEEN ACHIEVED (IC=0), THE STATEMENT GOTO 24
C     CAUSES AN ADDITIONAL RUN THROUGH STEPS 1 TO 4 SO THAT THE
C     OUTPUT ARRAYS WILL BE BASED ON THE FINAL C VALUES.
C
   23 RETURN$ END                                                        CHLOE 52
```

```
      SUBROUTINE MERSON(FUNC,N,X,Z,Y,F,H,HMIN,E)                        ****    1
      DIMENSION Y(1),F(1) $ DIMENSION T(100),G(100),S(100)              MERSON  2
      LOGICAL BC,BE,BH,BR,BX $ NT=N$ ZT=Z$ HMI=HMIN$ ET=ABS(E)          MERSON  3
      IF(HMI.LT.0.0)HMI=0.01*ABS(H) $ BH=BR=BX=.TRUE.                   MERSON  4
      BC=0.0.LT.ET.AND.ET.LT.1.0 $ E5=ET*5.0                            MERSON  5
      IF((ZT.GT.X.AND.H.LT.0.0).OR.(ZT.LT.X.AND.H.GT.0.0))H=-H          MERSON  6
      IF(NT.LE.100)GOTO100$ PRINT 1,NT$ STOP                            MERSON  7
    1 FORMAT(22H RUN ERROR, MERSON, N=,I10)                             MERSON  8
  100 XS=X $ DO 110 J=1,NT $ G(J)=Y(J)                                  MERSON  9
  110 CONTINUE                                                          MERSON10
  200 HS=H$ Q=X+H-ZT$ BE=.TRUE.                                         MERSON11
      IF((Q.LT.0.0.AND.H.GE.0.0).OR.(Q.GT.0.0.AND.H.LE.0.0)) GO TO 210  MERSON12
      H=ZT-X$ BR=.FALSE.                                                MERSON13
  210 H3=H/3.0 $ DO 510 ISW=1,5 $ CALL FUNC(NT,X,Y,F) $ DO 450 I=1,NT   MERSON14
      C=H3*F(I)$ GOTO(301,302,303,304,305),ISW                          MERSON15
  301 T(I)=R=C$ GOTO 400                                                MERSON16
  302 R=0.5*(C+T(I))$ GOTO 400                                          MERSON17
  303 S(I)=R=3.0*Q$ R=0.375*(R+T(I))$ GO TO 400                         MERSON18
  304 T(I)=R=T(I)+4.0*Q$ R=1.5*(R-S(I))$ GO TO 400                      MERSON19
  305 R=0.5*(Q+T(I))$ Q=ABS(2.0*R-1.5*(Q+S(I)))                         MERSON20
  400 Y(I)=G(I)+R $ IF(ISW.NE.5) GO TO 450 $ IF(.NOT.BC)GOTO 450 $ R=E5 MERSON21
      IF(ABS(Y(I)).GE.C.001)R=R*ABS(Y(I))                               MERSON22
      IF((Q.LT.R).OR..NOT.BX)GOTO 420 $ BR=.TRUE.$ BH=.FALSE.$ H=0.5*H  MERSON23
      IF(ABS(H).GE.HMI)GOTO 410 $ H=SIGN1(H)*HMI$ BX=.FALSE.            MERSON24
  410 DO 411 J=1,NT $ Y(J)=G(J)                                         MERSON25
  411 CONTINUE $ X=XS$ GOTO 200                                         MERSON26
  420 IF(Q.GE.0.03125*R)BE=.FALSE.                                      MERSON27
  450 CONTINUE $ GOTO(501,510,503,504,510),ISW                          MERSON28
  501 X=X+H3$ GOTO 510                                                  MERSON29
  503 X=X+0.5*H3$ GOTO 510                                              MERSON30
  504 X=X+0.5*H                                                         MERSON31
  510 CONTINUE $ IF(.NOT.BC) GO TO 521                                  MERSON32
      IF(.NOT.(BE.AND.BH.AND.BR)) GO TO 520 $ H=2.0*H $ BX=.TRUE.       MERSON33
  520 BH=.TRUE.                                                         MERSON34
  521 IF(BR) GO TO 100 $ H=HS$ RETURN $ END                             MERSON35
```

APPENDIX C

```
      SUBROUTINE MATINV(A,N,C,NMAX,K,DET)                                  ****   1
      DIMENSION A(NMAX,1), C(1)                                            MATINV 2
      NN=N $ KK=K $ IF(1-KK) 24,23,23                                      MATINV 3
   23 N3=NN $ IF(KK) 25,26,25                                              MATINV 4
   25 ASSIGN 8 TO N5 $ ASSIGN 16 TO N7 $ GO TO 27                          MATINV 5
   24 N3=KK+NN-1                                                           MATINV 6
   26 ASSIGN 9 TO N5 $ ASSIGN 17 TO N7                                     MATINV 7
   27 DET=1.0 $ DO 21 I=1,NN $ IF(A(I,I)) 2,3,2                            MATINV 8
    3 WRITE (6,4)                                                          MATINV 9
    4 FORMAT (16H SINGULAR MATRIX) $ DET=0.0 $ GO TO 22                    MATINV10
    2 T1=1.0/A(I,I) $ DET=DET*A(I,I) $ A(I,I)=1.0 $ DO 6 J=1,N3            MATINV11
    6 A(I,J)=A(I,J)*T1                                                     MATINV12
    7 GO TO N5,(8,9)                                                       MATINV13
    8 C(I)=C(I)*T1                                                         MATINV14
    9 DO 17 J=1,NN $ IF(I-J) 11,17,11                                      MATINV15
   11 T1=A(J,I) $ A(J,I)=0.0 $ DO 14 L=1,N3                                MATINV16
   14 A(J,L)=A(J,L)-T1*A(I,L)                                              MATINV17
   15 GO TO N7,(16,17)                                                     MATINV18
   16 C(J)=C(J)-T1*C(I)                                                    MATINV19
   17 CONTINUE                                                             MATINV20
   21 CONTINUE                                                             MATINV21
   22 RETURN $ END                                                         MATINV22
```