1968

ORTNRM - A Fortran Subroutine Package for the Solution of Linear Two-Point Boundary Value Problems

S. Silverston

Report Number:
68-018

ORTNRM - A Fortran Subroutine Package for the
Solution of Linear Two-Point Boundary Value Problems

S. Silverston
April, 1968

CSD TR 18

ORTNRM - A Fortran Subroutine Package for the Solution of Linear Two-Point Boundary Value Problems

S. Silverston

ORTNRM is used to solve linear two-point boundary value problems by the method of superposition with orthonormalization. See Reference.

Let x = the independent variable;

$u(x)$ = the vector of n dependent variables;

$f(x)$ = a given n-vector of functions of x;

$A(x)$ = an $n \times n$ matrix;

k = an integer $1 < k < n$;

B = a constant $(n-k)\times n$ matrix;

D = a constant $k \times n$ matrix;

C_1 = a constant $(n-k)$-vector;

C_2 = a constant k-vector.

We want to solve the problem:

$$\frac{d}{dx} u(x) = A(x)u(x) + f(x)$$
$$Bu(a) = C_1, \; Du(b) = C_2.$$

The subroutine is entered by the statement:

```
CALL ORTNRM (N,M,K,Y,DER,CO,
    A,NN,H,NP,NT,
    TEST, C,NX,
    NPO1,NPO2,ALT,
    NERR).
```

In the following discussion, we break the parameters into 5 groups.

I. System of equations:
N,M,K,Y,DER,CO

In the method of superposition, we actually obtain $k+1$ solutions $y^0, y^1, \ldots, y^k$ as follows:

Choose $y^0(a) \ni By^0(a) = C_1$

Choose $y^1, \ldots, y^k(a) \ni$

$$(y^i, y^j) = \delta_{ij}, \quad 0 \le i \le k, \quad 1 \le j \le k.$$

Solve: $\frac{d}{dx} y^0(x) = A(x)y^0(x) + f(x)$

$$\frac{d}{dx} y^i(x) = A(x)y^i(x), \quad 1 \le i \le k.$$

We then solve the system

$$D\left[\sum_{i=1}^{k} \beta_i y^i(b) + y^0(b)\right] = C_2$$

for the coefficients $\beta_1, \ldots, \beta_k$.

The solution to the original problem is then given by

$$u(x) = y^0(x) + \sum_{i=1}^{k} \beta_i y^i(x).$$

The solution $y^0(x)$ is called the particular solution. The solutions $y^i(x)$ are called base solutions.

If $C_1=0$ and $f=0$, the system is homogeneous. In this case, we let

$$By^i(a) = 0, \quad 1 \le i \le k$$

and omit the particular solution. We then will solve the system

$$D \sum_{i=1}^{k} \beta_i y^i(b) = C_2$$

for the coefficients $\beta_1, \ldots, \beta_k$.

If we also have $C_2 = 0$, then one of the β's must be chosen arbitrarily and the others computed in terms of it. In this case we only can determine $u(x)$ to within a constant multiplier. See discussion of parameter NPO2 under Output Options for the normalization convention used here.

The system of equations parameters should be set as follows:

N: Integer. No. of dependent variables n.

M: Integer. No. of solution vectors to be used for superposition. For inhomogeneous system, M=k+1. For homogeneous systems, M=k.

K: Integer. No. of base solution vectors k to be used.

Y: Real array dimensioned (N,M). Values of the vectors $y^0(a), y^1(a), \ldots, y^k(a)$, chosen as discussed above.

$$Y = \begin{matrix} y_1^0(a) & y_1^1(a) & \ldots & y_1^k(a) \\ \vdots & \vdots & & \\ y_n^0(a) & y_n^1(a) & \ldots & y_n^k(a) \end{matrix}$$

For homogeneous systems,

$$Y = \begin{matrix} y_1^1(a) & \ldots & y_1^k(a) \\ \vdots & & \vdots \\ y_n^1(a) & \ldots & y_n^k(a) \end{matrix}$$

DER: Name of subroutine for evaluation of the expressions

$$A(x)y^0(x) + f(x) \text{ and } A(x)y^i(x)$$

To be called by a statement of the form

CALL DER (X,Y,DY) with:

X = Real Value of independent variable x.

Y = Real array dimensioned (N,M). Values of solution vectors $y^0(x)$, $y^1(x), \ldots, y^k(x)$

DY = Real array dimensioned (N,M). Values of

$$\frac{d}{dx} y^0(x),\ \frac{d}{dx} y^1(x),\ \ldots,\ \frac{d}{dx} y^k(x)$$

The subroutine must (for inhomogeneous systems) compute

$$A(x)\ y^0(x) + f(x)$$

and store it in DY(1,1), DY(2,1),...,DY(N,1). It must similarly compute

$$A(x)y^i(x),\ i = 1,\ldots,k$$

and store it in DY(1,I), DY(2,I), ..., DY(N,I) , I = 2,...,M.

CO: Name of subroutine for computation of the values $\beta_1, \beta_2, \ldots, \beta_k$ in the equation

$$D[\sum_{i=1}^{k} \beta_i y^i(b) + y^0(b)] = C_2$$

To be called by a statement of the form

CALL CO (Y0,Y,BETA) for an inhomogeneous system

or by a statement of the form

CALL CO (Y,BETA) for a homogeneous system.

with:

YO: Real array dimensioned (N). Values of $y_0(b)$. Omitted for homogeneous system.

Y: Real array dimensioned (N,K). Values of $y^1(b), y^2(b), \ldots, y^k(b)$.

BETA: Real array dimensioned (K). Subroutine must compute the values of $\beta_1, \beta_2, \ldots, \beta_k$ and store them in the array BETA.

<u>None</u> of the parameters specifying system of equations, N,M,K,Y, are changed by ORTNRM.

II. <u>Interval and spacing: A,NN,H,NP,NT</u>

We solve the problem on the interval

$$S = [\min(a,b), \max(a,b)]$$

We may break S up into j sub-intervals $S_1, S_2, \ldots, S_j$

$$\{x_0=a,x_1\},\{x_1,x_2\},\ldots,\{x_{j-2},x_{j-1}\},\{x_{j-1},x_j=b\},$$

such that $a<x_1<x_2<\ldots<x_{j-1}<b$ for $a<b$,

$a>x_1>x_2>\ldots>x_{j-1}>b$ for $a>b$.

On each sub-interval S_i, $1\le i\le j$, the solution $u(x)$ will be computed <u>and</u> <u>stored</u> at n_i equally spaced points, i.e., letting

$$d_i = \frac{x_i - x_{i-1}}{n_i},$$

at the points $x_{i-1}+d_i, x_{i-1}+2d_i, \ldots, x_{i-1}+n_i d_i = x_i$.

The solution values stored at these solution points only will be available to the user for print-out as well as for use in further computation.

The intervals d_i between solution points are themselves divided into increments of integration. Specifically,

$$d_i = p_i h_i$$

where h_i is the length of the increment of integration, or step-size, for sub-interval S_i, and p_i is the number of integration steps between solution points for the sub-interval S_i.

We thus have the following relationships among $S_i, n_i, p_i, h_i, d_i, x_i, x_{i-1}$

$$|S_i| = |x_i - x_{i-1}| = n_i\ |d_i| = n_i p_i |h_i|$$

The total number of solution points on S is given by

$$t = \sum_{i=1}^{j} n_i + 1$$

The "+1" is because the initial point a is also taken as a solution point.

The interval and spacing parameters should be set as follows:

A: Real. Initial value (a) of the independent variable.

NN: Integer array dimensioned (J). The set of numbers $n_1, n_2, \ldots, n_j$ giving the number of solution points in each sub-interval S_i, respectively.

H: Real array dimensioned (J). The set of step-sizes $h_1, h_2, \ldots, h_j$ to be used on the sub-intervals S_i, respectively. The h's should be positive (>0) if a<b, and negative (<0) if a>b.

NP: Integer array dimensioned (J). The set of numbers $p_1, p_2, \ldots, p_j$ which specify the number of integration increments between solution points for the sub-intervals S_i, respectively.

NT: Integer. The total number t of solution points on S.

Note that the number of sub-intervals j does not appear in the parameter list at all. Of course if $j=1$, then NN,H, and NP need not be dimensioned in the calling program.

None of the interval and spacing parameters A,NN,H,NP,NT, are changed by ORTNRM.

This method of specifying interval and spacing is admittedly rather complicated. However, the flexibility it affords the user in varying step-size over the region as well as in specifying output, or solution, points, is quite useful for research purposes. The latter is especially valuable in the extension of this method to non-linear problems.

III. Orthonormalization
TEST,C,NX

As discussed in the reference, an orthonormalization of the form

$$Z(x) = Y(x)P,$$

where

$$Y(x) = [y^1, \ldots, y^k]$$

$$P \text{ is a } k \times k \text{ matrix}$$

$$Z(x) = [z^1, \ldots, z^k]$$

$$(z^i, z^j) = \delta_{ij}, 0 \le i \le k, \ 1 \le j \le k,$$

is performed whenever the solution vectors $y^0, y^1, \ldots, y^k$ meet some criterion to be specified. The two types of test considered here are

1) the <u>magnitude</u> test

Reorthonormalization is performed whenever $|y^i(x)| > C$ for some $i=0,1,2,\ldots,k$ Where C is a specified constant > 0.

2) the <u>angle</u> test

Reorthonormalization is performed whenever

$$57.3 \cos^{-1} \left| \frac{(y^i, y^j)}{\{(y^i,y^i)(y^j,y^j)\}^{1/2}} \right| < C, \quad 0 \le i \le k, \; 0 \le j \le k, \; i \ne j$$

where C is an angle specified in degrees.

In this program, the solution vectors $y^0, y^1, \ldots, y^k$ are tested at <u>all</u> points where they are computed, whether at "solution points" or points between the solution points.

The user also has the options of

1) reorthonormalizing at <u>every</u> point
2) not reorthonormalizing at all
3) always reorthonormalizing at the last point (x=b)

<u>TEST</u>: Integer.Flag for orthonormalization test as follows:

TEST = 0, no test (see below under <u>Iteration</u>)

= +1, magnitude test, always orthonormalize at last point (b)

= -1, magnitude test

= +2, angle test, always orthonormalize at last point (b)

= -2, angle test

C: Real array dimensioned (J). For J, see preceding section, Interval and Spacing. The orthonormalization criterion, either magnitude (for TEST = ±1) or angle (for TEST = ±2). C may also be set so as to either force orthonormalization at every point, or suppress orthonormalization. See chart below. The orthonormalization criteria can be varied for each sub-interval S_i. (The type of test made, as specified by TEST, is fixed for the whole interval S, however.) Thus C is actually the set of criteria $C_1, C_2, \ldots, C_j$ to be used on the sub-intervals S_i, respectively. Of course, as for NN, H, and NP, if j=1 C need not be dimensioned in the calling program.

	Mag. test TEST = ±1	Angle test TEST = ± 2
Test for re-orth.	$0. < C$	$0 \leq C < 90$
Re-orth. at every point	$C = 0.$	$90. \leq C$
Do not re-orth. at all	$C < 0.$	$C < 0.$

Note that the do-not-re-orthonormalize option allows ORTNRM to be used as a straight method-of-superposition package without reference to orthonormalization.

NX: Integer. The maximum number of re-orthonormalizations for which space has been allocated. (see below under Storage Space.) It is possible to have as many as

$$\sum_{i=1}^{j} n_i p_i$$

orthonormalizations.

None of the parameters TEST, C, NX, are changed by ORTNRM.

IV. Output options
NPO1, NPO2, ALT

The user may be interested in the following types of output from ORTNRM:

1) print-out of intermediate vectors. That is, the particular vector and base vectors $(y^0(x), y^1(x),\ldots,y^k(x))$, and, if orthonormalization occurred at x, the particular and base vectors $(z^0(x),z^1(x),\ldots,z^k(x))$ resulting after orthonormalization.

2) availability of the solution vector u(x) at the specified solution points to the calling program.

This may be stored, without print-out, for use in later computations. (See below under Storage Space.)

3) print-out of the solution vector at specified solution points.

In research work, it is often of interest to compare results obtained using a method being investigated with known "exact values", or with values obtained using another method. For this reason, the option of printing alternate values of the solution u(x), obtained independently of ORTNRM, along with the values computed by ORTNRM, is provided.

NPO1: Integer. Flag for the print-out of intermodiate vectors.

NPO1 = 0, omit intermediate vector print-out

= 1, print intermediate vectors at the initial point (a), last point (b), and at all points where orthonormalization has occurred.

= 2, print intermediato vectors at all points where orthonormalization has occurred and at all solution points.

When intermediate vectors are printed, the y-vectors are always given, the z-vectors are given whenever orthonormalization has occurred at the point in question.

NPO2: Integer. Flag for the output of solution vector u(x).

NPO2 = 0, solution vector u(x) is not generated. This option is useful mainly in iterative processes, when perhaps only u(b) is of interest for intermediate iterations. When this option is exercised, the subroutine may be re-entered subsequently to obtain u(x). See below under Alternate Entry.

= ± 1, solution vector u(x) generated and stored (see below under Storage Space) but not printed. When this option is exercised, the subroutine may be entered subsequently to obtain printout. See below under Alternate Entry.

= ± 2, solution vector u(x) generated, stored, and printed.

= ± 3, solution vector u(x) generated, stored, and printed along with an alternate solution vector $u_A(x)$ printed at the same solution points. This alternate solution vector is generated by a user-coded subroutine. See ALT below. The differences between the values computed by ORTNRM and the alternate values are also printed.

The sign of NPO2 serves as a solution-normalization flag for the case of a homogeneous system. Recall that when

$$f = 0,\ C_1 = 0,\ C_2 = 0$$

the system is homogeneous and the solution u is determined only to within a constant multiplier. If NPO2 is set negative, the solution u is normalized according to the convention that the first non-zero component of u(a) is made equal to 1. (If u(a)=0, of course, then u(x)=0 for all x.) If the conditions

$$f = 0 \text{ and } C_1 = 0$$

do not hold, a minus sign on NPO2 will be ignored. However, ORTNRM does not check for the third necessary condition for a homogeneous system, namely

$$C_2 = 0.$$

ALT: Name of subroutine for computing an alternate solution $u_A(x)$.

To be called by a statement of the form

CALL ALT(X,UA) with:

X: Real. Value of dependent variabl- x.

UA: Real array dimensioned (N). Values of alternate solution $u_A(x)$.

The subroutine must compute $u_A(x)$, given x, and store values in UA. This subroutine is called only if NPO2 is set to ± 3. If NPO2 is not set to ± 3, a dummy name may be used for ALT in the CALL ORTNRM statement.

Neither of the parameters specifying output, NPO1 or NPO2, are changed by ORTNRM.

V. Error flag
NERR

There is only one error condition to be flagged. This is the case when not enough space has been allocated for storage of re-orthonormalization parameters. In other words, the case in which NX is too small.

NERR: Integer variable. If there has been no error, NERR will have been set to 0 on return from ORTNRM. If the above error condition exists, NERR is set to 1 on return from ORTNRM. In case of error, a note is also printed giving details.

The user must include the names for DER, CO, and, if the alternate solution option is used, ALT in an EXTERNAL statement in the calling program.

Storage Space

The user must allocate working storage space for use by ORTNRM. This is done via the labeled COMMON block /SCRATCH/. The ORTNRM package will use the first L locations of /SCRATCH/, where

L = (NT+6)*N*M + (3*K+K*(K-1)/2+1)*NX + K + NT + 1

and the other variables are as in the CALL ORTNRM statement. (For homogeneous systems, L is actually less than the above, by K*NX locations.)

Upon return from ORTNRM, the first (N,NT) locations of /SCRATCH/ will contain the solution u(x), provided solution generation has been requested. Thus, the full solution u(x), as evaluated at the NT solution points, becomes available to the calling program. For example, suppose

N = 4, M = 3, K = 2, NT = 11, NX = 25.

The user might include the following statement in the calling program:

```
COMMON /SCRATCH/ U(4,11), S(374)
```

Auxilliary COMMON Blocks

ORTNRM uses, in addition to /SCRATCH/, COMMON blocks named

/KKKK/ and /MMMM/.

ORTNRM does not use blank COMMON.

Alternate Entry

ORTNRM may be re-entered to effect solution generation or solution print-out where this has been temporarily suppressed via the flag NPO2 as described above.

1) To resume processing after solution generation has been suppressed (by setting NPO2=0), use

```
CALL SOLN (...)
```

2) To resume processing after solution print-out has been suppressed (by setting NPO2=±1), use

```
CALL PRM (...)
```

The argument lists for SOLN and PRM are exactly the same as that for ORTNRM, <u>except</u> that the value of NPO2 must be changed.

For CALL SOLN, NPO2 must be ± 1, ± 2, or ± 3.

For CALL PRM, NPO2 must be ± 2 or ± 3.

Iteration

In certain iterative processes, it may be necessary to establish a set of orthonormalizing transformations on a first pass, and then use the <u>same</u> transformations at the same points on subsequent passes. (On these subsequent passes, the transformations may not actually affect <u>strict</u> orthonormalization; however, this may be desirable for purposes of keeping all iterations uniform.) This can be accomplished via the parameter TEST. If TEST is set to 0, transformations as established on a previous pass and stored in COMMON block /SCRATCH/ will be used. In this case, testing against orthonormalization criteria, as well as computation of new orthonormalizing transformation coefficients, will be omitted.

Backward Integration

Generated solutions will be stored and printed in the direction of <u>increasing</u> x, regardless of whether $a<b$ or $b<a$. This is done for the following reasons:

ORTNRM is primarily useful for problems in which there is some numerical instability. In problems of this type, the instability may often exist for one direction of integration but not for the other. Generating the solution always in the same direction facilitates comparison when the user wants to try solving a problem in both directions.

Another application of ORTNRM is in the area of unstable initial-value problems. Such problems can be worked backwards as boundary-value problems. In this case too it is convenient to have the solution stored and printed in the "forward" direction.

Deck Set-up

The ORTNRM package consists of the following subroutines:

ORTNRM
ORTSUB
RUNKUT
NUGO
ARRAY
RND
FLIP
BLOCK DATA

The largest of these subroutines, ORTSUB, needs 50600_8 locations to compile on the CDC 6500.

The package uses COMMON blocks named

/SCRATCH/ (discussed above)
/KKKK/
/MMMM/

The ORTNRM package should be placed after the calling program in the deck to allow proper loading of COMMON blocks.

Reference 1) Conte, The Numerical Solution of Linear Boundary Value Problems, SIAM Review, Vol. 8, No. 3, July, 1966.

APPENDIX

Fortran listing of ORTNRM package

```
      SUBROUTINE ORTNRM
C         PARAMETERS SPECIFYING SYSTEM OF EQUATIONS
     1         ( N, M, K, Y, DER, CO,
C         PARAMETERS DEFINING INTERVAL AND SPACING
     2         A, NM, H, NP, MT,
C         PARAMETERS SPECIFYING ORTHONORMALIZATION
     3         TEST, C, NX,
C         SPECIFICATION OF USERS OUTPUT OPTIONS
     4         NPO1, NPO2, ALT,
C         ERROR FLAG
     5         NERR )
      COMMON /SCRATCH/ S(1)        /KKKK/ L,K1,MTK
      EXTERNAL DER,CO,ALT
      NERR = 0
      X0 = A
      NXM = N*M
      NXN = NX*MT + K1 + 1
      NY = NXN + MT
      CALL ARRAY (Y,S(NY),N,M)
      NM = NY + 6*NXM
      KO = K*(K-1)/2
      NR = NM + KO*NX
      KRMX = K*NX
      NB = NR + KRMX
      NL = NB + KRMX + K
      NXL = NX + 1
      NA = NL + NXL
      CALL ORTSUB (X0,S(K1+1),S(NXN),S(NM),S(NR),S(NA),S(NB),S(NL),
     1 N,M,K,KO,MT,NXL,S(NY),DER,CO,ALT,TEST,NPO1,NPO2,NERR,NXM,NM,H,
     2 NP,C)
      RETURN
      END

      SUBROUTINE ORTSUB (X,Z,XN,OMEGA,R,ALPHA,BETA,LX,N,M,KR,KO,MT,
     1 NX,Y,DERIV,COEFFS,EXACT,MD,NPO1,NPO2,NERR,NXM,NM,H,NP,A)
      DIMENSION Z(N,MT),XN(MT),OMEGA(KO,NX),R(KR,NX),ALPHA(KR,NX),
     1 BETA(KR,NX),Y(N,M,6),NZ(40),NO(2),YY(2),XB(2),LX(NX)
     2 ,NM(1),H(1),NP(1),A(1)
      DIMENSION FMT(6),BEG(2),FE(6),HED(6)
      EQUIVALENCE (NO(2),NZ)
      INTEGER P,Q,S,T,U,V
      LOGICAL HOM,NEG, ORE,LAST,ENDPT,LASTBK
     1 ,OLDCO,MAGTST,NOPO
      DOUBLE PRECISION D,DG
      EXTERNAL DERIV
      DATA NO(1),NZ /1H0,1H1,1H2,1H3,1H4,1H5,1H6,1H7,1H8,1H9,2H10,2H11,
     1 2H12,2H13,2H14,2H15,2H16,2H17,2H18,2H19,2H20,2H21,2H22,2H23,
     1 2H24,2H25,2H26,2H27,2H28,2H29,2H30,2H31,2H32,2H33,2H34,2H35,
     1 2H36,2H37,2H38,2H39,2H40/,
     2 YY,XB /1HY,1HZ,1HX,1H /
      DATA FMT(1),FMT(4),H6,SP6 /5H(1H /, 5H/(12X,
     1 9H/(1H0 2 X, 9H/(A1,2 X /,
     1 FMT(6) /2H))/,
     1 BEG /6HF12.4, ,3H12X /,
     2 FE /6H E18.8,6H2E18.8,6H3E18.8,6H4F18.8,6H5E18.8,6H6E18.8/,
     3 HED /1 H(1H0 7X A1, 1 H, 12X 6(A1, 1 H, A2, 15X),
     4 10H /(1H0 2 X, 1 H 6(A1, A2,,7H 15X))) /
C
C                    INITIALIZE
C
```

```
      K = 2
      HOM = KR.EQ.M
      IF (HOM) K = 1
      K1 = K + 1
      KM1 = K - 1
      NORXS = 0
      U = 1
      V = 1
      IN = 1
      IL = NT
      MORE = KR.GT.1
      LAST = ND.GT.0
      OLDCO = ND.EQ.0
      MAGTST = IABS(ND).EQ.1 .OR. .NOT.MORE
      P = 0
      NTC = 1
      XN(1) = X
      NNC = 0
      NPLP = 1
      NHLP = 1
      GO TO 17
C
C                   START INTEGRATION LOOP
C
   10 NNC = NNC + 1
      NPLP = NN(NNC)
      NHLP = NP(NNC)
      DX = H(NNC)*FLOAT(NHLP)
      NTC = NTC + NPLP
      NF = 0
      IF (A(NNC).GE.0.) GO TO 13
      NF = -1
      GO TO 17
   13 IF (MAGTST) GO TO 15
      C = COS(A(NNC)/57.3)
      IF (A(NNC).GE.90.) NF = 1
      GO TO 17
   15 A2 = A(NNC)**2
      IF (A(NNC).EQ.0.) NF = 1
   17 DO 301 NPC = 1,NPLP
      LASTBK = NPC.EQ.NPLP .AND. NTC.EQ.NT
      XN(U+1) = XN(U) + DX
      DO 301 NHC = 1,NHLP
      P = P + 1
      NOPO = NHC.LT.NHLP
      IF (P.EQ.1) GO TO 200
      ENDPT = LASTBK .AND. .NOT.NOPO
      CALL RUNKUT (X,Y,Y(1,1,2),NXM,H(NNC),DERIV)
      U = U + 1

                    BRANCH ON ORTHONORMALIZATION

      IF (OLDCO) IF (P-LX(V)) 55,70,55
      IF (NF) 55,19,100
   19 IF (ENDPT .AND. LAST) GO TO 100

                    TEST FOR ORTHOGONALITY OR MAGNITUDE

      G = 0.
      MM = M
```

```
      DO 30 I = 1,M
      IF (MAGTST) MM = I
      DO 30 J = I,MM
      E = 0.
      DO 20 L = 1,N
   20 E = E + Y(L,I,1)*Y(L,J,1)
      Y(I,J,5) = E
   30 IF (E.GT.G) G = E
      IF (MAGTST) GO TO 52
      DO 40 I = 1,M
      DO 40 J = I,M
   40 Y(I,J,5) = Y(I,J,5)/G
      T = 1
      M1 = M - 1
      DO 45 I = 1,M1
      L = I + 1
      DO 45 J = L,M
   45 IF (Y(I,J,5)**2 .GT. C**2*Y(I,I,5)*Y(J,J,5)) GO TO 50
      T = 0
   50 IF (T) 100,55,100
   52 DO 54 I = 1,M
   54 IF (Y(I,I,5).GT.A2) GO TO 100
C
C                    NO RE-ORTHONORMALIZATION
C
   55 IF (NOPO) GO TO 65
   57 DO 60 I = 1,M
      DO 60 J = 1,N
   60 Z(J,I,U) = Y(J,I,1)
C                    IS PRINT-OUT INDICATED
   65 REG = .TRUE.
      IF (NPO1.GT.0 .AND. ( ENDPT   .OR. P.EQ.1)) GO TO 222
      IF (NPO1 - 2) 300,220,300
C
C                    RE-ORTHONORMALIZATION USING OLD COEFFICIENTS
C
C                         CHECK FOR SUFFICIENCY OF STORAGE
   70 IF (V.LT.NX) GO TO 74
      WRITE (6,71)
   71 FORMAT (95H0INSUFFICIENT STORAGE FOR ORTHONORMALIZATION PARAMETERS
     1 DISCOVERED DURING ATTEMPTED COMPUTATION / 78H WITH PREVIOUSLY DET
     2ERMINED PARAMETERS. ERROR RETURN TO CALLING PROGRAM GIVEN. /
     3 32H SOLUTION GENERATION SUPPRESSED. )
      NERR = 1
      RETURN
C                    ORTHOGONALIZATION
   74 DO 80 Q = 1,N
      L = 0
      DO 80 I = K,M
      Z(Q,I,U) = Y(Q,I,1)
      IF (I.EQ.K) GO TO 80
      I1 = I - 1
      DO 75 J = K,I1
      L = L + 1
   75 Z(Q,I,U) = Z(Q,I,U) - OMEGA(L,V)*Y(Q,J,1)
   80 CONTINUE
                     NORMALIZATION
      DO 85 I = K,M
      IR = I - KM1
      DO 85 J = 1,N
```

```
   85 Z(J,I,U) = R(IR,V)*Z(J,I,U)                                  ORT01800
C                    BRANCH ON HOMOGENEITY                          ORT01810
   90 IF (HOM) GO TO 190                                            ORT01820
      GO TO 183                                                     ORT01830
C                                                                   ORT01840
C                    RE-ORTHONORMALIZATION WITH NEW COEFFICIENTS    ORT01850
C                                                                   ORT01860
  100 IF (V.NE.NX) GO TO 105                                        ORT01870
      NORXS = NORXS + 1                                             ORT01880
      V = 1                                                         ORT01890
      IN = U                                                        ORT01900
  105 LX(V) = P                                                     ORT01910
C                    FIRST VECTOR AND MOD**2                        ORT01920
      E = 0.                                                        ORT01930
      DO 110 I = 1,N                                                ORT01940
      E = E + Y(I,K,1)**2                                           ORT01950
  110 Z(I,K,U) = Y(I,K,1)                                           ORT01960
      R(1,V) = 1./E                                                 ORT01970
C                    BEGIN MAJOR ORTHONORMALIZATION LOOP            ORT01980
      IF (.NOT.MORE) GO TO 165                                      ORT01990
      L = 0                                                         ORT02000
      DO 160 I = K1,M                                               ORT02010
      I1 = I - 1                                                    ORT02020
      L0 = L                                                        ORT02030
C                    BEGIN LOOP TO DETERMINE OMEGAS                 ORT02040
      DO 140 J = K,I1                                               ORT02050
      L = L + 1                                                     ORT02060
C                    OBTAIN FIRST TERM OF EXPRESSION FOR OMEGA (IN D. P.)ORT02070
      D = 0.                                                        ORT02080
      DO 120 Q = 1,N                                                ORT02090
  120 D = D + Y(Q,I,1)*Z(Q,J,U)                                     ORT02100
      IR = J - KM1                                                  ORT02110
      DG = D*R(IR,V)                                                ORT02120
C                    COMPUTE SUBSEQUENT TERMS IN OMEGA IF NECESSARY (IN DORT02130
      S = J + 1                                                     ORT02140
      IF (S.GT.I1) GO TO 140                                        ORT02150
      DO 130 Q = S,I1                                               ORT02160
      D = 0.                                                        ORT02170
      DO 125 T = 1,N                                                ORT02180
  125 D = D + Y(T,I,1)*Z(T,Q,U)                                     ORT02190
      IR = Q - KM1                                                  ORT02200
      IW = (IR-2)*(IR-1)/2 + J - KM1                                ORT02210
  130 DG = DG - D*R(IR,V)*OMEGA(IW,V)                               ORT02220
  140 OMEGA(L,V) = DG                                               ORT02230
C                    END OF OMEGA LOOP                              ORT02240
C                    ORTHOGONALIZATION                              ORT02250
      DO 150 Q = 1,N                                                ORT02260
      L = L0                                                        ORT02270
      Z(Q,I,U) = Y(Q,I,1)                                           ORT02280
      DO 150 J = K,I1                                               ORT02290
      L = L + 1                                                     ORT02300
  150 Z(Q,I,U) = Z(Q,I,U) - OMEGA(L,V)*Y(Q,J,1)                     ORT02310
      IR = I - KM1                                                  ORT02320
      E = 0.                                                        ORT02330
      DO 155 Q = 1,N                                                ORT02340
  155 E = E + Z(Q,I,U)**2                                           ORT02350
  160 R(IR,V) = 1./E                                                ORT02360
C                    END MAJOR ORTHONORMALIZATION LOOP              ORT02370
C                    NORMALIZATION                                  ORT02380
  165 DO 170 I = K,M                                                ORT02390
```

```
      IR = I - KM1                                                      ORT02400
      R(IR,V) = SQRT(R(IR,V))                                           ORT02410
      DO 170 J = 1,N                                                    ORT02420
  170 Z(J,I,U) = R(IR,V)*Z(J,I,U)                                       ORT02430
C                         CALCULATE ALPHAS (IN D. P.)                   ORT02440
      IF (HOM) GO TO 190                                                ORT02450
      DO 180 I = 2,M                                                    ORT02460
      D = 0.                                                            ORT02470
      DO 175 J = 1,N                                                    ORT02480
  175 D = D + Y(J,1,1)*Z(J,I,U)                                         ORT02490
  180 ALPHA(I-1,V) = D                                                  ORT02500
C                            ORTHOGONALIZE PARTICULAR SOLUTION          ORT02510
  183 DO 185 J = 1,N                                                    ORT02520
      Z(J,1,U) = Y(J,1,1)                                               ORT02530
      DO 185 I = 2,M                                                    ORT02540
  185 Z(J,1,U) = Z(J,1,U) - ALPHA(I-1,V)*Z(J,I,U)                       ORT02550
C                         IS PRINT-OUT INDICATED                        ORT02560
  190 REG = .FALSE.                                                     ORT02570
      V = V + 1                                                         ORT02580
      IF (NPO1) 222,290,222                                             ORT02590
C                                                                       ORT02600
C                         PRINT-OUT OF VECTORS                          ORT02610
C                                                                       ORT02620
C                         FIRST POINT - SET UP LIMITS - PRINT HEADING   ORT02630
  200 IF (NPO1.EQ.0) GO TO 57                                           ORT02640
      NK = 2 - K                                                        ORT02650
      NBK = (M-1)/6 + 1                                                 ORT02660
      NXS = M - 6*(NBK-1)                                               ORT02670
      HED(4) = H6                                                       ORT02680
      IF (M.EQ.6) HED(4) = SP6                                          ORT02690
      WRITE (6,205)                                                     ORT02700
  205 FORMAT (7H1ORTNRM 42X 20HINTERMEDIATE VECTORS)                    ORT02710
      DO 210 I = 1,2                                                    ORT02720
  210 WRITE (6,HED) XB(I),(YY(I),NZ(T),T=NK,KR)                         ORT02730
      GO TO 57                                                          ORT02740
C                         PRINT Y-VECTORS                               ORT02750
  220 IF (NOPO) GO TO 300                                               ORT02760
  222 J = -5                                                            ORT02770
      FMT(2) = BEG(1)                                                   ORT02780
      FMT(3) = EE(6)                                                    ORT02790
      FMT(5) = EE(6)                                                    ORT02800
      DO 240 I = 1,NBK                                                  ORT02810
      J = J + 6                                                         ORT02820
      L = J + 5                                                         ORT02830
      IF (I.NE.NBK) GO TO 225                                           ORT02840
      FMT(3) = EE(NXS)                                                  ORT02850
      FMT(5) = EE(NXS)                                                  ORT02860
      L = M                                                             ORT02870
  225 IF (I.NE.1) GO TO 230                                             ORT02880
      WRITE (6,FMT) X,((Y(S,T,1),T=J,L),S=1,N)                          ORT02890
      FMT(2) = BEG(2)                                                   ORT02900
      GO TO 240                                                         ORT02910
  230 WRITE(6,FMT) ((Y(S,T,1),T=J,L),S=1,N)                             ORT02920
  240 CONTINUE                                                          ORT02930
      IF (REG) GO TO 300                                                ORT02940
                          PRINT Z-VECTORS                               ORT02950
      J = -5                                                            ORT02960
      FMT(3) = EE(6)                                                    ORT02970
      FMT(5) = EE(6)                                                    ORT02980
      DO 250 I = 1,NBK                                                  ORT02990
```

```
      J = J + 6
      L = J + 5
      IF (I.NE.NBK) GO TO 250
      FMT(3) = EE(NXS)
      FMT(5) = EE(NXS)
      L = M
  250 WRITE (6,FMT) ((Z(S,T,U),T=J,L),S=1,N)
C
C                    RE-INITIALIZE
C
  290 DO 295 I = 1,M
      DO 295 J = 1,N
  295 Y(J,I,1) = Z(J,I,U)
      CALL NUGO
C
  300 IF (NOPO) U = U - 1
  301 CONTINUE
      IF (NTC.LT.NT) GO TO 10
      LX(V) = P + 1
      CALL NUGO
C
C                    END INTEGRATION LOOP
C
C                         CHECK FOR ERROR - INSUFFICIENT COEFFICIENT STORA
C
      IF (NORXS.EQ.0) GO TO 305
      NERR = 1
      NX1 = NX - 1
      NEED = NX1*NORXS + V - 1
      WRITE (6,302) NEED,NX1
  302 FORMAT (7H1ORTNRM // 75H INSUFFICIENT STORAGE HAS BEEN ALLOCATED F
     1OR ORTHONORMALIZATION PARAMETERS. / 10X I4, 14H BLOCKS NEEDED /
     2 10X I4, 17H BLOCKS ALLOCATED //
     1 43H THE SOLUTION GENERATED WILL BE INCOMPLETE.)
  305 IF (NPO2.EQ.0) RETURN
C
C                    CALCULATE BETAS AT END POINT
C
      ENTRY SOLN
      IF (HOM) CALL COEFFS (Z(1,1,NT),BETA(1,V))
      IF (.NOT. HOM) CALL COEFFS (Z(1,1,NT),Z(1,2,NT),BETA(1,V))
C
C                    CALCULATE INTERMEDIATE BETAS
C
      Q = V
  308 IF (Q.EQ.1) GO TO 340
      S = Q - 1
      DO 310 I = 1,KR
      E = BETA(I,Q)
      IF (.NOT. HOM) E = E - ALPHA(I,S)
  310 Y(I,1,1) = R(I,S)*E
      KO = 1
      DO 335 I = 1,KR
      BETA(I,S) = Y(I,1,1)
      IF (I.EQ.KR) GO TO 335
      KO = KO + 1
      DO 330 K = KO,KR
      L = (K-1)*(K-2)/2 + I
  330 BETA(I,S) = BETA(I,S) - OMEGA(L,S)*Y(K,1,1)
  335 CONTINUE
```

```
      Q = S
      GO TO 308
C
C                    REVERSE ARRAYS IF INTEGRATION WAS BACKWARDS
C
  340 IF (H(1).GT.0.) GO TO 350
      CALL FLIP (XN,NT,1)
      DO 342 I = 1,M
      DO 342 J = 1,N
  342 CALL FLIP (Z(J,I,1),NT,NXM)
      IF (V.EQ.1) GO TO 350
      DO 346 I = 1,KR
  346 CALL FLIP (BETA(I,1),V,KR)
      J = V - 1
      P = P + 2
      DO 348 I = 1,J
  348 LX(I) = P - LX(I)
      CALL FLIP (LX,J,1)
      IF (NORXS.EQ.0) GO TO 350
      IL = NT - IN + 1
      IN = 1
C
C                     NORMALIZATION FOR HOMOGENEOUS SYSTEM
C
  350 G = 1.
      IF (.NOT. HOM .OR. NPO2.GT.0) GO TO 370
      DO 360 I = 1,N
      G = 0.
      DO 355 J = 1,M
  355 G = G + BETA(J,1)*Z(I,J,1)
  360 IF (G.NE.0.) GO TO 365
      G = 1.
  365 G = 1./G
C
C                     COMP. AND PRINT LOOP
C
  370 REG = .FALSE.
  380 NPA2 = IABS(NPO2)
      NPG = 0
      NBK = 56/(N+1)
      NC = 0
      Q = 0
      V = 1
      MORE = N.GT.KR
      KR1 = KR + 1
      P = 1
      NNC = 1
      NPC = 0
      IF (IN.EQ.1) GO TO 388
      U = 1
      GO TO 384
  383 NNC = NNC + 1
  384 NPLP = NN(NNC)
      DO 386 NPC = 1,NPLP
      P = P + NP(NNC)
      U = U + 1
  386 IF (U.EQ.IN) GO TO 388
      GO TO 383
C
C                     CALCULATE SOLUTIONS
```

```
388 DO 600 I = IN,IL
390 IF (LX(V).GT.P) GO TO 400
    V = V + 1
    GO TO 390
400 IF (REG) GO TO 610
    IF (HOM) GO TO 420
    DO 410 J = 1,N
    Y(J,1,1) = Z(J,1,I)
    DO 410 K = 2,M
410 Y(J,1,1) = Y(J,1,1) + BETA(K-1,V)*Z(J,K,I)
    GO TO 440
420 DO 430 J = 1,N
    Y(J,1,1) = 0.
    DO 425 K = 1,M
425 Y(J,1,1) = Y(J,1,1) + BETA(K,V)*Z(J,K,I)
430 Y(J,1,1) = G*Y(J,1,1)
440 IF (NPC.LT.NN(NNC)) GO TO 445
    NNC = NNC + 1
    NPC = 0
445 NPC = NPC + 1
    P = P + NP(NNC)

                  PAGE HEADING

    IF (NPA2.EQ.1) GO TO 580
    NC = NC + 1
    IF (MOD(NC,NBK).NE.1) GO TO 480
    NPG = NPG + 1
    WRITE (6,450) NPG
450 FORMAT (7H1ORTNRM 10X 14HSOLUTION (PAGE I3, 1H) )
    IF (NPA2.EQ.2) WRITE (6,460)
    IF (NPA2.GE.3) WRITE (6,470)
460 FORMAT(1H07X1HX11X 4HBETA 16X 1HU)
470 FORMAT(1H07X1HX11X 4HBETA 16X 1HU 13X 9HU COMPARE 9X 4HDIFF )

                  PRINT SOLUTIONS

480 IF (NPA2.GE.3) GO TO 520
    WRITE (6,490) XN(I),(BETA(J,V),Y(J,1,1),J=1,KR)
490 FORMAT (1H0F11.4, 2E18.8 / (E30.8, E18.8) )
    IF (MORE) WRITE (6,500) (Y(J,1,1),J=KR1,N)
500 FORMAT (30X E18.8)
    IF (REG) GO TO 600
    GO TO 580
520 CALL EXACT (XN(I),Y(1,1,2))
    DO 530 J = 1,N
530 Y(J,1,3) = Y(J,1,1) - Y(J,1,2)
    WRITE (6,540) XN(I),(BETA(J,V),(Y(J,1,L),L=1,3),J=1,KR)
540 FORMAT (1H0F11.4, 3E18.8, E13.3 / (E30.8, 2E18.8, E13.3) )
    IF (MORE) WRITE (6,550) ((Y(J,1,L),L=1,3),J=KR1,N)
550 FORMAT (30X 2E18.8, E13.3)
    IF (REG) GO TO 600

                  STORE SOLUTIONS

580 DO 590 J = 1,N
    Q = Q + 1
590 Z(Q,1,1) = Y(J,1,1)
600 CONTINUE
```

```
C
      RETURN
C
C                                 PRINT-ONLY SECTION
C
      ENTRY PRM
      REG = .TRUE.
      GO TO 380
  610 DO 620 T = 1,N
      Q = Q + 1
  620 Y(T,1,1) = Z(Q,1,1)
      GO TO 440
      END

      SUBROUTINE RUNKUT (X,Y,D,N,H,DERIV)
      DIMENSION Y(N),D(N,5),E(2)
      COMMON /MMMM/ MID
      DOUBLE PRECISION XDP,H2,EDP
      LOGICAL MID
      EQUIVALENCE (EDP,E)
      H2 = 0.5*DBLE(H)
      H6 = H/6.
      IF (MID) GO TO 20
      DO 10 I = 1,N
      EDP = Y(I)
      D(I,1) = E(1)
   10 D(I,2) = E(2)
      MID = .TRUE.
      XDP = X
   20 CALL DERIV (X,Y,D(1,4))
      DO 30 I = 1,N
   30 D(I,3) = Y(I) + SNGL(H2)*D(I,4)
      XDP = XDP + H2
      CALL DERIV (SNGL(XDP),D(1,3),D(1,5))
      DO 40 I = 1,N
      D(I,4) = D(I,4) + 2.*D(I,5)
   40 D(I,3) = Y(I) + SNGL(H2)*D(I,5)
      CALL DERIV (SNGL(XDP),D(1,3),D(1,5))
      DO 50 I = 1,N
      D(I,4) = D(I,4) + 2.*D(I,5)
   50 Y(I) = Y(I) + H*D(I,5)
      XDP = XDP + H2
      X = RND(XDP)
      CALL DERIV (X,Y,D(1,5))
      DO 60 I = 1,N
      D(I,4) = D(I,4) + D(I,5)
      E(1) = D(I,1)
      E(2) = D(I,2)
      EDP = EDP + H6*D(I,4)
      D(I,1) = E(1)
      D(I,2) = E(2)
   60 Y(I) = RND(EDP)
      RETURN
      END

      SUBROUTINE NUGO
      COMMON /MMMM/ MID
      LOGICAL MID
      MID = .FALSE.
      RETURN
      END
```

```
      FUNCTION RND (D)
      DIMENSION D(2)
      EQUIVALENCE (A,J)
      RND = D
      IF ( (ABS(D(2)).A.4000000000000000B) .EQ. 0) RETURN
      A = ABS(D)
      J = J + 1
      A = A.O.4000000000000000B
      RND = SIGN(A,D)
      RETURN
      END
      SUBROUTINE ARRAY (Y,S,N,M)
      DIMENSION Y(N,M),S(N,M)
      DO 10 I = 1,M
      DO 10 J = 1,N
   10 S(J,I) = Y(J,I)
      RETURN
      END
      SUBROUTINE FLIP (Y,N,J)
      DIMENSION Y(1)
      L = (N/2 - 1)*J + 1
      M = N*J + 1
      DO 10 I = 1,L,J
      M = M - J
      E = Y(M)
      Y(M) = Y(I)
   10 Y(I) = E
      RETURN
      END
      BLOCK DATA
      COMMON /KKKK/ L,K1,NTK,NK  /MMMM/ MID
      LOGICAL MID
      DATA K1 /0/
      DATA MID /F/
      END
```