COO-1469-0205

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN
URBANA, ILLINOIS 61801

AN OPERATING MANUAL FOR DIFMF3*

A Program to Solve Initial Conditions in Simultaneous Differential Equations

by

Bill van Melle

*Supported in part by research grant US AEC AT(11-1)1469

UIUCDCS-F-72-870
April 1972

3.1.3 The Steady-State Package (B. van Melle)

DIFMF3 is a subroutine used to solve the steady-state problem of ordinary differential equations. It will operate on a mixture of differential and algebraic equations, and can be used merely to solve a system of simultaneous equations, which it handles in the same manner. To solve the system, a combination of the Newton method and a first-order predictor-corrector method is used.

The Method

The problem is presented to us in the form

$$f(y, y', t) = 0.$$

However, for the steady-state problem, t is constant, and in each component of the vectors y and y' either the y or the y' is held constant, so the problem can be written

$$f(y) = 0 \tag{1}$$

where the vector y contains the unknowns.

To solve (1) we invent an independent variable s, and attempt to integrate

$$\frac{df}{ds} = -f \tag{2}$$

from 0 to ∞. The exact solution of (2) is

$$f = f(y_0)e^{-s}$$

which approximates $f = 0$ as s becomes large. Hopefully, a reasonable solution is obtained while s is still finite.

To obtain an expression for $y' = dy/ds$, apply the chain rule to (2)

$$\frac{\partial f}{\partial y} \cdot \frac{dy}{ds} = -f(y)$$

$$y' = -\left(\frac{\partial f}{\partial y}\right)^{-1} f(y) \tag{3}$$

We use Euler's method for a predictor:

$$y_n = y_{n-1} + hy'_{n-1} \text{ (h the step size)} \tag{4}$$

For a corrector we use the formula:

$$y_n = y_{n-1} + h(\alpha y'_n + (1-\alpha)\, y'_{n-1}) \tag{5}$$

with $0 \leq \alpha \leq 1$. When $\alpha = 0$ the corrector has no effect, and when $\alpha = 0$ and $h = 1$, the entire method is simply Newton's method

$$y_n = y_{n-1} - \left(\frac{\partial f}{\partial y}\right)^{-1} f(y_{n-1}).$$

Rewriting (5) and substituting for y',

$$y_n - h\alpha y'_n = y_{n-1} + h(1-\alpha)\, y'_{n-1}$$

$$y_n + h\alpha\left(\frac{\partial f}{\partial y}\right)^{-1} f(y_n) = y_{n-1} + hy'_{n-1} - \alpha hy'_{n-1}$$

Now write $y_n = y_n^{(0)} + \Delta y_n$, where $y_n^{(0)}$ is the predicted value from (4) and Δy_n is what we seek an expression for.

$$y_n^{(0)} + \Delta y_n + h\alpha\left(\frac{\partial f}{\partial y}\right)^{-1} f(y_n^{(0)} + \Delta y_n) = y_n^{(0)} - \alpha hy'_{n-1}$$

Use a first-order Taylor expansion to approximate $f(y_n^{(0)} + \Delta y_n)$:

$$\Delta y_n + h\alpha(\frac{\partial f}{\partial y})^{-1} [f(y_n^{(0)}) + (\frac{\partial f}{\partial y}) \Delta y_n] = - \alpha h y'_{n-1}$$

$$(1+h\alpha) \Delta y_n = -h\alpha(\frac{\partial f}{\partial y})^{-1} f(y_n^{(0)}) - \alpha h y'_{n-1}$$

$$\Delta y_n = - \frac{\alpha}{1+h\alpha} [h(\frac{\partial f}{\partial y})^{-1} f(y_n^{(0)}) + h y'_{n-1}] \tag{5}$$

To obtain a similar expression for hy'_n, approximate (3) with a first-order Taylor expansion and substitute from (5)

$$\begin{aligned} hy'_n &= -h(\frac{\partial f}{\partial y})^{-1} f(y_n) \\ &= -h(\frac{\partial f}{\partial y})^{-1} (f(y_n^{(0)}) + (\frac{\partial f}{\partial y}) \Delta y_n) \\ &= -h(\frac{\partial f}{\partial y})^{-1} f(y_n^{(0)}) - h[- \frac{\alpha}{1+h\alpha} [h(\frac{\partial f}{\partial y})^{-1} f(y_n^{(0)}) + hy'_{n-1}]] \\ &= - \frac{1}{1+\alpha h} [h(\frac{\partial f}{\partial y})^{-1} f(y_n^{(0)})] + (\frac{\alpha h}{1+h\alpha}) hy'_{n-1} \\ &= hy'_{n-1} - \frac{1}{1+\alpha h} [h(\frac{\partial f}{\partial y})^{-1} f(y_n^{(0)}) - hy'_{n-1}] \end{aligned}$$

Thus, the corrector step can be written

$$\Delta = \frac{1}{1+h\alpha} [h(\frac{\partial f}{\partial y})^{-1} f(y_n^{(0)}) + hy'_{n-1}]$$

$$y_n = y_n^{(0)} - \alpha\Delta$$

$$hy'_n = hy'_{n-1} - \Delta$$

By varying α between 0 and 1 we can alternate between the Newton method (if h is near 1) and this predictor-corrector method, hopefully choosing the more effective scheme for the situation. The Newton method is quite efficient when in the neighborhood of a solution, but can be erratic elsewhere, while the other method generally finds a solution, but very slowly.

DIFMF3

A. Use

The user supplies several subroutines to be used by DIFMF3. The system to be solved is defined by the routines S1, S2, and DIFFUN. S1 and S2 perform computations involving only time and global variables, which remain constant during solution of this problem (these routines may be dummy). DIFFUN consists of equations of the form DY(J) = Jth equation. The vector DY then holds the values of the functions f(y, y', t) described in the previous section. The variables y are separated into linear variables which appear without derivatives, stored in the vector YL, and variables with derivatives, which are stored in the first row of an array Y (in Y(1,*)) and their first derivatives with respect to time stored in the second row (Y(2,*)).

A subroutine MATSET computes the Jacobian matrix $(\frac{\partial f}{\partial y})$, and routine MATINV computes its inverse. MATMUL performs the multiplication $(\frac{\partial f}{\partial y})^{-1}$ f(y). The present DIFMF3 uses routines which utilize sparse techniques, but ordinary dense matrix routines could also be used.

If any of the equations in DIFFUN inherently places a restriction on the range of values y can assume, e.g. a square-root function, another routine, RANGER, must be supplied. This routine checks the restricted variables. If they are out of bounds, they are adjusted to legal values and a return flag is set to reflect this.

In the simplest form of the problem, the derivatives in Y(2,*) are all set to 0, and some sort of initial guess is given to the variables in Y(1,*) and YL(*). The user may wish instead to give a Y(1,J) a constant value and solve for its derivative Y(2,J). The information about which variables to solve for he supplies in the vector IND, whose J-th component is 1 or 2, if he wants to solve for Y(1,J) or Y(2,J), respectively.

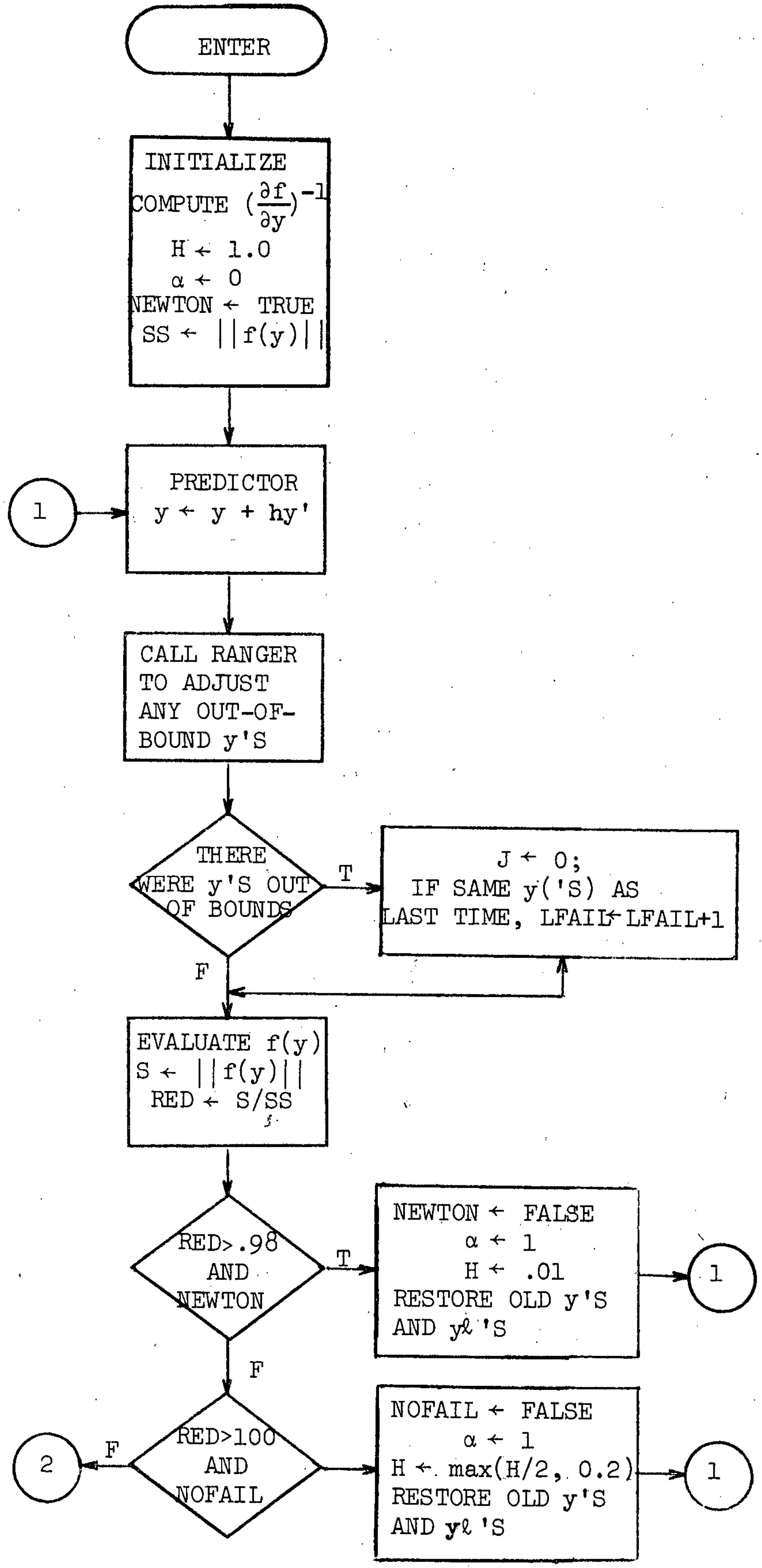
B. Strategy

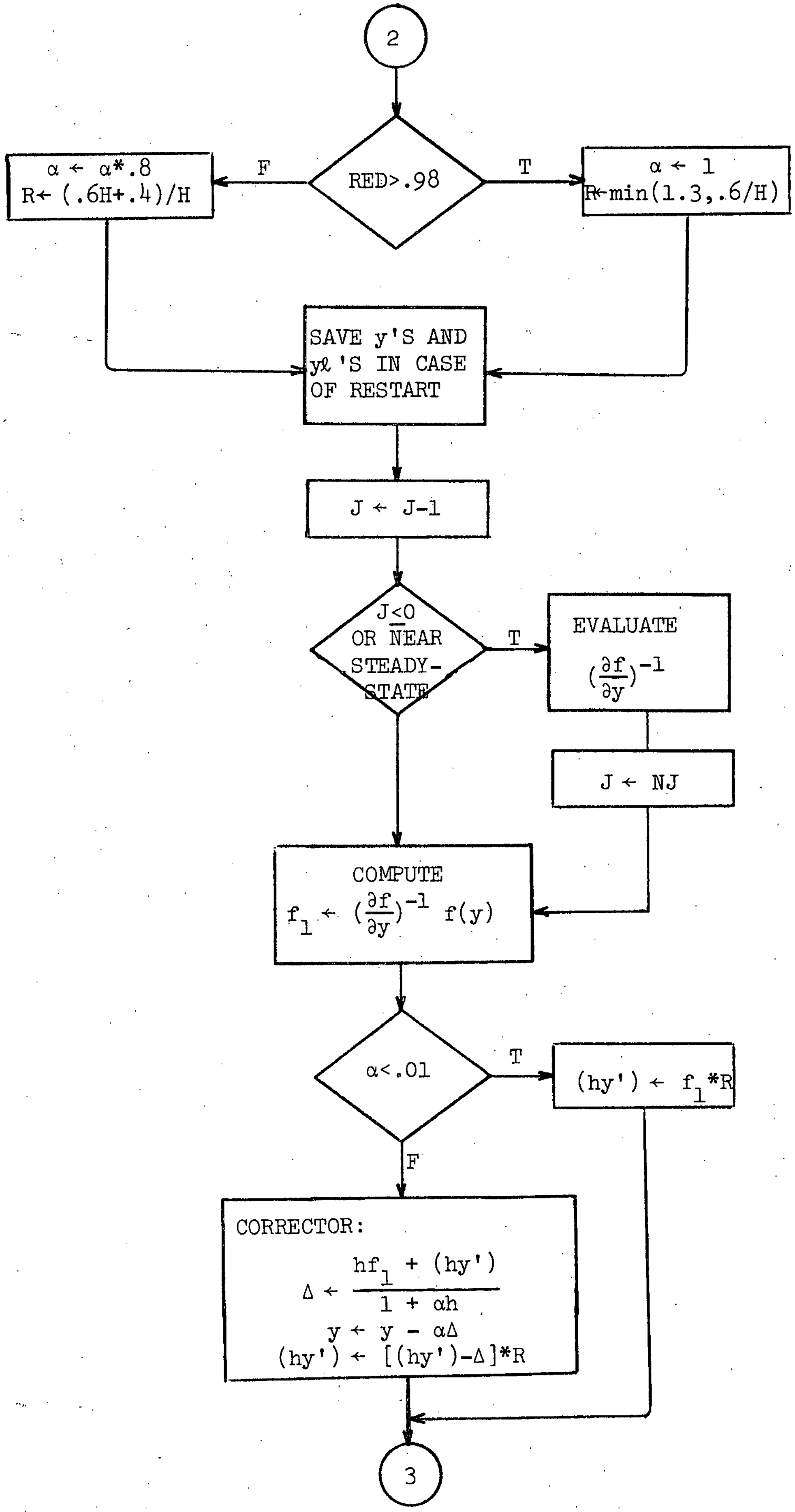
There are two major items of concern in writing DIFMF3: (1) how to vary α and h to achieve a solution as efficiently as possible, (2) how often to evaluate the Jacobian $\partial f/\partial y$.

1. We have seen that $\alpha = 0$, $h = 1$, corresponds to the Newton method, while $\alpha = 1$ is the more cautious integration procedure. The former method is quite effective for systems which are linear or nearly so, and also for most systems when near a solution; however, it is potentially quite erratic in other situations, where the latter method must be used. In order to cater to those systems best solved by Newton, we start out immediately with $\alpha = 0$, $h = 1$, and monitor convergence. We continue as long as the "error" $||f(y)||$ decreases.
If at any point the error increases, we switch over to the other method, setting $\alpha = 1$ and h small, and continue to monitor convergence. At each step, h is slightly increased. If the error starts reducing favorably, we increase h more rapidly and start phasing out the corrector step by reducing α. If the error later increases, we revert to the $\alpha = 1$ and small h method. The rate used to vary α and h was empirically determined.

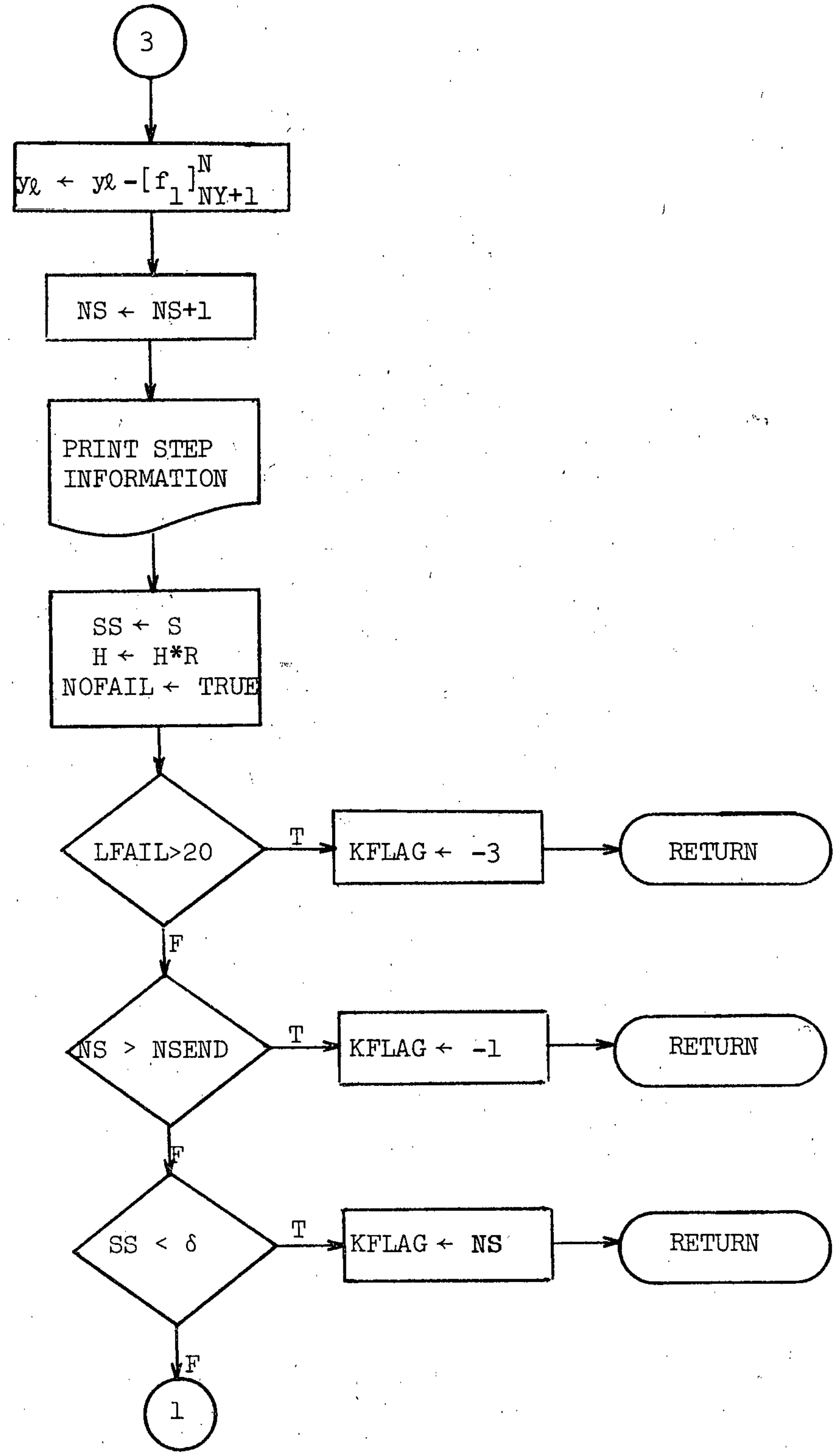
2. Computing and inverting the Jacobian is a fairly time-consuming process, so it is desirable to avoid it whenever possible. Fortunately, the Jacobian does not usually change too rapidly, so we can take several steps with the same one. The present strategy calls for recomputing the Jacobian every NY*5 steps (NY the number of columns in Y) unless the situation requires otherwise. Such situations include: a large increase in $||f(y)||$, variables going out of bounds, and apparently imminent approach of a solution. The recomputation occurs in the last case because a "fresh" Jacobian speeds convergence considerably when near a solution.

C. Flowchart


ENTER
INITIALIZE
COMPUTE $(\frac{\partial f}{\partial y})^{-1}$
H ← 1.0
α ← 0
NEWTON ← TRUE
SS ← ||f(y)||
1
PREDICTOR
y ← y + hy'
CALL RANGER TO ADJUST ANY OUT-OF-BOUND y'S
THERE WERE y'S OUT OF BOUNDS
T
F
J ← 0;
IF SAME y('S) AS LAST TIME, LFAIL←LFAIL+1
EVALUATE f(y)
S ← ||f(y)||
RED ← S/SS
RED>.98 AND NEWTON
T
F
NEWTON ← FALSE
α ← 1
H ← .01
RESTORE OLD y'S AND yℓ'S
1
RED>100 AND NOFAIL
F
2
NOFAIL ← FALSE
α ← 1
H ← max(H/2, 0.2)
RESTORE OLD y'S AND yℓ'S
1



2
RED>.98
F
T
α ← α*.8
R← (.6H+.4)/H
α ← 1
R←min(1.3,.6/H)
SAVE y'S AND
yℓ'S IN CASE
OF RESTART
J ← J-1
J≤0
OR NEAR
STEADY-
STATE
T
EVALUATE
(∂f/∂y)^-1
J ← NJ
COMPUTE
f1 ← (∂f/∂y)^-1 f(y)
α<.01
T
(hy') ← f1*R
F
CORRECTOR:
Δ ← (hf1 + (hy'))/(1 + αh)
y ← y - αΔ
(hy') ← [(hy')-Δ]*R
3


3
yℓ ← yℓ −[f1]N NY+1
NS ← NS+1
PRINT STEP INFORMATION
SS ← S
H ← H*R
NOFAIL ← TRUE
LFAIL>20
T
KFLAG ← -3
RETURN
F
NS > NSEND
T
KFLAG ← -1
RETURN
F
SS < δ
T
KFLAG ← NS
RETURN
F
1

D. Explanation of Symbols

(Symbol used in flowchart and/or previous discussion)	Symbol used in DIFMF3	Explanation
α	ALPHA	See part II
Δ	D	See part II
δ	DEL	Convergence criterion: equations are considered satisfied when S < DEL
f(y)	DY	Vector containing the values of the functions $\underline{f}(\underline{y})$
	EPS	Passed to MATSET
	EQN	Passed to MATSET
f_1	F1	Vector containing the product $(\partial f/\partial y)^{-1}\, f(y)$
	G	An array of global variables
h	H	The step size
	HINV	Held constant at 1.0 for this program
	HOLD	Value of H associated with the saved values of Y(3,*)
	IND	A vector indicating which of the Y variables to solve for
J	JACOB	A flag indicating when to re-evaluate the Jacobian. Decremented at each step, when it becomes zero or negative the Jacobian is computed and JACOB ← NJ.
MATINV	MATIN1 MATIN2 MATIN3	Subroutines which divide up the labor of computing the inverse Jacobian.
KFLAG	KFLAG	A return code (see program listing)

LFAIL	LFAIL	Number of times a given set of variables is found out of bounds. When excessive causes termination of DIFMF3.
	LFLAG	Return code from RANGER. Bits set correspond to variables out of bounds.
	LFLAG1	Previous non-zero value of LFLAG. It is AND'ed with LFLAG to see if the same variables are guilty as before.
	LIST	Describes bounds on Y variables; used by RANGER.
	M	Number of equations in DIFFUN.
	MF	Method indicator (see program listing).
	MM	Dimension of LIST, PLIM; used by RANGER
	N	NL + NY = total number of variables. Usually M = N.
NEWTON	NEWTON	A flag indicating whether we are still performing the NEWTON method which started the routine
NJ	NJ	The Jacobian is evaluated every NJ steps, barring trouble.
	NJJ	NJ-5. A comparison of NJ and NJJ is made to avoid too frequent evaluation of the Jacobian even in trouble situations.
	NL	Number of variables in YL.
NOFAIL	NOFAIL	Flag used to avoid a continuous loop when S suddenly increases.
NS	NS	The step number, incremented at each step.
NSEND	NSEND	The maximum number of steps to attempt before termination.

	NW	Number of calls to MATSET.
	NY	Number of columns in Y.
	PLIM	Contains bounds used by RANGER.
red	PRED	S/SS. An indication of the rate of convergence.
	PW	Contains the inverse Jacobian.
R	R	The factor by which H is changed at each step; used to scale Y(3,*).
S	S	$\|\|f(y)\|\| = \sum_{i=1}^{M} \|DY(i)\|$, a measure of the distance from the solution.
SS	SS	Value of S from previous step.
	SAVE	An array used to save the variables Y(IND(J), J), Y(3,J) in case of mishap.
t	T	Time and remainder variables, constants to DIFMF3.
	VAR	Used by MATSET.
y,hy'	Y	An array containing in Y(1,*) dependent variables Y(2,*) their derivatives with respect to t Y(3,*) hy' Y(IND(*),*) corresponds to y.
yl	YL	An array of variables appearing only linearly and without derivatives.
	YSLV	A vector used to save the YL variables.

```
'ILER OPTIONS - NAME=  MAIN,OPT=02,LINECNT=55,SOURCE,EBCDIC,NOLIST,DECK,LOAD,MAP,
      SUBROUTINE DIFMF3 (DEL,DY,EQN,F1,G,IND,KFLAG,LIST,M,MF,
     +                   MM,N,NL,NSEND,PLIM,PW,SAVE,T,VAR,Y,YL,YLSV)
      IMPLICIT REAL*8 (A-H,O-Z)
C*****************************************************************************
C*
C*    THE ROUTINE DIFMF3 SOLVES THE STEADY-STATE PROBLEM.
C*    THE PARAMETERS HAVE THE FOLLOWING MEANINGS:
C*
C*       M        THE NUMBER OF EQUATIONS.
C*       N        THE TOTAL NUMBER OF VARIABLES.
C*       NL       THE NUMBER OF LINEAR VARIABLES.
C*       DEL      CONVERGENCE CRITERION. ITERATION ENDS WHEN
C*                  SUM(|DY(J)|) < DEL .
C*       NSEND    THE MAXIMUM NUMBER OF STEPS TO BE ATTEMPTED.
C*       KFLAG    A COMPLETION CODE WITH THE FOLLOWING MEANINGS:
C*                   >0   CONVERGENCE WAS ACHIEVED IN KFLAG STEPS.
C*                   -1   CONVERGENCE NOT ACHIEVED IN NSEND STEPS.
C*                   -3   VARIABLE(S) REPEATEDLY GO OUT OF BOUNDS.
C*                          TRY NEW STARTING VALUES.
C*       MF       THE METHOD INDICATOR.
C*                   IF MF=4, SEE DESCRIPTION OF IND.
C*                   IF MF=3, THE VECTOR IND IS LOADED WITH 1 IN
C*                      EACH ELEMENT AND Y(2,J) IS CLEARED. THEN
C*                      MF IS SET TO 4.
C*
      DIMENSION T(1),G(1),Y(7,1),YL(1),SAVE(2,1),YLSV(1)
      DIMENSION PW(1),DY(1),F1(1),PLIM(1)
      INTEGER*2 VAR(3,2,1),EQN(1),IND(1),LIST(2,1)
C*
C*       T        THE INDEPENDENT VARIABLE.
C*       G        AN ARRAY OF GLOBAL VARIABLES.
C*       Y        A 7 BY N-NL ARRAY. Y(1,J) CONTAINS THE DEPENDENT
C*                  VARIABLES, Y(2,J) CONTAINS THEIR DERIVATIVES.
C*                  THE CALLER MUST SUPPLY INITIAL VALUES FOR
C*                  Y(1,J) AND, IF MF=4, Y(2,J).
C*       YL       AN ARRAY OF NL LINEAR VARIABLES.
C*       SAVE     AN ARRAY OF AT LEAST 2*(N-NL) VARIABLES.
C*       YLSV     AN ARRAY OF AT LEAST NL VARIABLES.
C*       PW       A SINGLE-PRECISION VECTOR
C*                  WHICH HOLDS THE INVERSE JACOBIAN.
C*       DY       A VECTOR OF LENGTH M, OUTPUT OF DIFFUN.
C*       F1       A VECTOR OF LENGTH MAX(M,N), OUTPUT OF MATMUL,
C*                ALSO USED BY MATSET AS SUBSTITUTE FOR SAVE.
C*       EQN,VAR   VECTORS USED BY MATSET.
C*       IND      AN INDICATOR VECTOR OF LENGTH N-NL. WHEN MF=4
C*                  WE SOLVE FOR Y(IND(J),J), KEEPING Y(3-IND(J),J)
C*                  CONSTANT. NOTE THAT THIS VECTOR MUST BE
C*                  SUPPLIED EVEN WHEN MF = 3.
C*       LIST,PLIM  ARRAYS PASSED TO ROUTINE RANGER.
C*       MM         DIMENSION OF LIST,PLIM.
C*
C*****************************************************************************
      DATA EPS /.05/, HINV /1.D0/
```

```
      LOGICAL NOFAIL,NEWTON
      WRITE (6,353) MF,N,NL,DEL
  353 FORMAT ('1MF =',I2,'    N =',I3,'    NL =',I3,'    DEL =',D10.2)
      NY = N-NL
      NJ = NY*5
      NJJ = NJ-10
      ALPHA = 0
      IF (MF .EQ. 4) GO TO 9
      DO 8 J = 1,NY
        IND(J) = 1
    8   Y(2,J) = 0
    9 CALL RANGER (Y,LIST,PLIM,MM,LFLAG,.FALSE.)
      H = 1
      NS = 0
      NW = 1
      WRITE (6,4)
    4 FORMAT ('0 NS NW  ALPHA      H',8X,'ERROR',6X,
     +        'Y(IND(J),J) AND YL(*)'//)
      NOFAIL = .TRUE.
      LFAIL = 0
C*    SET ALL BITS IN LFLAG1:
      LFLAG1 = 2147483647
      JACOB = 0
      R = 1
C****
C*    COMPUTE THE INITIAL JACOBIAN.
C*    DIFFUN EVALUATES THE DERIVATIVES DY.
C*    ROUTINE MATSET EVALUATES THE JACOBIAN MATRIX, MATIN1-3
C*    INVERTS IT, AND MATMUL MULTIPLIES THE INVERSE ON
C*    THE RIGHT BY THE VECTOR DY, PLACING THE PRODUCT IN F1.
C****
      CALL S1(T,G)
      CALL S2(T,G)
      CALL DIFFUN (T,G,DY,Y,YL,HINV)
      CALL MATSET (0D0,DY,EPS,EQN,G,HINV,IND,M,MF1,N,NY,1,
     +            PW,F1,T,VAR,Y,YL)
      CALL MATSET (0D0,DY,EPS,EQN,G,HINV,IND,M,MF1,N,NY,2,
     +            PW,F1,T,VAR,Y,YL)
      CALL MATSET (HINV,DY,EPS,EQN,G,HINV,IND,M,MF1,N,NY,3,
     +            PW,F1,T,VAR,Y,YL)
      CALL MATIN1 (PW)
      CALL MATIN2 (PW)
      CALL MATIN3 (PW)
      CALL MATMUL (PW,DY,F1)
      SS = 0
      DO 20 J = 1,M
   20   SS = SS+DABS(DY(J))
      WRITE (6,1) NS,NW,ALPHA,H,SS,(Y(IND(J),J),J=1,NY)
      IF (NL .GT. 0) WRITE (6,2) (YL(J),J=1,NL)
      DO 10 J = 1,NY
   10   Y(3,J) = -H*F1(J)
      IF (NL .LE. 0) GO TO 18
      DO 15 J = 1,NL
```

```
   15    YL(J) = YL(J)-F1(J+NY)
   18  CONTINUE
C*
C*     START WITH NEWTON METHOD. IN CASE
C*     THIS BOMBS, SAVE RESTART INFO.
C*
       DO 25 J = 1,NY
         SAVE(1,J) = Y(IND(J),J)
   25    SAVE(2,J) = Y(3,J)
       DO 26 J = 1,NL
   26    YLSV(J) = YL(J)
       NEWTON = .TRUE.
C****
C*     PREDICTOR STEP
C****
   30  DO 40 J = 1,NY
         KL = IND(J)
   40    Y(KL,J) = Y(KL,J)+Y(3,J)
C****
C*     BEFORE TAKING THE CORRECTOR STEP, CHECK TO SEE IF
C*     ANYTHING HAS GONE OUT OF BOUNDS. LFAIL IS THE NUMBER OF
C*     CONSECUTIVE TIMES A GIVEN VARIABLE OR SET OF VARIABLES
C*     HAS GONE OUT OF BOUNDS. LFLAG1 IS THE MOST RECENT NON-
C*     ZERO VALUE OF LFLAG.
C****
       CALL RANGER (Y,LIST,PLIM,MM,LFLAG,.TRUE.)
       IF (LFLAG .EQ. 0) GO TO 49
       IF (IAND(LFLAG1,LFLAG) .EQ. 0) GO TO 47
       LFAIL = LFAIL+1
       GO TO 48
   47  LFAIL = 0
   48  LFLAG1 = LFLAG
       IF (JACOB .LT. NJJ) JACOB = 0
   49  CONTINUE
C*
C*     CHECK ERROR. IF IT HAS INCREASED TOO MUCH (100-FOLD),
C*     MAKE CHANGE OF PLAN. NOFAIL PREVENTS AN INFINITE LOOP:
C*     WHEN THE CHANGES ARE MADE (AT LINE 100) NOFAIL IS SET
C*     TO .FALSE.
C*     IF ERROR INCREASES AT ALL WHILE USING THE STRAIGHT NEWTON
C*     METHOD (NEWTON = .TRUE.), WE CHANGE OVER TO PREDICTOR-
C*     CORRECTOR METHOD.
C*
       CALL DIFFUN (T,G,DY,Y,YL,HINV)
       S = 0
       DO 50 J = 1,M
   50    S = S+DABS(DY(J))
       PRED = S/SS
       IF (NEWTON) GO TO 150
       IF (PRED .LT. 100) GO TO 51
       IF (NOFAIL) GO TO 100
   51  CONTINUE
C*
```

```
C*     CHECK RATE OF CONVERGENCE. IF SLOW (OR IF ERROR IS
C*     INCREASING), CONTINUE USING THE PREDICTOR-CORRECTOR
C*     METHOD WITH ALPHA=1 AND SLOWLY-INCREASING H. OTHER-
C*     WISE, INCREASE H TOWARDS 1.0 AND DECREASE ALPHA.
C*
       IF (PRED .LT. .98) GO TO 53
       ALPHA = 1
       R = DMIN1 (1.3D0,0.6D0/H)
       GO TO 60
   53  R = .60+.40/H
       ALPHA = ALPHA*0.8
       GO TO 60
C****
C*     SAVE INFORMATION FOR POSSIBLE RESTART
C****
   60  DO 62 J = 1,NY
         SAVE(1,J) = Y(IND(J),J)
   62    SAVE(2,J) = Y(3,J)
       DO 63 J = 1,NL
   63    YLSV(J) = YL(J)
       HOLD = H
C*****
C*     IF WE ARE NEAR THE STEADY STATE, RE-EVALUATE THE
C*     JACOBIAN TO GIVE ONE FINAL PUSH.
C*****
       IF (SS .LT. 1.D0  .AND.  JACOB .LT. NJJ) GO TO 67
   65  JACOB = JACOB-1
       IF (JACOB .GT. 0) GO TO 70
C****
C*     IF THERE HAS BEEN TROUBLE WITH CONVERGENCE, OR IF IT HAS
C*     BEEN A LONG TIME SINCE THE LAST RE-EVALUATION, THE JACOBIAN
C*     IS RE-EVALUATED PRIOR TO THE CORRECTOR STEP.
C****
   67  CALL MATSET (HINV,DY,EPS,EQN,G,HINV,IND,M,MF1,N,NY,3,
     +             PW,F1,T,VAR,Y,YL)
       CALL MATIN3 (PW)
       NW = NW+1
       JACOB = NJ
   70  CALL MATMUL (PW,DY,F1)
C*****
C*     CORRECTOR STEP IS NOW PERFORMED. Y(3,J) (PSEUDO H*Y'(J))
C*     IS SIMULTANEOUSLY SCALED IN ACCORDANCE WITH THE NEW H.
C*     IF ALPHA IS VERY SMALL, THE EFFECT OF THE CORRECTOR IS
C*     SLIGHT, SO SKIP IT AND JUST SCALE.
C*****
       IF (ALPHA .LT. 0.01) GO TO 77
       DD = 1.D0/(1+H*ALPHA)
       DO 75 J = 1,NY
         D = (F1(J)*H+Y(3,J))*DD
         KL = IND(J)
         Y(KL,J) = Y(KL,J)-ALPHA*D
   75    Y(3,J) = R*(Y(3,J)-D)
       GO TO 79
```

```
   77 HH = -H*R
      DO 78 J = 1,NY
   78   Y(3,J) = HH*F1(J)
   79 CONTINUE
      IF (NL .LE. 0) GO TO 90
      DO 80 J = 1,NL
   80   YL(J) = YL(J)-F1(J+NY)
   90 SS = S
      NS = NS+1
      WRITE (6,1) NS,NW,ALPHA,H,SS,(Y(IND(J),J),J=1,NY)
    1 FORMAT (/I4,I3,F6.2,2D11.2,7D14.4/(35X,7D14.4))
      IF (NL .GT. 0) WRITE (6,2) (YL(J),J=1,NL)
    2 FORMAT (35X,7D14.4)
      H = H*R
      NOFAIL = .TRUE.
      IF (LFAIL .GT. 20) GO TO 350
      IF (NS .GE. NSEND) GO TO 200
      IF (SS .GT. DEL) GO TO 30
      KFLAG = NS
      RETURN
  100 CONTINUE
C*
C*    THE PREDICTER LOOP BLEW UP. ALPHA RETURNS TO 1.0, H IS
C*    REDUCED, THE SAVED VALUES OF Y & YL ARE RESTORED, AND
C*    THE STEP IS RETRIED. IN ADDITION, WE SIGNAL FOR JACOBIAN
C*    RE-EVALUATION UNLESS THIS HAS BEEN DONE RECENTLY.
C*
      ALPHA = 1.0
      IF (JACOB .LT. NJJ) JACOB = 0
      R = DMAX1(0.5D0,0.2D0/HOLD)
      H = HOLD*R
  105 DO 110 J = 1,NY
        Y(IND(J),J) = SAVE(1,J)
  110   Y(3,J) = SAVE(2,J)*R
      DO 115 J = 1,NL
  115   YL(J) = YLSV(J)
      NOFAIL = .FALSE.
      GO TO 30
C*
C*     IF NEWTON METHOD IS FAILING,
C*     WE MUST SET UP FOR PREDICTOR-CORRECTOR SCHEME.
C*
  150 IF (PRED .LT. 0.95) GO TO 60
      WRITE (6,5) S,(Y(IND(J),J),J=1,NY)
    5 FORMAT ('ONEWTON FAILED:',9X,D11.2,7D14.4/(35X,7D14.4))
      IF (NL .GT. 0) WRITE (6,2) (YL(J),J=1,NL)
      H = .01
      ALPHA = 1
      R = .01
      JACOB = 0
      NEWTON = .FALSE.
      GO TO 105
  200 KFLAG = -1
```

```
      WRITE (6,333) (J,DY(J),J=1,M)
  333 FORMAT ('-DY:'/(I5,D20.6))
      RETURN
  350 KFLAG = -3
      RETURN
      END
```

E. Examples

Some of the systems used to test DIFMF3 follow. Pseudo-random starting values were used, $t = 0$, and the convergence criterion was $\delta = 10^{-6}$.

Most of these systems are purely algebraic. The last three include restricted variables. Solutions to these often take longer because variables go out of range and are shoved around a lot by RANGER before they settle down. System (3) is linear, so the solution was obtained quite swiftly.

1. $4 + y_1 + y_2 - y_1^2 + 2y_1y_2 + 3y_2^2$

   $1 + 2y_1 - 3y_2 + y_1^2 + y_1y_2 - 2y_2^2$

   Starting values: $y_1 = -2.057$ $y_2 = -7.503$

   Solution obtained: $y_1 = 3.339$ $y_2 = -2.984$

   NS = 24, NW = 5

2. $y_1^2 + y_2^2 + y_3^2 - 5$

   $y_1 + y_2 - 1$

   $y_1 + y_3 - 3$

   Starting values: $y_1 = -2.057$ $y_2 = -7.503$ $y_3 = -4.834$

   Solution obtained: $y_1 = 1.667$ $y_2 = -.6667$ $y_3 = 1.333$

   NS = 28, NW = 4

3. $t_2 + y\ell_1 - 6y_3 - y_3'$

$y_3 - y_2^1$

$y_2 - y_1^1$

$y\ell_3 + 11y_2$

$y\ell_2 + 6y_1$

$y\ell_1 - y\ell_2 - y\ell_3$

$t_2 = \sin(t)$

Starting values: $y_1 = -2.057$, $y_2 = -7.503$, $y_3 = -4.834$

$y\ell_1 = 4.473$, $y\ell_2 = -6.240$, $y\ell_3 = 9.308$

Solution obtained: $y_1 = y_2 = y_3 = y\ell_1 = y\ell_2 = y\ell_3 = 0.000$

NS = 2, NW = 2

4. $\frac{1}{2}\sqrt{4-y_1^2} + y_2 - 1$

$2y_1^3 + \ln(y_2 + .8) - .136$

Restriction: $-2 \leq y_1 \leq 2$, $y_2 > -.8$

Starting values: $y_1 = -.9433$, $y_2 = 3.951$

Solution obtained: $y_1 = 5.394$, $y_2 = .03705$

NS = 27, NW = 5

5. $\tan(y_1) + y_2^3 - 3y\ell_1 - .5$

$\sin(2y_1) - 1/y_2 + 2y\ell_1 - 1$

$y_2 + y\ell_1 - 1.5$

Restriction: $\pi/2 < y_1 < \pi/2,\ y_2 > 0$

Starting values: $y_1 = -.2983,\ y_2 = 4.751,\ y\ell_1 = -4.834$

Solution obtained: $y_1 = .7854,\ y_2 = 1.000,\ y\ell_1 = .5000$

NS = 116, NW = 14

6. $y_1^2 + y_2^2 + y\ell_1 - 3$

$y_2^2 + y_3^2 + y\ell_2 - 10$

$\sqrt{2-y_1} + y\ell_1$

$y\ell_1 + y\ell_2 - 3$

$20y_3 - 9y\ell_2 + 5$

Restriction: $y_1 \leq 2$

Starting values: $y_1 = -2.057,\ y_2 = -7.503,\ y_3 = -4.834,$

$y\ell_1 = 4.473,\ y\ell_2 = -6.240$

Solution obtained: $y_1 = -2.000,\ y_2 = -1.000,\ y_3 = 2.000,$

$y\ell_1 = -2.000,\ y\ell_2 = 5.000$

NS = 20, NW = 3