mathematisch centrum

DEPARTMENT OF NUMERICAL MATHEMATICS OCTOBER

NUMAL, A LIBRARY OF NUMERICAL PROCEDURES IN ALGOL 60

VOLUME 5. ANALYTICAL PROBLEMS

amsterdam 1974

AUTHORS: J. C. P. BUS AND T. J. DEKKER.

CONTRIBUTOR: J. C. P. BUS.

INSTITUTE: MATHEMATICAL CENTRE.

RECEIVED: 740908.

BRIEF DESCRIPTION:

THIS SECTION CONTAINS TWO PROCEDURES FOR FINDING A ZERO OF A GIVEN FUNCTION IN A GIVEN INTERVAL:
ZEROIN APPROXIMATES A ZERO MAINLY BY LINEAR INTERPOLATION AND EXTRAPOLATION;
ZEROINRAT APPROXIMATES A ZERO BY INTERPOLATION WITH RATIONAL FUNCTIONS.

KEYWORDS:

ZERO SEARCHING,
ANALYTIC EQUATIONS,
SINGLE NON-LINEAR EQUATION.

SUBSECTION: ZEROIN.

CALLING SEQUENCE:

THE HEADING OF THIS PROCEDURE READS :
"BOOLEAN" "PROCEDURE" ZEROIN(X, Y, FX, TOLX);
"REAL" X, Y, FX, TOLX;

ZEROIN IS GIVEN THE VALUE TRUE IF AN INTERVAL IS FOUND WHICH IS SMALLER THAN TWICE THE VALUE OF TOLX AND IN WHICH THE FUNCTION CHANGES SIGN, OTHERWISE THE VALUE OF ZEROIN WILL BE FALSE;

THE MEANING OF THE FORMAL PARAMETERS IS;

X: <REAL VARIABLE>;
A JENSEN VARIABLE; THE ACTUAL PARAMETERS FOR FX AND TOLX ARE DEPENDING ON X;
ENTRY: ONE OF THE ENDPOINTS OF THE INTERVAL IN WHICH A ZERO IS SEARCHED FOR;
EXIT: ONE OF THE ENDPOINTS OF THE CALCULATED INTERVAL [XEND, YEND]; THIS INTERVAL IS SUCH THAT:
ABS(XEND - YEND) <= 2 * TOLX;
IF X AND Y ARE GIVEN SUCH THAT F(X) * F(Y) <= 0, WHERE F DENOTES THE GIVEN FUNCTION, THEN WE WILL BE SURE THAT THIS INTERVAL CONTAINS A ZERO OF F; THE VALUE OF X WILL BE EQUAL TO XEND IF ABS(F(XEND)) <= ABS(F(YEND)), ELSE X WILL BE EQUAL TO YEND;

Y: <REAL VARIABLE>;
ENTRY: THE OTHER ENDPOINT OF THE INTERVAL IN WHICH A ZERO IS SEARCHED FOR;
ONLY IF X AND Y ARE CHOSEN SUCH THAT F(X)*F(Y) <= 0 THEN CONVERGENCE TO A ZERO IS GUARANTEED AND THE VALUE OF ZEROIN WILL BE TRUE ON EXIT;
EXIT: THE VALUE OF THE OTHER ENDPOINT OF THE CALCULATED INTERVAL [XEND, YEND] (SEE PARAMETER X);

FX: <ARITHMETIC EXPRESSION>;
THE FUNCTION IS GIVEN BY THE ACTUAL PARAMETER FX, WHICH DEPENDS ON X;

TOLX: <ARITHMETIC EXPRESSION>;
THE TOLERANCE IS GIVEN BY THE ACTUAL PARAMETER TOLX, WHICH MAY DEPEND ON X; A SUITABLE TOLERANCE FUNCTION IS: ABS(X) * RE + AE, WHERE RE IS THE DESIRED RELATIVE PRECISION, WHICH SHOULD BE CHOSEN GREATER THAN OR EQUAL TO THE MACHINE PRECISION AND AE IS AN ABSOLUTE TOLERANCE, WHICH MUST BE CHOSEN UNEQUAL TO ZERO IF THE GIVEN INTERVAL CONTAINS THE ORIGIN.

EXECUTION FIELD LENGTH: NO AUXILIARY ARRAYS ARE DECLARED IN ZEROIN.

LANGUAGE: ALGOL 60.

METHOD AND PERFORMANCE:

FOR A DETAILED DESCRIPTION OF THIS PROCEDURE SEE [1].

EXAMPLE OF USE:

THE ZERO OF THE FUNCTION EXP(-X * 3) * (X - 1) + X ** 3, IN THE INTERVAL [0, 1], MAY BE CALCULATED BY THE FOLLOWING PROGRAM:

```
"BEGIN" "REAL" X, Y;
    "BOOLEAN" "PROCEDURE" ZEROIN(X, Y, FX, TOLX); "CODE" 34150;
    "REAL" "PROCEDURE" F(X); "VALUE" X; "REAL" X;
    F:= EXP(-X * 3) * ( X - 1) + X ** 3;
    X:= 0; Y:= 1;
    "IF" ZEROIN(X, Y, F(X), ABS(X) * "-14 + "-14) "THEN"
    OUTPUT(71, "("/48,"("CALCULATED ZERO:")"B+.15D"+3D")", X)
    "ELSE" OUTPUT(71, "("/48,"("NO ZERO FOUND")"")")
"END"
```

RESULT:

CALCULATED ZERO: +.4897027485482 40"+000

SUBSECTION: ZEROINRAT.

CALLING SEQUENCE:

THE HEADING OF THIS PROCEDURE IS:
"BOOLEAN" "PROCEDURE" ZEROINRAT(X, Y, FX, TOLX);
"REAL" X, Y, FX, TOLX;

ZEROINRAT IS GIVEN THE VALUE TRUE IF AN INTERVAL IS FOUND WHICH IS SMALLER THAN TWICE THE VALUE OF TOLX AND IN WHICH THE FUNCTION CHANGES SIGN, OTHERWISE THE VALUE OF ZEROINRAT WILL BE FALSE;

THE MEANING OF THE FORMAL PARAMETERS IS:

X: <REAL VARIABLE>;
A JENSEN VARIABLE; THE ACTUAL PARAMETERS FOR FX AND TOLX ARE DEPENDING ON X;
ENTRY: ONE OF THE ENDPOINTS OF THE INTERVAL IN WHICH A ZERO IS SEARCHED FOR;
EXIT: ONE OF THE ENDPOINTS OF THE CALCULATED INTERVAL [XEND, YEND]; THIS INTERVAL IS SUCH THAT:
ABS(XEND - YEND) <= 2 * TOLX;
IF X AND Y ARE GIVEN SUCH THAT F(X) * F(Y) <= 0, WHERE F DENOTES THE GIVEN FUNCTION, THEN WE WILL BE SURE THAT THIS INTERVAL CONTAINS A ZERO OF F; THE VALUE OF X WILL BE EQUAL TO XEND IF ABS(F(XEND)) <= ABS(F(YEND)), ELSE X WILL BE EQUAL TO YEND;

Y: <REAL VARIABLE>;
ENTRY: THE OTHER ENDPOINT OF THE INTERVAL IN WHICH A ZERO IS SEARCHED FOR;
ONLY IF X AND Y ARE CHOSEN SUCH THAT F(X)*F(Y) <= 0 THEN CONVERGENCE TO A ZERO IS GUARANTEED AND THE VALUE OF ZEROIN WILL BE TRUE ON EXIT;
EXIT: THE VALUE OF THE OTHER ENDPOINT OF THE CALCULATED INTERVAL [XEND, YEND] (SEE PARAMETER X);

FX: <ARITHMETIC EXPRESSION>;
THE FUNCTION IS GIVEN BY THE ACTUAL PARAMETER FX, WHICH DEPENDS ON X;

TOLX: <ARITHMETIC EXPRESSION>;
THE TOLERANCE IS GIVEN BY THE ACTUAL PARAMETER TOLX, WHICH MAY DEPEND ON X; A SUITABLE TOLERANCE FUNCTION IS: ABS(X) * RE + AE, WHERE RE IS THE DESIRED RELATIVE PRECISION, WHICH SHOULD BE CHOSEN GREATER THAN OR EQUAL TO THE MACHINE PRECISION AND AE IS AN ABSOLUTE TOLERANCE, WHICH MUST BE CHOSEN UNEQUAL TO ZERO IF THE GIVEN INTERVAL CONTAINS THE ORIGIN.

EXECUTION FIELD LENGTH: NO AUXILIARY ARRAYS ARE DECLARED IN ZEROINRAT.

LANGUAGE: ALGOL 60.

METHOD AND PERFORMANCE:

FOR A DETAILED DESCRIPTION OF THIS PROCEDURE SEE [1].

EXAMPLE OF USE:

THE ZERO OF THE FUNCTION EXP(-X * 3) * (X - 1) + X ** 3, IN THE INTERVAL [0, 1], MAY BE CALCULATED BY THE FOLLOWING PROGRAM:

```
"BEGIN" "REAL" X, Y;
    "BOOLEAN" "PROCEDURE" ZEROINRAT(X, Y, FX, TOLX); "CODE" 34436;
    "REAL" "PROCEDURE" F(X); "VALUE" X; "REAL" X;
    F:= EXP(-X * 3) * ( X - 1) + X ** 3;
    X:= 0; Y:= 1;
    "IF" ZEROINRAT(X, Y, F(X), ABS(X) * "-14 + "-14) "THEN"
    OUTPUT(71, "("/48,"("CALCULATED ZERO:")"B+.15D"+3D")", X)
    "ELSE" OUTPUT(71, "("/48,"("NO ZERO FOUND")""")")
"END"
```

RESULT:

CALCULATED ZERO: +.489702748548240"+000

REFERENCES:

[1]: BUS, J.C.P. AND DEKKER, T.J.
TWO EFFICIENT ALGORITHMS WITH GUARANTEED CONVERGENCE FOR FINDING A ZERO OF A FUNCTION.
MATHEMATICAL CENTRE, REPORT NW 13/74, AMSTERDAM (1974).

SOURCE TEXTS;

```
"CODE" 34150;
"BOOLEAN" "PROCEDURE" ZEROIN(X, Y, FX, TOLX);
"REAL" X, Y, FX, TOLX;
"BEGIN" "INTEGER" EXT;
    "REAL" C, FC, B, FB, A, FA, D, FD, FDB, FDA, W, MB,
    TOL, M, P, Q;
    B:= X; FB:= FX; A:= X:= Y; FA:= FX;
INTERPOLATE: C:= A; FC:= FA; EXT:= 0;
EXTRAPOLATE: "IF" ABS(FC) < ABS(FB) "THEN"
    "BEGIN" "IF" C ^= A "THEN" "BEGIN" D:= A; FD:= FA "END";
        A:= B; FA:= FB; B:= X:= C; FB:= FC; C:= A; FC:= FA
    "END" INTERCHANGE;
    TOL:= TOLX; M:= (C + B) * 0.5; MB:= M - B;
    "IF" ABS(MB) > TOL "THEN"
    "BEGIN" "IF" EXT > 2 "THEN" W:= MB "ELSE"
        "BEGIN" TOL:= TOL * SIGN(MB);
            P:= (B - A) * FB; "IF" EXT <= 1 "THEN"
            Q:= FA - FB "ELSE"
            "BEGIN" FDB:= (FD - FB) / (D - B);
                FDA:= (FD - FA) / (D - A);
                P:= FDA * P; Q:= FDB * FA - FDA * FB
            "END"; "IF" P < 0 "THEN"
            "BEGIN" P:= -P; Q:= -Q "END";
            W:= "IF" P * 1 = 0 "OR" P <= Q * TOL "THEN" TOL "ELSE"
            "IF" P < MB * Q "THEN" P / Q "ELSE" MB
        "END"; D:= A; FD:= FA; A:= B; FA:= FB;
        X:= B:= B + W; FB:= FX;
        "IF" ("IF" FC >= 0 "THEN" FB >= 0 "ELSE" FB <= 0) "THEN"
        "GOTO" INTERPOLATE "ELSE"
        "BEGIN" EXT:= "IF" W = MB "THEN" 0 "ELSE" EXT + 1;
            "GOTO" EXTRAPOLATE
        "END"
    "END"; Y:= C;
    ZEROIN:= "IF" FC >= 0 "THEN" FB <= 0 "ELSE" FB >= 0
"END" ZEROIN;
    "EOP"
```

```
"CODE" 34436;
"BOOLEAN" "PROCEDURE" ZEROINRAT(X, Y, FX, TOLX);
"REAL" X, Y, FX, TOLX;
"BEGIN" "INTEGER" EXT; "BOOLEAN" FIRST;
    "REAL" B, FB, A, FA, D, FD, C, FC, FDB, FDA, W,
    MB, TOL, M, P, Q;

    B:= X; FB:= FX; A:= X:= Y; FA:= FX; FIRST:= "TRUE";
INTERPOLATE: C:= A; FC:= FA; EXT:= 0;
EXTRAPOLATE: "IF" ABS(FC) < ABS(FB) "THEN"
    "BEGIN" "IF" C ^= A "THEN" "BEGIN" D:= A; FD:= FA "END";
        A:= B; FA:= FB; B:= X:= C; FB:= FC; C:= A; FC:= FA
    "END" INTERCHANGE;
    TOL:= TOLX; M:= (C + B) * .5; MB:= M - B;
    "IF" ABS(MB) > TOL "THEN"
    "BEGIN" "IF" EXT > 3 "THEN" W:= MB "ELSE"
        "BEGIN" TOL:= TOL * SIGN(MB);
            P:= (B - A) * FB; "IF" FIRST "THEN"
            "BEGIN" Q:= FA - FB; FIRST:= "FALSE" "END" "ELSE"
            "BEGIN" FDB:= (FD - FB) / (D - B);
                FDA:= (FD - FA) / (D - A);
                P:= FDA * P; Q:= FDB * FA - FDA * FB
            "END"; "IF" P < 0 "THEN"
            "BEGIN" P:= -P; Q:= -Q "END";
            "IF" EXT = 3 "THEN" P:= P * 2;
            W:= "IF" P * 1 = 0 "OR" P <= Q * TOL "THEN" TOL "ELSE"
            "IF" P < MB * Q "THEN" P / Q "ELSE" MB
        "END"; D:= A; FD:= FA; A:= B; FA:= FB;
        X:= B:= B + W; FB:= FX;
        "IF" ("IF" FC >= 0 "THEN" FB >= 0 "ELSE" FB <= 0) "THEN"
        "GOTO" INTERPOLATE "ELSE"
        "BEGIN" EXT:= "IF" W = MB "THEN" 0 "ELSE" EXT + 1;
            "GOTO" EXTRAPOLATE
        "END"
    "END"; Y:= C;
    ZEROINRAT:= "IF" FC >= 0 "THEN" FB <= 0 "ELSE" FB >= 0
"END" ZEROINRAT;
    "EOP"
```

AUTHORS: J. C. P. BUS AND T. J. DEKKER.

CONTRIBUTOR: J. C. P. BUS.

INSTITUTE: MATHEMATICAL CENTRE.

RECEIVED: 750615.

BRIEF DESCRIPTION:

THIS SECTION CONTAINS TWO PROCEDURES FOR FINDING A ZERO OF A GIVEN FUNCTION IN A GIVEN INTERVAL;
ZEROIN APPROXIMATES A ZERO MAINLY BY LINEAR INTERPOLATION AND EXTRAPOLATION;
ZEROINRAT APPROXIMATES A ZERO BY INTERPOLATION WITH RATIONAL FUNCTIONS.
ZEROIN IS PREFERABLE FOR SIMPLE (I.E. CHEAPLY TO CALCULATE) FUNCTIONS AND/OR WHEN NO HIGH PRECISION IS REQUIRED. ZEROINRAT IS PREFERABLE FOR COMPLICATED (I.E. EXPENSIVE) FUNCTIONS WHEN A ZERO IS REQUIRED IN RATHER HIGH PRECISION AND ALSO FOR FUNCTIONS HAVING A POLE NEAR THE ZERO. WHEN THE ANALYTIC DERIVATIVE OF THE FUNCTION IS EASILY OBTAINED, THEN ZEROINDER (SECTION 5.1.1.1.2) SHOULD BE TAKEN INTO CONSIDERATION.

KEYWORDS:

ZERO SEARCHING,
ANALYTIC EQUATIONS,
SINGLE NON-LINEAR EQUATION.

SUBSECTION: ZEROIN.

CALLING SEQUENCE:

```
THE HEADING OF THIS PROCEDURE READS :
"BOOLEAN" "PROCEDURE" ZEROIN(X, Y, FX, TOLX);
"REAL" X, Y, FX, TOLX;

ZEROIN  SEARCHES FOR A ZERO OF A REAL FUNCTION  F  DEFINED ON A
        CERTAIN INTERVAL J;
        ZEROIN := "TRUE" WHEN A (SUFFICIENTLY SMALL) SUBINTERVAL OF
        J CONTAINING A  ZERO OF  F HAS  BEEN  FOUND;  OTHERWISE,
        ZEROIN := "FALSE";
        LET A REAL FUNCTION T DEFINED ON J, DENOTE A TOLERANCE
        FUNCTION DEFINING THE REQUIRED PRECISION OF THE ZERO;
        (FOR INSTANCE
                        T(X) = ABS(X) * RE + AE,
        WHERE  RE  AND  AE  ARE THE REQUIRED  RELATIVE AND ABSOLUTE
        PRECISION RESPECTIVELY);

THE MEANING OF THE FORMAL PARAMETERS IS:
X:      <REAL VARIABLE>;
        A JENSEN VARIABLE;  THE ACTUAL PARAMETERS FOR  FX AND TOLX
        (MAY) DEPEND ON THE ACTUAL PARAMETER FOR X;
        ENTRY:  ONE  ENDPOINT OF INTERVAL  J  IN  WHICH  A  ZERO IS
                SEARCHED FOR;
        EXIT:   A VALUE APPROXIMATING THE ZERO WITHIN THE TOLERANCE
                2 * T(X) WHEN ZEROIN HAS THE VALUE "TRUE", AND A
                PRESUMABLY WORTHLESS ARGUMENT VALUE OTHERWISE;
Y:      <REAL VARIABLE>;
        ENTRY:  THE OTHER ENDPOINT OF INTERVAL J IN WHICH A ZERO IS
                SEARCHED FOR;  UPON ENTRY X < Y AS WELL AS Y < X IS
                ALLOWED;
        EXIT:   THE  OTHER  STRADDLING  APPROXIMATION  OF THE ZERO,
                I.E.  UPON  EXIT  THE  VALUES  OF  Y AND X  SATISFY
                1. F(X) * F(Y) <= 0,  2. ABS(X - Y) <= 2 * T(X) AND
                3. ABS(F(X)) <= ABS(F(Y)) WHEN ZEROIN HAS THE
                VALUE "TRUE", AND  A  PRESUMABLY WORTHLESS ARGUMENT
                VALUE  SATISFYING  CONDITIONS  2  AND  3  BUT NOT 1
                OTHERWISE;
FX:     <ARITHMETIC EXPRESSION>;
        DEFINES  FUNCTION F AS A FUNCTION  DEPENDING ON THE  ACTUAL
        PARAMETER FOR X;
TOLX:   <ARITHMETIC EXPRESSION>;
        DEFINES THE  TOLERANCE  FUNCTION T WHICH MAY  DEPEND ON THE
        ACTUAL PARAMETER FOR X;
        ONE SHOULD CHOOSE TOLX  POSITIVE AND NEVER SMALLER THAN THE
        PRECISION OF THE  MACHINE'S  ARITHMETIC  AT X, I.E. IN THIS
        ARITHMETIC  X + TOLX  AND  X - TOLX  SHOULD ALWAYS  YIELD
        VALUES  DISTINCT  FROM  X; OTHERWISE  THE PROCEDURE MAY GET
        INTO A LOOP.
```

PROCEDURES USED: DWARF = CP30003;

EXECUTION FIELD LENGTH: NO AUXILIARY ARRAYS ARE DECLARED IN ZEROIN.

LANGUAGE: ALGOL 60.

METHOD AND PERFORMANCE:

THE METHOD USED IS DESCRIBED IN DETAIL IN [1].
THE NUMBER OF EVALUATIONS OF FX AND TOLX IS AT MOST
4 * LOG(ABS(X - Y)) / TAU,
WHERE X AND Y ARE THE ARGUMENT VALUES GIVEN UPON ENTRY, LOG DENOTES THE BASE 2 LOGARITHM AND TAU IS THE MINIMUM OF THE TOLERANCE FUNCTION TOLX ON THE INITIAL INTERVAL. IF UPON ENTRY X AND Y SATISFY F(X) * F(Y) <= 0, THEN CONVERGENCE IS GUARANTEED AND THE ASYMPTOTIC ORDER OF CONVERGENCE IS 1.618 FOR SIMPLE ZEROES.

EXAMPLE OF USE:

THE ZERO OF THE FUNCTION EXP(-X * 3) * (X - 1) + X ** 3, IN THE INTERVAL [0, 1], MAY BE CALCULATED BY THE FOLLOWING PROGRAM:

```
"BEGIN" "REAL" X, Y;
    "BOOLEAN" "PROCEDURE" ZEROIN(X, Y, FX, TOLX); "CODE" 34150;
    "REAL" "PROCEDURE" F(X); "VALUE" X; "REAL" X;
    F:= EXP(-X * 3) * ( X - 1) + X ** 3;
    X:= 0; Y:= 1;
    "IF" ZEROIN(X, Y, F(X), ABS(X) * "-14 + "-14) "THEN"
    OUTPUT(71, "("/4B,"("CALCULATED ZERO:")"B+.15D"+3D")", X)
    "ELSE" OUTPUT(71, "("/4B,"("NO ZERO FOUND")""")")
"END"
```

RESULT:

```
CALCULATED ZERO: +.489702748548240"+000
```

SUBSECTION: ZEROINRAT.

CALLING SEQUENCE:

THE HEADING OF THIS PROCEDURE READS:
"BOOLEAN" "PROCEDURE" ZEROINRAT(X, Y, FX, TOLX);
"REAL" X, Y, FX, TOLX;

ZEROINRAT SEARCHES FOR A ZERO OF A REAL FUNCTION F DEFINED ON A CERTAIN INTERVAL J;
ZEROINRAT := "TRUE" WHEN A (SUFFICIENTLY SMALL) SUBINTERVAL OF J CONTAINING A ZERO OF F HAS BEEN FOUND; OTHERWISE, ZEROINRAT := "FALSE";
LET A REAL FUNCTION T DEFINED ON J, DENOTE A TOLERANCE FUNCTION DEFINING THE REQUIRED PRECISION OF THE ZERO;
(FOR INSTANCE

T(X) = ABS(X) * RE + AE,

WHERE RE AND AE ARE THE REQUIRED RELATIVE AND ABSOLUTE PRECISION RESPECTIVELY);

THE MEANING OF THE FORMAL PARAMETERS IS:

X: <REAL VARIABLE>;
A JENSEN VARIABLE; THE ACTUAL PARAMETERS FOR FX AND TOLX (MAY) DEPEND ON THE ACTUAL PARAMETER FOR X;
ENTRY: ONE ENDPOINT OF INTERVAL J IN WHICH A ZERO IS SEARCHED FOR;
EXIT: A VALUE APPROXIMATING THE ZERO WITHIN THE TOLERANCE 2 * T(X) WHEN ZEROINRAT HAS THE VALUE "TRUE", AND A PRESUMABLY WORTHLESS ARGUMENT VALUE OTHERWISE;

Y: <REAL VARIABLE>;
ENTRY: THE OTHER ENDPOINT OF INTERVAL J IN WHICH A ZERO IS SEARCHED FOR; UPON ENTRY X < Y AS WELL AS Y < X IS ALLOWED;
EXIT: THE OTHER STRADDLING APPROXIMATION OF THE ZERO, I.E. UPON EXIT THE VALUES OF Y AND X SATISFY 1. F(X) * F(Y) <= 0, 2. ABS(X - Y) <= 2 * T(X) AND 3. ABS(F(X)) <= ABS(F(Y)) WHEN ZEROINRAT HAS THE VALUE "TRUE", AND A PRESUMABLY WORTHLESS ARGUMENT VALUE SATISFYING CONDITIONS 2 AND 3 BUT NOT 1 OTHERWISE;

FX: <ARITHMETIC EXPRESSION>;
DEFINES FUNCTION F AS A FUNCTION DEPENDING ON THE ACTUAL PARAMETER FOR X;

TOLX: <ARITHMETIC EXPRESSION>;
DEFINES THE TOLERANCE FUNCTION T WHICH MAY DEPEND ON THE ACTUAL PARAMETER FOR X;
ONE SHOULD CHOOSE TOLX POSITIVE AND NEVER SMALLER THAN THE PRECISION OF THE MACHINE'S ARITHMETIC AT X, I.E. IN THIS ARITHMETIC X + TOLX AND X - TOLX SHOULD ALWAYS YIELD VALUES DISTINCT FROM X; OTHERWISE THE PROCEDURE MAY GET INTO A LOOP.

PROCEDURES USED: DWARF = CP30003;

EXECUTION FIELD LENGTH: NO AUXILIARY ARRAYS ARE DECLARED IN ZEROINRAT.

LANGUAGE: ALGOL 60.

METHOD AND PERFORMANCE:

THE METHOD USED IS DESCRIBED IN DETAIL IN [1].
THE NUMBER OF EVALUATIONS OF FX AND TOLX IS AT MOST
5 * LOG(ABS(X - Y)) / TAU,
WHERE X AND Y ARE THE ARGUMENT VALUES GIVEN UPON ENTRY, LOG DENOTES THE BASE 2 LOGARITHM AND TAU IS THE MINIMUM OF THE TOLERANCE FUNCTION TOLX ON THE INITIAL INTERVAL. IF UPON ENTRY X AND Y SATISFY F(X) * F(Y) <= 0, THEN CONVERGENCE IS GUARANTEED AND THE ASYMPTOTIC ORDER OF CONVERGENCE IS 1.839 FOR SIMPLE ZEROES.

EXAMPLE OF USE:

THE ZERO OF THE FUNCTION EXP(-X * 3) * (X - 1) + X ** 3, IN THE INTERVAL [0, 1], MAY BE CALCULATED BY THE FOLLOWING PROGRAM:

```
"BEGIN" "REAL" X, Y;
    "BOOLEAN" "PROCEDURE" ZEROINRAT(X, Y, FX, TOLX); "CODE" 34436;
    "REAL" "PROCEDURE" F(X); "VALUE" X; "REAL" X;
    F:= EXP(-X * 3) * ( X - 1) + X ** 3;
    X:= 0; Y:= 1;
    "IF" ZEROINRAT(X, Y, F(X), ABS(X) * "-14 + "-14) "THEN"
    OUTPUT(71, "("/4B,"("CALCULATED ZERO:")"B+.15D"+3D")", X)
    "ELSE" OUTPUT(71, "("/4B,"("NO ZERO FOUND")""")")
"END"
```

RESULT:

CALCULATED ZERO: +.489702748548240"+000

REFERENCES:

[1]: BUS, J.C.P. AND DEKKER, T.J.,
TWO EFFICIENT ALGORITHMS WITH GUARANTEED CONVERGENCE FOR FINDING A ZERO OF A FUNCTION.
MATHEMATICAL CENTRE, REPORT NW 13/74, AMSTERDAM (1974),
(ALSO TO APPEAR IN TOMS 1975).

SOURCE TEXT(S):

```
"CODE" 34150;
"BOOLEAN" "PROCEDURE" ZEROIN(X, Y, FX, TOLX);
"REAL" X, Y, FX, TOLX;
"BEGIN" "INTEGER" EXT;
        "REAL" C, FC, B, FB, A, FA, D, FD, FDB, FDA, W, MB,
        TOL, M, P, Q, DW;
        DW:= DWARF; B:= X; FB:= FX; A:= X:= Y; FA:= FX;
    INTERPOLATE: C:= A; FC:= FA; EXT:= 0;
    EXTRAPOLATE: "IF" ABS(FC) < ABS(FB) "THEN"
        "BEGIN" "IF" C ^= A "THEN" "BEGIN" D:= A; FD:= FA "END";
            A:= B; FA:= FB; B:= X:= C; FB:= FC; C:= A; FC:= FA
        "END" INTERCHANGE;
        TOL:= TOLX; M:= (C + B) * 0.5; MB:= M - B;
        "IF" ABS(MB) > TOL "THEN"
        "BEGIN" "IF" EXT > 2 "THEN" W:= MB "ELSE"
            "BEGIN" TOL:= TOL * SIGN(MB);
                P:= (B - A) * FB; "IF" EXT <= 1 "THEN"
                Q:= FA - FB "ELSE"
                "BEGIN" FDB:= (FD - FB) / (D - B);
                    FDA:= (FD - FA) / (D - A);
                    P:= FDA * P; Q:= FDB * FA - FDA * FB
                "END"; "IF" P < 0 "THEN"
                "BEGIN" P:= -P; Q:= -Q "END";
                W:= "IF" P < DW "OR" P <= Q * TOL "THEN" TOL "ELSE"
                "IF" P < MB * Q "THEN" P / Q "ELSE" MB
            "END"; D:= A; FD:= FA; A:= B; FA:= FB;
            X:= B:= B + W; FB:= FX;
            "IF" ("IF" FC >= 0 "THEN" FB >= 0 "ELSE" FB <= 0) "THEN"
            "GOTO" INTERPOLATE "ELSE"
            "BEGIN" EXT:= "IF" W = MB "THEN" 0 "ELSE" EXT + 1;
                "GOTO" EXTRAPOLATE
            "END"
        "END"; Y:= C;
        ZEROIN:= "IF" FC >= 0 "THEN" FB <= 0 "ELSE" FB >= 0
"END" ZEROIN;
        "EOP"
```

```
"CODE" 34436;
"BOOLEAN" "PROCEDURE" ZEROINRAT(X, Y, FX, TOLX);
"REAL" X, Y, FX, TOLX;
"BEGIN" "INTEGER" EXT; "BOOLEAN" FIRST;
        "REAL" B, FB, A, FA, D, FD, C, FC, FDB, FDA, W,
        MB, TOL, M, P, Q, DW;
        "REAL" "PROCEDURE" DWARF; "CODE" 30003;
        DW:= DWARF; B:= X; FB:= FX; A:= X:= Y; FA:= FX; FIRST:= "TRUE";
    INTERPOLATE: C:= A; FC:= FA; EXT:= 0;
    EXTRAPOLATE: "IF" ABS(FC) < ABS(FB) "THEN"
        "BEGIN" "IF" C ^= A "THEN" "BEGIN" D:= A; FD:= FA "END";
            A:= B; FA:= FB; B:= X:= C; FB:= FC; C:= A; FC:= FA
        "END" INTERCHANGE;
        TOL:= TOLX; M:= (C + B) * .5; MB:= M - B;
        "IF" ABS(MB) > TOL "THEN"
        "BEGIN" "IF" EXT > 3 "THEN" W:= MB "ELSE"
            "BEGIN" TOL:= TOL * SIGN(MB);
                P:= (B - A) * FB; "IF" FIRST "THEN"
                "BEGIN" Q:= FA - FB; FIRST:= "FALSE" "END" "ELSE"
                "BEGIN" FDB:= (FD - FB) / (D - B);
                    FDA:= (FD - FA) / (D - A);
                    P:= FDA * P; Q:= FDB * FA - FDA * FB
                "END"; "IF" P < 0 "THEN"
                "BEGIN" P:= -P; Q:= -Q "END";
                "IF" EXT = 3 "THEN" P:= P * 2;
                W:= "IF" P < DW "OR" P <= Q * TOL "THEN" TOL "ELSE"
                "IF" P < MB * Q "THEN" P / Q "ELSE" MB
        "END"; D:= A; FD:= FA; A:= B; FA:= FB;
        X:= B:= B + W; FB:= FX;
        "IF" ("IF" FC >= 0 "THEN" FB >= 0 "ELSE" FB <= 0) "THEN"
        "GOTO" INTERPOLATE "ELSE"
        "BEGIN" EXT:= "IF" W = MB "THEN" 0 "ELSE" EXT + 1;
            "GOTO" EXTRAPOLATE
        "END"
"END"; Y:= C;
ZEROINRAT:= "IF" FC >= 0 "THEN" FB <= 0 "ELSE" FB >= 0
"END" ZEROINRAT;
        "EOP"
```

AUTHOR: T.J. DEKKER.

CONTRIBUTORS: T.J. DEKKER AND T.H.P. REYMER.

INSTITUTE: UNIVERSITY OF AMSTERDAM.

RECEIVED: 750615.

BRIEF DESCRIPTION:

THIS SECTION CONTAINS ONE PROCEDURE FOR FINDING A ZERO OF A GIVEN DIFFERENTIABLE FUNCTION IN A GIVEN INTERVAL;
ZEROINDER APPROXIMATES A ZERO MAINLY BY MEANS OF CONFLUENT 3-POINT RATIONAL INTERPOLATION USING NOT ONLY VALUES OF THE GIVEN FUNCTION BUT ALSO OF ITS DERIVATIVE.
ZEROINDER IS TO PREFER TO ZEROIN OR ZEROINRAT (SECTION 5.1.1.1.1), IF THE DERIVATIVE IS (MUCH) CHEAPER TO EVALUATE THAN THE FUNCTION.

KEYWORDS:

ZERO SEARCHING,
ANALYTIC EQUATIONS,
SINGLE NONLINEAR EQUATION,
DERIVATIVE AVAILABLE.

CALLING SEQUENCE:

THE HEADING OF THIS PROCEDURE READS:
"BOOLEAN" "PROCEDURE" ZEROINDER(X, Y, FX, DFX, TOLX);
"REAL" X, Y, FX, DFX, TOLX;

ZEROINDER SEARCHES FOR A ZERO OF A DIFFERENTIABLE REAL FUNCTION F DEFINED ON A CERTAIN INTERVAL J;
ZEROINDER := "TRUE" WHEN A (SUFFICIENTLY SMALL) SUBINTERVAL OF J CONTAINING A ZERO OF F HAS BEEN FOUND; OTHERWISE, ZEROINDER := "FALSE";
LET DF AND T DENOTE REAL FUNCTIONS DEFINED ON J, WHERE DF IS THE DERIVATIVE OF F AND T A TOLERANCE FUNCTION DEFINING THE REQUIRED PRECISION OF THE ZERO; (FOR INSTANCE
T(X) = ABS(X) * RE + AE,
WHERE RE AND AE ARE THE REQUIRED RELATIVE AND ABSOLUTE PRECISION RESPECTIVELY);

THE MEANING OF THE FORMAL PARAMETERS IS:

X: <REAL VARIABLE>;
A JENSEN VARIABLE; THE ACTUAL PARAMETERS FOR FX, DFX AND TOLX (MAY) DEPEND ON THE ACTUAL PARAMETER FOR X;
ENTRY: ONE ENDPOINT OF INTERVAL J IN WHICH A ZERO IS SEARCHED FOR;
EXIT: A VALUE APPROXIMATING THE ZERO WITHIN THE TOLERANCE 2 * T(X) WHEN ZEROINDER HAS THE VALUE "TRUE", AND A PRESUMABLY WORTHLESS ARGUMENT VALUE OTHERWISE;

Y: <REAL VARIABLE>;
ENTRY: THE OTHER ENDPOINT OF INTERVAL J IN WHICH A ZERO IS SEARCHED FOR; UPON ENTRY X < Y AS WELL AS Y < X IS ALLOWED;
EXIT: THE OTHER STRADDLING APPROXIMATION OF THE ZERO, I.E. UPON EXIT THE VALUES OF Y AND X SATISFY 1. F(X) * F(Y) <= 0, 2. ABS(X - Y) <= 2 * T(X) AND 3. ABS(F(X)) <= ABS(F(Y)) WHEN ZEROINDER HAS THE VALUE "TRUE", AND A PRESUMABLY WORTHLESS ARGUMENT VALUE SATISFYING CONDITIONS 2 AND 3 BUT NOT 1 OTHERWISE;

FX: <ARITHMETIC EXPRESSION>;
DEFINES FUNCTION F AS A FUNCTION DEPENDING ON THE ACTUAL PARAMETER FOR X;

DFX: <ARITHMETIC EXPRESSION>;
DEFINES THE DERIVATIVE DF OF F AS A FUNCTION DEPENDING ON THE ACTUAL PARAMETER FOR X;

TOLX: <ARITHMETIC EXPRESSION>;
DEFINES THE TOLERANCE FUNCTION T WHICH MAY DEPEND ON THE ACTUAL PARAMETER FOR X;
ONE SHOULD CHOOSE TOLX POSITIVE AND NEVER SMALLER THAN THE PRECISION OF THE MACHINE'S ARITHMETIC AT X, I.E. IN THIS ARITHMETIC X + TOLX AND X - TOLX SHOULD ALWAYS YIELD VALUES DISTINCT FROM X; OTHERWISE THE PROCEDURE MAY GET INTO A LOOP.

PROCEDURES JSED: DWARF = CP30003;

REQUIRED CENTRAL MEMORY: NO AUXILIARY ARRAYS ARE DECLARED IN ZEROINDER.

RUNNING TIME: SEE METHOD AND PERFORMANCE.

LANGUAGE: ALGOL 60.

METHOD AND PERFORMANCE:

THE METHOD USED IS CONFLUENT 3-POINT RATIONAL INTERPOLATION, I.E. THE INTERPOLATION FUNCTION OF THE FORM (X - A) / (BX + C) IS FITTED EXACTLY AT 3 POINTS 2 OF WHICH ARE COINCIDING; MOREOVER, IN ORDER TO IMPROVE THE GLOBAL BEHAVIOUR OF THE PROCESS, LINEAR INTERPOLATION ON THE FUNCTION F / DF OR BISECTION ARE INCLUDED IN SOME STEPS;
THE PERFORMANCE IS AS FOLLOWS:
THE NUMBER OF EVALUATIONS OF FX, DFX AND TOLX IS AT MOST

4 LOG(ABS(X - Y)) / TAU,

WHERE X AND Y ARE THE ARGUMENT VALUES GIVEN UPON ENTRY, LOG DENOTES THE BASE 2 LOGARITHM, AND TAU IS THE MINIMUM OF THE TOLERANCE FUNCTION ON J (I.E. ZEROINDER REQUIRES AT MOST 4 TIMES THE NUMBER OF STEPS REQUIRED FOR BISECTION); IF UPON ENTRY X AND Y SATISFY F(X) * F(Y) <= 0, THEN CONVERGENCE TO A ZERO IS GUARANTEED, SO THAT UPON EXIT X AND Y SATISFY CONDITION 1 TO 3 MENTIONED ABOVE (SEE PARAMETER Y) AND ZEROINDER HAS THE VALUE "TRUE";
FOR A SIMPLE ZERO OF F, THE ASYMPTOTIC ORDER OF CONVERGENCE APPROXIMATELY EQUALS 2.414.

EXAMPLE OF USE:

THE ZERO OF THE FUNCTION EXP(-X * 3) * (X - 1) + X ** 3, IN THE INTERVAL [0, 1], MAY BE CALCULATED BY THE FOLLOWING PROGRAM:

```
"BEGIN" "REAL" X, Y;
    "BOOLEAN" "PROCEDURE" ZEROINDER(X,Y,FX,DFX,TOLX); "CODE" 34453;
    "REAL" "PROCEDURE" F(X); "VALUE" X; "REAL" X;
    F:= EXP(-X * 3) * ( X - 1) + X ** 3;
    "REAL" "PROCEDURE" DF(X); "VALUE" X; "REAL" X;
    DF:= EXP(-X * 3) * (-3 * X + 4) + 3 * X ** 2;
    X:= 0; Y:= 1;
    "IF" ZEROINDER(X, Y, F(X), DF(X), ABS(X) * "-14 + "-14) "THEN"
    OUTPUT(71, "("/4B,"("CALCULATED ZERO AND FUNCTION VALUE:")",
        /8B, 2(B+.15D"+3D4B), /4B,
        "("OTHER STRADDLING APPROXIMATION AND FUNCTION VALUE:")",
        /8B, 2(B+.15D"+3D4B)")", X, F(X), Y, F(Y))
  "ELSE" OUTPUT(71, "("/4B, "("NO ZERO FOUND")""")")
"END"
```

RESULT:

```
CALCULATED ZERO AND FUNCTION VALUE:
     +.489702748548240"+000      -.444089209850060"-015
OTHER STRADDLING APPROXIMATION AND FUNCTION VALUE:
     +.489702748548250"+000      +.177635683940030"-013
```

REFERENCES:

[1]: BUS, J.C.P. AND DEKKER, T.J.,
TWO EFFICIENT ALGORITHMS WITH GUARANTEED CONVERGENCE FOR FINDING A ZERO OF A FUNCTION.
MATHEMATICAL CENTRE, REPORT NW 13/74, AMSTERDAM (1974), (ALSO TO APPEAR IN TOMS 1975).

[2]: OSTROWSKI, A.M.,
SOLUTION OF EQUATIONS AND SYSTEMS OF EQUATIONS.
ACADEMIC PRESS 1966. CHAPTERS 3 AND 11.

SOURCE TEXT(S):

```
"CODE" 34453;
    "BOOLEAN" "PROCEDURE" ZEROINDER(X, Y, FX, DFX, TOLX);
    "REAL" X, Y, FX, DFX, TOLX;
    "BEGIN" "INTEGER" EXT;
    "REAL" B, FB, DFB, A, FA, DFA, C, FC, DFC, D, W, MB,
    TOL, M, P, Q, DW;
    "REAL" "PROCEDURE" DWARF; "CODE" 30003;
    DW:= DWARF;
    B:= X; FB:= FX; DFB:= DFX; A:= X:= Y; FA:= FX; DFA:= DFX;
INTERPOLATE: C:= A; FC:= FA; DFC:= DFA; EXT:= 0;
EXTRAPOLATE: "IF" ABS(FC) < ABS(FB) "THEN"
    "BEGIN" A:= B; FA:= FB; DFA:= DFB; B:= X:= C; FB:= FC;
        DFB:= DFC; C:= A; FC:= FA; DFC:= DFA
    "END" INTERCHANGE;
    TOL:= TOLX; M:= (C + B) * 0.5; MB:= M - B;
    "IF" ABS(MB) > TOL "THEN"
    "BEGIN" "IF" EXT > 2 "THEN" W:= MB "ELSE"
        "BEGIN" TOL:= TOL * SIGN(MB);
            D:= "IF" EXT = 2 "THEN" DFA "ELSE" (FB - FA) / (B - A);
            P:= FB * D * (B - A);
            Q:= FA * DFB - FB * D;
            "IF" P < 0 "THEN" "BEGIN" P:= -P; Q:= -Q "END";
            W:= "IF" P < DW "OR" P <= Q * TOL "THEN" TOL "ELSE"
                "IF" P < MB * Q "THEN" P / Q "ELSE" MB;
        "END"; A:= B; FA:= FB; DFA:= DFB;
        X:= B:= B + W; FB:= FX; DFB:= DFX;
        "IF" ("IF" FC >= 0 "THEN" FB >= 0 "ELSE" FB <= 0) "THEN"
        "GOTO" INTERPOLATE "ELSE"
        "BEGIN" EXT:= "IF" W = MB "THEN" 0 "ELSE" EXT + 1;
            "GOTO" EXTRAPOLATE
        "END"
    "END"; Y:= C;
    ZEROINDER:= "IF" FC >= 0 "THEN" FB <= 0 "ELSE" FB >= 0
"END" ZEROINDER;
```

AUTHOR: J.C.P.BUS.

INSTITUTE: MATHEMATICAL CENTRE.

RECEIVED: 740218.

BRIEF DESCRIPTION:

THIS SECTION CONTAINS TWO PROCEDURES FOR SOLVING SYSTEMS OF NON-LINEAR EQUATIONS, OF WHICH THE JACOBIAN IS KNOWN TO BE A BAND MATRIX.
QUANEWBND ASKS FOR AN APPROXIMATION OF THE JACOBIAN AT THE INITIAL GUESS.
QUANEWBND1 CALCULATES AN APPROXIMATION OF THE JACOBIAN AT THE INITIAL GUESS, USING FORWARD DIFFERENCES.

KEYWORDS:

NON-LINEAR EQUATIONS,
SYSTEMS OF EQUATIONS,
NO EXPLICIT JACOBIAN.

SUBSECTION: QUANEWBND.

CALLING SEQUENCE:

THE HEADING OF THIS PROCEDURE READS :

```
"PROCEDURE" QUANEWBND(N, LW, RW, X, F, JAC, FUNCT, IN, OUT);
"VALUE" N, LW, RW; "INTEGER" N, LW, RW;
"ARRAY" X, F, JAC, IN, OUT; "BOOLEAN" "PROCEDURE" FUNCT;
```

THE MEANING OF THE FORMAL PARAMETERS IS;

N: <ARITHMETIC EXPRESSION>;
THE NUMBER OF INDEPENDENT VARIABLES; THE NUMBER OF EQUATIONS SHOULD ALSO BE EQUAL TO N;

LW: <ARITHMETIC EXPRESSION>;
THE NUMBER OF CODIAGONALS TO THE LEFT OF THE MAIN DIAGONAL OF THE JACOBIAN;

RW: <ARITHMETIC EXPRESSION>;
THE NUMBER OF CODIAGONALS TO THE RIGHT OF THE MAIN DIAGONAL OF THE JACOBIAN;

X: <ARRAY IDENTIFIER>;
"ARRAY" X[1:N];
ENTRY: AN INITIAL ESTIMATE OF THE SOLUTION OF THE SYSTEM THAT HAS TO BE SOLVED;
EXIT: THE CALCULATED SOLUTION OF THE SYSTEM;

F: <ARRAY IDENTIFIER>;
"ARRAY" F[1:N];
ENTRY: THE VALUES OF THE FUNCTION COMPONENTS AT THE INITIAL GUESS;
EXIT: THE VALUES OF THE FUNCTION COMPONENTS AT THE CALCULATED SOLUTION;

JAC: <ARRAY IDENTIFIER>;
"ARRAY" JAC[1:(LW + RW) * (N - 1) + N];
ENTRY: AN APPROXIMATION OF THE JACOBIAN AT THE INITIAL ESTIMATE OF THE SOLUTION;
EXIT: AN APPROXIMATION OF THE JACOBIAN AT THE CALCULATED SOLUTION;
AN APPROXIMATION OF THE (I, J)-TH ELEMENT OF THE JACOBIAN IS GIVEN IN JAC[(LW + RW) * (I - 1) + J], FOR I = 1, ..., N AND J = MAX(1, I - LW), ..., MIN(N, I + RW);

```
FUNCT:  <PROCEDURE IDENTIFIER>;
        THE HEADING OF THIS PROCEDURE READS :
        "BOOLEAN" "PROCEDURE" FUNCT(N, L, U, X, F);
        "VALUE" N, L, U; "INTEGER" N, L, U; "ARRAY" X, F;
        THE MEANING OF THE FORMAL PARAMETERS IS :
        N:   <ARITHMETIC EXPRESSION>;
             THE NUMBER OF INDEPENDENT VARIABLES OF THE FUNCTION  F;
        L, U: <ARITHMETIC EXPRESSION>;
             LOWER  AND  UPPER  BOUND  OF  THE  FUNCTION  COMPONENT
             SUBSCRIPT;
        X:   <ARRAY IDENTIFIER>;
             THE INDEPENDENT VARIABLES ARE GIVEN IN X[1:N];
        F:   <ARRAY IDENTIFIER>;
             AFTER  A CALL OF  FUNCT  THE FUNCTION COMPONENTS  F[I],
             I = L, ..., U, SHOULD BE GIVEN IN F[L:U];
        IF   THE VALUE  OF  THE PROCEDURE IDENTIFIER  EQUALS  FALSE,
        THEN THE EXECUTION OF  QUANEWBND  WILL BE TERMINATED, WHILE
        THE VALUE OF OUT[5] IS SET EQUAL TO 2;
IN:     <ARRAY IDENTIFIER>;
        "ARRAY" IN[0:4];
        ENTRY :
        IN  THIS  AUXILIARY  ARRAY  ONE  SHOULD  GIVE THE FOLLOWING
        VALUES FOR CONTROLLING THE PROCESS:
        IN[0]:  THE MACHINE PRECISION;
        IN[1]:  THE RELATIVE PRECISION ASKED FOR;
        IN[2]:  THE ABSOLUTE PRECISION  ASKED FOR;  IF  THE  VALUE,
                DELIVERED IN  OUT[5]  EQUALS ZERO,  THEN  THE  LAST
                CORRECTION VECTOR  D, SAY,  WHICH  IS A MEASURE FOR
                THE ERROR IN THE SOLUTION,  SATIFIES THE INEQUALITY
                NORM(D) <= NORM(X) * IN[1] + IN[2],
                WHEREBY X DENOTES THE CALCULATED SOLUTION, GIVEN IN
                ARRAY  X  AND  NORM(.)  DENOTES THE EUCLIDIAN NORM;
                HOWEVER,WE CAN NOT GUARANTEE THAT THE TRUE ERROR IN
                THE SOLUTION SATISFIES THIS INEQUALITY,  ESPECIALLY
                IF  THE  JACOBIAN  IS  (NEARLY)  SINGULAR  AT  THE
                SOLUTION;
        IN[3]:  THE  MAXIMUM  VALUE  OF  THE NORM  OF  THE RESIDUAL
                VECTOR ALLOWED; IF  OUT[5] = 0,  THEN THIS RESIDUAL
                VECTOR F, SAY, SATIFIES; NORM(F) <= IN[3];
        IN[4]:  THE   MAXIMUM   NUMBER   OF   FUNCTION   COMPONENT
        EVALUATIONS ALLOWED;  L - U + 1  FUNCTION COMPONENT
        EVALUATIONS     ARE     COUNTED     EACH     CALL     OF
        FUNCT(N, L, U, X, F); IF OUT[5]=1, THEN THE PROCESS
        IS TERMINATED,  BECAUSE  THE NUMBER  OF EVALUATIONS
        EXCEEDED THE VALUE GIVEN IN IN[4];
```

```
OUT:      <ARRAY IDENTIFIER>;
          "ARRAY" OUT[1:5];
          EXIT ;
OUT[1]: THE EUCLIDIAN NORM OF THE LAST STEP ACCEPTED;
OUT[2]: THE EUCLIDIAN NORM  OF  THE RESIDUAL VECTOR  AT THE
        CALCULATED SOLUTION;
OUT[3]: THE  NUMBER   OF   FUNCTION  COMPONENT  EVALUATIONS
        PERFORMED;
OUT[4]: THE NUMBER OF ITERATIONS CARRIED OUT;
OUT[5]: THE INTEGER VALUE DELIVERED IN  OUT[5]  GIVES  SOME
        INFORMATION  ABOUT THE TERMINATION  OF THE PROCESS;
        OUT[5] = 0: THE PROCESS  IS TERMINATED  IN A NORMAL
                    WAY; THE LAST STEP AND  THE NORM OF THE
                    RESIDUAL VECTOR SATISFY  THE CONDITIONS
                    (SEE IN[2], IN[3]);
        IF  OUT[5] ^= 0,  THEN  THE PROCESS  IS  TERMINATED
        PREMATURALY;
        OUT[5] = 1: THE   NUMBER   OF   FUNCTION  COMPONENT
                    EVALUATIONS  EXCEEDS THE VALUE GIVEN IN
                    IN[4];
        OUT[5] = 2: A CALL OF  FUNCT  DELIVERED  THE  VALUE
                    FALSE;
        OUT[5] = 3: THE  APPROXIMATION   OF   THE  JACOBIAN
                    MATRIX TURNS OUT TO BE SINGULAR.
```

PROCEDURES USED:

```
MULVEC      = CP31020,
DUPVEC      = CP31030,
VECVEC      = CP34010,
ELMVEC      = CP34020,
DECSOLBND   = CP34322.
```

EXECUTION FIELD LENGTH:

THE MAXIMUM NUMBER OF WORDS, NECESSARY FOR THE ARRAYS DECLARED IN QUANEWBND EQUALS MAX(N * 3 + (N - 1) * (LW + RW), 4N).

RUNNING TIME: PROPORTIONAL TO N * LW * (LW + RW + 1).

LANGUAGE: ALGOL 60.

METHOD AND PERFORMANCE:

THE METHOD USED IN QUANEWBND IS THE SAME AS GIVEN IN [1]; THE SAME PROBLEMS HAVE BEEN TESTED AND THE RESULTS ARE THE SAME OR BETTER THAN THOSE REPORTED IN [1]; CITING [1], WE CAN SAY THAT "THIS METHOD OFFERS A USEFUL IF MODEST IMPROVEMENT UPON NEWTON'S METHOD, BUT THIS IMPROVEMENT TENDS TO VANISH AS THE NONLINEARITIES BECOME MORE PRONOUNCED".

EXAMPLE OF USE: SEE QUANEWBND1 (THIS SECTION).

REFERENCES:

[1] BROYDEN C.G.
THE CONVERGENCE OF AN ALGORITHM FOR SOLVING SPARSE NONLINEAR SYSTEMS,
MATH. COMP., VOL.25 (1971).

SUBSECTION: QUANEWBND1.

CALLING SEQUENCE:

THE HEADING OF THIS PROCEDURE READS :

```
"PROCEDURE" QUANEWBND1(N, LW, RW, X, F, FUNCT, IN, OUT);
"VALUE" N, LW, RW; "INTEGER" N, LW, RW;
"ARRAY" X, F, IN, OUT; "BOOLEAN" "PROCEDURE" FUNCT;
```

THE MEANING OF THE FORMAL PARAMETERS IS:

N: <ARITHMETIC EXPRESSION>;
THE NUMBER OF INDEPENDENT VARIABLES; THE NUMBER OF EQUATIONS SHOULD ALSO BE EQUAL TO N;

LW: <ARITHMETIC EXPRESSION>;
THE NUMBER OF CODIAGONALS TO THE LEFT OF THE MAIN DIAGONAL OF THE JACOBIAN;

RW: <ARITHMETIC EXPRESSION>;
THE NUMBER OF CODIAGONALS TO THE RIGHT OF THE MAIN DIAGONAL OF THE JACOBIAN;

X: <ARRAY IDENTIFIER>;
"ARRAY" X[1:N];
ENTRY: AN INITIAL ESTIMATE OF THE SOLUTION OF THE SYSTEM THAT HAS TO BE SOLVED;
EXIT: THE CALCULATED SOLUTION OF THE SYSTEM;

F: <ARRAY IDENTIFIER>;
"ARRAY" F[1:N];
EXIT: THE VALUES OF THE FUNCTION COMPONENTS AT THE CALCULATED SOLUTION;

FUNCT: <PROCEDURE IDENTIFIER>;
THE HEADING OF THIS PROCEDURE READS :
"BOOLEAN" "PROCEDURE" FUNCT(N, L, U, X, F);
"VALUE" N, L, U; "INTEGER" N, L, U; "ARRAY" X, F;
THE MEANING OF THE FORMAL PARAMETERS IS :
N: <ARITHMETIC EXPRESSION>;
THE NUMBER OF INDEPENDENT VARIABLES OF THE FUNCTION F;
L, U: <ARITHMETIC EXPRESSION>;
LOWER AND UPPER BOUND OF THE FUNCTION COMPONENT SUBSCRIPT;
X: <ARRAY IDENTIFIER>;
THE INDEPENDENT VARIABLES ARE GIVEN IN X[1:N];
F: <ARRAY IDENTIFIER>;
AFTER A CALL OF FUNCT THE FUNCTION COMPONENTS F[I], I = L, ..., U, SHOULD BE GIVEN IN F[L:U];
IF THE VALUE OF THE PROCEDURE IDENTIFIER EQUALS FALSE, THEN THE EXECUTION OF QUANEWBND WILL BE TERMINATED, WHILE THE VALUE OF OUT[5] IS SET EQUAL TO 2;

IN: <ARRAY IDENTIFIER>;
"ARRAY" IN[0:4];
ENTRY :
IN THIS AUXILIARY ARRAY ONE SHOULD GIVE THE FOLLOWING VALUES FOR CONTROLLING THE PROCESS:
IN[0]: THE MACHINE PRECISION;
IN[1]: THE RELATIVE PRECISION ASKED FOR;
IN[2]: THE ABSOLUTE PRECISION ASKED FOR; IF THE VALUE, DELIVERED IN OUT[5] EQUALS ZERO, THEN THE LAST CORRECTION VECTOR D, SAY, WHICH IS A MEASURE FOR THE ERROR IN THE SOLUTION, SATIFIES THE INEQUALITY NORM(D) <= NORM(X) * IN[1] + IN[2], WHEREBY X DENOTES THE CALCULATED SOLUTION, GIVEN IN ARRAY X AND NORM(.) DENOTES THE EUCLIDIAN NORM; HOWEVER,WE CAN NOT GUARANTEE THAT THE TRUE ERROR IN THE SOLUTION SATISFIES THIS INEQUALITY, ESPECIALLY IF THE JACOBIAN IS (NEARLY) SINGULAR AT THE SOLUTION;
IN[3]: THE MAXIMUM VALUE OF THE NORM OF THE RESIDUAL VECTOR ALLOWED; IF OUT[5] = 0, THEN THIS RESIDUAL VECTOR F, SAY, SATIFIES: NORM(F) <= IN[3];
IN[4]: THE MAXIMUM NUMBER OF FUNCTION COMPONENT EVALUATIONS ALLOWED; L - U + 1 FUNCTION COMPONENT EVALUATIONS ARE COUNTED EACH CALL OF FUNCT(N, L, U, X, F); IF OUT[5]=1, THEN THE PROCESS IS TERMINATED, BECAUSE THE NUMBER OF EVALUATIONS EXCEEDED THE VALUE GIVEN IN IN[4];
IN[5]: THE JACOBIAN MATRIX AT THE INITIAL GUESS IS APPROXIMATED USING FORWARD DIFFERENCES, WITH AN FIXED INCREMENT TO EACH VARIABLE THAT EQUALS THE VALUE GIVEN IN IN[5];

```
OUT:      <ARRAY IDENTIFIER>;
          "ARRAY" OUT[1:5];
          EXIT :
OUT[1]: THE EUCLIDIAN NORM OF THE LAST STEP ACCEPTED;
OUT[2]: THE EUCLIDIAN NORM  OF  THE RESIDUAL VECTOR  AT THE
        CALCULATED SOLUTION;
OUT[3]: THE  NUMBER   OF   FUNCTION  COMPONENT  EVALUATIONS
        PERFORMED;
OUT[4]: THE NUMBER OF ITERATIONS CARRIED OUT;
OUT[5]: THE INTEGER VALUE DELIVERED IN  OUT[5]  GIVES  SOME
        INFORMATION  ABOUT THE TERMINATION  OF THE PROCESS;
        OUT[5] = 0: THE PROCESS  IS TERMINATED  IN A NORMAL
                    WAY; THE LAST STEP AND  THE NORM OF THE
                    RESIDUAL VECTOR SATISFY  THE CONDITIONS
                    (SEE IN[2], IN[3]);
        IF  OUT[5] ^= 0,  THEN  THE PROCESS  IS  TERMINATED
        PREMATURALY;
        OUT[5] = 1: THE   NUMBER   OF   FUNCTION COMPONENT
                    EVALUATIONS  EXCEEDS THE VALUE GIVEN IN
                    IN[4];
        OUT[5] = 2: A CALL OF  FUNCT  DELIVERED  THE  VALUE
                    FALSE;
        OUT[5] = 3: THE  APPROXIMATION   OF   THE  JACOBIAN
                    MATRIX TURNS OUT TO BE SINGULAR.
```

PROCEDURES USED:

JACOBNBNDF = CP34439,
QUANEWBND = CP34430.

EXECUTION FIELD LENGTH:

QUANEWBND: DECLARES AN AUXILIARY ARRAY OF DIMENSION ONE AND ORDER N + (N - 1) * (LW + RW).

RUNNING TIME: PROPORTIONAL TO N * LW * (LW + RW + 1).

LANGUAGE: ALGOL 60.

METHOD AND PERFORMANCE:

QUANEWBND: USES JACOBNBNDF (SECTION 4.3.2.1) TO CALCULATE AN INITIAL APPROXIMATION OF THE JACOBIAN MATRIX AT THE INITIAL GUESS GIVEN IN X[1:N] AND SOLVES THE NONLINEAR SYSTEM BY CALLING QUANEWBND (THIS SECTION).

EXAMPLE OF USE:

LET THE FUNCTION F BE DEFINED BY (SEE [1]):
F[1] = (3 - 2 * X[1]) * X[1] + 1 - 2 * X[2],
F[I] = (3 - 2 * X[I]) * X[I] + 1 - X[I - 1] - 2 * X[I + 1], I = 2, ..., N - 1,
F[N] = (3 - 2 * X[N]) * X[N] + 1 - X[N - 1];
LET AN INITIAL ESTIMATE OF THE SOLUTION OF THE SYSTEM F(X) = 0 BE GIVEN BY X[I] = -1, I = 1, ..., N; THEN THE FOLLOWING PROGRAM MAY SOLVE THIS SYSTEM FOR N = 600 AND PRINTS SOME RESULTS.

```
"BEGIN"
    "PROCEDURE" QUANEWBND1(N, L, R, X, F, FU, I, 0); "CODE" 34431;
    "BOOLEAN" "PROCEDURE" FUN(N, L, U, X, F); "VALUE" N, L, U;
    "INTEGER" N, L, U; "ARRAY" X, F;
    "BEGIN" "INTEGER" I; "REAL" X1, X2, X3;
        X1:= "IF" L = 1 "THEN" 0 "ELSE" X[L - 1]; X2:= X[L];
        X3:= "IF" L = N "THEN" 0 "ELSE" X[L + 1];
        "FOR" I:= L "STEP" 1 "UNTIL" U "DO"
        "BEGIN" F[I]:= (3 - 2 * X2) * X2 + 1 - X1 - X3 * 2;
            X1:= X2; X2:= X3;
            X3:= "IF" I <= N - 2 "THEN" X[I + 2] "ELSE" 0
        "END"; FUN:= "TRUE"
    "END" FUN;

    "INTEGER" I; "ARRAY" X, F[1:600], IN[0:5], OUT[1:5];
    "FOR" I := 1 "STEP" 1 "UNTIL" 600 "DO" X[I]:= -1;
    IN[0]:= "-14; IN[1]:= IN[2]:= IN[3]:= "-6; IN[4]:= 20000;
    IN[5]:= 0.001;
    QUANEWBND1(600, 1, 1, X, F, FUN, IN, OUT);
    OUTPUT(71, "("//,"(" NORM RESIDUALVECTOR: ")"+.15D"+3D,/,
    "(" LENGTH OF LAST STEP: ")"+.15D"+3D,/,
    "(" NUMBER OF FUNCTION COMPONENT EVALUATIONS: ")"5ZD,/,
    "(" NUMBER OF ITERATIONS: ")"4ZD/,"("REPORT: ")"D/")",
    OUT[2], OUT[1], OUT[3], OUT[4], OUT[5])
"END"
```

RESULTS:

```
NORM RESIDUALVECTOR: +.221010684482660"-006
LENGTH OF LAST STEP: +.302712457332660"-006
NUMBER OF FUNCTION COMPONENT EVALUATIONS:   6598
NUMBER OF ITERATIONS:      7
REPORT: 0
```

REFERENCES:

[1] BROYDEN C.G.
THE CONVERGENCE OF AN ALGORITHM FOR SOLVING SPARSE NONLINEAR SYSTEMS,
MATH. COMP., VOL.25 (1971).

```
SOURCE TEXT(S);
"CODE" 34430;
"PROCEDURE" QUANEWBND(N, LW, RW, X, F, JAC, FUNCT, IN, OUT);
"VALUE" N, LW, RW; "INTEGER" N, LW, RW;
"ARRAY" X, F, JAC, IN, OUT; "BOOLEAN" "PROCEDURE" FUNCT;
"BEGIN" "INTEGER" L, IT, FCNT, FMAX, ERR, B;
    "REAL" MACHEPS, RELTOL, ABSTOL, TOLRES, ND, MZ, RES;
    "ARRAY" DELTA[1:N];

    "REAL" "PROCEDURE" VECVEC(L, U, SHIFT, A, B); "CODE" 34010;
    "REAL" "PROCEDURE" ELMVEC(L, U, SHIFT, A, B, X); "CODE" 34020;
    "PROCEDURE" MULVEC(L, U, SHIFT, A, B, X); "CODE" 31020;
    "PROCEDURE" DUPVEC(L, U, SHIFT, A, B); "CODE" 31030;
    "REAL" "PROCEDURE" DECSOLBND(A, N, LW, RW, AUX, B);
    "CODE" 34322;


    "REAL" "PROCEDURE" EVALUATE(N, X, F); "VALUE" N;
    "INTEGER" N; "ARRAY" X, F;
    "BEGIN" FCNT:= FCNT + N; "IF" ^ FUNCT(N, 1, N, X, F) "THEN"
        "BEGIN" ERR:= 2; "GOTO" EXIT "END";
    "IF" FCNT > FMAX "THEN" ERR:= 1;
    EVALUATE:= SQRT(VECVEC(1, N, 0, F, F))
"END" EVAL;

"BOOLEAN" "PROCEDURE" DIRECTION;
"BEGIN" "ARRAY" LU[1:L], AUX[1:5]; AUX[2]:= MACHEPS;
    MULVEC(1, N, 0, DELTA, F, -1); DUPVEC(1, L, 0, LU, JAC);
    DECSOLBND(LU, N, LW, RW, AUX, DELTA);
    DIRECTION:= AUX[3] = N
"END" SOLLINSYS;

"BOOLEAN" "PROCEDURE" TEST(ND, TOLD, NRES, TOLRES, ERR);
"VALUE" ND, TOLD; "INTEGER" ERR; "REAL" ND, TOLD, NRES, TOLRES;
TEST:= ERR ^= 0 "OR" (NRES < TOLRES "AND" ND < TOLD);

"PROCEDURE" UPDATE JAC;
"BEGIN" "INTEGER"  I, J, K, R, M; "REAL" MUL, CRIT;
    "ARRAY" PP, S[1:N];
    CRIT:= ND * MZ;
    "FOR" I:= 1 "STEP" 1 "UNTIL" N "DO" PP[I]:= DELTA[I] ** 2;
    R:= 1; K:= 1; M:= RW + 1;
    "FOR" I:= 1 "STEP" 1 "UNTIL" N "DO"
    "BEGIN" MUL:= 0; "FOR" J:= R "STEP" 1 "UNTIL" M "DO"
        MUL:= MUL + PP[J]; J:= R - K;
        "IF" ABS(MUL) > CRIT "THEN"
        ELMVEC(K, M - J, J, JAC, DELTA, F[I] / MUL); K:= K + B;
        "IF" I > LW "THEN" R:= R + 1 "ELSE" K:= K - 1;
        "IF" M < N "THEN" M:= M + 1
    "END"
"END" UPDATEJAC;
```

```
        MACHEPS:= IN[0]; RELTOL:= IN[1]; ABSTOL:= IN[2];
        TOLRES:= IN[3]; FMAX:= IN[4]; MZ:= MACHEPS ** 2;
        IT:= FCNT:= 0; B:= LW + RW; L:= (N - 1) * B + N; B:= B + 1;
        RES:= SQRT(VECVEC(1, N, 0, F, F)); ERR:= 0;
    ITERATE: "IF" ^ TEST(SQRT(ND), SQRT(VECVEC(1, N, 0, X, X)) * RELTOL
        + ABSTOL, RES, TOLRES, ERR) "THEN"
        "BEGIN" IT:= IT + 1; "IF" IT ^= 1 "THEN" UPDATEJAC;
            "IF" ^ DIRECTION "THEN" ERR:= 3 "ELSE"
            "BEGIN" ELMVEC(1, N, 0, X, DELTA, 1);
                ND:= VECVEC(1, N, 0, DELTA, DELTA);
                RES:= EVALUATE(N, X, F); "GOTO" ITERATE
             "END"
        "END";
    EXIT: OUT[1]:= SQRT(ND); OUT[2]:=RES; OUT[3]:= FCNT;
        OUT[4]:= IT; OUT[5]:= ERR
    "END" QUANEWBND;
        "EOP"

"CODE" 34431;
    "PROCEDURE" QUANEWBND1(N, LW, RW, X, F, FUNCT, IN, OUT);
    "VALUE" N, LW, RW; "INTEGER" N, LW, RW; "ARRAY" X, F, IN, OUT;
    "BOOLEAN" "PROCEDURE" FUNCT;
    "BEGIN" "INTEGER" I, K; "REAL" S;
        "PROCEDURE" QUANEWBND(N, L, R, X, F, J, G, I, O); "CODE" 34430;
        "ARRAY" JAC[1:(LW + RW) * (N - 1) + N];
        "PROCEDURE" JACOBNBNDF(N,L, R, X, F, J, I, D, F); "CODE" 34439;
        FUNCT(N, 1, N, X, F); S:= IN[5];
        K:= (LW + RW)*(N - 1) + N*2 - ((LW - 1)*LW + (RW - 1)*RW) // 2;
        IN[4]:= IN[4] - K;
        JACOBNBNDF(N, LW, RW, X, F, JAC, I, S, FUNCT);
        QUANEWBND(N, LW, RW, X, F, JAC, FUNCT, IN, OUT);
        IN[4]:= IN[4] + K; OUT[3]:= OUT[3] + K
    "END" QUANEWBND1;
        "EOP"
```

AUTHOR: J.C.P.BUS.

INSTITUTE: MATHEMATICAL CENTRE.

RECEIVED: 73-06-20.

BRIEF DESCRIPTION:

THIS SECTION CONTAINS TWO PROCEDURES, RNK1MIN AND FLEMIN, FOR MINIMIZING A GIVEN DIFFERENTIABLE FUNCTION OF SEVERAL VARIABLES; BOTH PROCEDURES USE A VARIABLE METRIC METHOD; THE USER HAS TO PROGRAM THE EVALUATION OF THE FUNCTION AND ITS GRADIENT; THE CHOICE OF RNK1MIN AND FLEMIN IS DEPENDENT ON THE PROBLEM INVOLVED; IF THE NUMBER OF VARIABLES OF THE FUNCTION TO BE MINIMIZED IS VERY LARGE AND THE CALCULATION OF THE FUNCTION AND ITS GRADIENT IS RELATIVELY CHEAP (THE NUMBER OF ARITHMETICAL OPERATIONS IS OF ORDER AT MOST N ** 2),THEN THE USER IS ADVISED TO USE FLEMIN; IF THE HESSIAN OF THE FUNCTION IS EXPECTED TO BE (ALMOST) SINGULAR AT THE MINIMUM, THEN RNK1MIN IS PREFERRED; FOUR AUXILIARY PROCEDURES ARE ALSO DESCRIBED: LINEMIN, RNK1UPD, DAVUPD AND FLEUPD.

KEYWORDS:

OPTIMIZATION,
HIGHER - DIMENSIONAL,
UNCONSTRAINED,
VARIABLE METRIC METHOD.

SUBSECTION: LINEMIN.

CALLING SEQUENCE:

THE HEADING OF THIS AUXILIARY PROCEDURE IS:
"PROCEDURE" LINEMIN(N, X, D, ND, ALFA, G, FUNCT, F0, F1, DF0, DF1,
EVLMAX, STRONGSEARCH, IN);
"VALUE" N, ND, F0, DF0, STRONGSEARCH;
"INTEGER" N, EVLMAX; "BOOLEAN" STRONGSEARCH;
"REAL" ND, ALFA, F0, F1, DF0, DF1;
"ARRAY" X, D, G, IN; "REAL" "PROCEDURE" FUNCT;

THE MEANING OF THE FORMAL PARAMETERS IS:
N: <ARITHMETICAL EXPRESSION>;
THE NUMBER OF VARIABLES OF THE GIVEN FUNCTION F;
X: <ARRAY IDENTIFIER>;
"ARRAY" X[1 : N];
ENTRY: A VECTOR X0, SUCH THAT F IS DECREASING IN X0, IN
THE DIRECTION GIVEN BY D;
EXIT: THE CALCULATED APPROXIMATION OF THE VECTOR FOR
WHICH F IS MINIMAL ON THE LINE DEFINED BY:
X0 + ALFA * D, (ALFA > 0);
D: <ARRAY IDENTIFIER>;
"ARRAY" D[1 : N];
ENTRY: THE DIRECTION OF THE LINE ON WHICH F HAS TO BE
MINIMIZED;
ND: <ARITHMETICAL EXPRESSION>;
ENTRY: THE EUCLIDIAN NORM OF THE VECTOR GIVEN IN D[1 : N];
ALFA: <VARIABLE>;
THE INDEPENDENT VARIABLE, THAT DEFINES THE POSITION ON THE
LINE ON WHICH F HAS TO BE MINIMIZED;
THIS LINE IS DEFINED BY X0 + ALFA * D, (ALFA > 0);
ENTRY: AN ESTIMATE ALFA0 OF THE VALUE FOR WHICH
H(ALFA) = F(X0 + ALFA * D), (ALFA > 0), IS MINIMAL;
EXIT: THE CALCULATED APPROXIMATION ALFAM OF THE VALUE FOR
WHICH H(ALFA) IS MINIMAL;
G: <ARRAY IDENTIFIER>;
"ARRAY" G[1 : N];
EXIT: THE GRADIENT OF F AT THE CALCULATED APPROXIMATION
OF THE MINIMUM;
FUNCT: <PROCEDURE IDENTIFIER>;
THE HEADING OF THIS PROCEDURE SHOULD BE:
"REAL" "PROCEDURE" FUNCT(N, X, G); "VALUE" N;
"INTEGER" N; "ARRAY" X, G;
A CALL OF FUNCT SHOULD EFFECTUATE IN:
1: FUNCT:= F(X);
2: THE VALUE OF G[I], (I = 1, ..., N), BECOMES THE VALUE
OF THE I - TH COMPONENT OF THE GRADIENT OF F AT X;

```
F0:       <ARITHMETICAL EXPRESSION>;
          ENTRY:  THE VALUE OF H(0), (SEE ALFA);
F1:       <VARIABLE>;
          ENTRY:  THE VALUE OF H(ALFA0);
          EXIT:   THE VALUE OF H(ALFAM), (SEE ALFA);
DF0:      <ARITHMETICAL EXPRESSION>;
          ENTRY:  THE VALUE OF THE DERIVATIVE OF H AT ALFA = 0;
DF1:      <VARIABLE>;
          ENTRY:  THE VALUE OF THE DERIVATIVE OF H AT ALFA = ALFA0;
          EXIT:   THE VALUE OF THE DERIVATIVE OF H AT ALFA = ALFAM;
EVLMAX:   <VARIABLE>;
          ENTRY:  THE MAXIMUM ALLOWED NUMBER OF CALLS OF FUNCT;
          EXIT:   THE NUMBER OF TIMES FUNCT HAS BEEN CALLED;
STRONGSEARCH:
          <BOOLEAN EXPRESSION>;
          IF  THE VALUE OF STRONGSEARCH IS  TRUE,  THEN  THE  PROCESS
          MAKES USE OF TWO STOPPING CRITERIA:
          A:  THE NUMBER OF TIMES FUNCT HAS BEEN CALLED EXCEEDS THE
              GIVEN VALUE OF EVLMAX;
          B:  AN INTERVAL IS FOUND  WITH LENGTH  LESS  THAN TWO TIMES
              THE PRESCRIBED PRECISION,ON WICH A MINIMUM IS EXPECTED;
          IF THE VALUE OF STRONGSEARCH  IS FALSE,  THE PROCESS  MAKES
          ALSO USE OF A THIRD STOPPING CRITERION :
          C:  MU <= (H(ALFAK) - H(ALFA0)) / (ALFAK * DF0) <= 1 - MU,
              WHEREBY ALFA IS THE CURRENT ITERATE AND MU A PRESCRIBED
              CONSTANT;
IN:       <ARRAY IDENTIFIER>;
          ENTRY:
          "ARRAY" IN[1:3];
          IN[1]:  THE  RELATIVE  PRECISION,  EPSR,  NECESSARY FOR THE
                  STOPPING CRITERION B, (SEE STRONGSEARCH);
          IN[2]:  THE  ABSOLUTE  PRECISION,  EPSA,  NECESSARY FOR THE
                  STOPPING CRITERION B, (SEE STRONGSEARCH);
          THE PRESCRIBED PRECISION, EPS, AT ALFA = ALFAK IS GIVEN BY:
          EPS = NORM ( X0 + ALPHA * D ) * EPSR + EPSA, WHERE
          NORM ( . ) DENOTES THE EUCLIDEAN NORM,
          IN[3]: THE PARAMETER MU NECESSARY FOR STOPPING CRITERION C;
                 THIS  PARAMETER  MUST  SATISFY:   0 < MU < 0,5 ; IN
                 PRACTICE,A CHOICE OF MU = 0,0001 IS ADVISED.
```

DATA AND RESULTS:

LINEMIN CALCULATES AN APPROXIMATION OF A MINIMUM OF A HIGHER - DIMENSIONAL FUNCTION ON A GIVEN LINE;
THE QUANTITY DF0 MUST SATIFY: DF0 < 0;
IF MOREOVER DF1 > 0, THEN THE PROCEDURE WILL YIELD A RESULT THAT SATISFIES ONE OF THE CHOSEN STOPPING CRITERIA, (SEE STRONGSEARCH), OTHERWISE WE CAN NOT GUARANTEE SUCH A RESULT.

PROCEDURES USED:

VECVEC = CP34010,
ELMVEC = CP34020,
DUPVEC = CP31030.

REQUIRED CENTRAL MEMORY:

EXECUTION FIELD LENGTH: N + 17 WORDS.

LANGUAGE: ALGOL 60.

METHOD AND PERFORMANCE:

AN APPROXIMATION TO THE MINIMUM ON THE GIVEN LINE IS CALCULATED WITH CUBIC INTERPOLATION ([2]);THE STOPPING CRITERION USED WHEN THE VALUE OF STRONGSEARCH IS FALSE IS DESCRIBED IN [3] AND [4]; A DETAILED DESCRIPTION OF THIS PROCEDURE IS GIVEN IN [1].

SUBSECTION: RNK1UPD.

CALLING SEQUENCE:

```
THE HEADING OF THIS AUXILIARY PROCEDURE IS:
"PROCEDURE" RNK1UPD(H, N, V, C); "VALUE" N, C;
"INTEGER" N; "REAL" C; "ARRAY" H, V;

THE MEANING OF THE FORMAL PARAMETERS IS:
N:      <ARITHMETICAL EXPRESSION>;
        THE ORDER OF THE  SYMMETRIC  MATRIX, WHOSE UPPERTRIANGLE IS
        STORED COLUMNWISE IN THE ONE - DIMENSIONAL ARRAY H;
C:      <ARITHMETICAL EXPRESSION>;
        SEE V;
V:      <ARRAY IDENTIFIER>;
        "ARRAY" V[1 : N];
        THE GIVEN MATRIX IS UPDATED (ANOTHER MATRIX IS ADDED TO IT)
        WITH A SYMMETRIC MATRIX , U, OF RANK ONE, DEFINED BY:
                U[I,J] = C * V[I] * V[J];
H:      <ARRAY IDENTIFIER>;
        "ARRAY" H[1 : N * (N + 1) // 2];
        ENTRY:  THE  UPPERTRIANGLE  (STORED  COLUMNWISE)  OF  THE
                SYMMETRIC MATRIX THAT HAS TO BE UPDATED;
        EXIT:   THE  UPPERTRIANGLE  (STORED  COLUMNWISE)  OF  THE
                UPDATED MATRIX.
```

PROCEDURES USED:

ELMVEC = CP34020.

REQUIRED CENTRAL MEMORY:

EXECUTION FIELD LENGTH: 2 WORDS.

LANGUAGE: ALGOL 60.

SUBSECTION: DAVUPD.

CALLING SEQUENCE:

```
THE HEADING OF THIS AUXILIARY PROCEDURE IS:
"PROCEDURE" DAVUPD(H, N, V, W, C1, C2); "VALUE" N, C1, C2;
"INTEGER" N; "REAL" C1, C2; "ARRAY" H, V, W;

THE MEANING OF THE FORMAL PARAMETERS IS:
N:      <ARITHMETICAL EXPRESSION>;
        THE ORDER OF THE SYMMETRIC MATRIX WHOSE UPPERTRIANGLE IS
        STORED COLUMNWISE IN THE ONE - DIMENSIONAL ARRAY H;
C1:     <ARITHMETICAL EXPRESSION>;
        SEE W;
C2:     <ARITHMETICAL EXPRESSION>;
        SEE W;
V:      <ARRAY IDENTIFIER>;
        "ARRAY" V[1 : N], SEE W;
W:      <ARRAY IDENTIFIER>;
        "ARRAY" W[1 : N];
        THE GIVEN MATRIX IS UPDATED WITH A SYMMETRIC MATRIX U OF
        RANK TWO, DEFINED BY:
        U[I,J] = C1 * V[I] * V[J] - C2 * W[I] * W[J];
H:      <ARRAY IDENTIFIER>;
        "ARRAY" H[1 : N * (N + 1) // 2];
        ENTRY: THE UPPERTRIANGLE (STORED COLUMNWISE) OF THE MATRIX
               THAT HAS TO BE UPDATED;
        EXIT:  THE UPPERTRIANGLE (STORED COLUMNWISE) OF THE
               UPDATED MATRIX.
```

PROCEDURES USED: NONE.

REQUIRED CENTRAL MEMORY:

EXECUTION FIELD LENGTH: 3 WORDS.

LANGUAGE: ALGOL 60.

SUBSECTION: FLEUPD.

CALLING SEQUENCE:

THE HEADING OF THIS AUXILIARY PROCEDURE IS:
"PROCEDURE" FLEUPD(H, N, V, W, C1, C2); "VALUE" N, C1, C2;
"INTEGER" N; "REAL" C1, C2; "ARRAY" H, V, W;

THE MEANING OF THE FORMAL PARAMETERS IS:

N: <ARITHMETICAL EXPRESSION>;
THE ORDER OF THE SYMMETRIC MATRIX WHOSE UPPERTRIANGLE IS STORED COLUMNWISE IN THE ONE - DIMENSIONAL ARRAY H;

C1: <ARITHMETICAL EXPRESSION>;
SEE W;

C2: <ARITHMETICAL EXPRESSION>;
SEE W;

V: <ARRAY IDENTIFIER>;
"ARRAY" V[1 : N], SEE W;

W: <ARRAY IDENTIFIER>;
"ARRAY" W[1 : N];
THE GIVEN MATRIX IS UPDATED WITH A SYMMETRIC MATRIX U OF RANK TWO, DEFINED BY:
U[I,J] = C2 * V[I] * V[J] - C1 *(V[I] * W[J] + W[I] * V[J]);

H: <ARRAY IDENTIFIER>;
"ARRAY" H[1 : N * (N + 1) // 2];
ENTRY: THE UPPERTRIANGLE (STORED COLUMNWISE) OF THE MATRIX THAT HAS TO BE UPDATED;
EXIT: THE UPPERTRIANGLE (STORED COLUMNWISE) OF THE UPDATED MATRIX.

PROCEDURE USED: NONE.

REQUIRED CENTRAL MEMORY:

EXECUTION FIELD LENGTH: 3 WORDS.

LANGUAGE: ALGOL 60.

SUBSECTION: RNK1MIN.

CALLING SEQUENCE:

```
THE HEADING OF THIS PROCEDURE IS:
"REAL" "PROCEDURE" RNK1MIN(N, X, G, H, FUNCT, IN, OUT);
"VALUE" N; "INTEGER" N;
"ARRAY" X, G, H, IN, OUT;
"REAL" "PROCEDURE" FUNCT;

RNK1MIN: DELIVERS THE CALCULATED LEAST VALUE OF THE GIVEN FUNCTION;

THE MEANING OF THE FORMAL PARAMETERS IS:
N:      <ARITHMETICAL EXPRESSION>;
        THE NUMBER OF VARIABLES OF THE FUNCTION TO BE MINIMIZED;
X:      <ARRAY IDENTIFIER>;
        "ARRAY" X[1 : N];
        THE INDEPENDENT VARIABLES;
        ENTRY:  AN APPROXIMATION OF A MINIMUM OF THE FUNCTION;
        EXIT:   THE CALCULATED MINIMUM OF THE FUNCTION;
G:      <ARRAY IDENTIFIER>;
        "ARRAY" G[1 : N];
        EXIT:   THE GRADIENT OF THE FUNCTION AT THE CALCULATED
                MINIMUM;
H:      <ARRAY IDENTIFIER>;
        "ARRAY" H[1 : N * (N + 1) // 2];
        THE UPPERTRIANGLE OF AN APPROXIMATION OF THE INVERSE
        HESSIAN IS STORED COLUMNWISE IN H;
        IF IN[6] > 0 INITIALIZING OF H WILL BE DONE AUTOMATICALLY
        AND THE INITIAL APPROXIMATION OF THE INVERSE HESSIAN WILL
        EQUAL THE UNITMATRIX MULTIPLIED WITH THE VALUE OF IN[6];
        IF IN[6] < 0 NO INITIALIZING OF H WILL BE DONE AND THE USER
        SHOULD GIVE IN H AN APPROXIMATION OF THE INVERSE HESSIAN,
        AT THE STARTING POINT;THE UPPERTRIANGLE OF AN APPROXIMATION
        OF THE INVERSE HESSIAN AT THE CALCULATED MINIMUM IS
        DELIVERED IN H;
FUNCT:  <PROCEDURE IDENTIFIER>;
        THE HEADING OF THIS PROCEDURE SHOULD BE:
        "REAL" "PROCEDURE" FUNCT(N, X, G); "VALUE" N;
        "INTEGER" N; "ARRAY" X, G;
        A CALL OF FUNCT MUST EFFECTUATE IN:
        1:  FUNCT BECOMES THE VALUE OF THE FUNCTION TO BE MINIMIZED
            AT THE POINT X;
        2:  THE VALUE OF G[I], (I = 1, ..., N), BECOMES THE VALUE
            OF THE I - TH COMPONENT OF THE GRADIENT OF THE FUNCTION
            AT X;
```

IN: <ARRAY IDENTIFIER>;
"ARRAY" IN[0 : 8];
ENTRY:
IN[0]: THE MACHINE PRECISION;
FOR THE CYBER 73-26 A SUITABLE VALUE IS "-14;
IN[1]: THE RELATIVE TOLERANCE FOR THE SOLUTION;
THIS TOLERANCE SHOULD NOT BE CHOSEN SMALLER THAN IN[0];
IN[2]: THE ASSOLUTE TOLERANCE FOR THE SOLUTION;
IN[3]; A PARAMETER USED FOR CONTROLLING LINEMINIMIZATION, ([3], [4]); USUALLY A SUITABLE VALUE IS: 0.0001;
IN[4]: THE ABSOLUTE TOLERANCE FOR THE EUCLIDIAN NORM OF THE GRADIENT AT THE SOLUTION;
IN[5]; A LOWERBOUND FOR THE FUNCTIONVALUE;
IN[6]; THIS PARAMETER CONTROLS THE INITIALIZATION OF THE APPROXIMATION OF THE INVERSE HESSIAN (METRIC), SEE H; USUALLY THE CHOICE IN[6] = 1 WILL GIVE GOOD RESULTS;
IN[7]; THE MAXIMUM ALLOWED NUMBER OF CALLS OF FUNCT;
IN[8]: A PARAMETER USED FOR CONTROLLING THE UPDATING OF THE METRIC; IT IS USED TO AVOID UNBOUNDEDNESS OF THE METRIC (SEE: [6], FORMULA (19));
THE VALUE OF IN[8] SHOULD SATISFY:
SQRT(IN[0] / IN[1]) / N < IN[8] < 1;
USUALLY A SUITABLE VALUE WILL BE 0.01;

OUT: <ARRAY IDENTIFIER>;
"ARRAY" OUT[0:4];
EXIT:
OUT[0]: THE EUCLIDIAN NORM OF THE PRODUCT OF THE METRIC AND THE GRADIENT AT THE CALCULATED MINIMUM;
OUT[1]: THE EUCLIDIAN NORM OF THE GRADIENT AT THE CALCULATED MINIMUM;
OUT[2]: THE NUMBER OF CALLS OF FUNCT, NECESSARY TO ATTAIN THIS RESULT;
OUT[3]: THE NUMBER OF ITERATIONS IN WHICH A LINESEARCH WAS NECESSARY;
OUT[4]: THE NUMBER OF ITERATIONS IN WHICH A DIRECTION HAD TO BE CALCULATED WITH THE METHOD GIVEN IN [5]; IN SUCH AN ITERATION A CALCULATION OF THE EIGENVALUES AND EIGENVECTORS OF THE METRIC IS NECESSARY.

DATA AND RESULTS:

USUALLY THE CALCULATED SOLUTION WILL SATISFY:
NORM (XMIN - XCAL) < NORM (XCAL) * IN[1] + IN[2].
WHERE XMIN IS A MINIMUM OF THE GIVEN FUNCTION, XCAL THE CALCULATED APPROXIMATION OF XMIN AND NORM(.) DENOTES THE EUCLIDIAN NORM OF X; HOWEVER, WE CANNOT GUARANTEE SUCH A RESULT; THE CALCULATED SOLUTION POSSIBLY WILL NOT SATISFY THE ABOVE INEQUALITY IF THE PROBLEM IS VERY ILL - CONDITIONED; THE USER CAN DISCOVER SUCH A SITUATION BY LOOKING AT THE EUCLIDIAN NORM OF THE METRIC, DELIVERED IN H; THE PROBLEM IS ILL - CONDITIONED IF THIS NORM IS LARGE RELATIVE TO 1.

PROCEDURES USED:

VECVEC = CP34010,
MATVEC = CP34011,
TAMVEC = CP34012,
SYMMATVEC = CP34018,
INIVEC = CP31010,.
INISYMD = CP31013,
MULVEC = CP31020,
DUPVEC = CP31030,
EIGSYM1 = CP34156,
LINEMIN = CP34210,
RNK1UPD = CP34211,
DAVUPD = CP34212,
FLEUPD = CP34213.

REQUIRED CENTRAL MEMORY:

EXECUTION FIELD LENGTH: VARIES FROM 5 * N + 26 TO
N ** 2 + N * (N + 1) // 2 + 5 * N + 35 WORDS.

RUNNING TIME:

DEPENDS STRONGLY ON THE PROBLEM TO BE SOLVED;

LANGUAGE: ALGOL 60.

METHOD AND PERFORMANCE:

RNK1MIN CALCULATES AN APPROXIMATION OF A MINIMUM OF A GIVEN FUNCTION BY MEANS OF A VARIABLE METRIC METHOD; THE RANK - ONE UPDATING FORMULA, USED IN THIS ALGORITHM IS GIVEN IN [6], (FORMULA (4)); TO AVOID UNBOUNDEDNESS OF THE METRIC (SEE [8]), SOMETIMES A RANK - TWO UPDATING FORMULA IS USED ([3], FORMULAS (1) AND (5)); TO AVOID LINESEARCHES AS MUCH AS POSSIBLE A STRATEGY GIVEN IN [4] IS USED; IF IN AN ITERATION THE FUNCTION IS INCREASING IN THE DIRECTION GIVEN BY THE VARIABLE METRIC ALGORITHM, BECAUSE THE METRIC IS NOT POSITIVE DEFINITE, THEN A METHOD GIVEN IN [5] IS USED TO CALCULATE A NEW DIRECTION; THIS METHOD REQUIRES THE CALCULATION OF THE EIGENVECTORS AND EIGENVALUES OF THE METRIC; USUALLY,THE NUMBER OF TIMES SUCH A CALCULATION IS NECESSARY IS VERY SMALL RELATIVE TO THE NUMBER OF ITERATIONS (AND OFTEN EQUALS ZERO); IF THE NUMBER OF VARIABLES OF THE FUNCTION IS VERY LARGE AND THE CALCULATION OF THE FUNCTION AND ITS GRADIENT IS RELATIVELY CHEAP (THE NUMBER OF ARITHMETICAL OPERATIONS IS OF ORDER AT MOST N ** 2), THEN THE USER IS ADVISED TO USE FLEMIN (CP32105);
A DETAILED DESCRIPTION OF THE ALGORITHM AND SOME RESULTS ABOUT ITS CONVERGENCE IS GIVEN IN [1].

SUBSECTION: FLEMIN.

CALLING SEQUENCE:

THE HEADING OF THIS PROCEDURE IS:
"REAL" "PROCEDURE" FLEMIN(N, X, G, H, FUNCT, IN, OUT);
"VALUE" N; "INTEGER" N;
"ARRAY" X, G, H, IN, OUT; "REAL" "PROCEDURE" FUNCT;

FLEMIN: DELIVERS THE CALCULATED LEAST VALUE OF THE GIVEN FUNCTION;

THE MEANING OF THE FORMAL PARAMETERS IS:
N: <ARITHMETICAL EXPRESSION>;
THE NUMBER OF VARIABLES OF THE FUNCTION TO BE MINIMIZED;
X: <ARRAY IDENTIFIER>;
"ARRAY" X[1 : N];
THE INDEPENDENT VARIABLES;
ENTRY: AN APPROXIMATION OF A MINIMUM OF THE FUNCTION;
EXIT: THE CALCULATED MINIMUM OF THE FUNCTION;
G: <ARRAY IDENTIFIER>;
"ARRAY" G[1 : N];
EXIT: THE GRADIENT OF THE FUNCTION AT THE CALCULATED MINIMUM;
H: <ARRAY IDENTIFIER>;
"ARRAY" H[1 : N * (N + 1) // 2];
THE UPPERTRIANGLE OF AN APPROXIMATION OF THE INVERSE HESSIAN IS STORED COLUMNWISE IN H;
IF IN[6] > 0 INITIALIZING OF H WILL BE DONE AUTOMATICALLY AND THE INITIAL APPROXIMATION OF THE INVERSE HESSIAN WILL EQUAL THE UNITMATRIX MULTIPLIED WITH THE VALUE OF IN[6]; IF IN[6] < 0 NO INITIALIZING OF H WILL BE DONE AND THE USER SHOULD GIVE IN H AN APPROXIMATION OF THE INVERSE HESSIAN AT THE STARTING POINT;
THE UPPERTRIANGLE OF AN APPROXIMATION OF THE INVERSE HESSIAN AT THE CALCULATED MINIMUM IS DELIVERED IN H;
FUNCT: <PROCEDURE IDENTIFIER>;
THE HEADING OF THIS PROCEDURE MUST BE:
"REAL" "PROCEDURE" FUNCT(N, X, G); "VALUE" N;
"INTEGER" N; "ARRAY" X, G;
A CALL OF FUNCT MUST EFFECTUATE IN:
1: FUNCT BECOMES THE VALUE OF THE FUNCTION TO BE MINIMIZED AT THE POINT X;
2: THE VALUE OF G[I], (I = 1, ..., N), BECOMES THE VALUE OF THE I - TH COMPONENT OF THE GRADIENT OF THE FUNCTION AT X;

IN: <ARRAY IDENTIFIER>;
"ARRAY" IN[1 : 7];
ENTRY:
IN[1]: THE RELATIVE TOLERANCE FOR THE SOLUTION;
IN[2]: THE ABSOLUTE TOLERANCE FOR THE SOLUTION;
IN[3]: A PARAMETER USED FOR CONTROLLING LINEMINIMIZATION ([3], [4]); USUALLY A SUITABLE VALUE IS 0.0001;
IN[4]: THE ABSOLUTE TOLERANCE FOR THE EUCLIDIAN NORM OF THE GRADIENT AT THE SOLUTION;
IN[5]: A LOWERBOUND FOR THE FUNCTION VALUE;
IN[6]: THIS PARAMETER CONTROLS THE INITIALIZATION OF THE APPROXIMATION OF THE INVERSE HESSIAN (METRIC) (SEE H); USUALLY IN[6] = 1 WILL GIVE GOOD RESULTS;
IN[7]: THE MAXIMUM ALLOWED NUMBER OF CALLS OF FUNCT;

OUT: <ARRAY IDENTIFIER>;
"ARRAY" OUT[0:4];
EXIT:
OUT[0]: THE EUCLIDIAN NORM OF THE PRODUCT OF THE METRIC AND THE GRADIENT AT THE CALCULATED MINIMUM;
OUT[1]: THE EUCLIDIAN NORM OF THE GRADIENT AT THE CALCULATED MINIMUM;
OUT[2]: THE NUMBER OF CALLS OF FUNCT, NECESSARY TO ATTAIN THESE RESULTS;
OUT[3]: THE NUMBER OF ITERATIONS IN WHICH A LINESEARCH WAS NECESSARY;
OUT[4]: IF OUT[4] = - 1, THEN THE PROCESS IS BROKEN OFF BECAUSE NO DOWNHILL DIRECTION COULD BE CALCULATED; THE PRECISION ASKED FOR MAY NOT BE ATTAINED AND IS POSSIBLY CHOSEN TOO HIGH;
NORMALLY OUT[4] = 0;

DATA AND RESULTS:

USUALLY THE CALCULATED SOLUTION WILL SATISFY:
NORM (XMIN - XCAL) < NORM (XCAL) * IN[1] + IN[2],
WHERE XMIN IS A MINIMUM OF THE GIVEN FUNCTION, XCAL THE CALCULATED APPROXIMATION OF XMIN AND NORM (.) DENOTES THE EUCLIDIAN NORM OF X; HOWEVER, WE CAN NOT GUARANTEE SUCH A RESULT; THE CALCULATED SOLUTION POSSIBLY WILL NOT SATISFY THE ABOVE INEQUALITY IF THE PROBLEM IS VERY ILL - CONDITIONED; THE USER CAN DISCOVER SUCH A SITUATION BY LOOKING AT THE EUCLIDIAN NORM OF THE METRIC, DELIVERED IN H; THE PROBLEM IS ILL - CONDITIONED IF THIS NORM IS LARGE RELATIVE TO 1.

PROCEDURES USED:

VECVEC = CP34010,
ELMVEC = CP34020,
SYMMATVEC = CP34018,
INIVEC = CP31010,
INISYMD = CP31013,
MULVEC = CP31020,
DUPVEC = CP31030,
LINEMIN = CP34210,
DAVUPD = CP34212,
FLEUPD = CP34213.

REQUIRED CENTRAL MEMORY:

EXECUTION FIELD LENGTH: 3 * N + 19 WORDS.

RUNNING TIME:

DEPENDS STRONGLY ON THE PROBLEM TO BE SOLVED;

LANGUAGE: ALGOL 60.

METHOD AND PERFORMANCE:

FLEMIN CALCULATES AN APPROXIMATION OF A MINIMUM OF A GIVEN FUNCTION BY MEANS OF THE VARIABLE METRIC ALGORITHM GIVEN IN [3], EXCEPT FOR SOME DETAILS (SEE [1]).

REFERENCES:

[1] BUS, J. C. P.
MINIMIZATION OF FUNCTIONS OF SEVERAL VARIABLES (DUTCH).
MATHEMATICAL CENTRE, AMSTERDAM, NR 29/72 (1972).
[2] DAVIDON, W. C.
VARIABLE METRIC METHOD FOR MINIMIZATION.
ARGONNE NAT. LAB. REPORT, ANL 5990 (1959).
[3] FLETCHER, R.
A NEW APPROACH TO VARIABLE METRIC ALGORITHMS.
COMP. J. 6, (1963), P.163 - 168.
[4] GOLDSTEIN, A. A. AND PRICE, J. F.
AN EFFECTIVE ALGORITHM FOR MINIMIZATION.
NUMER. MATH. 10, (1967), P.184 - 189.
[5] GREENSTADT, J. L.
ON THE RELATIVE EFFICIENCIES OF GRADIENT METHODS.
MATH. COMP. 21, (1967), P.360 - 367.
[6] POWELL, M. J. D.
RANK ONE METHODS FOR UNCONSTRAINED OPTIMIZATION.
IN: ABADIE, J. (ED.)
INTEGER AND NONLINEAR PROGRAMMING.
NORTH - HOLLAND, (1970).

EXAMPLE OF USE:

THE MINIMUM OF THE FUNCTION:
F(X) = (X[2] - X[1] ** 2) ** 2 * 100 + (1 - X[1]) ** 2,
CALCULATED WITH BOTH RNK1MIN AND FLEMIN MAY BE OBTAINED BY THE FOLLOWING PROGRAM:

```
"BEGIN"
    "REAL" "PROCEDURE" RNK1MIN(N, X, G, H, FUNCT, IN, OUT);
    "CODE" 34214;
    "REAL" "PROCEDURE" FLEMIN(N, X, G, H, FUNCT, IN, OUT);
    "CODE" 34215;
    "REAL" "PROCEDURE" ROSENBROCK(N, X, G); "VALUE" N;
    "INTEGER" N; "ARRAY" X, G;
    "BEGIN" ROSENBROCK:= (X[2] - X[1] ** 2) ** 2 * 100
        + (1 - X[1]) ** 2;
        G[1]:= ((X[1] ** 2 - X[2]) * 400 + 2) * X[1] - 2;
        G[2]:=(X[2] - X[1] ** 2) * 200
    "END" ROSENBROCK;
    "INTEGER" I; "BOOLEAN" AGAIN; "REAL" F;
    "ARRAY" X, G[1:2], H[1:3], IN[0:8], OUT[0:4];
    IN[0]:= "-14; IN[1]:= "-5; IN[2]:= "-5; IN[3]:= "-4;
    IN[4]:= "-5; IN[5]:= -10; IN[6]:= 1; IN[7]:= 100; IN[8]:= 0.01;
    X[1]:= -1.2; X[2]:= 1; AGAIN:= "TRUE";
    F:= RNK1MIN(2, X, G, H, ROSENBROCK, IN, OUT);
    "GOTO" PRINT;
NEXT: X[1]:= -1.2; X[2]:= 1; AGAIN:= "FALSE";
    F:= FLEMIN(2, X, G, H, ROSENBROCK, IN, OUT);
PRINT: OUTPUT(61, "(""("LEAST VALUE:")"B+.15D"+3D,//, "("X:")",
    2(B+.15D"+3DB),//,"("GRADIENT:")", 2(B+.15D"+3DB),//, "("METRIC:")"
    ,2(B+.15D"+3DB),/,32B+.15D"+3D,//,"("OUT:")", 5(B+.15D"+3DB,/),///
    ")",F, X[1], X[2], G[1], G[2], H[1], H[2], H[3], OUT[0], OUT[1],
    OUT[2], OUT[3], OUT[4]);
    "IF" AGAIN "THEN" "GOTO" NEXT
"END"
        "EOP"
```

DELIVERS:

```
LEAST VALUE: +.2006997988011 80"-018

X: +.9999999994484 0"+000  +.9999999998452 20"+000

GRADIENT: +.1767313051459 50"-007  -.8891731795301 90"-008

METRIC: +.4999824148632 50"+000  +.9999573838102 30"+000
                                 +.2004897576792 90"+001

OUT: +.1641571237746 60"-009
     +.1978389336064 80"-007
     +.5500000000000 00"+002
     +.8000000000000 00"+001
     +.4000000000000 00"+001

LEAST VALUE: +.8119734999212 90"-016

X: +.9999999997587 70"+000  +.9999999986167 80"+000

GRADIENT: +.3598266573590 10"-006  -.1801545579382 90"-006

METRIC: +.5010853569755 50"+000  +.1001981391996 00"+001
                                 +.2008616555435 10"+001

OUT: +.1338022893878 30"-008
     +.4024063718333 70"-006
     +.4400000000000 00"+002
     +.7000000000000 00"+001
     +.0000000000000 00"+000
```

SOURCE TEXT(S):

```
"CODE" 34210;
    "PROCEDURE" LINEMIN(N, X, D, ND, ALFA, G, FUNCT, F0, F1, DF0, DF1,
    EVLMAX, STRONGSEARCH, IN); "VALUE" N, ND, F0, DF0, STRONGSEARCH;
    "INTEGER" N, EVLMAX; "BOOLEAN" STRONGSEARCH;
    "REAL" ND, ALFA, F0, F1, DF0, DF1;
    "ARRAY" X, D, G, IN;
    "REAL" "PROCEDURE" FUNCT;
    "BEGIN" "INTEGER" I, EVL;
        "BOOLEAN" NOTININT;
        "REAL" F,OLDF,DF,OLDDF,MU,ALFA0,Q,W,Y,Z,RELTOL,ABSTOL
        ,EPS, AID;
        "ARRAY" X0[1:N];
        "REAL" "PROCEDURE" VECVEC(L, U, SHIFT, A, B); "CODE" 34010;
        "PROCEDURE" ELMVEC(L, U, SHIFT, A, B, X); "CODE" 34020;
        "PROCEDURE" DUPVEC(L, U, SHIFT, A, B); "CODE" 31030;

        RELTOL:= IN[1]; ABSTOL:= IN[2]; MU:= IN[3]; EVL:= 0;
        ALFA0:= 0; OLDF:= F0; OLDDF:= DF0; Y:= ALFA; NOTININT:= "TRUE";
        DUPVEC(1, N, 0, X0, X);
        EPS:= (SQRT(VECVEC(1, N, 0, X, X)) * RELTOL + ABSTOL) / ND;
        Q:= (F1 - F0) / (ALFA * DF0);
INT:    "IF" NOTININT "THEN" NOTININT:= DF1 < 0 "AND" Q > MU;
        AID:= ALFA; "IF" DF1 >= 0 "THEN"
        "BEGIN" Z:= 3 * (OLDF - F1) / ALFA + OLDDF + DF1;
            W:= SQRT(Z ** 2 - OLDDF * DF1);
            ALFA:= ALFA * (1 - (DF1 + W - Z) / (DF1 - OLDDF + W * 2));
            "IF" ALFA < EPS "THEN" ALFA:= EPS "ELSE"
            "IF" AID - ALFA < EPS "THEN" ALFA:= AID - EPS
        "END" CUBIC INTERPOLATION
        "ELSE" "IF" NOTININT "THEN"
        "BEGIN" ALFA0:= ALFA:= Y; OLDDF:= DF1; OLDF:= F1 "END"
        "ELSE" ALFA:= 0.5 * ALFA; Y:= ALFA + ALFA0;
        DUPVEC(1, N, 0, X, X0); ELMVEC(1, N, 0, X, D, Y);
        EPS:= (SQRT(VECVEC(1, N, 0, X, X)) * RELTOL + ABSTOL) / ND;
        F:= FUNCT(N, X, G); EVL:= EVL +1 ; DF:= VECVEC(1, N, 0, D, G);
        Q:= (F - F0) / (Y * DF0);
        "IF" ("IF" NOTININT "OR" STRONGSEARCH "THEN" "TRUE" "ELSE"
        Q < MU "OR" Q > 1 - MU) "AND" EVL < EVLMAX "THEN"
        "BEGIN" "IF" NOTININT "OR" DF > 0 "OR" Q < MU "THEN"
            "BEGIN" DF1:= DF; F1:= F "END"
            "ELSE"
            "BEGIN" ALFA0:= Y; ALFA:= AID - ALFA; OLDDF:= DF; OLDF:= F
            "END";
            "IF" ALFA > EPS * 2 "THEN" "GOTO" INT
    "END";
    ALFA:= Y; EVLMAX:= EVL; DF1:= DF; F1:= F
```

```
"CODE" 34211;
    "PROCEDURE" RNK1UPD(H, N, V, C);"VALUE" N, C; "INTEGER" N;
    "REAL" C;"ARRAY" H, V;
     "BEGIN" "INTEGER" J, K;
          "PROCEDURE" ELMVEC(L, U, SHIFT, A, B, X); "CODE" 34020;
          K:= 0;
          "FOR" J:= 1, J + K "WHILE" K < N "DO"
          "BEGIN" K:= K +1 ;
              ELMVEC(J, J + K - 1, 1 - J, H, V, V[K] * C)
          "END"
     "END" RNK1UPD;
          "EOP"

"CODE" 34212;
     "PROCEDURE" DAVUPD(H, N, V, W, C1, C2); "VALUE" N, C1, C2;
     "INTEGER" N; "REAL" C1, C2; "ARRAY" H, V, W;
     "BEGIN" "INTEGER" I, J, K;
          "REAL" VK, WK;
          K:= 0;
          "FOR" J:= 1, J + K "WHILE" K < N "DO"
          "BEGIN" K:= K +1 ; VK:= V[K] * C1; WK:= W[K] * C2;
              "FOR" I:= 0 "STEP" 1 "UNTIL" K -1  "DO"
              H[I + J]:= H[I + J] + V[I + 1] * VK - W[I + 1] * WK
          "END"
     "END" DAVUPD;
          "EOP"

"CODE" 34213;
     "PROCEDURE" FLEUPD(H, N, V, W, C1, C2); "VALUE" N, C1, C2;
     "INTEGER" N; "REAL" C1, C2; "ARRAY" H, V, W;
     "BEGIN" "INTEGER" I, J, K;
          "REAL" VK, WK;
          K:= 0; "FOR" J:= 1, J + K "WHILE" K < N "DO"
          "BEGIN" K:= K +1; VK:= - W[K] * C1 + V[K] * C2; WK:= V[K] * C1;
              "FOR" I:= 0 "STEP" 1 "UNTIL" K - 1  "DO"
              H[I + J]:= H[I + J] + V[I + 1] * VK -W[I + 1] * WK
     "END"
"END" FLEUPD;
     "EOP"
```

```
"CODE" 34214;
     "REAL" "PROCEDURE" RNK1MIN(N, X, G, H, FUNCT, IN, OUT);
     "VALUE" N;
     "INTEGER" N; "ARRAY" X, G, H, IN, OUT;
     "REAL" "PROCEDURE" FUNCT;
"BEGIN" "INTEGER" I, IT, N2, CNTL, CNTE, EVL, EVLMAX;
     "BOOLEAN" OK;
     "REAL" F, F0, FMIN, MU, DG, DG0, GHG, GS, NRMDELTA, ALFA,
     MACHEPS, RELTOL, ABSTOL, EPS, TOLG, ORTH, AID;
     "ARRAY" V, DELTA, GAMMA, S, P[1:N];
     "REAL" "PROCEDURE" VECVEC(L, U, SHIFT, A, B); "CODE" 34010;
     "REAL" "PROCEDURE" MATVEC(L, U, I, A, B); "CODE" 34011;
     "REAL" "PROCEDURE" TAMVEC(L, U, I, A, B); "CODE" 34012;
     "PROCEDURE" ELMVEC(L, U, SHIFT, A, B, X); "CODE" 34020;
     "REAL" "PROCEDURE" SYMMATVEC(L, U, I, A, B); "CODE" 34018;
     "PROCEDURE" INIVEC(L, U, A, X); "CODE" 31010;
     "PROCEDURE" INISYMD(LR, UR, SHIFT, A, X); "CODE" 31013;
     "PROCEDURE" MULVEC(L, U, SHIFT, A, B, X); "CODE" 31020;
     "PROCEDURE" DUPVEC(L, U, SHIFT, A, B); "CODE" 31030;
     "PROCEDURE" EIGSYM1(A, N, NUMVAL, VAL, VEC, EM); "CODE" 34156;
     "PROCEDURE" LINEMIN(N, X, D, ND, A, G, F, F0, F1, DF0, DF1,
     E, S, IN); "CODE" 34210;
     "PROCEDURE" RNK1UPD(H, N, V, C); "CODE" 34211;
     "PROCEDURE" DAVUPD(H, N, V, W, C1, C2); "CODE" 34212;
     "PROCEDURE" FLEUPD(H, N, V, W, C1, C2); "CODE" 34213;
     MACHEPS:= IN[0]; RELTOL:= IN[1]; ABSTOL:= IN[2];
     MU:= IN[3]; TOLG:= IN[4]; FMIN:= IN[5];
     ALFA:= IN[6]; EVLMAX:= IN[7]; ORTH:= IN[8];
     N2:= N * (N + 1) // 2; CNTL:= CNTE:= 0; "IF" ALFA > 0 "THEN"
     "BEGIN" INIVEC(1, N2, H, 0); INISYMD(1, N, 0, H, ALFA) "END";
     F:= FUNCT(N, X, G); EVL:= 1; DG:= SQRT(VECVEC(1, N, 0, G, G));
     "FOR" I:= 1 "STEP" 1 "UNTIL" N "DO"
     DELTA[I]:= - SYMMATVEC(1, N, I, H, G);
     NRMDELTA:= SQRT(VECVEC(1, N, 0, DELTA, DELTA));
     DG0:= VECVEC(1, N, 0, DELTA, G); OK:= DG0 <= 0;
     EPS:= SQRT(VECVEC(1, N, 0, X, X)) * RELTOL + ABSTOL;
     "FOR" IT:= 1, IT +1  "WHILE"
     (NRMDELTA > EPS "OR" DG > TOLG "OR" ~ OK) "AND" EVL < EVLMAX
     "DO"
     "BEGIN" "IF" ~OK "THEN"
          "BEGIN" "ARRAY" VEC[1:N,1:N], TH[1:N2], EM[0:9];
               EM[0]:= MACHEPS; EM[2]:= AID:= SQRT(MACHEPS * RELTOL);
               EM[4]:= ORTH; EM[6]:= AID * N; EM[8]:= 5;
               CNTE:= CNTE + 1; DUPVEC(1, N2, 0, TH, H);
               EIGSYM1(TH,N,N,V,VEC,EM);
               "FOR" I:= 1 "STEP" 1 "UNTIL" N "DO"
               "BEGIN" AID:= - TAMVEC(1, N, I, VEC, G);
                    S[I]:= AID * ABS(V[I]); V[I]:= AID * SIGN(V[I])
               "END"
```

```
"FOR" I:= 1 "STEP" 1 "UNTIL" N "DO"
"BEGIN" DELTA[I]:= MATVEC(1, N, I, VEC, S):
    P[I]:= MATVEC(1, N, I, VEC, V)
"END";
    DG0:= VECVEC(1, N, 0, DELTA, G);
    NRMDELTA:= SQRT(VECVEC(1, N, 0, DELTA, DELTA))
"END" CALCULATING GREENSTADTS DIRECTION;
DUPVEC(1, N, 0, S, X); DUPVEC(1, N, 0, V, G);
"IF" IT > N "THEN" ALFA:= 1 "ELSE"
"BEGIN" "IF" IT ¬= 1 "THEN" ALFA:= ALFA / NRMDELTA "ELSE"
    "BEGIN" ALFA:= 2 * (FMIN - F) / DG0;
    "IF" ALFA > 1 "THEN" ALFA:= 1
    "END"
"END";
ELMVEC(1, N, 0, X, DELTA, ALFA);
F0:= F; F:= FUNCT(N, X, G); EVL:= EVL +1 ;
DG:= VECVEC(1, N, 0, DELTA, G);
"IF" IT = 1 "OR" F0 - F < -MU * DG0 * ALFA "THEN"
"BEGIN" I:= EVLMAX - EVL; CNTL:= CNTL +1 ;
    LINEMIN(N, S, DELTA, NRMDELTA, ALFA, G, FUNCT, F0, F,
    DG0, DG, I, "FALSE", IN); EVL:= EVL + I;
    DUPVEC(1, N, 0, X, S);
"END" LINEMINIMIZATION;
DUPVEC(1, N, 0, GAMMA, G); ELMVEC(1, N, 0, GAMMA, V, -1);
"IF" ¬ OK "THEN" MULVEC(1, N, 0, V, P, -1);
DG:= DG - DG0; "IF" ALFA ¬= 1 "THEN"
"BEGIN" MULVEC(1, N, 0, DELTA, DELTA, ALFA);
    MULVEC(1, N, 0, V, V, ALFA);
    NRMDELTA:= NRMDELTA * ALFA; DG:= DG * ALFA
"END";
DUPVEC(1, N, 0, P, GAMMA); ELMVEC(1, N, 0, P, V, 1);
"FOR" I:= 1 "STEP" 1 "UNTIL" N "DO"
V[I]:= SYMMATVEC(1, N, I, H, GAMMA);
DUPVEC(1, N, 0, S, DELTA); ELMVEC(1, N, 0, S, V, -1);
GS:= VECVEC(1, N, 0, GAMMA, S);
GHG:= VECVEC(1, N, 0, V, GAMMA);
AID:= DG / GS;
"IF" VECVEC(1, N, 0, DELTA, P) ** 2 > VECVEC(1, N, 0, P, P)
* (ORTH * NRMDELTA) ** 2 "THEN" RNK1UPD(H, N, S, 1 / GS)
"ELSE" "IF" AID >= 0 "THEN"
        FLEUPD(H, N, DELTA, V, 1 / DG, (1 + GHG / DG) / DG) "ELSE"
        DAVUPD(H, N, DELTA, V, 1 / DG, 1 / GHG);
        "FOR" I:= 1 "STEP" 1 "UNTIL" N "DO"
        DELTA[I]:= -SYMMATVEC(1, N, I, H, G);
        ALFA:= NRMDELTA;
        NRMDELTA:= SQRT(VECVEC(1, N, 0, DELTA, DELTA));
        EPS:= SQRT(VECVEC(1, N, 0, X, X)) * RELTOL + ABSTOL;
        DG:= SQRT(VECVEC(1, N, 0, G, G));
        DG0:= VECVEC(1, N, 0, DELTA, G); OK:= DG0 <= 0
    "END" ITERATION;
    OUT[0]:= NRMDELTA; OUT[1]:= DG; OUT[2]:= EVL;
    OUT[3]:= CNTL; OUT[4]:= CNTE; RNK1MIN:= F
"END" RNK1MIN;
    "EOP"
```

```
"CODE" 34215;
     "REAL" "PROCEDURE" FLEMIN(N, X, G, H, FUNCT, IN, OUT);
     "VALUE" N;
"INTEGER" N; "ARRAY" X, G, H, IN, OUT;
"REAL" "PROCEDURE" FUNCT;
"BEGIN" "INTEGER" I, IT, CNTL, EVL, EVLMAX;
     "REAL" F,F0,FMIN,MU,DG,DG0,NRMDELTA,ALFA,RELTOL,ABSTOL,
     EPS, TOLG, AID;
     "ARRAY" V, DELTA, S[1:N];
     "REAL" "PROCEDURE" VECVEC(L, U. SHIFT, A, B); "CODE" 34010;
     "PROCEDURE" ELMVEC(L, U, SHIFT, A, B, X); "CODE" 34020;
     "REAL" "PROCEDURE" SYMMATVEC(L, U, I, A, B); "CODE" 34018;
     "PROCEDURE" INIVEC(L, U, A, X); "CODE" 31010;
     "PROCEDURE" INISYMD(LR, UR, SHIFT, A, X); "CODE" 31013;
     "PROCEDURE" MULVEC(L, U, SHIFT, A, B, XB); "CODE" 31020;
     "PROCEDURE" DUPVEC(L, U, SHIFT, A, B); "CODE" 31030;
     "PROCEDURE" LINEMIN(N, X, D, ND, A, G, F, F0, F1, DF0, DF1,
     E, S, IN); "CODE" 34210;
     "PROCEDURE" DAVUPD(H, N, V, W, C1, C2); "CODE" 34212;
     "PROCEDURE" FLEUPD(H, N, V, W, C1, C2); "CODE" 34213;

RELTOL:= IN[1]; ABSTOL:= IN[2]; MU:= IN[3];
TOLG:= IN[4]; FMIN:= IN[5]; ALFA:= IN[6];
EVLMAX:= IN[7]; OUT[4]:= 0;
F:= FUNCT(N, X, G); EVL:= 1; CNTL:= 0;"IF" ALFA > 0 "THEN"
"BEGIN" INIVEC(1, N * (N + 1) // 2, H, 0);
     INISYMD(1, N, 0, H, ALFA)
"END";
"FOR" I:= 1 "STEP" 1 "UNTIL" N "DO"
DELTA[I]:= - SYMMATVEC(1, N, I, H, G);
DG:= SQRT(VECVEC(1, N, 0, G, G));
NRMDELTA:= SQRT(VECVEC(1, N, 0, DELTA, DELTA));
EPS:= SQRT(VECVEC(1, N, 0, X, X)) * RELTOL + ABSTOL;
DG0:= VECVEC(1, N, 0, DELTA, G);"COMMENT"
```

```
        "FOR" IT:= 1, IT +1   "WHILE"
        (NRMDELTA > EPS "OR" DG > TOLG ) "AND" EVL < EVLMAX "DO"
    "BEGIN" DUPVEC(1, N, 0, S, X); DUPVEC(1, N, 0, V, G);
        "IF" IT >= N "THEN" ALFA:= 1 "ELSE"
        "BEGIN" "IF" IT ^= 1 "THEN" ALFA:= ALFA / NRMDELTA "ELSE"
            "BEGIN" ALFA:= 2 * (FMIN - F) / DG0;
                "IF" ALFA > 1 "THEN" ALFA:= 1
            "END"
        "END";
        ELMVEC(1, N, 0, X, DELTA, ALFA);
        F0:= F; F:= FUNCT(N, X, G); EVL:= EVL +1 ;
        DG:= VECVEC(1, N, 0, DELTA, G);
        "IF" IT = 1 "OR" F0 - F <= - MU * DG0 * ALFA "THEN"
        "BEGIN" I:= EVLMAX - EVL; CNTL:= CNTL +1 ;
            LINEMIN(N, S, DELTA, NRMDELTA, ALFA, G, FUNCT, F0, F,
            DG0, DG, I, "FALSE", IN); EVL:= EVL + I;
            DUPVEC(1, N, 0, X, S);
        "END" LINEMINIMIZATION;
        "IF" ALFA ^= 1 "THEN" MULVEC(1, N, 0, DELTA, DELTA, ALFA);
        MULVEC(1, N, 0, V, V, -1); ELMVEC(1, N, 0, V, G, 1);
            "FOR" I:= 1 "STEP" 1 "UNTIL" N "DO"
            S[I]:= SYMMATVEC(1, N, I, H, V);
            AID:= VECVEC(1, N, 0, V, S); DG:= (DG - DG0) * ALFA;
            "IF" DG > 0 "THEN"
            "BEGIN" "IF" DG >= AID "THEN"
                FLEUPD(H, N, DELTA, S, 1 / DG, (1 + AID / DG) / DG)
                "ELSE" DAVUPD(H, N, DELTA, S, 1 / DG, 1 / AID)
            "END" UPDATING;
            "FOR" I:= 1 "STEP" 1 "UNTIL" N "DO"
            DELTA[I]:= -SYMMATVEC(1, N, I, H, G);
            ALFA:= NRMDELTA * ALFA;
            NRMDELTA:= SQRT(VECVEC(1, N, 0, DELTA, DELTA));
            EPS:= SQRT(VECVEC(1, N, 0, X, X)) * RELTOL + ABSTOL;
            DG:= SQRT(VECVEC(1, N, 0, G, G));
            DG0:= VECVEC(1, N, 0, DELTA, G); "IF" DG0 > 0 "THEN"
            "BEGIN" OUT[4]:= -1 ; "GOTO" EXIT "END"
    "END" ITERATION;
EXIT: OUT[0]:= NRMDELTA; OUT[1]:= DG; OUT[2]:= EVL;
    OUT[3]:= CNTL; FLEMIN:= F
"END" FLEMIN;
    "EOP"
```

AUTHOR: J. C. P. BUS.

INSTITUTE: MATHEMATICAL CENTRE.

RECEIVED: 741101.

BRIEF DESCRIPTION:

THIS SECTION CONTAINS THE PROCEDURE MININ, FOR MINIMIZING A FUNCTION OF ONE VARIABLE IN A GIVEN INTERVAL;

KEYWORDS:

MINIMIZATION,
FUNCTIONS OF ONE VARIABLE.

CALLING SEQUENCE:

THE HEADING OF THIS PROCEDURE IS:
"REAL" "PROCEDURE" MININ(X, A, B, FX, TOLX);
"REAL" X, A, B, FX, TOLX;

MININ DELIVERS THE CALCULATED MINIMUM VALUE OF THE FUNCTION, DEFINED BY FX, ON THE INTERVAL WITH END POINTS A AND B.

THE MEANING OF THE FORMAL PARAMETERS IS:

X: <REAL VARIABLE>;
A JENSEN VARIABLE; THE ACTUAL PARAMETERS FOR FX AND TOLX DEPEND ON X;
EXIT: THE CALCULATED APPROXIMATION OF THE POSITION OF THE MINIMUM;

A, B: <REAL VARIABLE>;
ENTRY: THE END POINTS OF THE INTERVAL ON WHICH A MINIMUM IS SEARCHED FOR;
EXIT: THE END POINTS OF AN INTERVAL WITH LENGTH LESS THAN 4 * TOL(X) SUCH THAT A < X < B;

FX: <ARITHMETIC EXPRESSION>;
THE FUNCTION IS GIVEN BY THE ACTUAL PARAMETER FX, WHICH DEPENDS ON X;

TOLX: <ARITHMETIC EXPRESSION>;
THE TOLERANCE IS GIVEN BY THE ACTUAL PARAMETER TOLX, WHICH MAY DEPEND ON X; A SUITABLE TOLERANCE FUNCTION IS: ABS(X) * RE + AE, WHERE RE IS THE RELATIVE PRECISION DESIRED AND AE IS AN ABSOLUTE PRECISION WHICH SHOULD NOT BE CHOSEN EQUAL TO ZERO.

DATA AND RESULTS:
THE USER SHOULD BE AWARE OF THE FACT THAT THE CHOICE OF TOLX MAY HIGHLY AFFECT THE BEHAVIOUR OF THE ALGORITHM, ALTHOUGH CONVERGENCE TO A POINT FOR WHICH THE GIVEN FUNCTION IS MINIMAL ON THE GIVEN INTERVAL IS ASSURED; THE ASYMPTOTIC BEHAVIOUR WILL USUALLY BE FINE AS LONG AS THE NUMERICAL FUNCTION IS STRICTLY DELTA-UNIMODAL ON THE GIVEN INTERVAL (SEE [1]) AND THE TOLERANCE FUNCTION SATISFIES TOL(X) >= DELTA, FOR ALL X IN THE GIVEN INTERVAL.

PROCEDURES USED: NONE.

REQUIRED CENTRAL MEMORY: NO AUXILIARY ARRAYS ARE DECLARED IN MININ.

LANGUAGE: ALGOL 60.

METHOD AND PERFORMANCE:
MININ IS A SLIGHTLY MODIFIED VERSION OF THE ALGORITHM GIVEN IN [1].

REFERENCES :
[1]: BRENT, R.P.
ALGORITHMS FOR MINIMIZATION WITHOUT DERIVATIVES. CH.5.
PRENTICE HALL, 1973.

EXAMPLE OF USE:
THE FOLLOWING PROGRAM MAY BE USED TO CALCULATE THE MINIMUM OF THE FUNCTION F(X) = SUM(((I * 2 - 5)/(X - I ** 2)) ** 2; I = 1 (1) 20) ON THE INTERVAL [1,4] (SEE [1]).

```
"BEGIN"
    "REAL" "PROCEDURE" MININ(X, A, B, FX, TOLX); "CODE" 34433;
    "REAL" M, X, A, B; "INTEGER" CNT;
    "REAL" "PROCEDURE" F(X); "VALUE" X; "REAL" X;
    "BEGIN" "INTEGER" I; "REAL" S;
        S:= 0; "FOR" I:= 1 "STEP" 1 "UNTIL" 20 "DO"
        S:= S + ((I * 2 - 5) / (X - I ** 2)) ** 2;
        CNT:= CNT + 1; F:= S
    "END" F;
    "REAL" "PROCEDURE" TOL(X); "VALUE" X; "REAL" X;
    TOL:= ABS(X) * "-7 + "-7;

    A:= 1; B:= 4; CNT:= 0;
    M:= MININ(X, A, B, F(X), TOL(X));
    OUTPUT(61 ,"("4B,"("MINIMUM IS ")",N,/4B,
    "("FOR X IS ")",N,/4B,
    "("IN THE INTERVAL WITH ENDPOINTS ")", /8B,2(N),/4B,
    "("THE NUMBER OF FUNCTION EVALUATIONS NEEDED IS ")",2ZD,/")",
    M, X, A, B, CNT)
"END"
```

```
RESULTS:

MINIMUM IS +3.676699016902O"+000
FOR X IS +3.022915424094B"+000
IN THE INTERVAL WITH ENDPOINTS
    +3.022915021803 3"+000  +3.022915826386 3"+000
THE NUMBER OF FUNCTION EVALUATIONS NEEDED IS  11
```

SOURCE TEXT(S):

```
"CODE" 34433;
    "REAL" "PROCEDURE" MININ(X, A, B, FX, TOLX);
    "REAL" X, A, B, FX, TOLX;
    "BEGIN" "COMMENT" SEE BRENT, 1973, P79;
        "REAL" Z, C, D, E, M, P, Q, R, TOL, T2, U, V, W, FU, FV,
        FW, FZ;
        C:= (3 - SQRT(5)) / 2; "IF" A > B "THEN"
        "BEGIN" Z:= A; A:= B; B:= Z "END";
        V:= W:= Z:= X:= A + C * (B - A); E:= 0;
    FV:= FW:= FZ:= FX;
LOOP: M:= (A + B) * 0.5; TOL:= TOLX; T2:= TOL * 2;
    "IF" ABS(Z - M) > T2 - 0.5 * (B - A) "THEN"
    "BEGIN" P:= Q:= R:= 0; "IF" ABS(E) > TOL "THEN"
        "BEGIN" R:= (Z - W) * (FZ - FV);
            Q:= (Z - V) * (FZ - FW); P:= (Z - V) * Q - (Z - W) * R;
            Q:= (Q - R) * 2; "IF" Q > 0 "THEN" P:= -P "ELSE" Q:=-Q;
            R:= E; E:= D;
        "END";
        "IF" ABS(P) < ABS(0.5 * Q * R) "AND" P > Q * (A - Z)
        "AND" P < Q * (B - Z) "THEN"
        "BEGIN" D:= P / Q; U:= Z + D;
            "IF" U - A < T2 "OR" B - U < T2 "THEN"
            D:= "IF" Z < M "THEN" TOL "ELSE" -TOL
        "END" "ELSE"
        "BEGIN" E:= ("IF" Z < M "THEN" B "ELSE" A) - Z; D:= C * E
        "END";
        U:= X:= Z + ("IF" ABS(D) >= TOL "THEN" D "ELSE" "IF" D > 0
        "THEN" TOL "ELSE" -TOL); FU:= FX;
        "IF" FU <= FZ "THEN"
        "BEGIN" "IF" U < Z "THEN" B:= Z "ELSE" A:= Z;
            V:= W; FV:= FW; W:= Z; FW:= FZ; Z:= U; FZ:= FU
        "END" "ELSE"
        "BEGIN" "IF" U < Z "THEN" A:= U "ELSE" B:= U;
            "IF" FU <= FW "OR" W = Z "THEN"
            "BEGIN" V:= W; FV:= FW; W:= U; FW:= FU "END"
            "ELSE" "IF" FU <= FV "OR" V = Z "OR" V = W "THEN"
            "BEGIN" V:= U; FV:= FU "END"
        "END"; "GOTO" LOOP
    "END"; X:= Z; MININ:= FZ
"END" MININ;
    "EOP"
```

AUTHOR: J. C. P. BUS.

INSTITUTE: MATHEMATICAL CENTRE.

RECEIVED: 741101.

BRIEF DESCRIPTION:

THIS SECTION CONTAINS THE PROCEDURE MININDER, FOR MINIMIZING A FUNCTION OF ONE VARIABLE IN A GIVEN INTERVAL, WHEN THE ANALYTICAL DERIVATIVE OF THE FUNCTION IS AVAILABLE.

KEYWORDS :

MINIMIZATION,
FUNCTIONS OF ONE VARIABLE.

CALLING SEQUENCE:

THE HEADING OF THIS PROCEDURE IS:
"REAL" "PROCEDURE" MININDER(X, Y, FX, DFX, TOLX);
"REAL" X, Y, FX, DFX, TOLX;

MININDER DELIVERS THE CALCULATED MINIMUM VALUE OF THE FUNCTION, DEFINED BY FX, ON THE INTERVAL WITH END POINTS A AND B.

THE MEANING OF THE FORMAL PARAMETERS IS:

X: <REAL VARIABLE>;
A JENSEN VARIABLE; THE ACTUAL PARAMETERS FOR FX, DFX AND TOLX DEPEND ON X;
ENTRY: ONE OF THE END POINTS OF THE INTERVAL ON WHICH THE FUNCTION HAS TO BE MINIMIZED;
EXIT: THE CALCULATED APPROXIMATION OF THE POSITION OF THE MINIMUM;

Y: <REAL VARIABLE>;
ENTRY: THE OTHER END POINT OF THE INTERVAL ON WHICH THE FUNCTION HAS TO BE MINIMIZED;
EXIT: A VALUE SUCH THAT ABS(X - Y) <= TOL(X) * 3;

FX: <ARITHMETIC EXPRESSION>;
THE FUNCTION IS GIVEN BY THE ACTUAL PARAMETER FX WHICH DEPENDS ON X;

DFX: <ARITHMETIC EXPRESSION>;
THE DERIVATIVE OF THE FUNCTION IS GIVEN BY THE ACTUAL PARAMETER DFX WHICH DEPENDS ON X; FX AND DFX ARE EVALUATED SUCCESSIVELY FOR A CERTAIN VALUE OF X;

TOLX: <ARITHMETIC EXPRESSION>;
THE TOLERANCE IS GIVEN BY THE ACTUAL PARAMETER TOLX, WHICH MAY DEPEND ON X; A SUITABLE TOLERANCE FUNCTION IS: ABS(X) * RE + AE, WHERE RE IS THE RELATIVE PRECISION DESIRED AND AE IS AN ABSOLUTE PRECISION WHICH SHOULD NOT BE CHOSEN EQUAL TO ZERO.

DATA AND RESULTS:

THE USER SHOULD BE AWARE OF THE FACT THAT THE CHOICE OF TOLX MAY HIGHLY AFFECT THE BEHAVIOUR OF THE ALGORITHM, ALTHOUGH CONVERGENCE TO A POINT FOR WHICH THE GIVEN FUNCTION IS MINIMAL ON THE GIVEN INTERVAL IS ASSURED; THE ASYMPTOTIC BEHAVIOUR WILL USUALLY BE FINE AS LONG AS THE NUMERICAL FUNCTION IS STRICTLY DELTA-UNIMODAL ON THE GIVEN INTERVAL (SEE [1]) AND THE TOLERANCE FUNCTION SATISFIES TOL(X) >= DELTA, FOR ALL X IN THE GIVEN INTERVAL; LET THE VALUE OF DFX AT THE BEGIN AND END POINT OF THE INITIAL INTERVAL BE DENOTED BY DFA AND DFB, RESPECTIVELY, THEN, FINDING A GLOBAL MINIMUM IS ONLY GUARANTEED IF THE FUNCTION IS CONVEX AND DFA <= 0 AND DFB >= 0; IF THESE CONDITIONS ARE NOT SATISFIED, THEN A LOCAL MINIMUM OR A MINIMUM AT ONE OF THE END POINTS MIGHT BE FOUND.

PROCEDURES USED: NONE.

REQUIRED CENTRAL MEMORY: NO AUXILIARY ARRAYS ARE DECLARED IN MININDER.

LANGUAGE: ALGOL 60.

METHOD AND PERFORMANCE:

MININDER HAS ALMOST THE SAME STRUCTURE AS THE PROCEDURE GIVEN IN [1]; HOWEVER, CUBIC INTERPOLATION (SEE [2]) IS USED INSTEAD OF QUADRATIC INTERPOLATION TO APPROXIMATE THE MINIMIUM.

REFERENCES:

[1]: BRENT, R.P.
ALGORITHMS FOR MINIMIZATION WITHOUT DERIVATIVES. CH.5.
PRENTICE HALL, 1973.

[2]: DAVIDON, W.C.
VARIABLE METRIC METHODS FOR MINIMIZATION.
REP. A.N.L. 5990, 1959.

EXAMPLE OF USE:

THE FOLLOWING PROGRAM MAY BE USED TO CALCULATE THE MINIMUM OF THE FUNCTION F(X) = SUM(((I * 2 - 5)/(X - I ** 2)) ** 2; I = 1 (1) 20) ON THE INTERVAL [1.01,3.99] (SEE [1]).

```
"BEGIN"
    "REAL" "PROCEDURE" MININDER(X, Y, FX, DFX, TOLX); "CODE" 34435;
    "REAL" M, X, Y; "INTEGER" CNT;

    "REAL" "PROCEDURE" F(X); "VALUE" X; "REAL" X;
    "BEGIN" "INTEGER" I; "REAL" S;
        S:= 0; "FOR" I:= 1 "STEP" 1 "UNTIL" 20 "DO"
        S:= S + ((I * 2 - 5) / (X - I ** 2)) ** 2;
        CNT:= CNT + 1; F:= S
    "END" F;

    "REAL" "PROCEDURE" DF(X); "VALUE" X; "REAL" X;
    "BEGIN" "INTEGER" I; "REAL" S;
        S:= 0; "FOR" I:= 1 "STEP" 1 "UNTIL" 20 "DO"
        S:= S + (I * 2 - 5) ** 2 / (X - I ** 2) ** 3;
        DF:= -S * 2
    "END" DF;

    "REAL" "PROCEDURE" TOL(X); "VALUE" X; "REAL" X;
    TOL:= ABS(X) * "-7 + "-7;

    X:= 1.01; Y:= 3.99; CNT:= 0;
    M:= MININDER(X, Y, F(X), DF(X),TOL(X));
    OUTPUT(61 ,"("4B,"("MINIMUM IS ")",N,/4B,
    "("FOR X IS ")",N,/4B,
    "("AND Y IS ")",N,/4B,
    "("THE NUMBER OF FUNCTION EVALUATIONS NEEDED IS ")",2ZD,/")",
    M, X, Y, CNT)
"END"
```

RESULTS:

```
MINIMUM IS +3.676699016902l"+000
FOR X IS +3.022915525030 2"+000
AND Y IS +3.022915122738 6"+000
THE NUMBER OF FUNCTION EVALUATIONS NEEDED IS    9
```

```
SOURCE TEXT(9);

     "CODE"34435;
"REAL" "PROCEDURE" MININDER(X, Y, FX, DFX, TOLX);
"REAL" X, Y, FX, DFX, TOLX;
"BEGIN" "COMMENT" THE FUNCTION IS APPROXIMATED BY A CUBIC AS
     NIVEN BY DAVIDON, 1958, THE STRUCTURE IS SIMILAR TO THE
     STRUCTURE OF THE PROGRAM GIVEN BY BRENT, 1973, THIS IS
     A REVISION OF 760407;

     "INTEGER" SGN;
     "REAL" A, B, C, FA, FB, FU, DFA, DFB, DFU, E, D, TOL, BA,
     Z, P, Q, S;

     "IF" X <= Y "THEN"
     "BEGIN" A:= X; FA:= FX; DFA:= DFX;
          B:= X:= Y; FB:= FX; DFB:= DFX
     "END" "ELSE"
     "BEGIN" B:= X; FB:= FX; DFB:= DFX;
          A:= X:= Y; FA:= FX; DFA:= DFX
     "END";
     C:= (3 - SQRT(5)) / 2; D:= B - A; E:= D * 2; Z:= E * 2;
LOOP: BA:= B - A; TOL:= TOLX; "IF" BA >= TOL * 3 "THEN"
     "BEGIN" "IF" ABS(DFA) <= ABS(DFB) "THEN"
          "BEGIN" X:=A; SGN:= 1 "END" "ELSE"
          "BEGIN" X:= B; SGN:= -1 "END";
          "IF" DFA <= 0 "AND" DFB >= 0 "THEN"
          "BEGIN" Z:= (FA - FB) * 3 / BA + DFA + DFB;
               S:= SQRT(Z ** 2 - DFA * DFB);
               P:= "IF" SGN = 1 "THEN" DFA - S - Z "ELSE"
               DFB + S - Z; P:= P * BA;
               Q:= DFB - DFA + S * 2; Z:= E; E:= D;
               D:= "IF" ABS(P) <= ABS(Q) * TOL "THEN" TOL * SGN
               "ELSE" -P / Q
          "END" "ELSE" D:= BA;
          "IF" ABS(D) >= ABS(Z * 0.5) "OR" ABS(D) > BA * 0.5 "THEN"
          "BEGIN" E:= BA; D:= C * BA * SGN "END";
          X:= X + D; FU:= FX; DFU:= DFX;
          "IF" DFU >= 0 "OR" (FU >= FA "AND" DFA <= 0) "THEN"
          "BEGIN" B:= X; FB:= FU; DFB:= DFU "END" "ELSE"
          "BEGIN" A:= X; FA:= FU; DFA:= DFU "END";
          "GOTO" LOOP
     "END"; "IF" FA <= FB "THEN"
     "BEGIN" X:= A; Y:= B; MININDER:= FA "END" "ELSE"
     "BEGIN" X:= B; Y:= A; MININDER:= FB "END"
"END" MININDER;
     "EOP"
```

AUTHOR: J.C.P.BUS.

INSTITUTE: MATHEMATICAL CENTRE.

RECEIVED: 730620.

BRIEF DESCRIPTION:
THIS SECTION CONTAINS FOUR PROCEDURES, LINEMIN, RNK1UPD, DAVUPD AND FLEUPD, THAT ARE AUXILIARY PROCEDURES FOR THE PROCEDURES RNK1MIN AND FLEMIN (SECTION 5.1.2.2.2).

KEYWORDS:
AUXILIARY PROCEDURE.

SUBSECTION: LINEMIN.

CALLING SEQJENCE:

```
THE HEADING OF THIS AUXILIARY PROCEDURE IS:
"PROCEDURE" LINEMIN(N, X, D, ND, ALFA, G, FUNCT, F0, F1, DF0, DF1,
EVLMAX, STRONGSEARCH, IN);
"VALUE" N, ND, F0, DF0, STRONGSEARCH;
"INTEGER" N, EVLMAX; "BOOLEAN" STRONGSEARCH;
"REAL" ND, ALFA, F0, F1, DF0, DF1;
"ARRAY" X, D, G, IN; "REAL" "PROCEDURE" FUNCT;

THE MEANING OF THE FORMAL PARAMETERS IS:
N:      <ARITHMETIC EXPRESSION>;
        THE NUMBER OF VARIABLES OF THE GIVEN FUNCTION F;
X:      <ARRAY IDENTIFIER>;
        "ARRAY" X[1 : N];
        ENTRY:  A VECTOR  X0, SUCH THAT  F  IS DECREASING IN X0, IN
                THE DIRECTION GIVEN BY D;
        EXIT:   THE  CALCULATED  APPROXIMATION  OF  THE  VECTOR FOR
                WHICH F IS MINIMAL ON THE LINE DEFINED BY:
                X0 + ALFA * D, (ALFA > 0);
D:      <ARRAY IDENTIFIER>;
        "ARRAY" D[1 : N];
        ENTRY:  THE DIRECTION  OF THE LINE  ON WHICH  F  HAS  TO BE
                MINIMIZED;
ND:     <ARITHMETIC EXPRESSION>;
        ENTRY:  THE EUCLIDEAN NORM OF THE VECTOR GIVEN IN D[1 : N];
ALFA:   <VARIABLE>;
        THE INDEPENDENT VARIABLE, THAT DEFINES  THE POSITION ON THE
        LINE ON WHICH F HAS TO BE MINIMIZED;
        THIS LINE IS DEFINED BY X0 + ALFA * D, (ALFA > 0);
        ENTRY:  AN ESTIMATE ALFA0 OF THE VALUE FOR WHICH
                H(ALFA) = F(X0 + ALFA * D), (ALFA > 0), IS MINIMAL;
        EXIT:   THE CALCULATED APPROXIMATION ALFAM OF THE VALUE FOR
                WHICH H(ALFA) IS MINIMAL;
```

```
G:        <ARRAY IDENTIFIER>;
          "ARRAY" G[1 : N];
          EXIT:   THE GRADIENT OF  F  AT THE CALCULATED APPROXIMATION
                  OF THE MINIMUM;
FUNCT:    <PROCEDURE IDENTIFIER>;
          THE HEADING OF THIS PROCEDURE SHOULD BE:
          "REAL" "PROCEDURE" FUNCT(N, X, G); "VALUE" N;
          "INTEGER" N; "ARRAY" X, G;
          A CALL OF FUNCT SHOULD EFFECTUATE IN:
          1:  FUNCT:= F(X);
          2:  THE VALUE OF G[I], (I = 1, ..., N),  BECOMES  THE VALUE
              OF THE I - TH COMPONENT OF THE GRADIENT OF F AT X;
F0:       <ARITHMETIC EXPRESSION>;
          ENTRY:  THE VALUE OF H(0), (SEE ALFA);
F1:       <VARIABLE>;
          ENTRY:  THE VALUE OF H(ALFA0);
          EXIT:   THE VALUE OF H(ALFAM), (SEE ALFA);
DF0:      <ARITHMETIC EXPRESSION>;
          ENTRY:  THE VALUE OF THE DERIVATIVE OF H AT ALFA = 0;
DF1:      <VARIABLE>;
          ENTRY:  THE VALUE OF THE DERIVATIVE OF H AT ALFA = ALFA0;
          EXIT:   THE VALUE OF THE DERIVATIVE OF H AT ALFA = ALFAM;
EVLMAX:   <VARIABLE>;
          ENTRY:  THE MAXIMUM ALLOWED NUMBER OF CALLS OF FUNCT;
          EXIT:   THE NUMBER OF TIMES FUNCT HAS BEEN CALLED;
STRONGSEARCH:
          <BOOLEAN EXPRESSION>;
          IF  THE VALUE OF STRONGSEARCH IS  TRUE,  THEN  THE  PROCESS
          MAKES USE OF TWO STOPPING CRITERIA:
          A:  THE NUMBER OF TIMES FUNCT HAS BEEN CALLED EXCEEDS THE
              GIVEN VALUE OF EVLMAX;
          B:  AN INTERVAL IS FOUND  WITH LENGTH  LESS  THAN TWO TIMES
              THE PRESCRIBED PRECISION,ON WICH A MINIMUM IS EXPECTED;
          IF THE VALUE OF STRONGSEARCH  IS FALSE,  THE PROCESS  MAKES
          ALSO USE OF A THIRD STOPPING CRITERION :
          C:  MU <= (H(ALFAK) - H(ALFA0)) / (ALFAK * DF0) <= 1 - MU,
              WHEREBY ALFAK IS THE CURRENT ITERATE AND MU A
              PRESCRIBED CONSTANT;
IN:       <ARRAY IDENTIFIER>;
          ENTRY:
          "ARRAY" IN[1:3];
          IN[1]:  THE  RELATIVE  PRECISION,  EPSR,  NECESSARY FOR THE
                  STOPPING CRITERION B, (SEE STRONGSEARCH);
          IN[2]:  THE  ABSOLUTE  PRECISION,  EPSA,  NECESSARY FOR THE
                  STOPPING CRITERION B, (SEE STRONGSEARCH);
          THE PRESCRIBED PRECISION, EPS, AT ALFA = ALFAK IS GIVEN BY:
          EPS = NORM ( X0 + ALPHA * D ) * EPSR + EPSA, WHERE
          NORM ( . ) DENOTES THE EUCLIDEAN NORM.
          IN[3]: THE PARAMETER MU NECESSARY FOR STOPPING CRITERION C;
                 THIS  PARAMETER  MUST  SATISFY:   0 < MU < 0.5 ; IN
                 PRACTICE,A CHOICE OF MU = 0.0001 IS ADVISED.
```

DATA AND RESULTS:

LINEMIN CALCULATES AN APPROXIMATION OF A MINIMUM OF A HIGHER - DIMENSIONAL FUNCTION ON A GIVEN LINE;
THE QUANTITY DF0 MUST SATIFY: DF0 < 0;
IF MOREOVER DF1 > 0, THEN THE PROCEDURE WILL YIELD A RESULT THAT SATISFIES ONE OF THE CHOSEN STOPPING CRITERIA, (SEE STRONGSEARCH), OTHERWISE WE CAN NOT GUARANTEE SUCH A RESULT.

PROCEDURES USED:

VECVEC = CP34010,
ELMVEC = CP34020,
DUPVEC = CP31030.

REQUIRED CENTRAL MEMORY:

EXECUTION FIELD LENGTH: N + 17 WORDS.

LANGUAGE: ALGOL 60.

METHOD AND PERFORMANCE:

AN APPROXIMATION TO THE MINIMUM ON THE GIVEN LINE IS CALCULATED WITH CUBIC INTERPOLATION ([2]);THE STOPPING CRITERION USED WHEN THE VALUE OF STRONGSEARCH IS FALSE IS DESCRIBED IN [3] AND [4]; A DETAILED DESCRIPTION OF THIS PROCEDURE IS GIVEN IN [1].

SUBSECTION: RNK1UPD.

CALLING SEQJENCE:

THE HEADING OF THIS AUXILIARY PROCEDJRE IS:
"PROCEDJRE" RNK1UPD(H, N, V, C); "VALUE" N, C;
"INTEGER" N; "REAL" C; "ARRAY" H, V;

THE MEANING OF THE FORMAL PARAMETERS IS:
N: <ARITHMETIC EXPRESSION>;
THE ORDER OF THE SYMMETRIC MATRIX, WHOSE UPPERTRIANGLE IS STORED COLUMNWISE IN THE ONE - DIMENSIONAL ARRAY H;
C: <ARITHMETIC EXPRESSION>;
SEE V;
V: <ARRAY IDENTIFIER>;
"ARRAY" V[1 : N];
THE GIVEN MATRIX IS UPDATED (ANOTHER MATRIX IS ADDED TO IT) WITH A SYMMETRIC MATRIX , U, OF RANK ONE, DEFINED BY:
U[I,J] = C * V[I] * V[J];
H: <ARRAY IDENTIFIER>;
"ARRAY" H[1 : N * (N + 1) // 2];
ENTRY: THE UPPERTRIANGLE (STORED COLUMNWISE, I.E. :
A[I,J] = H[(J-1)*J//2+I], 1 <= I <= J <= N)
OF THE SYMMETRIC MATRIX THAT HAS TO BE UPDATED;
EXIT: THE UPPERTRIANGLE (STORED COLUMNWISE) OF THE UPDATED MATRIX.

PROCEDURES JSED:

ELMVEC = CP34020.

REQUIRED CENTRAL MEMORY:

EXECUTION FIELD LENGTH: 2 WORDS.

LANGUAGE: ALGOL 60.

SUBSECTION: DAVUPD.

CALLING SEQUENCE:

THE HEADING OF THIS AUXILIARY PROCEDURE IS:
"PROCEDURE" DAVUPD(H, N, V, W, C1, C2); "VALUE" N, C1, C2;
"INTEGER" N; "REAL" C1, C2; "ARRAY" H, V, W;

THE MEANING OF THE FORMAL PARAMETERS IS:

N: <ARITHMETIC EXPRESSION>;
THE ORDER OF THE SYMMETRIC MATRIX WHOSE UPPERTRIANGLE IS STORED COLUMNWISE IN THE ONE - DIMENSIONAL ARRAY H;

C1: <ARITHMETIC EXPRESSION>;
SEE W;

C2: <ARITHMETIC EXPRESSION>;
SEE W;

V: <ARRAY IDENTIFIER>;
"ARRAY" V[1 : N];
SEE W;

W: <ARRAY IDENTIFIER>;
"ARRAY" W[1 : N];
THE GIVEN MATRIX IS UPDATED WITH A SYMMETRIC MATRIX U OF RANK TWO, DEFINED BY:
U[I,J] = C1 * V[I] * V[J] - C2 * W[I] * W[J];

H: <ARRAY IDENTIFIER>;
"ARRAY" H[1 : N * (N + 1) // 2];
ENTRY: THE UPPERTRIANGLE (STORED COLUMNWISE, I.E. :
A[I,J] = H[(J - 1) * J // 2 + I], 1 <= I <= J <= N)
OF THE MATRIX THAT HAS TO BE UPDATED;
EXIT: THE UPPERTRIANGLE (STORED COLUMNWISE) OF THE UPDATED MATRIX.

PROCEDURES USED: NONE.

REQUIRED CENTRAL MEMORY:

EXECUTION FIELD LENGTH: 3 WORDS.

LANGUAGE: ALGOL 60.

SUBSECTION: FLEUPD.

CALLING SEQUENCE:

THE HEADING OF THIS AUXILIARY PROCEDURE IS:
"PROCEDURE" FLEUPD(H, N, V, W, C1, C2); "VALUE" N, C1, C2;
"INTEGER" N; "REAL" C1, C2; "ARRAY" H, V, W;

THE MEANING OF THE FORMAL PARAMETERS IS:

N: <ARITHMETIC EXPRESSION>;
THE ORDER OF THE SYMMETRIC MATRIX WHOSE UPPERTRIANGLE IS STORED COLUMNWISE IN THE ONE - DIMENSIONAL ARRAY H;

C1: <ARITHMETIC EXPRESSION>;
SEE W;

C2: <ARITHMETIC EXPRESSION>;
SEE W;

V: <ARRAY IDENTIFIER>;
"ARRAY" V[1 : N];
SEE W;

W: <ARRAY IDENTIFIER>;
"ARRAY" W[1 : N];
THE GIVEN MATRIX IS UPDATED WITH A SYMMETRIC MATRIX U OF RANK TWO, DEFINED BY:
U[I,J]= C2 * V[I] * V[J] - C1 *(V[I] * W[J] + W[I] * V[J]);

H: <ARRAY IDENTIFIER>;
"ARRAY" H[1 : N * (N + 1) // 2];
ENTRY: THE UPPERTRIANGLE (STORED COLUMNWISE, I.E. :
A[I,J] = H[(J - 1) * J // 2 + I], 1 <= I <= J <= N)
OF THE MATRIX THAT HAS TO BE UPDATED;
EXIT: THE UPPERTRIANGLE (STORED COLUMNWISE) OF THE UPOATED MATRIX.

PROCEDURE USED: NONE.

REQUIRED CENTRAL MEMORY:

EXECUTION FIELD LENGTH: 3 WORDS.

LANGUAGE: ALGOL 60.

SOURCE TEXT(S):

```
"CODE" 34210;
    "PROCEDURE" LINEMIN(N, X, D, ND, ALFA, G, FUNCT, F0, F1, DF0, DF1,
    EVLMAX, STRONGSEARCH, IN); "VALUE" N, ND, F0, DF0, STRONGSEARCH;
    "INTEGER" N, EVLMAX; "BOOLEAN" STRONGSEARCH;
    "REAL" ND, ALFA, F0, F1, DF0, DF1;
    "ARRAY" X, D, G, IN;
    "REAL" "PROCEDURE" FUNCT;
    "BEGIN" "INTEGER" I, EVL;
        "BOOLEAN" NOTININT;
        "REAL" F,OLDF,DF,OLDDF,MU,ALFA0,Q,W,Y,Z,RELTOL,ABSTOL
        ,EPS, AID;
        "ARRAY" X0[1:N];
        "REAL" "PROCEDURE" VECVEC(L, U, SHIFT, A, B); "CODE" 34010;
        "PROCEDURE" ELMVEC(L, U, SHIFT, A, B, X); "CODE" 34020;
        "PROCEDURE" DUPVEC(L, U, SHIFT, A, B); "CODE" 31030;

        RELTOL:= IN[1]; ABSTOL:= IN[2]; MU:= IN[3]; EVL:= 0;
        ALFA0:= 0; OLDF:= F0; OLDDF:= DF0; Y:= ALFA; NOTININT:= "TRUE";
        DUPVEC(1, N, 0, X0, X);
        EPS:= (SQRT(VECVEC(1, N, 0, X, X)) * RELTOL + ABSTOL) / ND;
        Q:= (F1 - F0) / (ALFA * DF0);
INT:    "IF" NOTININT "THEN" NOTININT:= DF1 < 0 "AND" Q > MU;
        AID:= ALFA; "IF" DF1 >= 0 "THEN"
        "BEGIN" Z:= 3 * (OLDF - F1) / ALFA + OLDDF + DF1;
            W:= SQRT(Z ** 2 - OLDDF * DF1);
            ALFA:= ALFA * (1 - (DF1 + W - Z) / (DF1 - OLDDF + W * 2));
            "IF" ALFA < EPS "THEN" ALFA:= EPS "ELSE"
            "IF" AID - ALFA < EPS "THEN" ALFA:= AID - EPS
        "END" CUBIC INTERPOLATION
        "ELSE" "IF" NOTININT "THEN"
        "BEGIN" ALFA0:= ALFA:= Y; OLDDF:= DF1; OLDF:= F1 "END"
        "ELSE" ALFA:= 0.5 * ALFA; Y:= ALFA + ALFA0;
        DUPVEC(1, N, 0, X, X0); ELMVEC(1, N, 0, X, D, Y);
        EPS:= (SQRT(VECVEC(1, N, 0, X, X)) * RELTOL + ABSTOL) / ND;
        F:= FUNCT(N, X, G); EVL:= EVL +1 ; DF:= VECVEC(1, N, 0, D, G);
        Q:= (F - F0) / (Y * DF0);
        "IF" ("IF" NOTININT "OR" STRONGSEARCH "THEN" "TRUE" "ELSE"
        Q < MU "OR" Q > 1 - MU) "AND" EVL < EVLMAX "THEN"
        "BEGIN" "IF" NOTININT "OR" DF > 0 "OR" Q < MU "THEN"
            "BEGIN" DF1:= DF; F1:= F "END"
            "ELSE"
            "BEGIN" ALFA0:= Y; ALFA:= AID - ALFA; OLDDF:= DF; OLDF:= F
            "END";
            "IF" ALFA > EPS * 2 "THEN" "GOTO" INT
        "END";
        ALFA:= Y; EVLMAX:= EVL; DF1:= DF; F1:= F
"END" LINEMIN;
    "EOP"
```

```
"CODE" 34211;
    "PROCEDURE" RNK1UPD(H, N, V, C);"VALUE" N, C; "INTEGER" N;
    "REAL" C;"ARRAY" H, V;
     "BEGIN" "INTEGER" J, K;
         "PROCEDURE" ELMVEC(L, U, SHIFT, A, B, X); "CODE" 34020;
         K:= 0;
         "FOR" J:= 1, J + K "WHILE" K < N "DO"
         "BEGIN" K:= K +1 ;
             ELMVEC(J, J + K - 1, 1 - J, H, V, V[K] * C)
         "END"
    "END" RNK1UPD;
         "EOP"

"CODE" 34212;
    "PROCEDURE" DAVUPD(H, N, V, W, C1, C2); "VALUE" N, C1, C2;
    "INTEGER" N; "REAL" C1, C2; "ARRAY" H, V, W;
    "BEGIN" "INTEGER" I, J, K;
         "REAL" VK, WK;
         K:= 0;
         "FOR" J:= 1, J + K "WHILE" K < N "DO"
         "BEGIN" K:= K +1 ; VK:= V[K] * C1; WK:= W[K] * C2;
             "FOR" I:= 0 "STEP" 1 "UNTIL" K -1  "DO"
             H[I + J]:= H[I + J] + V[I + 1] * VK - W[I + 1] * WK
         "END"
    "END" DAVUPD;
         "EOP"

"CODE" 34213;
    "PROCEDURE" FLEUPD(H, N, V, W, C1, C2); "VALUE" N, C1, C2;
    "INTEGER" N; "REAL" C1, C2; "ARRAY" H, V, W;
    "BEGIN" "INTEGER" I, J, K;
         "REAL" VK, WK;
         K:= 0; "FOR" J:= 1, J + K "WHILE" K < N "DO"
         "BEGIN" K:= K +1; VK:= - W[K] * C1 + V[K] * C2; WK:= V[K] * C1;
             "FOR" I:= 0 "STEP" 1 "UNTIL" K - 1  "DO"
             H[I + J]:= H[I + J] + V[I + 1] * VK -W[I + 1] * WK
         "END"
    "END" FLEUPD;
         "EOP"
```

AUTHOR: J.C.P. BUS.

INSTITUTE: MATHEMATICAL CENTRE.

RECEIVED: 741101.

BRIEF DESCRIPTION:

THIS SECTION CONTAINS THE PROCEDURE PRAXIS;
PRAXIS MINIMIZES A FUNCTION OF SEVERAL VARIABLES.

KEYWORDS:

MINIMIZATION,
FUNCTION OF SEVERAL VARIABLES.

CALLING SEQUENCE:

THE HEADING OF THIS PROCEDURE IS:
"PROCEDURE" PRAXIS(N, X, FUNCT, IN, OUT); "VALUE" N;
"INTEGER" N; "ARRAY" X, IN, OUT; "REAL" "PROCEDURE" FJNCT;

THE MEANING OF THE FORMAL PARAMETERS IS:
N: <ARITHMETIC EXPRESSION>;
THE NUMBER OF VARIABLES OF THE FUNCTION TO BE MINIMIZED;
X: <ARRAY IDENTIFIER>;
"ARRAY" X[1 : N];
THE VARIABLES OF THE FUNCTION;
ENTRY: AN APPROXIMATION OF THE POSITION OF THE MINIMJM;
EXIT: THE CALCULATED POSITION OF THE MINIMUM;
FUNCT: <PROCEDURE IDENTIFIER>;
THE HEADING OF THIS PROCEDURE SHOULD BE:
"REAL" "PROCEDURE" FUNCT(N, X); "VALUE" N;
"INTEGER" N; "ARRAY" X;
FUNCT SHOULD DELIVER THE VALUE OF THE FUNCTION TO BE
MINIMIZED, AT THE POINT GIVEN BY X[1:N];

THE MEANING OF THE FORMAL PARAMETERS IS:
N: <ARITHMETIC EXPRESSION>;
THE NUMBER OF VARIABLES;
X: <ARRAY IDENTIFIER>; "ARRAY" X[1:N];
THE VALUES OF THE VARIABLES FOR WHICH THE FUNCTION HAS
TO BE EVALUATED;
IN: <ARRAY IDENTIFIER>;
"ARRAY" IN[0:9];
ENTRY:
IN[0]: THE MACHINE PRECISION; FOR THE CYBER 73 A
SUITABLE VALUE IS "-14;

IN[1], IN[2]: RELATIVE AND ABSOLUTE TOLERANCE, RESPECTIVELY, FOR THE STEPVECTOR (RELATIVE TO THE CURRENT ESTIMATES OF THE VARIABLES); THE PROCESS IS TERMINATED WHEN IN IN[8] + 1 SUCCESSIVE ITERATION STEPS THE EUCLIDEAN NORM OF THE STEP VECTOR IS LESS THAN (IN[1] * NORM(X) + IN[2]) * 0.5;
IN[1] SHOULD BE CHOSEN IN AGREEMENT WITH THE PRECISION IN WHICH THE FUNCTION IS CALCULATED; USUALLY IN[1] SHOULD BE CHOSEN SUCH THAT IN[1] >= SQRT(IN[0]); IN[0] SHOULD BE CHOSEN DIFFERENT FROM ZERO.

IN[3], IN[4] ARE NEITHER USED NOR CHANGED;

IN[5]: THE MAXIMUM NUMBER OF FUNCTION EVALUATIONS ALLOWED (I.E. CALLS OF FUNCT);

IN[6]: THE MAXIMUM STEP SIZE; IN[6] SHOULD BE EQUAL TO THE MAXIMUM EXPECTED DISTANCE BETWEEN THE GUESS AND THE MINIMUM; IF IN[6] IS TOO SMALL OR TOO LARGE, THEN THE INITIAL RATE OF CONVERGENCE WILL BE SLOW;

IN[7]: THE MAXIMUM SCALING FACTOR; THE VALUE OF IN[7] MAY BE USED TO OBTAIN AUTOMATIC SCALING OF THE VARIABLES; HOWEVER, THIS SCALING IS WORTHWHILE BUT MAY BE UNRELIABLE; THEREFORE, THE USER SHOULD TRY TO SCALE HIS PROBLEM HIMSELF AS WELL AS POSSIBLE AND SET IN[7]:= 1; IN EITHER CASE, IN[7] SHOULD NOT BE CHOSEN GREATER THAN 10;

IN[8]: THE PROCESS TERMINATES IF NO SUBSTANTIAL IMPROVEMENT OF THE VALUES OF THE VARIABLES IS OBTAINED IN IN[8] + 1 SUCCESSIVE ITERATION STEPS (SEE IN[1], IN[2]); IN[8] = 4 IS VERY CAUTIOUS; USUALLY, IN[8] = 1 IS SATISFACTORY;

IN[9]: IF THE PROBLEM IS KNOWN TO BE ILL-CONDITIONED (SEE [1]), THEN THE VALUE OF IN[9] SHOULD BE NEGATIVE, OTHERWISE IN[9] >= 0;

OUT: <ARRAY IDENTIFIER>;
"ARRAY" OUT[1:6];
EXIT:

OUT[1]: THIS VALUE GIVES INFORMATION ABOUT THE TERMINATION OF THE PROCESS;
OUT[1] = 0: NORMAL TERMINATION;
OUT[1] = 1: THE PROCESS IS BROKEN OFF, BECAUSE, AT THE END OF AN ITERATION STEP, THE NUMBER OF CALLS OF FUNCT EXCEEDED THE VALUE GIVEN IN IN[5];
OUT[1] = 2: THE PROCESS IS BROKEN OFF, BECAUSE THE CONDITION OF THE PROBLEM IS TOO BAD;

OUT[2]: THE CALCULATED MINIMUM OF THE FUNCTION;
OUT[3]: THE VALUE OF THE FUNCTION AT THE INITIAL GUESS;
OUT[4]: THE NUMBER OF FUNCTION EVALUATIONS NEEDED TO OBTAIN THIS RESULT;
OUT[5]: THE NUMBER OF LINE SEARCHES (SEE [1]);
OUT[6]: THE STEP SIZE IN THE LAST ITERATION STEP.

PROCEDURES USED:

INIVEC = CP31010,
INIMAT = CP31011,
DUPVEC = CP31030,
DUPMAT = CP31035,
DUPCOLVEC = CP31034,
MULROW = CP31021,
MULCOL = CP31022,
VECVEC = CP34010,
TAMMAT = CP34014,
MATTAM = CP34015,
ICHROWCOL = CP34033,
ELMVECCOL = CP34021,
QRISNGVALDEC = CP34273,
SETRANDOM = CP11014,
RANDOM = CP11015.

REQUIRED CENTRAL MEMORY:

EXECUTION FIELD LENGTH: ONE ARRAY OF LENGTH N SQUARED AND FIVE ARRAYS OF LENGTH N ARE DECLARED;

RUNNING TIME:

THE NUMBER OF ITERATION STEPS DEPENDS STRONGLY ON THE PROBLEM TO BE SOLVED.

LANGUAGE: ALGOL 60.

METHOD AND PERFORMANCE:

THIS PROCEDURE IS ADOPTED FROM [1].

REFERENCES:

[1] R. P. BRENT,
ALGORITHMS FOR MINIMIZATION WITHOUT DERIVATIVES, CH. 7.
PRENTICE HALL, 1973.

EXAMPLE OF USE:

THE FOLLOWING PROGRAM MAY BE USED TO CALCULATE THE MINIMUM OF THE FUNCTION F(X) = 100 * (X[2] - X[1] ** 2) ** 2 + (1 - X[1]) ** 2, USING (-1.2, 1) AS AN INITIAL ESTIMATE.

```
"BEGIN"
   "PROCEDURE" PRAXIS(N, X, FUNCT, IN, OUT); "CODE" 34432;

   "ARRAY" X[1:2], IN[0:9], OUT[1:6];

   "REAL" "PROCEDURE" F(N, X); "VALUE" N; "INTEGER" N; "ARRAY" X;
   F:= (X[2] - X[1] ** 2) ** 2 * 100 + (1 - X[1]) ** 2;

   IN[0]:= "-14; IN[1]:= IN[2]:= "-6; IN[5]:= 250;
   IN[6]:= 1; IN[7]:= 1; IN[8]:= 1; IN[9]:= 1;

   X[1]:= -1.2; X[2]:= 1;
   PRAXIS(2, X, F, IN, OUT);
   "IF" OUT[1] = 0 "THEN" OUTPUT(61,"("""("    NORMAL TERMINATION")"
   ,//")");
   OUTPUT(61, "("4B,"("MINIMUM IS ")",N,/4B,
   "("FOR X IS ")", 2(N),/4B,
   "("THE INITIAL FUNCTION VALUE WAS ")" ,N,/4B,
   "("THE NUMBER OF FUNCTION EVALUATIONS NEEDED WAS ")",3ZD/4B,
   "("THE NUMBER OF LINE SEARCHES WAS ")",3ZD/4B,
   "("THE STEP SIZE IN THE LAST ITERATION STEP WAS ")", N,/")",
   OUT[2], X[1], X[2], OUT[3], OUT[4], OUT[5], OUT[6])
"END"
```

RESULTS:

```
NORMAL TERMINATION

MINIMUM IS +1.5694986738789"-021
FOR X IS +1.0000000000389"+000  +1.0000000000785"+000
THE INITIAL FUNCTION VALUE WAS +2.4200000000001"+001
THE NUMBER OF FUNCTION EVALUATIONS NEEDED WAS  189
THE NUMBER OF LINE SEARCHES WAS   72
THE STEP SIZE IN THE LAST ITERATION STEP WAS +5.3830998470105"-009
```

SOURCE TEXT(S):

```
"CODE" 34432;
 "PROCEDURE" PRAXIS(N, X, FUNCT, IN, OUT);
 "VALUE" N; "INTEGER" N;
 "ARRAY" X, IN, OUT;
 "REAL" ."PROCEDURE" FUNCT;
 "BEGIN"
    "COMMENT"THIS PROCEDURE MINIMIZES FUNCT(N,X),WITH THE
    PRINCIPAL AXIS METHOD (SEE BRENT,R.P, 1973, ALGORITHMS
    FOR MINIMIZATION WITHOUT DERIVATIVES,CH.7);

    "PROCEDURE" INIVEC(L, U, A, X);                          "CODE" 31010;
    "PROCEDURE" INIMAT(L, U, K, V, A, X);                    "CODE" 31011;
    "PROCEDURE" DUPVEC(L, U, K, A, X);                       "CODE" 31030;
    "PROCEDURE" DUPMAT(L, U, K, V, A, B);                    "CODE" 31035;
    "PROCEDURE" DUPCOLVEC(L, U, K, A, B);                    "CODE" 31034;
    "PROCEDURE" MULROW(L, U, I, J, A, B, X);                 "CODE" 31021;
    "PROCEDURE" MULCOL(L, U, I, J, A, B, X);                 "CODE" 31022;
    "REAL" "PROCEDURE" VECVEC(L, U, S, A, B);                "CODE" 34010;
    "REAL" "PROCEDURE" TAMMAT(L, U, I, J, A, B);             "CODE" 34014;
    "REAL" "PROCEDURE" MATTAM(L, U, I, J, A, B);             "CODE" 34015;
    "PROCEDURE" ICHROWCOL(L, U, I, J, A);                    "CODE" 34033;
    "PROCEDURE" ELMVECCOL(L, U, I, A, B, X);                 "CODE" 34021;
    "INTEGER" "PROCEDURE" QRISNGVALDEC(A,M,N,VAL,V,EM);      "CODE" 34273;
    "PROCEDURE" SETRANDOM(X);                                "CODE" 11014;
    "REAL" "PROCEDURE" RANDOM;                               "CODE" 11015;

    "PROCEDURE" SORT;
    "BEGIN" "INTEGER" I, J, K; "REAL" S;
        "FOR" I:= 1 "STEP" 1 "UNTIL" N - 1 "DO"
        "BEGIN" K:= I; S:= D[I];
            "FOR" J:= I+1 "STEP" 1 "UNTIL" N "DO" "IF" D[J]>S "THEN"
            "BEGIN" K:= J; S:= D[J] "END";
            "IF" K>I "THEN"
            "BEGIN" D[K]:= D[I]; D[I]:= S;
                "FOR" J:= 1 "STEP" 1 "UNTIL" N "DO"
                "BEGIN" S:=V[J,I]; V[J,I]:= V[J,K]; V[J,K]:= S
                "END"
            "END"
        "END"
    "END" SORT
```

```
      "PROCEDURE" MIN(J, NITS, D2, X1, F1, FK); "VALUE" J, NITS, FK;
      "INTEGER" J, NITS; "REAL" D2, X1, F1; "BOOLEAN" FK;
      "BEGIN"
          "REAL" "PROCEDURE" FLIN(L); "VALUE" L; "REAL" L;
          "BEGIN" "INTEGER" I; "ARRAY" T[1:N];
              "IF" J > 0 "THEN"
          "BEGIN" "FOR" I:= 1 "STEP" 1 "UNTIL" N "DO"
              T[I]:= X[I] + L * V[I,J]
          "END" "ELSE"
          "BEGIN" "COMMENT" SEARCH ALONG PARABOLIC SPACE CURVE;
              QA:= L * (L - QD1) / (QD0 * (QD0 + QD1));
              QB:= (L + QD0) * (QD1 - L) /(QD0 * QD1);
              QC:= L * (L + QD0) / (QD1 * (QD0 + QD1));
              "FOR" I:= 1 "STEP" 1 "UNTIL" N "DO"
              T[I]:= QA * Q0[I] +QB * X[I] + QC * Q1[I]
          "END";
          NF:= NF + 1; FLIN:= FUNCT(N, T)
      "END" FLIN;

      "INTEGER" K; "BOOLEAN" DZ;
      "REAL" X2, XM, F0, F2, FM, D1, T2, S, SF1, SX1;
      SF1:= F1; SX1:= X1;
          K:= 0; XM:= 0; F0:= FM:= Fx; DZ:= D2 < RELTOL;
          S:= SQRT(VECVEC(1,N,0,X,X));
          T2:= M4 * SQRT(ABS(FX) / ("IF" DZ "THEN" DMIN "ELSE" D2)
          + S * LDT) + M2 * LDT; S:= S * M4 + ABSTOL;
          "IF" DZ "AND" T2 > S "THEN" T2:= S;
          "IF"T2<SMALL"THEN"T2:= SMALL;
          "IF"T2>0.01*H "THEN"T2:= 0.01*H;
          "IF"FK"AND"F1<=FM "THEN"
          "BEGIN"XM:=X1; FM:= F1 "END";
          "IF" ¬ FK"OR"ABS(X1)<T2"THEN"
          "BEGIN"X1:="IF"X1>0 "THEN"T2"ELSE"-T2;
              F1:= FLIN(X1)
          "END";
          "IF"F1<= FM"THEN"
          "BEGIN"XM:= X1; FM:= F1 "END";
      L0: "IF" DZ "THEN"
          "BEGIN" "COMMENT"EVALUATE FLIN AT ANOTHER POINT
              AND ESTIMATE THE SECOND DERIVATIVE;
          X2:= "IF" F0 < F1 "THEN" -X1 "ELSE" X1 * 2;
          F2:= FLIN(X2); "IF"F2 <= FM "THEN"
          "BEGIN" XM:= X2; FM:= F2 "END";
          D2:=(X2*(F1-F0)-X1*(F2-F0))/(X1*X2*(X1-X2))
      "END";
      "COMMENT"ESTIMATE FIRST DERIVATIVE AT 0;
      D1:=(F1-F0)/X1-X1*D2; DZ:="TRUE";
      X2:= "IF"D2<=SMALL"THEN"
      ("IF"D1<0"THEN"H"ELSE"-H)
      "ELSE"-0.5*D1/D2;
      "IF"ABS(X2)>H"THEN"X2:="IF"X2>0"THEN"H"ELSE"-H;
                                                  "COMMENT"
```

```
;


    L1: F2:=FLIN(X2);
        "IF"K<NITS"AND"F2>F0"THEN"
        "BEGIN"K:=K+1;
            "IF"F0<F1"AND"X1*X2>0"THEN" "GOTO"L0;
            X2:= 0.5*X2; "GOTO"L1
        "END";
        NL:= NL+1;
        "IF"F2>FM"THEN"X2:=XM"ELSE"FM:=F2;
        D2:="IF"ABS(X2*(X2-X1))>SMALL"THEN"
        (X2*(F1-F0)-X1*(FM-F0))/(X1*X2*(X1-X2))
        "ELSE" "IF"K>0"THEN"0"ELSE"D2;
        "IF"D2<=SMALL"THEN"D2:=SMALL;
        X1:=X2; FX:=FM;
        "IF"SF1<FX"THEN"
        "BEGIN" FX:=SF1; X1:=SX1 "END";
        "IF"J>0"THEN"ELMVECCOL(1,N,J,X,V,X1)
"END" MIN;

"PROCEDURE"QUAD;
"BEGIN" "INTEGER" I; "REAL" L, S;
        S:= FX; FX:= QF1; QF1:= S; QD1:= 0;
        "FOR" I:= 1 "STEP" 1 "UNTIL" N "DO"
        "BEGIN"S:=X[I]; X[I]:= L:= Q1[I]; Q1[I]:= S;
            QD1:= QD1 + (S - L) ** 2
        "END";
        L:=QD1:=SQRT(QD1); S:= 0;
        "IF"QD0>0"AND"QD1>0"AND"NL>=3*N*N"THEN"
        "BEGIN"MIN(0,2,S,L,QF1,"TRUE");
            QA:= L*(L-QD1)/(QD0*(QD0+QD1));
            QB:=(L+QD0)*(QD1-L)/(QD0*QD1);
            QC:= L*(L+QD0)/(QD1*(QD0+QD1))
        "END" "ELSE"
        "BEGIN" FX:= QF1; QA:= QB:= 0; QC:= 1 "END";
        QD0:= QD1;"FOR"I:= 1"STEP"1"UNTIL"N"DO"
        "BEGIN"S:=Q0[I]; Q0[I]:=X[I];
            X[I]:= QA*S + QB*X[I]+QC*Q1[I]
        "END"
"END" QUAD;

"BOOLEAN" ILLC;
"INTEGER" I, J, K, K2, NL, MAXF, NF, KL, KT, KTM;
"REAL" S, SL, DN, DMIN, FX, F1, LDS, LDT, SF, DF, QF1, QD0,
QD1, QA, QB, QC, M2, M4, SMALL, VSMALL, LARGE, VLARGE, SCBD,
LDFAC,T2, MACHEPS, RELTOL, ABSTOL, H;
"ARRAY" V[1:N,1:N], D, Y, Z, Q0, Q1[1:N];

MACHEPS:= IN[0]; RELTOL:= IN[1]; ABSTOL:= IN[2]; MAXF:= IN[5];
H:= IN[6]; SCBD:= IN[7]; KTM:= IN[8]; ILLC:= IN[9] < 0;
SMALL:= MACHEPS ** 2; VSMALL:= SMALL ** 2;
LARGE:= 1/SMALL; VLARGE:= 1/VSMALL;
M2:= RELTOL; M4:= SQRT(M2); SETRANDOM(0.5);
LDFAC:= "IF" ILLC "THEN" 0.1 "ELSE" 0.01;
KT:=NL:=0; NF:=1; OUT[3]:= QF1:=FX:=FUNCT(N,X);
                                                "COMMENT"
```

```
      ABSTOL:=T2:= SMALL+ABS(ABSTOL); DMIN:= SMALL;
      "IF" H<ABSTOL*100"THEN"H:=ABSTOL*100; LDT:=H;
      INIMAT(1,N,1,N,V,0);
      "FOR"I:=1"STEP"1"UNTIL"N"DO"V[I,I]:= 1;
      D[1]:= QD0:= 0; DUPVEC(1,N,0,Q1,X);


      "COMMENT"MAIN LOOP;
L0:   SF:=D[1]; D[1]:= S:= 0;
      MIN(1,2,D[1],S,FX,"FALSE");
      "IF" S <= 0 "THEN" MULCOL(1, N, 1, 1, V, V, -1);
      "IF" SF <= 0.9 * D[1] "OR" 0.9 * SF >= D[1] "THEN"
      INIVEC(2,N,D,0);
      "FOR" K:= 2"STEP"1"UNTIL"N"DO"
      "BEGIN" DUPVEC(1,N,0,Y,X); SF:=FX;
          ILLC:= ILLC "OR" KT>0;
      L1: KL:=K; DF:= 0; "IF" ILLC "THEN"
          "BEGIN" "COMMENT"RANDOM STOP TO GET OFF
              RESULTION VALLEY;
              "FOR"I:= 1 "STEP"1"UNTIL"N"DO"
              "BEGIN"S:=Z[I]:=(0.1*LDT+T2*10**KT)
                  *(RANDOM-0.5);
              ELMVECCOL(1,N,I,X,V,S)
          "END";
          FX:= FUNCT(N,X); NF:= NF+1
      "END";
      "FOR"K2:= K "STEP" 1 "UNTIL" N "DO"
      "BEGIN" SL:=FX; S:= 0;
          MIN (K2, 2, D[K2], S, FX, "FALSE");
          S:="IF" ILLC "THEN" D[K2] * (S + Z[K2]) ** 2
          "ELSE"SL-FX;"IF"DF<S"THEN"
          "BEGIN"DF:=S;KL:= K2"END";
      "END";
      "IF" ^ILLC "AND" DF < ABS(100 * MACHEPS * FX) "THEN"
      "BEGIN" ILLC:= "TRUE"; "GOTO" L1 "END";
      "FOR" K2:= 1"STEP" 1"UNTIL"K-1"DO"
      "BEGIN" S:= 0; MIN(K2, 2, D[K2], S, FX, "FALSE") "END";
      F1:= FX; FX:= SF; LDS:= 0;
      "FOR" I:= 1 "STEP" 1 "UNTIL" N "DO"
      "BEGIN" SL:= X[I]; X[I]:= Y[I]; SL:= Y[I]:= SL - Y[I];
          LDS:= LDS + SL * SL
      "END"; LDS:= SQRT(LDS);
      "IF" LDS > SMALL "THEN"
      "BEGIN" "FOR" I:= KL - 1 "STEP" -1 "UNTIL" K "DO"
          "BEGIN" "FOR" J:= 1 "STEP" 1 "UNTIL" N "DO"
              V[J, I + 1]:= V[J,I]; D[I + 1]:= D[I]
          "END";
          D[K]:= 0; DUPCOLVEC(1, N, K, V, Y);
          MULCOL(1, N, K, K, V, V, 1 / LDS);
          MIN(K, 4, D[K], LDS, F1, "TRUE"); "IF" LDS <= 0 "THEN"
          "BEGIN" LDS:= LDS; MULCOL(1, N, K, K, V, V, -1) "END"
      "END";
      LDT:= LDFAC * LDT; "IF" LDT < LDS "THEN" LDT:= LDS;
      T2:= M2 * SQRT(VECVEC(1, N, 0, X, X)) + ABSTOL;
                                                  "COMMENT"
```

```
        KT:= "IF" LDT > 0.5 * T2 "THEN" 0 "ELSE" KT + 1;
        "IF" KT > KTM "THEN" "BEGIN" OUT[1]:= 0; "GOTO" L2 "END"
    "END";
    QUAD;
    DN:= 0;"FOR"I:= 1"STEP"1"UNTIL"N"DO"
    "BEGIN"D[I]:= 1/SQRT(D[I]);
        "IF"DN<D[I]"THEN"DN:=D[I]
    "END";
    "FOR"J:= 1"STEP"1"UNTIL"N"DO"
    "BEGIN"S:= D[J]/DN; MULCOL(1,N,J,J,V,V,S)"END";
    "IF"SCBD>1"THEN"
    "BEGIN"S:=VLARGE; "FOR"I:=1 "STEP"1"UNTIL"N"DO"
        "BEGIN" SL:= Z[I]:= SQRT(MATTAM(1, N, I, I, V, V));
            "IF"SL<M4"THEN"Z[I]:= M4;
            "IF" S>SL "THEN" S:= SL
        "END";
        "FOR"I:=1"STEP"1"UNTIL"N"DO"
        "BEGIN"SL:=S/Z[I];Z[I]:= 1/SL;
            "IF"Z[I]>SCBD"THEN"
            "BEGIN"SL:=1/SCBD; Z[I]:= SCBD"END";
            MULROW(1, N, I, I, V, V, SL)
        "END"
    "END";
    "FOR" I:= 1 "STEP" 1 "UNTIL" N "DO"
    ICHROWCOL(I + 1, N, I, I, V);
    "BEGIN" "ARRAY" A[1:N,1:N], EM[0:7];
        EM[0]:= EM[2]:= MACHEPS;
        EM[4]:= 10 * N; EM[6]:= VSMALL;
        DUPMAT(1, N, 1, N, A, V);
        "IF" QRISNGVALDEC(A, N, N, D, V, EM) ^= 0 "THEN"
        "BEGIN" OUT[1]:= 2; "GOTO" L2 "END";
    "END";
    "IF"SCBD>1"THEN"
    "BEGIN" "FOR"I:=1"STEP"1"UNTIL"N"DO"
        MULROW(1,N,I,I,V,V,Z[I]);
        "FOR"I:= 1"STEP"1"UNTIL"N"DO"
        "BEGIN"S:= SQRT(TAMMAT(1,N,I,I,V,V));
            D[I]:= S*D[I]; S:= 1/S;
            MULCOL(1,N,I,I,V,V,S)
            "END"
        "END";
        "FOR" I:= 1 "STEP" 1 "UNTIL" N "DO"
        "BEGIN" S:= DN * D[I];
            D[I]:= "IF" S > LARGE "THEN" VSMALL "ELSE"
            "IF" S < SMALL "THEN" VLARGE "ELSE" S ** (-2)
        "END";
        SORT;
        DMIN:= D[N]; "IF" DMIN < SMALL "THEN" DMIN:= SMALL;
        ILLC:= (M2 * D[1]) > DMIN;
        "IF" NF < MAXF "THEN" "GOTO" L0 "ELSE" OUT[1]:= 1;
L2:     OUT[2]:= FX;
        OUT[4]:= NF; OUT[5]:= NL; OUT[6]:= LDT
"END"PRAXIS;
            "EOP"
```

AUTHOR: J.C.P.BUS.

INSTITUTE: MATHEMATICAL CENTRE.

RECEIVED: 730620.

BRIEF DESCRIPTION:

THIS SECTION CONTAINS TWO PROCEDURES, RNK1MIN AND FLEMIN, FOR MINIMIZING A GIVEN DIFFERENTIABLE FUNCTION OF SEVERAL VARIABLES; BOTH PROCEDURES USE A VARIABLE METRIC METHOD; THE USER HAS TO PROGRAM THE EVALUATION OF THE FUNCTION AND ITS GRADIENT; THE CHOICE OF RNK1MIN AND FLEMIN IS DEPENDENT ON THE PROBLEM INVOLVED; IF THE NUMBER OF VARIABLES OF THE FUNCTION TO BE MINIMIZED IS VERY LARGE AND THE CALCULATION OF THE FUNCTION AND ITS GRADIENT IS RELATIVELY CHEAP (THE NUMBER OF ARITHMETIC OPERATIONS IS OF ORDER AT MOST N ** 2),THEN THE USER IS ADVISED TO USE FLEMIN; IF THE HESSIAN OF THE FUNCTION IS EXPECTED TO BE (ALMOST) SINGULAR AT THE MINIMUM, THEN RNK1MIN IS PREFERRED;

KEYWORDS:

OPTIMIZATION,
HIGHER - DIMENSIONAL,
UNCONSTRAINED,
VARIABLE METRIC METHOD.

SUBSECTION: RNK1MIN.

CALLING SEQUENCE:

THE HEADING OF THIS PROCEDURE IS:
"REAL" "PROCEDURE" RNK1MIN(N, X, G, H, FUNCT, IN, OUT);
"VALUE" N; "INTEGER" N;
"ARRAY" X, G, H, IN, OUT;
"REAL" "PROCEDURE" FUNCT;

RNK1MIN: DELIVERS THE CALCULATED LEAST VALUE OF THE GIVEN FUNCTION;

THE MEANING OF THE FORMAL PARAMETERS IS:
N: <ARITHMETIC EXPRESSION>;
THE NUMBER OF VARIABLES OF THE FUNCTION TO BE MINIMIZED;
X: <ARRAY IDENTIFIER>;
"ARRAY" X[1 : N];
THE INDEPENDENT VARIABLES;
ENTRY: AN APPROXIMATION OF A MINIMUM OF THE FUNCTION;
EXIT: THE CALCULATED MINIMUM OF THE FUNCTION;
G: <ARRAY IDENTIFIER>;
"ARRAY" G[1 : N];
EXIT: THE GRADIENT OF THE FUNCTION AT THE CALCULATED MINIMUM;
H: <ARRAY IDENTIFIER>;
"ARRAY" H[1 : N * (N + 1) // 2];
THE UPPERTRIANGLE OF AN APPROXIMATION OF THE INVERSE HESSIAN IS STORED COLUMNWISE IN H (I.E. THE I,J-TH ELEMENT= H[(J - 1) * J // 2 + I], 1 <= I<= J <= N);
IF IN[6] > 0 INITIALIZING OF H WILL BE DONE AUTOMATICALLY AND THE INITIAL APPROXIMATION OF THE INVERSE HESSIAN WILL EQUAL THE UNITMATRIX MULTIPLIED WITH THE VALUE OF IN[6];
IF IN[6] < 0 NO INITIALIZING OF H WILL BE DONE AND THE USER SHOULD GIVE IN H AN APPROXIMATION OF THE INVERSE HESSIAN, AT THE STARTING POINT;THE UPPERTRIANGLE OF AN APPROXIMATION OF THE INVERSE HESSIAN AT THE CALCULATED MINIMUM IS DELIVERED IN H;
FUNCT: <PROCEDURE IDENTIFIER>;
THE HEADING OF THIS PROCEDURE SHOULD BE:
"REAL" "PROCEDURE" FUNCT(N, X, G); "VALUE" N;
"INTEGER" N; "ARRAY" X, G;
A CALL OF FUNCT MUST EFFECTUATE IN:
1: FUNCT BECOMES THE VALUE OF THE FUNCTION TO BE MINIMIZED AT THE POINT X;
2: THE VALUE OF G[I], (I = 1, ..., N), BECOMES THE VALUE OF THE I - TH COMPONENT OF THE GRADIENT OF THE FUNCTION AT X;

IN: <ARRAY IDENTIFIER>;
"ARRAY" IN[0 : 8];
ENTRY:
IN[0]: THE MACHINE PRECISION;
FOR THE CYBER 73-26 A SUITABLE VALUE IS "-14;
IN[1]: THE RELATIVE TOLERANCE FOR THE SOLUTION;
THIS TOLERANCE SHOULD NOT BE CHOSEN SMALLER THAN IN[0];
IN[2]: THE ABSOLUTE TOLERANCE FOR THE SOLUTION;
IN[3]; A PARAMETER USED FOR CONTROLLING LINEMINIMIZATION, ([3], [4]); USUALLY A SUITABLE VALUE IS: 0.0001;
IN[4]: THE ABSOLUTE TOLERANCE FOR THE EUCLIDEAN NORM OF THE GRADIENT AT THE SOLUTION;
IN[5]: A LOWERBOUND FOR THE FUNCTIONVALUE;
IN[6]: THIS PARAMETER CONTROLS THE INITIALIZATION OF THE APPROXIMATION OF THE INVERSE HESSIAN (METRIC), SEE H; USUALLY THE CHOICE IN[6] = 1 WILL GIVE GOOD RESULTS;
IN[7]: THE MAXIMUM ALLOWED NUMBER OF CALLS OF FUNCT;
IN[8]: A PARAMETER USED FOR CONTROLLING THE UPDATING OF THE METRIC; IT IS USED TO AVOID UNBOUNDEDNESS OF THE METRIC (SEE: [6], FORMULA (19));
THE VALUE OF IN[8] SHOULD SATISFY:
SQRT(IN[0] / IN[1]) / N < IN[8] < 1;
USUALLY A SUITABLE VALUE WILL BE 0.01;

OUT: <ARRAY IDENTIFIER>;
"ARRAY" OUT[0:4];
EXIT:
OUT[0]: THE EUCLIDEAN NORM OF THE PRODUCT OF THE METRIC AND THE GRADIENT AT THE CALCULATED MINIMUM;
OUT[1]: THE EUCLIDEAN NORM OF THE GRADIENT AT THE CALCULATED MINIMUM;
OUT[2]: THE NUMBER OF CALLS OF FUNCT, NECESSARY TO ATTAIN THIS RESULT;
OUT[3]: THE NUMBER OF ITERATIONS IN WHICH A LINESEARCH WAS NECESSARY;
OUT[4]: THE NUMBER OF ITERATIONS IN WHICH A DIRECTION HAD TO BE CALCULATED WITH THE METHOD GIVEN IN [5]; IN SUCH AN ITERATION A CALCULATION OF THE EIGENVALUES AND EIGENVECTORS OF THE METRIC IS NECESSARY.

DATA AND RESULTS:

USUALLY THE CALCULATED SOLUTION WILL SATISFY:
NORM (XMIN - XCAL) < NORM (XCAL) * IN[1] + IN[2].
WHERE AT XMIN THE GIVEN FUNCTION IS MINIMAL, XCAL THE CALCULATED APPROXIMATION OF XMIN AND NORM(.) DENOTES THE EUCLIDEAN NORM OF X; HOWEVER, WE CANNOT GUARANTEE SUCH A RESULT; THE CALCULATED SOLUTION POSSIBLY WILL NOT SATISFY THE ABOVE INEQUALITY IF THE PROBLEM IS VERY ILL - CONDITIONED; THE USER CAN DISCOVER SUCH A SITUATION BY LOOKING AT THE EUCLIDEAN NORM OF THE METRIC, DELIVERED IN H; THE PROBLEM IS ILL - CONDITIONED IF THIS NORM IS LARGE RELATIVE TO 1.

PROCEDURES USED:

VECVEC = CP34010,
MATVEC = CP34011,
TAMVEC = CP34012,
SYMMATVEC = CP34018,
INIVEC = CP31010,
INISYMD = CP31013,
MULVEC = CP31020,
DUPVEC = CP31030,
EIGSYM1 = CP34156,
LINEMIN = CP34210,
RNK1UPD = CP34211,
DAVUPD = CP34212,
FLEUPD = CP34213.

REQUIRED CENTRAL MEMORY:

EXECUTION FIELD LENGTH: VARIES FROM 5 * N + 26 TO
N ** 2 + N * (N + 1) // 2 + 5 * N + 35 WORDS.

RUNNING TIME:

DEPENDS STRONGLY ON THE PROBLEM TO BE SOLVED;

LANGUAGE: ALGOL 60.

METHOD AND PERFORMANCE:

RNK1MIN CALCULATES AN APPROXIMATION OF A MINIMUM OF A GIVEN FUNCTION BY MEANS OF A VARIABLE METRIC METHOD; THE RANK - ONE UPDATING FORMULA, USED IN THIS ALGORITHM IS GIVEN IN [6], (FORMULA (4)); TO AVOID UNBOUNDEDNESS OF THE METRIC (SEE [8]), SOMETIMES A RANK - TWO UPDATING FORMULA IS USED ([3], FORMULAS (1) AND (5)); TO AVOID LINESEARCHES AS MUCH AS POSSIBLE A STRATEGY GIVEN IN [4] IS USED; IF IN AN ITERATION THE FUNCTION IS INCREASING IN THE DIRECTION GIVEN BY THE VARIABLE METRIC ALGORITHM, BECAUSE THE METRIC IS NOT POSITIVE DEFINITE, THEN A METHOD GIVEN IN [5] IS USED TO CALCULATE A NEW DIRECTION; THIS METHOD REQUIRES THE CALCULATION OF THE EIGENVECTORS AND EIGENVALUES OF THE METRIC; USUALLY,THE NUMBER OF TIMES SUCH A CALCULATION IS NECESSARY IS VERY SMALL RELATIVE TO THE NUMBER OF ITERATIONS (AND OFTEN EQUALS ZERO); IF THE NUMBER OF VARIABLES OF THE FUNCTION IS VERY LARGE AND THE CALCULATION OF THE FUNCTION AND ITS GRADIENT IS RELATIVELY CHEAP (THE NUMBER OF ARITHMETICAL OPERATIONS IS OF ORDER AT MOST N ** 2), THEN THE USER IS ADVISED TO USE FLEMIN (CP32105);
A DETAILED DESCRIPTION OF THE ALGORITHM AND SOME RESULTS ABOUT ITS CONVERGENCE IS GIVEN IN [1].

SUBSECTION: FLEMIN.

CALLING SEQUENCE:

```
THE HEADING OF THIS PROCEDURE IS:
"REAL" "PROCEDURE" FLEMIN(N, X, G, H, FUNCT, IN, OUT);
"VALUE" N; "INTEGER" N;
"ARRAY" X, G, H, IN, OUT; "REAL" "PROCEDURE" FUNCT;

FLEMIN: DELIVERS THE CALCULATED LEAST VALUE OF THE GIVEN FUNCTION;

THE MEANING OF THE FORMAL PARAMETERS IS:
N:      <ARITHMETIC EXPRESSION>;
        THE NUMBER OF VARIABLES OF THE FUNCTION TO BE MINIMIZED;
X:      <ARRAY IDENTIFIER>;
        "ARRAY" X[1 : N];
        THE INDEPENDENT VARIABLES;
        ENTRY:  AN APPROXIMATION OF A MINIMUM OF THE FUNCTION;
        EXIT:   THE CALCULATED MINIMUM OF THE FUNCTION;
G:      <ARRAY IDENTIFIER>;
        "ARRAY" G[1 : N];
        EXIT:   THE GRADIENT OF THE FUNCTION AT THE CALCULATED
                MINIMUM;
H:      <ARRAY IDENTIFIER>;
        "ARRAY" H[1 : N * (N + 1) // 2];
        THE UPPERTRIANGLE OF AN APPROXIMATION OF THE INVERSE
        HESSIAN IS STORED COLUMNWISE IN H (I.E. THE I,J-TH ELEMENT=
        H[ )J - 1) * J // 2 + I], 1 <= I <= J <= N);
        IF IN[6] > 0  INITIALIZING OF  H WILL BE DONE AUTOMATICALLY
        AND  THE INITIAL APPROXIMATION  OF THE INVERSE HESSIAN WILL
        EQUAL THE UNITMATRIX MULTIPLIED WITH THE VALUE OF IN[6]; IF
        IN[6] < 0  NO INITIALIZING OF  H  WILL BE DONE AND THE USER
        SHOULD GIVE IN  H  AN APPROXIMATION OF  THE INVERSE HESSIAN
        AT THE STARTING POINT;
        THE  UPPERTRIANGLE  OF  AN  APPROXIMATION  OF  THE  INVERSE
        HESSIAN AT THE CALCULATED MINIMUM IS DELIVERED IN H;
FUNCT:  <PROCEDURE IDENTIFIER>;
        THE HEADING OF THIS PROCEDURE SHOULD BE :
        "REAL" "PROCEDURE" FUNCT(N, X, G); "VALUE" N;
        "INTEGER" N; "ARRAY" X, G;
        A CALL OF FUNCT SHOULD EFFECTUATE IN:
        1:  FUNCT BECOMES THE VALUE OF THE FUNCTION TO BE MINIMIZED
            AT THE POINT X;
        2:  THE VALUE OF G[I], (I = 1, ..., N),  BECOMES  THE VALUE
            OF THE I - TH COMPONENT OF THE GRADIENT OF THE FUNCTION
            AT X;
```

```
IN:     <ARRAY IDENTIFIER>;
        "ARRAY" IN[1 : 7];
        ENTRY:
        IN[1]:  THE RELATIVE TOLERANCE FOR THE SOLUTION;
        IN[2]:  THE ABSOLUTE TOLERANCE FOR THE SOLUTION;
        IN[3]:  A PARAMETER  USED FOR CONTROLLING LINEMINIMIZATION
                ([3], [4]); USUALLY A SUITABLE VALUE IS 0.0001;
        IN[4]:  THE ABSOLUTE TOLERANCE  FOR  THE EUCLIDEAN  NORM OF
                THE GRADIENT AT THE SOLUTION;
        IN[5]:  A LOWERBOUND FOR THE FUNCTION VALUE;
        IN[6]:  THIS  PARAMETER  CONTROLS THE INITIALIZATION OF THE
                APPROXIMATION OF THE INVERSE HESSIAN (METRIC) (SEE
                H); USUALLY IN[6] = 1 WILL GIVE GOOD RESULTS;
        IN[7]:  THE MAXIMUM ALLOWED NUMBER OF CALLS OF FUNCT;
OUT:    <ARRAY IDENTIFIER>;
        "ARRAY" OUT[0:4];
        EXIT:
        OUT[0]: THE EUCLIDEAN NORM OF THE PRODUCT OF THE METRIC AND
                THE GRADIENT AT THE CALCULATED MINIMUM;
        OUT[1]: THE  EUCLIDEAN  NORM   OF  THE  GRADIENT  AT  THE
                CALCULATED MINIMUM;
        OUT[2]: THE NUMBER OF CALLS OF FUNCT,  NECESSARY  TO ATTAIN
                THESE RESULTS;
        OUT[3]: THE NUMBER OF ITERATIONS IN WHICH A LINESEARCH  WAS
                NECESSARY;
        OUT[4]: IF  OUT[4] = - 1,  THEN  THE  PROCESS  IS BROKEN OFF
                BECAUSE  NO DOWNHILL DIRECTION COULD BE CALCULATED;
                THE PRECISION  ASKED FOR MAY NOT BE ATTAINED AND IS
                POSSIBLY CHOSEN TOO HIGH;
                NORMALLY OUT[4] = 0;
```

DATA AND RESULTS:

USUALLY THE CALCULATED SOLUTION WILL SATISFY:
NORM (XMIN - XCAL) < NORM (XCAL) * IN[1] + IN[2].
WHERE AT XMIN THE GIVEN FUNCTION IS MINIMAL, XCAL THE CALCULATED APPROXIMATION OF XMIN AND NORM (.) DENOTES THE EUCLIDEAN NORM OF X; HOWEVER, WE CAN NOT GUARANTEE SUCH A RESULT; THE CALCULATED SOLUTION POSSIBLY WILL NOT SATISFY THE ABOVE INEQUALITY IF THE PROBLEM IS VERY ILL - CONDITIONED; THE USER CAN DISCOVER SUCH A SITUATION BY LOOKING AT THE EUCLIDEAN NORM OF THE METRIC, DELIVERED IN H; THE PROBLEM IS ILL - CONDITIONED IF THIS NORM IS LARGE RELATIVE TO 1.

PROCEDURES USED:

VECVEC = CP34010,
ELMVEC = CP34020,
SYMMATVEC = CP34018,
INIVEC = CP31010,
INISYMD = CP31013,
MULVEC = CP31020,
DUPVEC = CP31030,
LINEMIN = CP34210,
DAVUPD = CP34212,
FLEUPD = CP34213.

REQUIRED CENTRAL MEMORY:

EXECUTION FIELD LENGTH: 3 * N + 19 WORDS.

RUNNING TIME:

DEPENDS STRONGLY ON THE PROBLEM TO BE SOLVED;

LANGUAGE: ALGOL 60.

METHOD AND PERFORMANCE:

FLEMIN CALCULATES AN APPROXIMATION OF A MINIMUM OF A GIVEN FUNCTION BY MEANS OF THE VARIABLE METRIC ALGORITHM GIVEN IN [3], EXCEPT FOR SOME DETAILS (SEE [1]).

REFERENCES:

[1] BUS, J. C. P.
MINIMIZATION OF FUNCTIONS OF SEVERAL VARIABLES (DUTCH).
MATHEMATICAL CENTRE, AMSTERDAM, NR 29/72 (1972).
[2] DAVIDON, W. C.
VARIABLE METRIC METHOD FOR MINIMIZATION.
ARGONNE NAT. LAB. REPORT, ANL 5990 (1959).
[3] FLETCHER, R.
A NEW APPROACH TO VARIABLE METRIC ALGORITHMS.
COMP. J. 6, (1963), P.163 - 168.
[4] GOLDSTEIN, A. A. AND PRICE, J. F.
AN EFFECTIVE ALGORITHM FOR MINIMIZATION.
NUMER. MATH. 10, (1967), P.184 - 189.
[5] GREENSTADT, J. L.
ON THE RELATIVE EFFICIENCIES OF GRADIENT METHODS.
MATH. COMP. 21, (1967), P.360 - 367.
[6] POWELL, M. J. D.
RANK ONE METHODS FOR UNCONSTRAINED OPTIMIZATION.
IN: ABADIE, J. (ED.)
INTEGER AND NONLINEAR PROGRAMMING.
NORTH - HOLLAND, (1970).

EXAMPLE OF USE:

```
    THE MINIMUM OF THE FUNCTION:
    F(X) = (X[2] - X[1] ** 2) ** 2 * 100 + (1 - X[1]) ** 2,
    CALCULATED WITH BOTH  RNK1MIN  AND  FLEMIN  MAY  BE OBTAINED BY THE
    FOLLOWING PROGRAM:

"BEGIN"
    "REAL" "PROCEDURE" RNK1MIN(N, X, G, H, FUNCT, IN, OUT);
    "CODE" 34214;
    "REAL" "PROCEDURE" FLEMIN(N, X, G, H, FUNCT, IN, OUT);
    "CODE" 34215;
    "REAL" "PROCEDURE" ROSENBROCK(N, X, G); "VALUE" N;
    "INTEGER" N; "ARRAY" X, G;
    "BEGIN" ROSENBROCK:= (X[2] - X[1] ** 2) ** 2 * 100
        + (1 - X[1]) ** 2;
        G[1]:= ((X[1] ** 2 - X[2]) * 400 + 2) * X[1] - 2;
        G[2]:=(X[2] - X[1] ** 2) * 200
    "END" ROSENBROCK;
    "INTEGER" I; "BOOLEAN" AGAIN; "REAL" F;
    "ARRAY" X, G[1:2], H[1:3], IN[0:8], OUT[0:4];

    IN[0]:= "-14; IN[1]:= "-5; IN[2]:= "-5; IN[3]:= "-4;
    IN[4]:= "-5; IN[5]:= -10; IN[6]:= 1; IN[7]:= 100; IN[8]:= 0.01;
    X[1]:= -1.2; X[2]:= 1; AGAIN:= "TRUE";
    F:= RNK1MIN(2, X, G, H, ROSENBROCK, IN, OUT);
    "GOTO" PRINT;
NEXT: X[1]:= -1.2; X[2]:= 1; AGAIN:= "FALSE";
    F:= FLEMIN(2, X, G, H, ROSENBROCK, IN, OUT);
PRINT: OUTPUT(61, "("""("LEAST VALUE:")"B+.15D"+3D,//, "("X:")",
    2(B+.15D"+3DB),//,"("GRADIENT:")", 2(B+.15D"+3DB),//, "("METRIC:")"
    ,2(B+.15D"+3DB),/,32B+.15D"+3D,//,"("OUT:")", 5(B+.15D"+3DB,/),///
    ")",F, X[1], X[2], G[1], G[2], H[1], H[2], H[3], OUT[0], OUT[1],
    OUT[2], OUT[3], OUT[4]);
    "IF" AGAIN "THEN" "GOTO" NEXT
"END"
        "EOP"
```

DELIVERS:

```
    LEAST VALUE: +.20069798801180"-018

    X: +.99999999944840"+000  +.99999999845220"+000

    GRADIENT: +.176731305145950"-007  -.889173179530190"-008

    METRIC: +.499982414863250"+000  +.999957383810230"+000
                                    +.200489757679290"+001

    OUT: +.164157123774660"-009
         +.197838933606480"-007
         +.550000000000000"+002
         +.800000000000000"+001
         +.400000000000000"+001


    LEAST VALUE: +.811973499921290"-016

    X: +.99999999758770"+000  +.99999999816780"+000

    GRADIENT: +.359826657359010"-006  -.180154557938290"-006

    METRIC: +.501085356975550"+000  +.100198139199600"+001
                                    +.200861655543510"+001

    OUT: +.133802289387830"-008
         +.402406371833370"-006
         +.440000000000000"+002
         +.700000000000000"+001
         +.000000000000000"+000
```

SOURCE TEXT(S):

```
"CODE" 34214;
    "REAL" "PROCEDURE" RNK1MIN(N, X, G, H, FUNCT, IN, OUT);
"VALUE" N;
"INTEGER" N; "ARRAY" X, G, H, IN, OUT;
"REAL" "PROCEDURE" FUNCT;
"BEGIN" "INTEGER" I, IT, N2, CNTL, CNTE, EVL, EVLMAX;
    "BOOLEAN" OK;
    "REAL" F, F0, FMIN, MU, DG, DG0, GHG, GS, NRMDELTA, ALFA,
    MACHEPS, RELTOL, ABSTOL, EPS, TOLG, ORTH, AID;
    "ARRAY" V, DELTA, GAMMA, S, P[1:N];
    "REAL" "PROCEDURE" VECVEC(L, U, SHIFT, A, B); "CODE" 34010;
    "REAL" "PROCEDURE" MATVEC(L, U, I, A, B); "CODE" 34011;
    "REAL" "PROCEDURE" TAMVEC(L, U, I, A, B); "CODE" 34012;
    "PROCEDURE" ELMVEC(L, U, SHIFT, A, B, X); "CODE" 34020;
    "REAL" "PROCEDURE" SYMMATVEC(L, U, I, A, B); "CODE" 34018;
    "PROCEDURE" INIVEC(L, U, A, X); "CODE" 31010;
    "PROCEDURE" INISYMD(LR, UR, SHIFT, A, X); "CODE" 31013;
    "PROCEDURE" MULVEC(L, U, SHIFT, A, B, X); "CODE" 31020;
    "PROCEDURE" DUPVEC(L, U, SHIFT, A, B); "CODE" 31030;
    "PROCEDURE" EIGSYM1(A, N, NUMVAL, VAL, VEC, EM); "CODE" 34156;
    "PROCEDURE" LINEMIN(N, X, D, ND, A, G, F, F0, F1, DF0, DF1,
    E, S, IN); "CODE" 34210;
    "PROCEDURE" RNK1UPD(H, N, V, C); "CODE" 34211;
    "PROCEDURE" DAVUPD(H, N, V, W, C1, C2); "CODE" 34212;
    "PROCEDURE" FLEUPD(H, N, V, W, C1, C2); "CODE" 34213;

    MACHEPS:= IN[0]; RELTOL:= IN[1]; ABSTOL:= IN[2];
    MU:= IN[3]; TOLG:= IN[4]; FMIN:= IN[5]; IT := 0;
    ALFA:= IN[6]; EVLMAX:= IN[7]; ORTH:= IN[8];
    N2:= N * (N + 1) // 2; CNTL:= CNTE:= 0; "IF" ALFA > 0 "THEN"
    "BEGIN" INIVEC(1, N2, H, 0); INISYMD(1, N, 0, H, ALFA) "END";
    F:= FUNCT(N, X, G); EVL:= 1; DG:= SQRT(VECVEC(1, N, 0, G, G));
    "FOR" I:= 1 "STEP" 1 "UNTIL" N "DO"
    DELTA[I]:= - SYMMATVEC(1, N, I, H, G);
    NRMDELTA:= SQRT(VECVEC(1, N, 0, DELTA, DELTA));
    DG0:= VECVEC(1, N, 0, DELTA, G); OK:= DG0 < 0;
    EPS:= SQRT(VECVEC(1, N, 0, X, X)) * RELTOL + ABSTOL;
    "FOR" IT:= IT + 1  "WHILE"
    (NRMDELTA > EPS "OR" DG > TOLG "OR" ^ OK) "AND" EVL < EVLMAX
    "DO"
    "BEGIN" "IF" ^OK "THEN"
        "BEGIN" "ARRAY" VEC[1:N,1:N], TH[1:N2], EM[0:9];
            EM[0]:= MACHEPS; EM[2]:= AID:= SQRT(MACHEPS * RELTOL);
            EM[4]:= ORTH; EM[6]:= AID * N; EM[8]:= 5;
            CNTE:= CNTE + 1; DUPVEC(1, N2, 0, TH, H);
            EIGSYM1(TH,N,N,V,VEC,EM);
            "FOR" I:= 1 "STEP" 1 "UNTIL" N "DO"
            "BEGIN" AID:= - TAMVEC(1, N, I, VEC, G);
                S[I]:= AID * ABS(V[I]); V[I]:= AID * SIGN(V[I])
            "END"
```

```
        "FOR" I:= 1 "STEP" 1 "UNTIL" N "DO"
        "BEGIN" DELTA[I]:= MATVEC(1, N, I, VEC, S);
            P[I]:= MATVEC(1, N, I, VEC, V)
        "END";
        DG0:= VECVEC(1, N, 0, DELTA, G);
            NRMDELTA:= SQRT(VECVEC(1, N, 0, DELTA, DELTA))
        "END" CALCULATING GREENSTADTS DIRECTION;
        DUPVEC(1, N, 0, S, X); DUPVEC(1, N, 0, V, G);
        "IF" IT > N "THEN" ALFA:= 1 "ELSE"
        "BEGIN" "IF" IT ^= 1 "THEN" ALFA:= ALFA / NRMDELTA "ELSE"
            "BEGIN" ALFA:= 2 * (FMIN - F) / DG0;
            "IF" ALFA > 1 "THEN" ALFA:= 1
            "END"
        "END";
        ELMVEC(1, N, 0, X, DELTA, ALFA);
        F0:= F; F:= FUNCT(N, X, G); EVL:= EVL +1 ;
        DG:= VECVEC(1, N, 0, DELTA, G);
        "IF" IT = 1 "OR" F0 - F < -MU * DG0 * ALFA "THEN"
        "BEGIN" I:= EVLMAX - EVL; CNTL:= CNTL +1 ;
            LINEMIN(N, S, DELTA, NRMDELTA, ALFA, G, FUNCT, F0, F,
            DG0, DG, I, "FALSE", IN); EVL:= EVL + I;
            DUPVEC(1, N, 0, X, S);
        "END" LINEMINIMIZATION;
        DUPVEC(1, N, 0, GAMMA, G); ELMVEC(1, N, 0, GAMMA, V, -1);
        "IF" ^ OK "THEN" MULVEC(1, N, 0, V, P, -1);
        DG:= DG - DG0; "IF" ALFA ^= 1 "THEN"
        "BEGIN" MULVEC(1, N, 0, DELTA, DELTA, ALFA);
            MULVEC(1, N, 0, V, V, ALFA);
            NRMDELTA:= NRMDELTA * ALFA; DG:= DG * ALFA
        "END";
        DUPVEC(1, N, 0, P, GAMMA); ELMVEC(1, N, 0, P, V, 1);
        "FOR" I:= 1 "STEP" 1 "UNTIL" N "DO"
        V[I]:= SYMMATVEC(1, N, I, H, GAMMA);
        DUPVEC(1, N, 0, S, DELTA); ELMVEC(1, N, 0, S, V, -1);
        GS:= VECVEC(1, N, 0, GAMMA, S);
        GHG:= VECVEC(1, N, 0, V, GAMMA);
        AID:= DG / GS;
        "IF" VECVEC(1, N, 0, DELTA, P) ** 2 > VECVEC(1, N, 0, P, P)
        * (ORTH * NRMDELTA) ** 2 "THEN" RNK1UPD(H, N, S, 1 / GS)
        "ELSE" "IF" AID >= 0 "THEN"
            FLEUPD(H, N, DELTA, V, 1 / DG, (1 + GHG / DG) / DG) "ELSE"
            DAVUPD(H, N, DELTA, V, 1 / DG, 1 / GHG);
            "FOR" I:= 1 "STEP" 1 "UNTIL" N "DO"
            DELTA[I]:= -SYMMATVEC(1, N, I, H, G);
            ALFA:= NRMDELTA;
            NRMDELTA:= SQRT(VECVEC(1, N, 0, DELTA, DELTA));
            EPS:= SQRT(VECVEC(1, N, 0, X, X)) * RELTOL + ABSTOL;
            DG:= SQRT(VECVEC(1, N, 0, G, G));
            DG0:= VECVEC(1, N, 0, DELTA, G); OK:= DG0 <= 0
        "END" ITERATION;
        OUT[0]:= NRMDELTA; OUT[1]:= DG; OUT[2]:= EVL;
        OUT[3]:= CNTL; OUT[4]:= CNTE; RNK1MIN:= F
    "END" RNK1MIN;
        "EOP"
```

```
"CODE" 34215;
    "REAL" "PROCEDURE" FLEMIN(N, X, G, H, FUNCT, IN, OUT);
    "VALUE" N;
"INTEGER" N; "ARRAY" X, G, H, IN, OUT;
"REAL" "PROCEDURE" FUNCT;
"BEGIN" "INTEGER" I, IT, CNTL, EVL, EVLMAX;
    "REAL" F,F0,FMIN,MU,DG,DG0,NRMDELTA,ALFA,RELTOL,ABSTOL,
    EPS, TOLG, AID;
    "ARRAY" V, DELTA, S[1:N];
    "REAL" "PROCEDURE" VECVEC(L, U, SHIFT, A, B); "CODE" 34010;
    "PROCEDURE" ELMVEC(L, U, SHIFT, A, B, X); "CODE" 34020;
    "REAL" "PROCEDURE" SYMMATVEC(L, U, I, A, B); "CODE" 34018;
    "PROCEDURE" INIVEC(L, U, A, X); "CODE" 31010;
    "PROCEDURE" INISYMD(LR, UR, SHIFT, A, X); "CODE" 31013;
    "PROCEDURE" MULVEC(L, U, SHIFT, A, B, XB); "CODE" 31020;
    "PROCEDURE" DUPVEC(L, U, SHIFT, A, B); "CODE" 31030;
    "PROCEDURE" LINEMIN(N, X, D, ND, A, G, F, F0, F1, DF0, DF1,
    E, S, IN); "CODE" 34210;
    "PROCEDURE" DAVUPD(H, N, V, W, C1, C2); "CODE" 34212;
    "PROCEDURE" FLEUPD(H, N, V, W, C1, C2); "CODE" 34213;
RELTOL:= IN[1]; ABSTOL:= IN[2]; MU:= IN[3];
TOLG:= IN[4]; FMIN:= IN[5]; ALFA:= IN[6];
EVLMAX:= IN[7]; OUT[4]:= 0; IT := 0;
F:= FUNCT(N, X, G); EVL:= 1; CNTL:= 0;"IF" ALFA > 0 "THEN"
"BEGIN" INIVEC(1, N * (N + 1) // 2, H, 0);
    INISYMD(1, N, 0, H, ALFA)
"END";
"FOR" I:= 1 "STEP" 1 "UNTIL" N "DO"
DELTA[I]:= - SYMMATVEC(1, N, I, H, G);
DG:= SQRT(VECVEC(1, N, 0, G, G));
NRMDELTA:= SQRT(VECVEC(1, N, 0, DELTA, DELTA));
EPS:= SQRT(VECVEC(1, N, 0, X, X)) * RELTOL + ABSTOL;
DG0:= VECVEC(1, N, 0, DELTA, G);"COMMENT"
```

```
;

"FOR" IT := IT +1  "WHILE"
(NRMDELTA > EPS "OR" DG > TOLG ) "AND" EVL < EVLMAX "DO"
"BEGIN" DUPVEC(1, N, 0, S, X); DUPVEC(1, N, 0, V, G);
    "IF" IT >= N "THEN" ALFA:= 1 "ELSE"
"BEGIN" "IF" IT ^= 1 "THEN" ALFA:= ALFA / NRMDELTA "ELSE"
    "BEGIN" ALFA:= 2 * (FMIN - F) / DG0;
        "IF" ALFA > 1 "THEN" ALFA:= 1
    "END"
"END";
ELMVEC(1, N, 0, X, DELTA, ALFA);
F0:= F; F:= FUNCT(N, X, G); EVL:= EVL +1 ;
DG:= VECVEC(1, N, 0, DELTA, G);
"IF" IT = 1 "OR" F0 - F < - MU * DG0 * ALFA "THEN"
"BEGIN" I:= EVLMAX - EVL; CNTL:= CNTL +1 ;
    LINEMIN(N, S, DELTA, NRMDELTA, ALFA, G, FUNCT, F0, F,
    DG0, DG, I, "FALSE", IN); EVL:= EVL + I;
    DUPVEC(1, N, 0, X, S);
"END" LINEMINIMIZATION;
"IF" ALFA ^= 1 "THEN" MULVEC(1, N, 0, DELTA, DELTA, ALFA);
MULVEC(1, N, 0, V, V, -1); ELMVEC(1, N, 0, V, G, 1);
"FOR" I:= 1 "STEP" 1 "UNTIL" N "DO"
        S[I]:= SYMMATVEC(1, N, I, H, V);
        AID:= VECVEC(1, N, 0, V, S); DG:= (DG - DG0) * ALFA;
        "IF" DG > 0 "THEN"
        "BEGIN" "IF" DG >= AID "THEN"
            FLEUPD(H, N, DELTA, S, 1 / DG, (1 + AID / DG) / DG)
            "ELSE" DAVUPD(H, N, DELTA, S, 1 / DG, 1 / AID)
        "END" UPDATING;
        "FOR" I:= 1 "STEP" 1 "UNTIL" N "DO"
        DELTA[I]:= -SYMMATVEC(1, N, I, H, G);
        ALFA:= NRMDELTA * ALFA;
        NRMDELTA:= SQRT(VECVEC(1, N, 0, DELTA, DELTA));
        EPS:= SQRT(VECVEC(1, N, 0, X, X)) * RELTOL + ABSTOL;
        DG:= SQRT(VECVEC(1, N, 0, G, G));
        DG0:= VECVEC(1, N, 0, DELTA, G); "IF" DG0 > 0 "THEN"
        "BEGIN" OUT[4]:= -1 ; "GOTO" EXIT "END"
    "END" ITERATION;
EXIT: OUT[0]:= NRMDELTA; OUT[1]:= DG; OUT[2]:= EVL;
    OUT[3]:= CNTL; FLEMIN:= F
"END" FLEMIN;
    "EOP"
```

AUTHORS: J.C.P. BUS, B. VAN DOMSELAAR, J. KOK.

CONTRIBUTORS: B. VAN DOMSELAAR, J. KOK.

INSTITUTE: MATHEMATICAL CENTRE.

RECEIVED: 741101.

BRIEF DESCRIPTION:

THIS SECTION CONTAINS TWO PROCEDURES;
MARQUARDT CALCULATES THE LEAST SQUARES SOLUTION OF AN OVERDETERMINED SYSTEM OF NONLINEAR EQUATIONS WITH MARQUARDT'S METHOD;
GSSNEWTON CALCULATES THE LEAST SQUARES SOLUTION OF AN OVERDETERMINED SYSTEM OF NONLINEAR EQUATIONS WITH THE GAUSS-NEWTON METHOD;
THE USER HAS TO PROGRAM THE EVALUATION OF THE RESIDUAL VECTOR OF THE SYSTEM AND THE JACOBIAN MATRIX OF ITS PARTIAL DERIVATIVES.

KEYWORDS:

NONLINEAR EQUATIONS,
LEAST SQUARES SOLUTION,
OVERDETERMINED SYSTEM,
MARQUARDT'S METHOD,
GAUSS-NEWTON METHOD,
CURVE FITTING.

SUBSECTION: MARQUARDT.

CALLING SEQUENCE:

```
THE HEADING OF THIS PROCEDURE IS:
"PROCEDURE" MARQUARDT(M, N, PAR, RV, JJINV, FUNCT, JACOBIAN, IN,
   OUT); "VALUE" M, N; "INTEGER" M, N;
"ARRAY" PAR, RV, JJINV, IN, OUT; "BOOLEAN" "PROCEDURE" FUNCT;
"PROCEDURE" JACOBIAN;
"CODE" 34440;

THE MEANING OF THE FORMAL PARAMETERS IS:
M:          <ARITHMETIC EXPRESSION>;
            THE NUMBER OF EQUATIONS;
N:          <ARITHMETIC EXPRESSION>;
            THE NUMBER OF UNKNOWN VARIABLES; N SHOULD SATISFY N<=M;
PAR:        <ARRAY IDENTIFIER>;
            "ARRAY" PAR[1 : N];
            THE UNKNOWN VARIABLES OF THE SYSTEM;
            ENTRY: AN APPROXIMATION TO A LEAST SQUARES SOLUTION
                   OF THE SYSTEM;
            EXIT:  THE CALCULATED LEAST SQUARES SOLUTION;
RV:         <ARRAY IDENTIFIER>;
            "ARRAY" RV[1 : M];
            EXIT:  THE RESIDUAL VECTOR AT  THE CALCULATED SOLUTION;
JJINV:      <ARRAY IDENTIFIER>;
            "ARRAY" JJINV[1 : N, 1 : N];
            EXIT:  THE INVERSE OF THE MATRIX  J' * J  WHERE J DENOTES
                   THE MATRIX OF PARTIAL DERIVATIVES DRV[I] / DPAR[J]
                   (I=1,...,M; J=1,...,N) AND  J'  DENOTES THE
                   TRANSPOSE OF J.
FUNCT:      <PROCEDURE IDENTIFIER>;
            THE HEADING OF THIS PROCEDURE SHOULD BE:
            "BOOLEAN" "PROCEDURE" FUNCT(M, N, PAR, RV); "VALUE" M, N;
            "INTEGER" M, N; "ARRAY" PAR, RV;
            ENTRY: M, N, PAR;
                   M, N  HAVE THE SAME MEANING AS IN THE PROCEDURE
                   MARQUARDT;
                   "ARRAY" PAR[1:N] CONTAINS THE CURRENT VALUES OF
                   THE UNKNOWNS AND SHOULD NOT BE ALTERED;
            EXIT:  "ARRAY" RV[1 : M];
                   UPON COMPLETION OF A CALL OF FUNCT, THIS ARRAY  RV
                   SHOULD CONTAIN THE RESIDUAL VECTOR, OBTAINED WITH
                   THE CURRENT VALUES OF THE UNKNOWNS;
                   E.G. IN CURVE FITTING PROBLEMS:
                   RV[I] := THEORETICAL VALUE  F(X[I], PAR) -
                            OBSERVED VALUE  Y[I];
AFTER A SUCCESSFUL CALL OF FUNCT, THE BOOLEAN PROCEDURE
SHOULD DELIVER THE VALUE TRUE;
HOWEVER, IF FUNCT DELIVERS THE VALUE FALSE, THEN IT IS
ASSUMED THAT THE CURRENT ESTIMATES OF THE UNKNOWNS LIE
OUTSIDE A FEASIBLE REGION AND THE PROCESS IS TERMINATED
(SEE OUT[1]);
```

HENCE, PROPER PROGRAMMING OF FUNCT MAKES IT POSSIBLE TO AVOID CALCULATION OF A RESIDUAL VECTOR WITH VALUES OF THE UNKNOWN VARIABLES WHICH MAKE NO SENSE OR WHICH EVEN MAY CAUSE OVERFLOW IN THE COMPUTATION;

JACOBIAN: <PROCEDURE IDENTIFIER>;
THE HEADING OF THIS PROCEDURE SHOULD BE:
"PROCEDURE" JACOBIAN(M, N, PAR, RV, JAC, LOCFUNCT);
"VALUE" M, N; "INTEGER" M, N; "ARRAY" PAR, RV, JAC;
"PROCEDURE" LOCFUNCT;
ENTRY: M, N, PAR, RV, LOCFUNCT;
FOR M,N,PAR SEE: FUNCT;
RV CONTAINS THE RESIDUAL VECTOR OBTAINED WITH THE CURRENT VALUES OF THE UNKNOWNS AND SHOULD NOT BE ALTERED;
A CALL OF LOCFUNCT(M,N,PAR,RV) IS EQUIVALENT WITH A CALL OF THE USER-DEFINED PROCEDURE FUNCT(M,N,PAR,RV), BUT, IN ADDITION, THIS CALL IS COUNTED TO THE TOTAL NUMBER OF CALLS OF FUNCT (SEE OUT[4]) AND, MOREOVER, IF FUNCT DELIVERS THE VALUE FALSE THEN THE PROCESS IS TERMINATED;
EXIT: "ARRAY" JAC[1 : M, 1 : N];
UPON COMPLETION OF A CALL OF JACOBIAN, JAC SHOULD CONTAIN THE PARTIAL DERIVATIVES DRV[I] / DPAR[J], OBTAINED WITH THE CURRENT VALUES OF THE UNKNOWN VARIABLES GIVEN IN PAR[1:N];
IT IS A PREREQUISITE FOR THE PROPER OPERATION OF THE PROCEDURE MARQUARDT THAT THE PRECISION OF THE ELEMENTS OF THE MATRIX JAC IS AT LEAST THE PRECISION DEFINED BY IN[3] AND IN[4];

IN: <ARRAY IDENTIFIER>;
"ARRAY" IN[0 : 6];
ENTRY: IN THIS ARRAY THE USER SHOULD GIVE SOME DATA TO CONTROL THE PROCESS;
IN[0]: THE MACHINE PRECISION;
FOR THE CYBER 73 A SUITABLE VALUE IS "-14;
IN[1], IN[2] ARE NOT USED BY THE PROCEDURE MARQUARDT;
IN[3], IN[4]:
THE RELATIVE AND ABSOLUTE TOLERANCE FOR THE DIFFERENCE BETWEEN THE EUCLIDEAN NORM OF THE ULTIMATE AND PENULTIMATE RESIDUAL VECTOR;
THE PROCESS IS TERMINATED IF THE IMPROVEMENT OF THE SUM OF SQUARES IS LESS THAN
IN[3] * (SUM OF SQUARES) + IN[4] * IN[4];
THESE TOLERANCES SHOULD BE CHOSEN GREATER THAN THE CORRESPONDING ERRORS OF THE CALCULATED RESIDUAL VECTOR;
NOTE THAT THE EUCLIDEAN NORM OF THE RESIDUAL VECTOR IS DEFINED AS THE SQUARE ROOT OF THE SUM OF SQUARES;
IN[5]: THE MAXIMUM NUMBER OF CALLS OF FUNCT ALLOWED;

IN[6]: A STARTING VALUE USED FOR THE RELATION BETWEEN THE GRADIENT AND THE GAUSS-NEWTON DIRECTION (SEE [2]); IF THE PROBLEM IS WELL CONDITIONED THEN A SUITABLE VALUE FOR IN[6] WILL BE 0.01; IF THE PROBLEM IS ILL CONDITIONED THEN IN[6] SHOULD BE GREATER, BUT THE VALUE OF IN[6] SHOULD SATISFY: IN[0] < IN[6] <= 1/IN[0];

OUT: <ARRAY IDENTIFIER>; "ARRAY" OUT[1 : 7];
EXIT : IN ARRAY OUT SOME BY-PRODUCTS ARE DELIVERED;
OUT[1]: THIS VALUE GIVES INFORMATION ABOUT THE TERMINATION OF THE PROCESS;
OUT[1]=0: NORMAL TERMINATION;
OUT[1]=1: THE PROCESS HAS BEEN BROKEN OFF, BECAUSE THE NUMBER OF CALLS OF FUNCT EXCEEDED THE NUMBER GIVEN IN IN[5];
OUT[1]=2: THE PROCESS HAS BEEN BROKEN OFF, BECAJSE A CALL OF FUNCT DELIVERED THE VALUE FALSE;
OUT[1]=3: FUNCT BECAME FALSE WHEN CALLED WITH THE INITIAL ESTIMATES OF PAR[1:N]; THE ITERATION PROCESS WAS NOT STARTED AND SO JJINV[1:N,1:N] CAN NOT BE USED;
OUT[1]=4: THE PROCESS HAS BEEN BROKEN OFF, BECAUSE THE PRECISION ASKED FOR CAN NOT BE ATTAINED; THIS PRECISION IS POSSIBLY CHOSEN TOO HIGH, RELATIVE TO THE PRECISION IN WHICH THE RESIDUAL VECTOR IS CALCULATED (SEE IN[3]);
OUT[2]: THE EUCLIDEAN NORM OF THE RESIDUAL VECTOR CALCULATED WITH VALUES OF THE UNKNOWNS DELIVERED;
OUT[3]: THE EUCLIDEAN NORM OF THE RESIDUAL VECTOR CALCULATED WITH THE INITIAL VALUES OF THE UNKNOWN VARIABLES;
OUT[4]: THE NUMBER OF CALLS OF FUNCT NECESSARY TO OBTAIN THE CALCULATED RESULT;
OUT[5]: THE TOTAL NUMBER OF ITERATIONS PERFORMED; NOTE THAT IN EACH ITERATION ONE EVALUATION OF THE JACOBIAN MATRIX HAD TO BE MADE;
OUT[6]: THE IMPROVEMENT OF THE EUCLIDEAN NORM OF THE RESIDUAL VECTOR IN THE LAST ITERATION STEP;
OUT[7]: THE CONDITION NUMBER OF J' * J , I.E. THE RATIO OF ITS LARGEST TO SMALLEST EIGENVALUES;

DATA AND RESULTS:

IF THIS PROCEDURE IS USED FOR CURVE FITTING THEN THE RELATIVE ACCURACY IN THE CALCULATION OF THE RESIDUAL VECTOR DEPENDS STRONGLY ON THE ERRORS IN THE EXPERIMENTAL DATA AND THIS SHOULD BE REFLECTED IN THE PARAMETERS IN[3] AND IN[4];
THE MATRIX JJINV CAN BE USED IF SOME STATISTICAL INFORMATION ABOUT THE FITTED PARAMETERS IS REQUIRED; THE STANDARD DEVIATION, COVARIANCE MATRIX AND CORRELATION MATRIX MAY BE CALCULATED EASILY FROM JJINV ;

PROCEDURES USED:

MULCOL = CP31022,
DUPVEC = CP31030,
VECVEC = CP34010,
MATVEC = CP34011,
TAMVEC = CP34012,
MATTAM = CP34015,
QRISNGVALDEC = CP34273.

REQUIRED CENTRAL MEMORY:

EXECUTION FIELD LENGTH: ONE ARRAY OF LENGTH M * N AND FOUR ARRAYS OF LENGTH N ARE DECLARED;

RUNNING TIME:

THE NUMBER OF ITERATION STEPS DEPENDS STRONGLY ON THE PROBLEM TO BE SOLVED; HOWEVER, THE WORK DONE EACH ITERATION STEP IS PROPORTIONAL TO N CUBED;

LANGUAGE: ALGOL 60.

METHOD AND PERFORMANCE:

MARQUARDT USES MARQUARDT'S METHOD; FOR DETAILS SEE [2];

REFERENCES:

[1] D. W. MARQUARDT,
AN ALGORITHM FOR LEAST-SQUARES ESTIMATION OF NONLINEAR PARAMETERS,
J. SIAM 11 (1963), PP.431-441.

[2] J. C. P. BUS, B. VAN DOMSELAAR, J. KOK,
NONLINEAR LEAST SQUARES ESTIMATION,
MATHEMATICAL CENTRE, AMSTERDAM. (TO APPEAR)

EXAMPLE OF USE:

THE PARAMETERS PAR[1 : 3] IN THE CURVE FITTING PROBLEM:
RV[I] = PAR[1] + PAR[2] * EXP(PAR[3] * X[I]) - Y[I]
WITH (X[I], Y[I]), I=1,...,6 AS THE EXPERIMENTAL DATA, MAY BE
OBTAINED BY THE FOLLOWING PROGRAM:

```
"BEGIN" "PROCEDURE" MARQUARDT(M,N,P,RV,JJINV,F,J,I,O); "CODE"34440;
   "INTEGER" I;
   "ARRAY" IN[0:6],OUT[0:7],X,Y,RV[1:6],JJINV[1:3,1:3],PAR[1:3];

   "BOOLEAN" "PROCEDURE" EXPFUNCT(M,N,PAR,RV);
       "VALUE" M,N; "INTEGER" M,N; "ARRAY" PAR,RV;
       "BEGIN" "INTEGER" I;
               "FOR" I:=1 "STEP" 1 "JNTIL" M "DO"
               "BEGIN" "IF" PAR[3]*X[I]>680 "THEN"
                       "BEGIN" EXPFUNCT:="FALSE";
                               "GOTO" STOP
                       "END";
                       RV[I]:=PAR[1]+PAR[2]*EXP(PAR[3]*X[I])-Y[I]
               "END"; EXPFUNCT:="TRUE";
         STOP:
   "END" EXPFUNCT;

"PROCEDURE" JACOBIAN(M,N,PAR,RV,JAC,LOCFUNCT);
   "VALUE" M,N; "INTEGER" M,N; "ARRAY" PAR,RV,JAC;
   "PROCEDURE" LOCFUNCT;
   "BEGIN" "INTEGER" I; "REAL" EX;
           "FOR" I:=1 "STEP" 1 "UNTIL" M "DO"
           "BEGIN" JAC[I,1]:=1;
                   JAC[I,2]:=EX:=EXP(PAR[3]*X[I]);
                   JAC[I,3]:=X[I]*PAR[2]*EX
           "END"
   "END" JACOBIAN;

IN[0]:="-14; IN[3]:="-4; IN[4]:="-1; IN[5]:=75; IN[6]:="-2;
"FOR" I:=1 "STEP" 1 "UNTIL" 6 "DO"
INPUT(60,"("2(N),/")",X[I],Y[I]);
   PAR[1]:=580; PAR[2]:=-180; PAR[3]:=-0.160;
   MARQUARDT(6,3,PAR,RV,JJINV,EXPFUNCT,JACOBIAN,IN,OUT);
   OUTPUT(61,"("3/,"("X[I]      Y[I]")",/,6(B,+D,5B,3D.D,/),2/,
   "("PARAMETERS")",/,3(+D.3D"+ZD,/),2/,"("OUT:")",/,7(5B+D.6D"+ZD,
   /),2/,"("LAST RESIDUAL VECTOR")",/,6(6B+3Z.D,/)")",X[1],Y[1],
   X[2],Y[2],X[3],Y[3],X[4],Y[4],X[5],Y[5],X[6],Y[6],PAR[1],PAR[2],
   PAR[3],OUT[7],OUT[2],OUT[6],OUT[3],OUT[4],OUT[5],OUT[1],RV[1],
   RV[2],RV[3],RV[4],RV[5],RV[6])
"END"
    "EOP"
```

WITH THE DATA GIVEN IN THE X - Y - TABLE THIS PROGRAM DELIVERS:

X[I]	Y[I]
-5	127.0
-3	151.0
-1	379.0
+1	421.0
+3	460.0
+5	426.0

PARAMETERS
+5.232" +2
-1.568" +2
-1.998" -1

OUT:
+7.220828" +7
+1.157156" +2
+1.728008" -3
+1.654588" +2
+2.300000" +1
+2.200000" +1
+0.000000" +0

LAST RESIDUAL VECTOR
-29.6
+86.5
-47.3
-26.2
-22.9
+39.5

SUBSECTION : GSSNEWTON.

CALLING SEQUENCE :

THE HEADING OF THE PROCEDURE READS :

```
"PROCEDURE" GSSNEWTON(M, N, PAR, RV, JJINV, FUNCT, JACOBIAN, IN,
    OUT);
"VALUE" M, N; "INTEGER" M, N; "ARRAY" PAR, RV, JJINV, IN, OUT;
"BOOLEAN""PROCEDURE" FUNCT; "PROCEDURE" JACOBIAN;
"CODE" 34441;
```

THE MEANING OF THE FORMAL PARAMETERS IS :

M : <ARITHMETIC EXPRESSION>;
THE NUMBER OF EQUATIONS;

N : <ARITHMETIC EXPRESSION>;
THE NUMBER OF UNKNOWNS IN THE M EQUATIONS (N <= M);

PAR : <ARRAY IDENTIFIER>; "ARRAY" PAR[1 : N];
THE UNKNOWNS OF THE EQUATIONS.
ENTRY : AN APPROXIMATION TO A LEAST SQUARES SOLUTION
OF THE SYSTEM.
EXIT : THE CALULATED LEAST SQUARES SOLUTION;

RV : <ARRAY IDENTIFIER>; "ARRAY" RV[1 : M];
EXIT : THE RESIDUAL VECTOR OF THE SYSTEM AT THE
CALCULATED SOLUTION;

JJINV : <ARRAY IDENTIFIER>; "ARRAY" JJINV[1 : N,1 : N];
EXIT : THE INVERSE OF THE MATRIX J' * J, WHERE J
IS THE JACOBIAN MATRIX AT THE SOLUTION AND J' IS
J TRANSPOSED;

FUNCT : <PROCEDURE IDENTIFIER>;
THE HEADING OF THIS PROCEDURE SHOULD BE :

```
"BOOLEAN""PROCEDURE" FUNCT(M, N, PAR, RV); "VALUE" M,
N; "INTEGER" M, N; "ARRAY" PAR, RV;
```

ENTRY: M, N, PAR;
M, N HAVE THE SAME MEANING AS IN THE PROCEDURE
GSSNEWTON;
"ARRAY" PAR[1:N] CONTAINS THE CURRENT VALUES OF
THE UNKNOWNS AND SHOULD NOT BE ALTERED.

EXIT: "ARRAY" RV[1 : M];
UPON COMPLETION OF A CALL OF FUNCT, THIS ARRAY RV
SHOULD CONTAIN THE RESIDUAL VECTOR, OBTAINED WITH
THE CURRENT VALUES OF THE UNKNOWNS.

THE PROGRAMMER OF FUNCT MAY DECIDE THAT SOME CURRENT ESTIMATES OF THE UNKNOWNS LIE OUTSIDE A FEASIBLE REGION; IN THIS CASE FUNCT SHOULD DELIVER THE VALUE FALSE AND THE PROCESS IS TERMINATED (SEE OUT[1]). OTHERWISE FUNCT SHOJLD DELIVER THE VALUE TRUE;

JACOBIAN : <PROCEDURE IDENTIFIER>;
THE HEADING OF THIS PROCEDURE SHOULD BE :

"PROCEDURE" JACOBIAN(M, N, PAR, RV, JAC, LOCFUNCT);
"VALUE" M, N; "INTEGER" M, N; "ARRAY" PAR, RV, JAC;
"PROCEDURE" LOCFUNCT;

THE MEANING OF THE PARAMETERS OF JACOBIAN IS :

M, N : SEE GSSNEWTON.
PAR : <ARRAY IDENTIFIER>; "ARRAY" PAR[1 : N];
ENTRY : CURRENT ESTIMATES OF THE UNKNOWNS.
THESE VALUES SHOULD NOT BE CHANGED.
RV : <ARRAY IDENTIFIER>; "ARRAY" RV[1 : M];
ENTRY : THE RESIDUAL VECTOR OF THE SYSTEM OF
EQUATIONS CORRESPONDING TO THE VECTOR OF UNKNOWNS
AS GIVEN IN PAR.
EXIT : THE ENTRY VALUES.
JAC : <ARRAY IDENTIFIER>; "ARRAY" JAC[1 : M, 1 : N];
EXIT : THE JACOBIAN MATRIX AT THE CURRENT
ESTIMATES GIVEN IN PAR, I.E. THE MATRIX OF PARTIAL
DERIVATIVES
D(RV)[I] / DPAR[J], I = 1(1)M, J = 1(1)N.
LOCFUNCT : <PROCEDURE IDENTIFIER>; THE HEADING OF THIS
PROCEDURE IS THE SAME AS THE HEADING OF FUNCT.

A CALL OF THE PROCEDURE JACOBIAN SHOULD DELIVER THE
JACOBIAN MATRIX EVALUATED WITH THE CURRENT ESTIMATES
OF THE UNKNOWN VARIABLES GIVEN IN PAR
IN SUCH A WAY, THAT THE PARTIAL DERIVATIVE
D(RV)[I] / DPAR[J] IS DELIVERED IN JAC[I,J], I = 1(1)M,
J = 1(1)N.
FOR THE CALCULATION OF THE DERIVATIVES ONE CAN USE THE
VALUES OF THE CURRENT ESTIMATES OF THE
UNKNOWNS AS GIVEN IN PAR AND THE RESIDUAL VECTOR AS
GIVEN IN RV.
ONE CAN ALSO USE THE PROCEDURE FUNCT
(PARAMETER OF GSSNEWTON) THROUGH CALLS OF THE PROCEDURE
LOCFUNCT (PARAMETER OF JACOBIAN). THIS PARAMETER OF
JACOBIAN MAY BE USED WHEN THE JACOBIAN MATRIX IS
APPROXIMATED USING (FORWARD) DIFFERENCES.
AN APPROPRIATE PROCEDURE TO THIS PURPOSE IS JACOBNMF
(SECTION 4.3.2.1). SUCH A PROCEDURE MAY BE USED ONLY IF
THE MATRIX ELEMENTS ARE COMPUTED SUFFICIENTLY ACCURATE;

IN : <ARRAY IDENTIFIER>; "ARRAY" IN[0 : 7];
IN THIS ARRAY TOLERANCES AND CONTROL PARAMETERS SHOULD
BE GIVEN.
ENTRY :
IN[0] : THE MACHINE PRECISION. FOR CALCULATION ON THE
CYBER 73 A SUITABLE VALUE IS "-14.

IN[1], IN[2] :
RELATIVE AND ABSOLUTE TOLERANCE FOR THE STEP VECTOR (RELATIVE TO THE VECTOR OF CURRENT ESTIMATES IN PAR).
THE PROCESS IS TERMINATED IF IN SOME ITERATION (BUT NOT THE FIRST) THE EUCLIDEAN NORM OF THE CALCULATED NEWTON STEP IS LESS THAN IN[1] * NORM(PAR) + IN[2]. IN[1] SHOULD NOT BE CHOSEN SMALLER THAN IN[0].
IN[3] IS NOT USED BY THE PROCEDURE GSSNEWTON;
IN[4] : ABSOLUTE TOLERANCE FOR THE EUCLIDEAN NORM OF THE RESIDUAL VECTOR. THE PROCESS IS TERMINATED WHEN THIS NORM IS LESS THAN IN[4].
IN[5] : THE MAXIMUM ALLOWED NUMBER OF FUNCTION EVALUATIONS (I.E. CALLS OF FUNCT).
IN[6] : THE MAXIMUM ALLOWED NUMBER OF HALVINGS OF A CALCULATED NEWTON STEP VECTOR (SEE METHOD AND PERFORMANCE). A SUITABLE VALUE IS 15.
IN[7] : THE MAXIMUM ALLOWED NUMBER OF SUCCESSIVE IN[6] TIMES HALVED STEP VECTORS. SUITABLE VALUES ARE 1 AND 2;

OUT : <ARRAY IDENTIFIER>; "ARRAY" OUT[1 : 9];
IN ARRAY OUT INFORMATION ABOUT THE TERMINATION OF THE PROCESS IS DELIVERED.

EXIT :

OUT[1] :
THE PROCESS WAS TERMINATED BECAUSE (OUT[1] =)
1. THE NORM OF THE RESIDUAL VECTOR IS SMALL WITH RESPECT TO IN[4],
2. THE CALCULATED NEWTON STEP IS SUFFICIENTLY SMALL (SEE IN[1], IN[2]),
3. THE CALCULATED STEP WAS COMPLETELY DAMPED (HALVED) IN IN[7] SUCCESSIVE ITERATIONS,
4. OUT[4] EXCEEDS IN[5], THE MAXIMUM ALLOWED NUMBER OF CALLS OF FUNCT,
5. THE JACOBIAN WAS NOT FULL-RANK (SEE OUT[8]),
6. FUNCT DELIVERED FALSE AT A NEW VECTOR OF ESTIMATES OF THE UNKNOWNS,
7. FUNCT DELIVERED FALSE IN A CALL FROM JACOBIAN.

OUT[2] : THE EUCLIDEAN NORM OF THE LAST RESIDUAL VECTOR.
OUT[3] : THE EUCLIDEAN NORM OF THE INITIAL RESIDUAL VECTOR.
OUT[4] : THE TOTAL NUMBER OF CALLS OF FUNCT.
OUT[4] WILL BE LESS THAN IN[5] + IN[6].
OUT[5] : THE TOTAL NUMBER OF ITERATIONS.
OUT[6] : THE EUCLIDEAN NORM OF THE LAST STEP VECTOR.
OUT[7] : ITERATION NUMBER OF THE LAST ITERATION IN WHICH THE NEWTON STEP WAS HALVED.
OUT[8], OUT[9] :
RANK AND MAXIMUM COLUMN NORM OF THE JACOBIAN MATRIX IN THE LAST ITERATION, AS DELIVERED BY LSQORTDEC (SECTION 3.1.1.2.1.1) IN AUX[3] AND AUX[5].

DATA AND RESULTS :

THE PROCEDURE GSSNEWTON CAN BE USED FOR APPROXIMATING AN EXACT OR A LEAST SQUARES SOLUTION OF A SYSTEM OF NONLINEAR EQUATIONS. WHEN AN EXACT SOLUTION IS REQUIRED, THE PROCEDURE MAY TERMINATE ONLY WITH OUT[1] = 1, AND VERY SMALL VALUES SHOULD BE ASSIGNED TO IN[1] AND IN[2]. WHEN A LEAST SQUARES SOLUTION IS REQUIRED, POSITIVE RESULTS OF THE PROCEDURE ARE SIGNALED BY OUT[1] = 1 OR 2. WHENEVER THE PROCEDURE TERMINATES WITH OUT[1] < 5, THEN THE INVERSE OF J' * J (SEE MEANING OF THE PARAMETER JJINV) IS DELIVERED IN JJINV. IN THAT CASE THE COVARIANCE MATRIX AND THE STANDARD DEVIATIONS OF THE SOLUTION CAN BE CALCULATED.

FOR A CURVE FITTING PROBLEM, SAY :
"ESTIMATE PARAMETERS PAR[1], ... , PAR[N] OF A FUNCTION
"Y = F(X; PAR[1], ... , PAR[N]), WHEN A SET OF DATA (X[I],Y[I]),
"I = 1(1)M, HAS TO BE FITTED,"
THE FOLLOWING SYSTEM OF M EQUATIONS IN THE N UNKNOWN PARAMETERS PAR[1], ... , PAR[N] CAN BE DERIVED :

F(X[I]; PAR[1], ... , PAR[N]) - Y[I] = 0, I = 1(1)M.

PROCEDURES USED :

VECVEC = CP34010,
DUPVEC = CP31030,
ELMVEC = CP34020,
LSQORTDEC = CP34134,
LSQSOL = CP34131,
LSQINV = CP34136.

REQUIRED CENTRAL MEMORY :
EXECUTION FIELD LENGTH : AN ARRAY OF (M + 1) * N ELEMENTS, FOUR ARRAYS OF N ELEMENTS AND ONE ARRAY OF M ELEMENTS ARE DECLARED.

RUNNING TIME :
THE RUNNING TIME IS PROPORTIONAL TO THE TOTAL NUMBER OF CALLS OF FUNCT. BESIDES, IN EACH ITERATION A LINEAR LEAST SQUARES SYSTEM IS SOLVED (NUMBER OF OPERATIONS PROPORTIONAL TO M * (N SQUARED)).

LANGUAGE : ALGOL 60.

METHOD AND PERFORMANCE :
THE PROCEDURE GSSNEWTON IS BASED UPON THE GAUSS-NEWTON METHOD FOR CALCULATING A LEAST SQUARES SOLUTION OF A SYSTEM OF NONLINEAR EQUATIONS (SEE E.G. [1], [2]). IN SEVERAL ITERATIONS A STEP VECTOR (FOR THE VECTOR OF UNKNOWNS) IS OBTAINED BY SOLVING A LINEARIZED SYSTEM OF EQUATIONS. THIS STEP IS PERFORMED ONLY IF THE NORM OF THE RESIDUAL VECTOR DECREASES. OTHERWISE THE NEWTON STEP VECTOR IS HALVED A NUMBER OF TIMES UNTIL THE NORM OF THE RESIDUAL VECTOR IS SMALLER THAN THE NORMS OF THE LAST AND SUBSEQUENT RESIDUAL VECTORS (THIS PROCESS IS CALLED STEP SIZE CONTROL).
FOR FURTHER DETAILS SEE [3] (SEE ALSO [2]).

REFERENCES :

[1] H.O. HARTLEY :
THE MODIFIED GAUSS-NEWTON METHOD.
TECHNOMETRICS, V.3 (1961), PP. 269 - 280.

[2] H. SPAETH :
THE DAMPED TAYLOR'S SERIES METHOD FOR MINIMIZING A SUM OF SQUARES AND FOR SOLVING SYSTEMS OF NONLINEAR EQUATIONS.
ALGORITHM 315, COLLECTED ALGORITHMS FROM CACM,
COMMUNICATIONS OF THE ACM, VOL. 10 (NOV. 1967), PP. 726 - 728.

[3] J.C.P. BUS, B. VAN DOMSELAAR, J. KOK :
NONLINEAR LEAST SQUARES ESTIMATION.
MATHEMATICAL CENTRE (TO APPEAR).

EXAMPLE OF USE :

THE PARAMETERS PAR[1 : 3] IN THE CURVE FITTING PROBLEM:
G[I] = PAR[1] + PAR[2] * EXP(PAR[3] * X[I]) - Y[I]
WITH (X[I], Y[I]), I=1,...,6 AS THE EXPERIMENTAL DATA, MAY BE OBTAINED BY THE FOLLOWING PROGRAM:

```
"BEGIN""PROCEDURE" GSSNEWTON(M, N, P, F, C, TF, JAC, I, O);
        "CODE" 34441;
    "INTEGER" I;
    "ARRAY" IN[0:7], OUT[1:9], X, Y, G[1:6], V[1:3, 1:3], PAR[1:3];

    "BOOLEAN""PROCEDURE" EXPFUNCT(M, N, PAR, G);
    "VALUE" M, N; "INTEGER" M, N; "ARRAY" PAR, G;
    "BEGIN""INTEGER" I;
        "FOR" I:= 1 "STEP" 1 "UNTIL" M "DO"
        "BEGIN""IF" PAR[3] * X[I] > 680 "THEN"
            "BEGIN" EXPFUNCT:= "FALSE"; "GO TO" STOP "END";
            G[I]:= PAR[1] + PAR[2] * EXP(PAR[3] * X[I]) - Y[I]
        "END"; EXPFUNCT:="TRUE";
    STOP:
    "END" EXPFUNCT;

"PROCEDURE" JACOBIAN(M, N, PAR, G, JAC, LOCFUNCT);
"VALUE" M, N; "INTEGER" M, N; "ARRAY" PAR, G, JAC;
"PROCEDURE" LOCFUNCT;
"BEGIN" "INTEGER" I; "REAL" EX;
    "FOR" I:= 1 "STEP" 1 "UNTIL" M "DO"
    "BEGIN" JAC[I,1]:=1; JAC[I,2]:= EX:= EXP(PAR[3] * X[I]);
        JAC[I,3]:= X[I] * PAR[2] * EX
    "END"
"END" JACOBIAN;
```

```
      IN[0]:= "-14; IN[1]:= IN[2]:= "-6; IN[5]:= 75; IN[4]:="-10;
      IN[6]:= 14; IN[7]:= 1;
      "FOR" I:= 1 "STEP" 1 "UNTIL" 6 "DO"
      INPUT(1, "("2(N),/")",X[I],Y[I]);
        PAR[1]:= 580; PAR[2]:= - 180; PAR[3]:= - 0.160;
        GSSNEWTON(6, 3, PAR, G, V, EXPFUNCT, JACOBIAN, IN, OUT);
        OUTPUT(61,"("3/4B,"("X[I]      Y[I]")",/, 6(5B+D, 5B3D.D, /), 2/,
        4B"("PARAMETERS")",/,3(4B+D.3D"+ZD,/),2/,4B"("OUT:")",/,
        3(9B+D.6D"+ZD,/), 5(14B3ZD,/), 9B+D.6D"+ZD,2/4B,
        "("LAST RESIDUAL VECTOR")",/,6(10B+3Z.D,/)")", X[1], Y[1],
        X[2],Y[2],X[3],Y[3],X[4],Y[4],X[5],Y[5],X[6],Y[6],PAR[1],PAR[2],
        PAR[3],OUT[6],OUT[2],OUT[3],OUT[4],OUT[5],OUT[1],OUT[7],OUT[8],
        OUT[9], G[1],G[2],G[3],G[4],G[5],G[6])
"END"
    "EOP"
```

WITH THE DATA GIVEN IN THE X - Y - TABLE THIS PROGRAM DELIVERS:

X[I]	Y[I]
-5	127.0
-3	151.0
-1	379.0
+1	421.0
+3	460.0
+5	426.0

```
PARAMETERS
+5.233" +2
-1.569" +2
-1.997" -1


OUT:
      +5.260478" -4
      +1.157156" +2
      +1.654588" +2
                16
                16
                 2
                 0
                 3
      +2.339529" +3

LAST RESIDUAL VECTOR
        -29.6
        +86.6
        -47.3
        -26.2
        -22.9
        +39.5
```

SOURCE TEXTS :

```
"CODE" 34440;
"PROCEDURE" MARQUARDT(M,N,PAR,G,V,FUNCT,JACOBIAN,IN,OUT);
"VALUE" M,N; "INTEGER" M,N; "ARRAY" PAR,G,V,IN,OUT;
"BOOLEAN" "PROCEDURE" FUNCT; "PROCEDURE" JACOBIAN;
"BEGIN" "INTEGER" MAXFE,FE,IT,I,J,ERR;
        "REAL" VV,WW,W,MU,RES,FPAR,FPARPRES,LAMBDA,LAMBDAMIN,
               P,PW,RELTOLRES,ABSTOLRES;
        "ARRAY" EM[0:7],VAL,B,BB,PARPRES[1:N],JAC[1:M,1:N];

        "PROCEDURE" MULCOL(L,U,S,T,A,B,X); "CODE" 31022;
        "PROCEDURE" DUPVEC(L,U,S,A,B); "CODE" 31030;
        "REAL" "PROCEDURE" VECVEC(L,U,S,A,B); "CODE" 34010;
        "REAL" "PROCEDURE" MATVEC(L,U,S,A,B); "CODE" 34011;
        "REAL" "PROCEDURE" TAMVEC(L,U,S,A,B); "CODE" 34012;
        "REAL" "PROCEDURE" MATTAM(L,U,S,T,A,B); "CODE" 34015;
        "INTEGER" "PROCEDURE" QRISNGVALDEC(A,M,N,VAL,V,EM);
        "CODE" 34273;

      "PROCEDURE" LOCFUNCT(M,N,PAR,G);
      "INTEGER" M,N; "ARRAY" PAR,G;
      "BEGIN" FE:= FE+1; "IF" FE >= MAXFE "THEN" ERR:= 1 "ELSE"
              "IF" "NOT" FUNCT(M,N,PAR,G) "THEN" ERR:= 2;
              "IF" ERR^=0 "THEN" "GOTO" EXIT
      "END" LOCFUNCT;

      VV:=10; W:=0.5; MU:= 0.01;
      WW:=("IF" IN[6]<"-7 "THEN" "-8 "ELSE" "-1*IN[6]);
      EM[0]:=EM[2]:=EM[6]:=IN[0]; EM[4]:=10*N;
      RELTOLRES:=IN[3]; ABSTOLRES:=IN[4]**2; MAXFE:=IN[5];
      ERR:= 0; FE:= IT:= 1; P:=FPAR:= RES:= 0;
      PW:=-LN(WW*IN[0])/2.30;

      "IF" "NOT" FUNCT(M,N,PAR,G) "THEN"
      "BEGIN" ERR:= 3; "GOTO" ESCAPE "END";
      FPAR:= VECVEC(1,M,0,G,G); OUT[3]:=SQRT(FPAR);

      "FOR" IT:= 1, IT+1 "WHILE" FPAR > ABSTOLRES "AND"
                    RES > RELTOLRES*FPAR+ABSTOLRES "DO"
      "BEGIN" JACOBIAN(M,N,PAR,G,JAC,LOCFUNCT);
              I:=QRISNGVALDEC(JAC,M,N,VAL,V,EM);
              "IF" IT=1 "THEN"
                    LAMBDA:= IN[6] * VECVEC(1,N,0,VAL,VAL) "ELSE"
              "IF" P =0 "THEN" LAMBDA:= LAMBDA*W "ELSE" P:= 0;

              "FOR" I:=1 "STEP" 1 "UNTIL" N "DO"
              B[I]:=VAL[I]*TAMVEC(1,M,I,JAC,G);
                                                    "COMMENT"
```

```
;
      L:    "FOR" I:=1 "STEP" 1 "UNTIL" N "DO"
            BB[I]:=B[I]/(VAL[I]*VAL[I]+LAMBDA);
      "FOR" I:=1 "STEP" 1 "UNTIL" N "DO"
      PARPRES[I]:= PAR[I] - MATVEC(1,N,I,V,BB);
      LOCFUNCT(M,N,PARPRES,G);
      FPARPRES:= VECVEC(1,M,0,G,G);
      RES:=FPAR-FPARPRES;
      "IF" RES < MU * VECVEC(1,N,0,B,BB) "THEN"
      "BEGIN" P:= P+1; LAMBDA:= VV * LAMBDA;
          "IF" P=1 "THEN"
          "BEGIN" LAMBDAMIN:= WW * VECVEC(1,N,0,VAL,VAL);
              "IF" LAMBDA<LAMBDAMIN "THEN" LAMBDA:= LAMBDAMIN
          "END";
          "IF" P<PW "THEN" "GOTO" L "ELSE"
          "BEGIN" ERR:= 4;
                  "GOTO" EXIT
          "END";
      "END";

      DUPVEC(1,N,0,PAR,PARPRES);
            FPAR:=FPARPRES
      "END" ITERATION;

EXIT:
      "FOR" I:=1 "STEP" 1 "UNTIL" N "DO"
      MULCOL(1,N,I,I,JAC,V,1/(VAL[I]+IN[0]));
      "FOR" I:=1 "STEP" 1 "UNTIL" N "DO"
      "FOR" J:=1 "STEP" 1 "UNTIL" I "DO"
      V[I,J]:= V[J,I]:= MATTAM(1,N,I,J,JAC,JAC);

      LAMBDA:= LAMBDAMIN:= VAL[1];
      "FOR" I:= 2 "STEP" 1 "UNTIL" N "DO"
      "IF" VAL[I]>LAMBDA     "THEN" LAMBDA    := VAL[I] "ELSE"
      "IF" VAL[I]<LAMBDAMIN "THEN" LAMBDAMIN:= VAL[I];

      OUT[7]:=(LAMBDA/(LAMBDAMIN+IN[0]))**2;
      OUT[2]:=SQRT(FPAR);
      OUT[6]:=SQRT(RES+FPAR)-OUT[2];
ESCAPE:
      OUT[4]:=FE;
      OUT[5]:=IT-1;
      OUT[1]:=ERR
"END" MARQUARDT;
   "EOP"
```

```
"CODE" 34441;
"PROCEDURE" GSSNEWTON(M, N, PAR, RV, JJINV, FUNCT, JACOBIAN,
  IN, OUT);
"VALUE" M, N; "INTEGER" M, N;
"ARRAY" PAR, RV, JJINV, IN, OUT;
"BOOLEAN" "PROCEDURE" FUNCT;
"PROCEDURE" JACOBIAN;

"BEGIN" "INTEGER" I, J, INR, MIT, TEXT,
    IT, ITMAX, INRMAX, TIM, FEVAL, FEVALMAX;
    "REAL" RHO, RES1, RES2, RN, RELTOLPAR, ABSTOLPAR, ABSTOLRES,
       STAP, NORMX;
    "BOOLEAN" CONV, TESTTHF, DAMPING ON;
    "ARRAY" JAC[1:M + 1,1:N], PR, AID, SOL[1 : N], FU2[1 : M],
       AUX[2 : 5];
    "INTEGER""ARRAY" CI[1:N];

    "REAL""PROCEDURE" VECVEC(L, U, SHIFT, A, B); "CODE" 34010;
    "PROCEDURE" DUPVEC(L, U, S, A, B); "CODE" 31030;
    "PROCEDURE" ELMVEC(L, U, S, A, B, X); "CODE" 34020;
    "PROCEDURE" LSQORTDEC(A, M, N, AUX, AID, CI); "CODE" 34134;
    "PROCEDURE" LSQSOL(A, M, N, AID, CI, B); "CODE" 34131;
    "PROCEDURE" LSQINV(A, N, AID, CI); "CODE" 34136;

    "BOOLEAN""PROCEDURE" LOC FUNCT(M, N, PAR, RV);
    "VALUE" M, N; "INTEGER" M, N; "ARRAY" PAR, RV;
    "BEGIN" LOC FUNCT:= TEST THF:= FUNCT(M, N, PAR, RV)
        "AND" TEST THF; FEVAL:= FEVAL + 1
    "END" LOC FUNCT;


    ITMAX:= FEVALMAX:= IN[5]; AUX[2]:= N * IN[0]; TIM:= IN[7];
    RELTOLPAR:= IN[1] ** 2; ABSTOLPAR:= IN[2] ** 2;
    ABSTOLRES:= IN[4] ** 2; INRMAX:= IN[6];
    DUPVEC(1, N, 0, PR, PAR);
    "IF" M < N "THEN"
    "FOR" I:= 1 "STEP" 1 "UNTIL" N "DO" JAC[M + 1, I]:= 0;
    TEXT:= 4; MIT:= 0; TEST THF:= "TRUE";
    RES2:= STAP:= OUT[5]:= OUT[6]:= OUT[7]:= 0;
    FUNCT(M, N, PAR, FU2); RN:= VECVEC(1, M, 0, FU2, FU2);
    OUT[3]:= SQRT(RN); FEVAL:= 1; DAMPING ON:= "FALSE";
    "FOR" IT:= 1, IT + 1 "WHILE" IT <= ITMAX "AND"
          FEVAL < FEVALMAX "DO"
    "BEGIN" OUT[5]:= IT; JACOBIAN(M, N, PAR, FU2, JAC, LOCFUNCT);
       "IF" "NOT" TEST THF "THEN"
       "BEGIN" TEXT:= 7; "GO TO" FAIL "END";
       LSQORTDEC(JAC, M, N, AUX, AID, CI);
       "IF" AUX[3] ^= N "THEN"
       "BEGIN" TEXT:= 5; "GO TO" FAIL "END";
       LSQSOL(JAC, M, N, AID, CI, FU2); DUPVEC(1, N, 0, SOL, FU2);
       STAP:= VECVEC(1, N, 0, SOL, SOL);
       RHO:= 2; NORMX:= VECVEC(1, N, 0, PAR, PAR);
                                                    "COMMENT"
```

```
        "IF" STAP > RELTOLPAR * NORMX + ABSTOLPAR
           "OR" IT = 1 "AND" STAP > 0 "THEN"
        "BEGIN""FOR" INR:= 0, INR + 1
           "WHILE""IF" INR = 1 "THEN" DAMPING ON "OR" RES2 >= RN
           "ELSE""NOT" CONV "AND" (RN <= RES1 "OR" RES2 < RES1) "DO"
        "BEGIN""COMMENT" DAMPING STOPS WHEN
           R0 > R1 "AND" R1 <= R2 (BEST RESULT IS X1, R1)
           WITH  X1 = X0 + I * DX, I:= 1, .5, .25, .125, ETC.  ;
           RHO:= RHO / 2; "IF" INR > 0 "THEN"
           "BEGIN" RES1:= RES2; DUPVEC(1, M, 0, RV, FU2);
              DAMPING ON:= INR > 1
           "END";
           "FOR" I:= 1 "STEP" 1 "UNTIL" N "DO"
           PR[I]:= PAR[I] - SOL[I] * RHO;
           FEVAL:= FEVAL + 1;
           "IF" "NOT" FUNCT(M, N, PR, FU2) "THEN"
           "BEGIN" TEXT:= 6; "GO TO" FAIL "END";
           RES2:= VECVEC(1, M, 0, FU2, FU2); CONV:= INR >= INRMAX
        "END" DAMPING OF STEP VECTOR;
        "IF" CONV "THEN"
        "BEGIN""COMMENT" RESIDUE CONSTANT; MIT:= MIT + 1;
           "IF" MIT < TIM "THEN" CONV:= "FALSE"
           "END" "ELSE" MIT:= 0;
           "IF" INR > 1 "THEN"
           "BEGIN" RHO:= RHO * 2; ELMVEC(1, N, 0, PAR, SOL, - RHO);
              RN:= RES1; "IF" INR > 2 "THEN" OUT[7]:= IT
           "END""ELSE"
           "BEGIN" DUPVEC(1, N, 0, PAR, PR); RN:= RES2;
              DUPVEC(1, M, 0, RV, FU2)
           "END";

           "IF" RN <= ABSTOLRES "THEN"
           "BEGIN" TEXT:= 1; ITMAX:= IT "END""ELSE"
           "IF" CONV "AND" INRMAX > 0 "THEN"
           "BEGIN" TEXT:= 3; ITMAX:= IT "END"
           "ELSE" DUPVEC(1, M, 0, FU2, RV)
        "END" ITERATION WITH DAMPING AND TESTS "ELSE"
        "BEGIN" TEXT:= 2; RHO:= 1; ITMAX:= IT "END"
     "END" OF ITERATIONS;

        LSQINV(JAC, N, AID, CI);
        "FOR" I:= 1 "STEP" 1 "UNTIL" N "DO"
        "BEGIN" JJINV[I,I]:= JAC[I,I];
            "FOR" J:= I + 1 "STEP" 1 "UNTIL" N "DO"
            JJINV[I,J]:= JJINV[J,I]:= JAC[I,J]
        "END" CALCULATION OF INVERSE MATRIX OF NORMAL EQUATIONS;
FAIL :
        OUT[6]:= SQRT(STAP) * RHO; OUT[2]:= SQRT(RN); OUT[4]:= FEVAL;
        OUT[1]:= TEXT; OUT[8]:= AUX[3]; OUT[9]:= AUX[5]
"END" GSSNEWTON;
        "EOP"
```

SECTION 5.2.1.1.1.1 CONTAINS NINE ALTERNATIVE PROCEDURES FOR SOLVING FIRST-ORDER INITIAL-VALUE PROBLEMS WITH NO DERIVATIVES OF THE RIGHT-HAND SIDE AVAILABLE.

A. RK1 SOLVES AN INITIAL-VALUE PROBLEM GIVEN AS A SINGLE FIRST ORDER DIFFERENTIAL EQUATION
DY / DX = F(X,Y)
BY MEANS OF A 5-TH ORDER RUNGE-KUTTA METHOD.

B. RKE SOLVES AN INITIAL-VALUE PROBLEM GIVEN AS A SYSTEM OF FIRST ORDER DIFFERENTIAL EQUATIONS BY MEANS OF A 5-TH ORDER RUNGE-KUTTA METHOD.

C. RK4A SOLVES THE INITIAL VALUE PROBLEM
DY/DX=F(X,Y), X>=A, Y(A)=YA ;
IF ABS(F(X,Y))<=1 RK4A USES X AS INDEPENDENT INTEGRATION VARIABLE,OTHERWISE Y;
THE INTEGRATION IS TERMINATED AS SOON AS A CONDITION ON X AND Y, WHICH IS SUPPLIED BY THE USER,IS FULFILLED,E.G. X=Y.

D. RK4NA SOLVES THE SYSTEM OF FIRST ORDER DIFFERENTIAL EQAUTIONS:
DX[J]/DX[0]=F(J,X[0],...,X[N]) , X[0]>=XA[0],
X[J]=XA[J] , X[0]=XA[0], J=1,...,N,
WHERE X[0] IS THE INDEPENDENT VARIABLE AND X[1],...,X[N] ARE THE DEPENDENT VARIABLES;RK4NA CHOOSES FROM AMONGST X[0],...,X[N] AS INTEGRATION VARIABLE ,A VARIABLE SUCH THAT THE ABSOLUTE VALUE OF THE DERIVATIVE OF THE OTHER VARIABLES WITH RESPECT TO THE ONE CHOSEN ARE <=1;THE INTEGRATION IS STOPPED,IF A CERTAIN CONDITION, SUPPLIED BY THE USER, ON X[0],...,X[N] IS SATISFIED.

E. RK5NA SOLVES THE SYSTEM OF FIRST ORDER DIFFERENTIAL EQAUTIONS:

$$DX[J] / DX[0] = F(J, X[0], \ldots, X[N]) / F(0, X[0], \ldots, X[N]), \quad J = 1, \ldots, N,$$

WHERE F(J, X[0], ..., X[N]) AND F(0, X[0], ..., X[N]) REMAIN FINITE; THE ARC LENGTH S IS INTRODUCED AS INTEGRATION VARIABLE ;
THE SYSTEM SOLVED IS

$$DX[J] / DS = F(J, X[0], \ldots, X[N]) / SQRT(A), \quad J = 0, \ldots, N,$$
$$A = SUM(I, 0, N, F(I, X[0], \ldots, X[N]) ** 2) ;$$

THE INTEGRATION STOPS IF SOME CONDITION ON X[0], ..., X[N] , TO BE SUPPLIED BY THE USER, IS SATISFIED WITHIN CERTAIN TOLERANCES.

F. MULTISTEP SOLVES AN INITIAL VALUE PROBLEM, GIVEN AS A SYSTEM OF FIRST ORDER DIFFERENTIAL EQUATIONS, BY MEANS OF A MULTISTEP METHOD: GEARS METHOD, ADAMS-MOULTON OR ADAMS-BASHFORTH-METHOD.

G. DIFFSYS SOLVES INITIAL VALUE PROBLEMS ,GIVEN AS A SYSTEM OF FIRST ORDER DIFFERENTIAL EQUATIONS, EXTRAPOLATION, APPLIED TO LOWER ORDER RESULTS,DELIVERS A HIGH ORDER SOLUTION. AUTOMATIC STEP SIZE CONTROL IS PROVIDED.

H. ARK SOLVES AN INITIAL VALUE PROBLEM, GIVEN AS A SYSTEM OF FIRST ORDER (NON-LINEAR) DIFFERENTIAL EQUATIONS BY MEANS OF A STABILIZED RUNGE KUTTA METHOD WITH LIMITED STORAGE REQUIREMENTS.

I. EFRK SOLVES AN INITIAL VALUE PROBLEM , GIVEN AS A SYSTEM OF FIRST ORDER DIFFERENTIAL EQUATIONS , BY MEANS OF AN EXPONENTIALLY FITTED, EXPLICIT RUNGE-KUTTA METHOD OF FIRST , SECOND OR THIRD ORDER . AUTOMATIC STEPSIZE CONTROL IS NOT PROVIDED; HOWEVER, FOR REASONS OF THE USER PRESCRIBED STEPSIZE CAN BE ADJUSTED.

RK1, RKE, RK4A, RK4NA, RK5NA AND DIFFSYS ARE WELL FITTED FOR NONSTIFF SYSTEMS OR EQUATIONS.
DIFFSYS SHOULD BE USED WHEN HIGH ACCURACY IS DESIRED.
MULTISTEP MIGHT BE USED FOR BOTH STIFF AND NONSTIFF SYSTEMS.
WHEN LARGE SYSTEMS (E.G. DERIVED FROM PARTIAL DIFFERENTIAL EQUATIONS) ARE TO BE SOLVED, THEN ARK AND EFRK ARE
RECOMMENDED, THE FORMER FOR NONSTIFF, THE LATTER FOR STIFF EQUATIONS.

AUTHOR: J.A.ZONNEVELD.

CONTRIBUTORS: M.BAKKER AND I.BRINK.

INSTITUTE: MATHEMATICAL CENTRE.

RECEIVED: 730715.

BRIEF DESCRIPTION:

RK1 INTEGRATES A SINGLE FIRST ORDER DIFFERENTIAL EQUATION

$$DY / DX = F(X,Y)$$

BY MEANS OF A 5-TH ORDER RUNGE-KUTTA METHOD.

KEYWORDS:
RUNGE-KUTTA METHODS,
FIRST ORDER DIFFERENTIAL EQUATION,
INITIAL VALUE PROBLEM.

CALLING SEQUENCE:

```
THE HEADING OF THE PROCEDURE READS:
"PROCEDURE" RK1(X,A,B,Y,YA,FXY,E,D,FI);
"VALUE" B,FI;
"REAL" X,A,B,Y,YA,FXY;
"BOOLEAN" FI;
"ARRAY" E,D;

THE MEANING OF THE FORMAL PARAMETERS IS:
X:    <VARIABLE>;
      THE INDEPENDENT VARIABLE.
      UPON COMPLETION OF A CALL OF RK1,
      IT IS EQUAL TO B:
A:    <ARITHMETIC EXPRESSION>;
      ENTRY: THE STARTING VALUE OF X;
B:    <ARITHMETIC EXPRESSION>;
      ENTRY: A VALUE PARAMETER,GIVING THE END VALUE OF X;
Y:    <VARIABLE>;
      THE DEPENDENT VARIABLE;
YA:   <ARITHMETIC EXPRESSION>;
      THE VALUE OF Y AT X=A;
FXY:  <ARITHMETIC EXPRESSION>;
      AN EXPRESSION,DEPENDING ON X AND Y,GIVING THE VALUE OF DY/DX;
E:    <ARRAY IDENTIFIER>;
      "ARRAY" E[1:2];
      ENTRY:
      E[1] IS A RELATIVE TOLERANCE,
      E[2] IS AN ABSOLUTE TOLERANCE ASSOCIATED TO Y;
D:    <ARRAY IDENTIFIER>;
      "ARRAY" D[1:4];
      EXIT:
      ENTIER(D[1]+.5) IS THE NUMBER OF STEPS SKIPPED;
      D[2] IS THE STEP LENGTH;
      D[3] IS EQUAL TO B;
      D[4] IS EQUAL TO Y(B);
FI:   <BOOLEAN EXPRESSION>;
      IF FI="TRUE" THEN INTEGRATION STARTS AT X=A,WITH TRIAL STEP
      B-A;IF FI="FALSE" THEN INTEGRATION IS CONTINUED WITH
      INITIAL CONDITIONS:X=D[3],Y=D[4] AND STEP LENGTH H=D[2]*
      SIGN(B-D[3]);A AND YA ARE IGNORED.

DATA AND RESULTS:
    RK1 INTEGRATES DY/DX=FXY TO X=B,WITH IF FI="TRUE" THEN  X=A,Y(A)=A.
    IF FI="FALSE" THEN X=D[3],Y(D[3])=D[4].
    UPON COMPLETION OF A CALL OF RK1 WE HAVE X=D[3]=B,Y=D[4]=Y(B).
    RK1 USES AS ITS MINIMAL ABSOLUTE STEP LENGTH :HMIN=E[1]*INT+E[2],
    WHERE INT=ABS(B-("IF" FI "THEN" A "ELSE" D[3])).
    IF A STEP OF LENGTH ABS(H)<=HMIN IS REJECTED.
    A STEP SIGN(H)*HMIN IS SKIPPED.
    A STEP IS REJECTED IF THE ABSOLUTE VALUE OF THE
    LAST TERM TAKEN INTO ACCOUNT IS GREATER THEN :
    (ABS(FXY)*E[1]+E[2])*ABS(H)/INT.
    SEE REF[1].
```

PROCEDURES USED: NONE.

RUNNING TIME:DEPENDS STRONGLY ON THE DIFFERENTIAL EQUATION TO SOLVE.

LANGUAGE: ALGOL 60.

METHOD AND PERFORMANCE:SEE REF[1].

REFERENCES:

[1] J.A.ZONNEVELD.
AUTOMATIC NUMERICAL INTEGRATION.
MATHEMATICAL CENTRE TRACT 8(1970).

EXAMPLE OF USE:

THE SOLUTION OF THE DIFFERENTIAL EQUATION DY/DX=-Y
WITH INITIAL CONDITION Y(0)=1 AT X=1 IS COMPUTED
BY MEANS OF THE FOLLOWING PROGRAM:

```
"BEGIN"
"REAL" X,Y;
"BOOLEAN" FI,FIRST;
"REAL" "ARRAY" E[1:2],D[1:4];

"PROCEDURE" RK1(X,A,B,Y,YA,FXY,E,D,FI); "CODE" 33010;

E[1]:=+"-4;E[2]:=+"-4;FIRST:="TRUE";
RK1(X,0,1,Y,1,-Y,E,D,FIRST);
OUTPUT(61,"("//10B"("X=")",12D"2D,//10B"("Y=")",12D"2D,
10B"("YEXACT=")",12D"2D")",X,Y,EXP(-X));
"END"
    "EOP"
```

IT DELIVERS WITH E[1]=E[2]="-4;

X=.100000000000"01

Y=.367876846355"00 YEXACT=.367879441171"00

SOURCE TEXT(S):

```
"CODE"" 33010;
   "PROCEDURE" RK1(X, A, B, Y, YA, FXY, E, D, FI);
   "VALUE" B, FI; "REAL" X, A, B, Y, YA, FXY; "BOOLEAN" FI;
   "ARRAY" E, D;
   "BEGIN" "REAL" E1, E2, XL, YL, H, INT, HMIN, ABSH, K0, K1,
      K2, K3, K4, K5, DISCR, TOL, MU, MU1, FH, HL;
      "BOOLEAN" LAST, FIRST, REJECT;                    "COMMENT"
```

```
;


    "IF" FI "THEN"
    "BEGIN" D[3]:= A; D[4]:= YA "END";
    D[1]:= 0; XL:= D[3]; YL:= D[4];
    "IF" FI "THEN" D[2]:= B - D[3]; ABSH:= H:= ABS(D[2]);
    "IF" B - XL < 0 "THEN" H:= - H; INT:= ABS(B - XL);
    HMIN:= INT * E[1] + E[2]; E1:= E[1] / INT;
    E2:= E[2] / INT; FIRST:= "TRUE"; "IF" FI "THEN"
    "BEGIN" LAST:= "TRUE"; "GOTO" STEP "END";
TEST: ABSH:= ABS(H); "IF" ABSH < HMIN "THEN"
    "BEGIN" H:= "IF" H > 0 "THEN" HMIN "ELSE" - HMIN; ABSH:= HMIN
    "END";
    "IF" H >= B - XL "EQUIV" H >= 0 "THEN"
    "BEGIN" D[2]:= H; LAST:= "TRUE"; H:= B - XL;
       ABSH:= ABS(H)
    "END"
    "ELSE" LAST:= "FALSE";
STEP: X:= XL; Y:= YL; K0:= FXY * H;
    X:= XL + H / 4.5; Y:= YL + K0 / 4.5;
    K1:= FXY * H; X:= XL + H / 3;
    Y:= YL + (K0 + K1 * 3) / 12; K2:= FXY * H;
    X:= XL + H * .5; Y:= YL + (K0 + K2 * 3) / 8;
    K3:= FXY * H; X:= XL + H * .8;
    Y:= YL + (K0 * 53 - K1 * 135 + K2 * 126 + K3 * 56)
    / 125; K4:= FXY * H; X:= "IF" LAST "THEN" B "ELSE" XL + H;
    Y:= YL + (K0 * 133 - K1 * 378 + K2 * 276 + K3 * 112
    + K4 * 25) / 168; K5:= FXY * H;
    DISCR:= ABS(K0 * 21 - K2 * 162 + K3 * 224 - K4 * 125
    + K5 * 42) / 14; TOL:= ABS(K0) * E1 + ABSH * E2;
    REJECT:= DISCR > TOL; MU:= TOL / (TOL + DISCR) + .45;
    "IF" REJECT "THEN"
    "BEGIN" "IF" ABSH <= HMIN "THEN"
      "BEGIN" D[1]:= D[1] + 1; Y:= YL; FIRST:= "TRUE";
         "GOTO" NEXT
       "END";
       H:= MU * H; "GOTO" TEST
    "END";
    "IF" FIRST "THEN"
    "BEGIN" FIRST:= "FALSE"; HL:= H; H:= MU * H; "GOTO" ACC
    "END";
       FH:= MU * H / HL + MU - MU1; HL:= H; H:= FH * H;
    ACC: MU1:= MU;
       Y:= YL + ( - K0 * 63 + K1 * 189 - K2 * 36 - K3 * 112
       + K4 * 50) / 28; K5:= FXY * HL;
       Y:= YL + (K0 * 35 + K2 * 162 + K4 * 125 + K5 * 14)
       / 336;

    NEXT: "IF" B ^= X "THEN"
       "BEGIN" XL:= X; YL:= Y; "GOTO" TEST "END";
       "IF" "NOT"LAST "THEN" D[2]:= H; D[3]:= X; D[4]:= Y
    "END" RK1;
          "EOP"
```

AUTHOR: P.A. BEENTJES.

INSTITUTE: MATHEMATICAL CENTRE.

RECEIVED: 740520.

BRIEF DESCRIPTION:

RKE INTEGRATES A SYSTEM OF FIRST ORDER DIFFERENTIAL EQUATIONS (INITIAL VALUES BEING GIVEN) BY MEANS OF A FIFTH ORDER EXPLICIT RUNGE KUTTA METHOD.

KEYWORDS:

RUNGE KUTTA METHODS,
DIFFERENTIAL EQUATIONS,
INITIAL VALUE PROBLEMS.

CALLING SEQUENCE:

THE HEADING OF THE PROCEDURE READS:

```
"PROCEDURE" RKE (X, XE, N, Y, DER, DATA, FI, OUT);
"VALUE" N, FI; "INTEGER" N; "REAL" X, XE; "BOOLEAN" FI;
"ARRAY" Y, DATA;
"PROCEDURE" DER, OUT;
```

RKE : INTEGRATES THE SYSTEM OF ORDINARY DIFFERENTIAL EQUATIONS
DY / DX = F(X, Y), FROM X = X0 TO X = XE WHERE Y(X0) = Y0.

THE MEANING OF THE FORMAL PARAMETERS IS:

X: <VARIABLE>;
THE INDEPENDENT VARIABLE;
ENTRY: THE INITIAL VALUE X0;

XE: <ARITHMETIC EXPRESSION>;
THE FINAL VALUE OF X;

N: <ARITHMETIC EXPRESSION>;
THE NUMBER OF EQUATIONS OF THE SYSTEM;

Y: <ARRAY IDENTIFIER>;
"ARRAY" Y[1 : N];
THE DEPENDENT VARIABLES;
ENTRY: THE INITIAL VALUES OF Y AT X = X0;
EXIT : THE VALUES OF THE SOLUTION AT X = XE;

DER: <PROCEDURE IDENTIFIER>;
THE HEADING OF THIS PROCEDURE READS:
"PROCEDURE" DER (T, V); "VALUE" T; "REAL" T; "ARRAY" V;
THIS PROCEDURE PERFORMS AN EVALUATION OF THE RIGHT HAND SIDE OF THE SYSTEM WITH DEPENDENT VARIABLES V[1 : N] AND INDEPENDENT VARIABLE T; UPON COMPLETION OF DER, THE RIGHT HAND SIDE SHOULD BE OVERWRITTEN ON V[1 : N];

DATA: <ARRAY IDENTIFIER>;
"ARRAY" DATA[1 : 6];
IN ARRAY DATA ONE SHOULD GIVE:
DATA[1]: THE RELATIVE TOLERANCE;
DATA[2]: THE ABSOLUTE TOLERANCE;
AFTER EACH STEP THE FOLLOWING BY-PRODUCTS ARE DELIVERED:
DATA[3]: THE STEPLENGTH USED FOR THE LAST STEP;
DATA[4]: THE NUMBER OF INTEGRATION STEPS PERFORMED;
DATA[5]: THE NUMBER OF INTEGRATION STEPS REJECTED;
DATA[6]: THE NUMBER OF INTEGRATION STEPS SKIPPED;
IF UPON COMPLETION OF RKE DATA[6] > 0 , RESULTS SHOULD BE CONSIDERED MOST CRITICALLY;

FI: <BOOLEAN EXPRESSION>;
IF FI = "TRUE" THE INTEGRATION STARTS AT X0 WITH A TRIAL STEP XE - X0; IF FI = "FALSE" THE INTEGRATION IS CONTINUED WITH A STEPLENGTH DATA[3] * SIGN(XE - X0);

OUT: <PROCEDURE IDENTIFIER>;
THE HEADING OF THIS PROCEDURE READS:
"PROCEDURE" OUT;
AFTER EACH INTEGRATION STEP PERFORMED, INFORMATION CAN BE OBTAINED OR UPDATED BY THIS PROCEDURE, E.G. THE VALUES OF X, Y[1 : N] AND DATA[3 : 6].

DATA AND RESULTS:

SEE REFERENCES [1] AND [3].

PROCEDURES USED:

NONE

REQUIRED CENTRAL MEMORY:

EXECUTION FIELD LENGTH: CIRCA 5 * N.

RUNNING TIME:

DEPENDS STRONGLY ON THE SYSTEM OF DIFFERENTIAL EQUATIONS TO BE SOLVED.

LANGUAGE: ALGOL60.

METHOD AND PERFORMANCE:

THE SCHEME UPON WHICH THE METHOD IS BASED, IS A MEMBER OF A CLASS OF FIFTH ORDER RUNGE KUTTA FORMULAS PRESENTED IN REFERENCE [1].
AUTOMATIC STEPSIZE CONTROL IS IMPLEMENTED IN A WAY AS PROPOSED IN REFERENCE [2].
FOR TESTRESULTS AND FURTHER INFORMATION SEE REFERENCE [3].

REFERENCES:

[1]. R. ENGLAND.
ERROR ESTIMATES FOR RUNGE KUTTA TYPE SOLUTIONS TO SYSTEMS OF ORDINARY DIFFERENTIAL EQUATIONS.
THE COMPUTER JOURNAL , VOLUME 12, P 166 - 169, 1969.

[2]. J.A. ZONNEVELD.
AUTOMATIC NUMERICAL INTEGRATION.
MATH. CENTRE TRACT 8(1970).

[3]. P.A. BEENTJES.
SOME SPECIAL FORMULAS OF THE ENGLAND CLASS OF FIFTH ORDER RUNGE - KUTTA SCHEMES.
MATH. CENTRE REPORT NW 24/74.

EXAMPLE OF USE:

THE SOLUTION AT T = 1 AND T = -1 OF THE SYSTEM

```
DX / DT = Y - Z,
DY / DT = X * X + 2 * Y + 4 * T,
DZ / DT = X * X + 5 * X + Z * Z + 4 * T,

WITH  X = Y = 0  AND  Z = 2  AT  T = 0,
```

CAN BE OBTAINED BY THE FOLLOWING PROGRAM:

```
"BEGIN" "REAL" T, TE; "ARRAY" Y[1 : 3], DATA[1 : 6];

   "PROCEDURE" RKE(X, XE, N, Y, DER, DATA, FI, OUT);
   "CODE" 33033;

   "PROCEDURE" RHS(T, Y); "VALUE" T; "REAL" T; "ARRAY" Y;
   "BEGIN" "REAL" XX, YY, ZZ;
      XX:= Y[1]; YY:= Y[2]; ZZ:= Y[3];
      Y[1]:= YY - ZZ;
      Y[2]:= XX * XX + 2 * YY + 4 * T;
      Y[3]:= XX * (XX + 5) + 2 * ZZ + 4 * T
"END" RHS;

"PROCEDURE" INFO;
"IF" T = TE "THEN"
"BEGIN" "REAL" ET, T2, AEX, AEY, AEZ, REX, REY, REZ;
   ET:= EXP(T); T2:= 2 * T;
   REX:= -ET * SIN(T2); AEX:= REX - Y[1]; REX:= ABS(AEX / REX);
   REY:= ET * ET * (8 + 2 * T2 - SIN(2 * T2)) / 8 - T2 - 1;
   REZ:= ET * (SIN(T2) + 2 * COS(T2)) + REY;
   AEY:= REY - Y[2]; REY:= ABS(AEY / REY); AEZ:= REZ - Y[3];
   REZ:= ABS(AEZ / REZ);
   OUTPUT(61, "(""(" T = ")", +D, //,
   "(" RELATIVE AND ABSOLUTE ERRORS IN X, Y AND Z :")", //,
   "("   RE(X)   RE(Y)   RE(Z)   AE(X)   AE(Y)   AE(Z)")", //,
   6(B,-.20"+D), //,
      "(" NUMBER OF INTEGRATION STEPS PERFORMED :")",4ZD,/,
      "(" NUMBER OF INTEGRATION STEPS SKIPPED   :")",4ZD,/,
      "(" NUMBER OF INTEGRATION STEPS REJECTED  :")",4ZD,///")",
      T, REX, REY, REZ, ABS(AEX), ABS(AEY), ABS(AEZ),
      DATA[4], DATA[6], DATA[5])
   "END" INFO;

   TE:= 1;
LEFT:
   Y[1]:= Y[2]:= 0; Y[3]:= 2; T:=0;
   DATA[1]:= DATA[2]:= "-5;
   RKE(T, TE, 3, Y, RHS, DATA, "TRUE", INFO);
   "IF" TE = 1 "THEN" "BEGIN" TE:= -1; "GOTO" LEFT "END"
"END"
```

THIS PROGRAM DELIVERS:

T = +1

RELATIVE AND ABSOLUTE ERRORS IN X, Y AND Z :

RE(X)	RE(Y)	RE(Z)	AE(X)	AE(Y)	AE(Z)
.37"-6	.15"-5	.13"-5	.91"-6	.13"-4	.11"-4

NUMBER OF INTEGRATION STEPS PERFORMED : 9
NUMBER OF INTEGRATION STEPS SKIPPED : 0
NUMBER OF INTEGRATION STEPS REJECTED : 5

T = -1

RELATIVE AND ABSOLUTE ERRORS IN X, Y AND Z :

RE(X)	RE(Y)	RE(Z)	AE(X)	AE(Y)	AE(Z)
.22"-6	.52"-7	.19"-6	.75"-7	.55"-7	.77"-7

NUMBER OF INTEGRATION STEPS PERFORMED : 10
NUMBER OF INTEGRATION STEPS SKIPPED : 0
NUMBER OF INTEGRATION STEPS REJECTED : 7

SOURCE TEXT(S):

```
"CODE" 33 033;
   "PROCEDURE" RKE (X, XE, N, Y, DER, DATA, FI, OUT);
   "VALUE"  FI, N; "INTEGER" N; "REAL" X, XE;
   "BOOLEAN" FI; "ARRAY" Y, DATA;
   "PROCEDURE" DER, OUT;
   "BEGIN" "INTEGER" J;
      "REAL" XT, H, HMIN, INT, HL, HT, ABSH, FHM, DISCR, TOL, MU,
      MU1, FH, E1, E2;
      "BOOLEAN" LAST, FIRST, REJECT;
      "ARRAY" K0, K1, K2, K3, K4[1:N];
      "IF" FI "THEN"
      "BEGIN" DATA[3]:= XE - X; DATA[4]:= DATA[5]:= DATA[6]:= 0 "END";
      ABSH:= H:= ABS(DATA[3]);
      "IF" XE < X "THEN" H:= - H; INT:= ABS(XE - X);
      HMIN:= INT * DATA[1] + DATA[2];
      E1:= 12 * DATA[1] / INT; E2:= 12 * DATA[2] / INT;
      FIRST:= "TRUE"; REJECT:= "FALSE"; "IF" FI "THEN"
      "BEGIN" LAST:= "TRUE"; "GOTO" STEP "END";
TEST: ABSH:= ABS(H); "IF" ABSH < HMIN "THEN"
      "BEGIN" H:= SIGN (XE - X) * HMIN; ABSH:= HMIN "END";
      "IF" H >= XE - X "EQUIV" H >= 0 "THEN"
      "BEGIN" LAST:= "TRUE"; H:= XE - X; ABSH:= ABS(H) "END"
      "ELSE" LAST:= "FALSE";
                                                          "COMMENT"
```

```
    STEP: "IF" ^ REJECT "THEN"
       "BEGIN" "FOR" J:= 1 "STEP" 1 "UNTIL" N "DO" K0[J]:= Y[J];
           DER(X, K0)
       "END";
    HT:= .184262134833347 * H; XT:= X + HT;
    "FOR" J:= 1 "STEP" 1 "UNTIL" N "DO" K1[J]:= K0[J] * HT + Y[J];
    DER(XT, K1);
    HT:= .690983005625053"-1 * H; XT:= 4 * HT + X;
    "FOR" J:= 1 "STEP" 1 "UNTIL" N "DO" K2[J]:=
    (3 * K1[J] + K0[J]) * HT + Y[J];
    DER(XT, K2);
    XT:= .5 * H + X; HT:= .1875 * H;
    "FOR" J:= 1 "STEP" 1 "UNTIL" N "DO" K3[J]:=((1.74535599249993
    * K2[J] - K1[J]) * 2.23606797749979 + K0[J]) * HT + Y[J];
    DER(XT, K3);
    XT:= .723606797749979 * H + X; HT:= .4 * H;
    "FOR" J:= 1 "STEP" 1 "UNTIL" N "DO" K4[J]:= (((.517595468166681
    * K0[J] - K1[J]) * .927050983124840 + K2[J]) * 1.46352549156242
    + K3[J]) * HT + Y[J];
    DER(XT, K4);
    XT:= "IF" LAST "THEN" XE "ELSE" X + H; HT:= 2 * H;
    "FOR" J:= 1 "STEP" 1 "UNTIL" N "DO" K1[J]:= ((((2 * K4[J] +
    K2[J]) * .412022659166595 + K1[J]) * 2.23606797749979 -
    K0[J]) * .375 - K3[J]) * HT + Y[J];
    DER(XT, K1);
    REJECT:= "FALSE"; FHM:= 0;
    "FOR" J:= 1 "STEP" 1 "UNTIL" N "DO"
    "BEGIN" DISCR:= ABS((1.6 * K3[J] - K2[J] - K4[J]) * 5 +
        K0[J] + K1[J]);
       TOL:= ABS(K0[J]) * E1 + E2;
       REJECT:= DISCR > TOL "OR" REJECT;
       FH:= DISCR / TOL; "IF" FH > FHM "THEN" FHM:= FH
    "END";
    MU:= 1 / (1 + FHM) + .45; "IF" REJECT "THEN"
    "BEGIN" DATA[5]:= DATA[5] + 1; "IF" ABSH <= HMIN "THEN"
       "BEGIN" DATA[6]:= DATA[6] + 1; HL:= H; REJECT:= "FALSE";
          FIRST:= "TRUE"; "GOTO" NEXT
       "END";
       H:= MU * H; "GOTO" TEST
    "END";
       "IF" FIRST "THEN"
       "BEGIN" FIRST:= "FALSE"; HL:= H; H:= MU * H; "GOTO" ACC
       "END";
       FH:= MU * H / HL + MU - MU1; HL:= H; H:= FH * H;
    ACC: MU1:= MU; HT:= HL / 12;
       "FOR" J:= 1 "STEP" 1 "UNTIL" N "DO" Y[J]:=
       ((K2[J] + K4[J]) * 5 + K0[J] + K1[J]) * HT + Y[J];
    NEXT: DATA[3]:= HL; DATA[4]:= DATA[4] + 1; X:= XT; OUT;
       "IF" X ^= XE "THEN" "GOTO" TEST
    "END" FKE;
          "EOP"
```

AUTHOR: J.A.ZONNEVELD.

CONTRIBUTORS: M.BAKKER AND I.BRINK.

INSTITUTE: MATHEMATICAL CENTRE,

RECEIVED: 730715.

BRIEF DESCRIPTION:

RK4A IS USED TO INTEGRATE THE INITIAL VALUE PROBLEM
DY/DX=F(X,Y), X>=A, Y(A)=YA ;
IF ABS(F(X,Y))<=1 RK4A USES X AS INDEPENDENT INTEGRATION
VARIABLE,OTHERWISE Y;
THE INTEGRATION IS TERMINATED AS SOON AS A CONDITION ON X AND Y,
WHICH IS SUPPLIED BY THE USER,IS FULFILLED,E.G. X=Y.

KEYWORDS:

RUNGE-KUTTA METHODS,
INITIAL VALUE PROBLEM,
CHANGE OF DEPENDENT VARIABLE,
DEPENDENT INTEGRATION INTERVAL
(USER SUPPLIED END CONDITION).

CALLING SEQUENCE:

THE HEADING OF THE PROCEDURE READS:

```
"PROCEDURE" RK4A(X,XA,B,Y,YA,FXY,E,D,FI,XDIR,POS);
"VALUE" FI,XDIR,POS;
"BOOLEAN" FI,XDIR,POS;
"REAL" X,XA,B,Y,YA,FXY;
"ARRAY" E,D;
```

THE MEANING OF THE FORMAL PARAMETERS IS:

X: <VARIABLE>;
THE INDEPENDENT VARIABLE ;
X CAN BE USED AS A JENSEN PARAMETER DURING THE EVALUATION OF FXY;
UPON COMPLETION OF A CALL OF RK4A ,ITS VALUE IS THE LATEST VALUE OF X REACHED BY THE INTEGRATION;

XA: <ARITHMETIC EXPRESSION>;
ENTRY: THE START VALUE OF X;

B: <ARITHMETIC EXPRESSION>;
B DEPENDS ON X AND Y.
THE EQUATION B=0 ,FULFILLED WITHIN CERTAIN TOLERANCES, SPECIFIES THE END OF THE INTEGRATION.
AT THE END OF EACH INTEGRATION STEP B IS EVALUATED AND IS TESTED FOR CHANGE OF SIGN;

Y; <VARIABLE>;
THE DEPENDENT VARIABLE;
Y CAN BE USED AS A JENSEN PARAMETER DURING THE EVALUATION OF FXY;

YA: <ATITHMETIC EXPRESSION>;
ENTRY: THE VALUE OF Y AT X=XA;

FXY; <ARITHMETIC EXPRESSION>;
FXY,DEPENDING ON X AND Y ,GIVES THE VALUE OF DY/DX;

E; <ARRAY IDENTIFIER>;
"ARRAY" E[0:5];
ENTRY:
E[0] AND E[2] ARE RELATIVE TOLERANCES ,
E[1] AND E[3] ARE ABSOLUTE TOLREANCES ASSOCIATED WITH X AND Y RESPECTIVELY;
E[4] AND E[5] ARE TOLERANCES USED IN THE DETERMINATION OF THE ZERO OF B;

D; <ARRAY IDENTIFIER>;
"ARRAY" D[0:4];
AFTER COMPLETION OF EACH STEP OF INTEGRATION WE HAVE:
IF D[0]>0 THEN X IS THE INTEGRATION VARIABLE;
IF D[0]<0 THEN Y IS THE INTEGRATION VARIABLE;
D[1] IS THE NUMBER OF STEPS SKIPPED;
D[2]IS THE STEPSIZE;
D[3] IS EQUAL TO THE LAST VALUE OF X;
D[4] IS EQUAL TO THE LAST VALUE OF Y;

FI: <BOOLEAN EXPRESSION>;
IF FI="TRUE" THEN THE INTEGRATION IS STARTED WITH INITIAL CONDITIONS X=XA,Y=YA.
IF FI="FALSE" THEN THE INTEGRATION IS STARTED WITH X=D[3], Y=D[4];

XDIR, POS : <BOOLEAN EXPRESSION>;
IF FI="TRUE" THEN THE INTEGRATION STARTS IN SUCH A WAY THAT
IF POS="TRUE" AND XDIR="TRUE" THEN X INCREASES,
IF POS="TRUE" AND XDIR="FALSE" THEN Y INCREASES,
IF POS="FALSE" AND XDIR="TRUE" THEN X DECREASES,
IF POS="FALSE" AND XDIR="FALSE" THEN Y DECREASES.

PROCEDURES USED:

ZEROIN = CP34150.

RUNNING TIME:DEPENDS STRONGLY ON THE DIFFERENTIAL EQUATION TO SOLVE.

LANGUAGE:ALGOL 60.

METHOD AND PERFORMANCE: A 5-TH ORDER RUNGE-KUTTA METHOD IS USED; SEE REF [1] (RK4A IS AN ADAPTED VERSION OF RK4).

REFERENCES:

[1]J.A.ZONNEVELD,
AUTOMATIC NUMERICAL INTEGRATION.
MATHEMATICAL CENTRE TRACT 8(1970).

EXAMPLE OF USE:

THE SOLUTION OF THE DIFFERENTIAL EQUATION
DY/DX=1-2*(X**2+Y), X>=0,
Y=0 , X =0,
IS REPRESENTED BY THE PARABOLA Y=X*(1-X);
WE WOULD LIKE TO FIND THE VALUE OF X FOR WHICH THE CURVE
OF THE SOLUTION INTERSECTS THE LINE Y+X=0.
THE SOLUTION CAN BE OBTAINED BY THE FOLLOWING PROGRAM:

```
"BEGIN" "COMMENT" INTEGRATION OF DY/DX=1-2*(Y+X**2), X>=0,
    Y(0)=0,UNTIL THE CONDITION Y+X=0 IS SATISFIED;
    "PROCEDURE" RK4A(X,XA,B,Y,YA,FXY,E,D,FI,XDIR,POS);
    "CODE" 33016;
    "REAL" X,Y;"ARRAY" D[0:4],E[0:5];
    E[0]:=E[1]:=E[2]:=E[3]:=E[4]:=E[5]:="-6;
    RK4A(X,0,X+Y,Y,0,1-2*(X*X+Y),E,D,"TRUE","TRUE","TRUE");
    OUTPUT(61,"("10B"("X="")"+D.9D,10B"("EXACTLY:")"2B+D.9D/,
    10B"("Y="")"+D.9D/,10B"("Y-X*(1-X)="")"+.10D")",X,2,
    Y,Y-X*(1-X))
"END"
    "EOP"
```

THE PROGRAM PRINTS THE FOLLOWING RESULTS:

```
X=1.9999998554      EXACTLY: 2.0000000000

Y=-1.9999995347427

Y-X*(1-X)=0.0000000313
```

SOURCE TEXT(S):

```
"CODE"" 33016 ;
   "PROCEDURE" RK4A(X, XA, B, Y, YA, FXY, E, D, FI, XDIR,
   POS); "VALUE" FI, XDIR, POS; "BOOLEAN" FI, XDIR, POS;
   "REAL" X, XA, B, Y, YA, FXY; "ARRAY" E, D;
   "BEGIN" "INTEGER" I;
      "BOOLEAN" IV, FIRST, FIR, REJ;
      "REAL" K0, K1, K2, K3, K4, K5, FHM, ABSH, DISCR, S, XL,
      COND0, S1, COND1, YL, HMIN, H, ZL, TOL, HL, MU, MU1;
      "ARRAY" E1[1:2];

      "BOOLEAN" "PROCEDURE" ZEROIN(X,Y,FX,EPS) ; "REAL" X,Y,FX,EPS ;
      "CODE"              34150 ;

      "PROCEDURE" RKSTEP(X, XL, H, Y, YL, ZL, FXY, D);
      "VALUE" XL, YL, ZL, H; "REAL" X, XL, H, Y, YL, ZL, FXY;
      "INTEGER" D;
      "BEGIN" "IF" D = 2 "THEN" "GOTO" INTEGRATE; "IF" D = 3 "THEN"
      "BEGIN" X:= XL; Y:= YL; K0:= FXY * H "END"
      "ELSE" "IF" D = 1 "THEN" K0:= ZL * H "ELSE" K0:= K0 * MU;
      X:= XL + H / 4.5; Y:= YL + K0 / 4.5; K1:= FXY * H;
      X:= XL + H / 3; Y:= YL + (K0 + K1 * 3) / 12;
      K2:= FXY * H; X:= XL + H * .5;
      Y:= YL + (K0 + K2 * 3) / 8; K3:= H * FXY;
      X:= XL + H * .8;
      Y:= YL + (K0 * 53 - K1 * 135 + K2 * 126 + K3 *
      56) / 125; K4:= FXY * H; "IF" D <= 1 "THEN"
      "BEGIN" X:= XL + H;
         Y:= YL + (K0 * 133 - K1 * 378 + K2 * 276 + K3
         * 112 + K4 * 25) / 168; K5:= FXY * H;
         DISCR:= ABS(K0 * 21 - K2 * 162 + K3 * 224 - K4
         * 125 + K5 * 42) / 14; "GOTO" END
      "END";
   INTEGRATE: X:= XL + H;
      Y:= YL + ( - K0 * 63 + K1 * 189 - K2 * 36 - K3 *
      112 + K4 * 50) / 28; K5:= FXY * H;
      Y:= YL + (K0 * 35 + K2 * 162 + K4 * 125 + K5 *
      14) / 336;
   END:
   "END" RKSTEP;                                    "COMMENT"
```

```
;

        "REAL" "PROCEDURE" FZERO;
        "BEGIN" "IF" IV "THEN"
           "BEGIN" "IF" S = XL "THEN" FZERO:= COND0 "ELSE" "IF" S = S1
              "THEN" FZERO:= COND1 "ELSE"
              "BEGIN" RKSTEP(X, XL, S - XL, Y, YL, ZL, FXY, 3);
                 FZERO:= B
              "END"
           "END"
           "ELSE"
           "BEGIN" "IF" S = YL "THEN" FZERO:= COND0 "ELSE" "IF" S = S1
              "THEN" FZERO:= COND1 "ELSE"
              "BEGIN" RKSTEP(Y, YL, S - YL, X, XL, ZL, 1 /
                 FXY, 3); FZERO:= B
              "END"
           "END"
     "END" FZERO;

     "IF" FI "THEN"
     "BEGIN" D[3]:= XA; D[4]:= YA; D[0]:= 1 "END";
     D[1]:= 0; X:= XL:= D[3]; Y:= YL:= D[4]; IV:= D[0] > 0;
     FIRST:= FIR:= "TRUE"; HMIN:= E[0] + E[1];
     H:= E[2] + E[3]; "IF" H < HMIN "THEN" HMIN:= H;
CHANGE: ZL:= FXY; "IF" ABS(ZL) <= 1 "THEN"
"BEGIN" "IF" "NOT"IV "THEN"
   "BEGIN" D[2]:= H:= H / ZL; D[0]:= 1;
      IV:= FIRST:= "TRUE"
   "END";
   "IF" FIR "THEN" "GOTO" A; I:= 1; "GOTO" AGAIN
"END"
"ELSE"
"BEGIN" "IF" IV "THEN"
   "BEGIN" "IF" "NOT"FIR "THEN" D[2]:= H:= H * ZL; D[0]:= - 1;
      IV:= "FALSE"; FIRST:= "TRUE"
   "END";
   "IF" FIR "THEN"
   "BEGIN" H:= E[0] + E[1];
   A: "IF" ("IF" FI "THEN" ("IF" IV "EQUIV" XDIR "THEN" H "ELSE"
      H * ZL) < 0 "EQUIV" POS "ELSE" H * D[2] < 0) "THEN" H:= - H
   "END";
      I:= 1
"END";
```

"COMMENT"

```
    AGAIN: ABSH:= ABS(H); "IF" ABSH < HMIN "THEN"
       "BEGIN" H:= SIGN(H) * HMIN; ABSH:= HMIN "END";
       "IF" IV "THEN"
    "BEGIN" RKSTEP(X, XL, H, Y, YL, ZL, FXY, I);
       TOL:= E[2] * ABS(K0) + E[3] * ABSH
    "END"
    "ELSE"
    "BEGIN" RKSTEP(Y, YL, H, X, XL, 1 / ZL, 1 / FXY, I);
       TOL:= E[0] * ABS(K0) + E[1] * ABSH
    "END";
    REJ:= DISCR > TOL; MU:= TOL / (TOL + DISCR) + .45;
    "IF" REJ "THEN"
    "BEGIN" "IF" ABSH <= HMIN "THEN"
       "BEGIN" "IF" IV "THEN"
          "BEGIN" X:= XL + H; Y:= YL + K0 "END"
          "ELSE"
          "BEGIN" X:= XL + K0; Y:= YL + H "END";
          D[1]:= D[1] + 1; FIRST:= "TRUE"; "GOTO" NEXT
       "END";
       H:= H * MU; I:= 0; "GOTO" AGAIN
       "END" REJ;
       "IF" FIRST "THEN"
       "BEGIN" FIRST:= FIR; HL:= H; H:= MU * H; "GOTO" ACCEPT
       "END";
       FHM:= MU * H / HL + MU = MU1; HL:= H; H:= FHM * H;
    ACCEPT: "IF" IV "THEN" RKSTEP(X, XL, HL, Y, YL, ZL, FXY,
       2) "ELSE" RKSTEP(Y, YL, HL, X, XL, ZL, 1 / FXY, 2);
       MU1:= MU;
    NEXT: "IF" FIR "THEN"
       "BEGIN" FIR:= "FALSE"; COND0:= B;
          "IF" "NOT"(FI "OR" REJ) "THEN" H:= D[2]
       "END"
       "ELSE"
       "BEGIN" D[2]:= H; COND1:= B;
          "IF" COND0 * COND1 <= 0 "THEN" "GOTO" ZERO;
          COND0:= COND1
       "END";
       D[3]:= XL:= X; D[4]:= YL:= Y; "GOTO" CHANGE;
    ZERO: E1[1]:= E[4]; E1[2]:= E[5];
       S1:= "IF" IV "THEN" X "ELSE" Y;
       S:= "IF" IV "THEN" XL "ELSE" YL ;
       ZEROIN(S,S1,FZERO,ABS(E1[1]*S)+ABS(E1[2])) ;
       S1:= "IF" IV "THEN" X "ELSE" Y ;
       "IF" IV "THEN" RKSTEP(X, XL, S - XL, Y, YL, ZL, FXY, 3)
       "ELSE" RKSTEP(Y, YL, S - YL, X, XL, ZL, 1 / FXY,
       3); D[3]:= X; D[4]:= Y
    "END" RK4A;
          "EOP"
```

AUTHOR: J.A.ZONNEVELD.

CONTRIBUTORS: M.BAKKER AND I.BRINK.

INSTITUTE: MATHEMATICAL CENTRE.

RECEIVED: 730715.

BRIEF DESCRIPTION:

RK4NA IS USED TO INTEGRATE THE VECTOR INITAL VALUE PROBLEM:
DX[J]/DX[0]=F(J,X[0],...,X[N]) , X[0]>=XA[0],
X[J]=XA[J] , X[0]=XA[0], J=1,...,N,
WHERE X[0] IS THE INDEPENDENT VARIABLE AND X[1],...,X[N] ARE
THE DEPENDENT VARIABLES;RK4NA CHOOSES FROM AMONGST X[0],...,X[N]
AS INTEGRATION VARIABLE ,A VARIABLE SUCH THAT THE ABSOLUTE VALUE
OF THE DERIVATIVE OF THE OTHER VARIABLES WITH RESPECT TO THE ONE
CHOSEN ARE <=1;THE INTEGRATION IS STOPPED,IF A CERTAIN CONDITION,
TO BE SUPPLIED BY THE USER,
ON X[0],...,X[N] IS SATISFIED.

KEYWORDS:

RUNGE-KUTTA METHODS,
SYSTEM OF FIRST ORDER DIFFERENTIAL EQUATIONS,
INITIAL VALUE PROBLEM,
DYNAMIC INTEGRATION INTERVAL,

CALLING SEQUENCE:

THE HEADING OF THE PROCEDURE READS:

```
"PROCEDURE" RK4NA(X,XA,B,FXJ,J,E,D,FI,N,L,POS);
"VALUE" FI,N,L,POS;
"INTEGER" J,N,L;
"BOOLEAN" FI,POS;
"REAL" B,FXJ;
"ARRAY" X,XA,E,D;
```

THE MEANING OF THE FORMAL PARAMETERS IS:

```
X:      <ARRAY IDENTIFIER>;
        "ARRAY" X[0:N];
        X[0] IS THE INDEPENDENT VARIABLE,
        X[1],...,X[N] ARE THE DEPENDENT VARIABLES;
        X[0],...,X[N] ARE USED AS JENSEN PARAMETERS;
        EXIT:THE SOLUTION AT B=0;
XA:     <ARRAY IDENTIFIER>;
        "ARRAY" XA[0:N];
        ENTRY: THE INITIAL VALUES OF X[0],...,X[N];
B:      <ARITHMETIC EXPRESSION>;
        B DEPENDS ON X[0],...,X[N];
        IF THE EQUATION B=0 IS SATISFIED WITHIN A
        CERTAIN TOLERANCE,THE INTEGRATION IS TERMINATED;
        B IS EVALUATED AND TESTED FOR CHANGE OF SIGN AT THE
        END OF EACH STEP;
        FOR THE TOLERANCE SEE ALSO THE EXPLICATION OF E;
FXJ:    <ARITHMETIC EXPRESSION>;
        FXJ DEPENDS ON X[0],...,X[N] AND J, DEFINING THE RIGHT
        HAND SIDE OF THE DIFFERENTIAL EQUATION;
        AT EACH CALL IT DELIVERS :DX[J]/DX[0];
J:      <VARIABLE>;
        J IS USED AS A JENSEN PARAMETER IN THE ACTUAL PARAMETER
        CORRESPONDING TO FXJ, TO DENOTE THE NUMBER OF THE EQUATION
        REQUIRED;
E:      <ARRAY IDENTIFIER>;
        "ARRAY" E[0:2*N+3];
        ENTRY: E[2*J] AND E[2*J+1] , 0<=J<=N ,
        ARE THE RELATIVE AND THE ABSOLUTE TOLERANCE ,
        RESPECTIVELY, ASSOCIATED WITH X[J];
        E[2*N+2] AND E[2*N+3] ARE THE RELATIVE AND ABSOLUTE
        TOLERANCE  USED IN THE DETERMINATION OF THE ZERO OF B;
D:      <ARRAY IDENTIFIER>;
        "ARRAY" D[0:N+3];
        AFTER COMPLETION OF EACH STEP WE HAVE :
        ENTIER(D[0]+.5) IS THE NUMBER OF STEPS SKIPPED;
        D[2] IS THE STEP LENGTH;
        D[J+3] IS THE LAST VALUE OF X[J], J=0,...,J=N;
```

FI: <BOOLEAN EXPRESSION>F
IF FI="TRUE" THEN THE INTEGRATION IS STARTED WITH INITIAL CONDITIONS X[J]=XA[J];IF FI="FALSE" THEN THE INTEGRATION IS CONTINUED WITH X[J]=D[J+3];

N: <ARITHMETIC EXPRESSION>;
THE NUMBER OF EQUATIONS;

L: <ARITHMETIC EXPRESSION>;
AN INTEGER TO BE SUPPLIED BY THE USER, 0<=L<=N.
SEE ALSO THE EXPLICATION OF POS;

POS: <BOOLEAN EXPRESSION>F
IF FI="TRUE" THEN THE INTEGRATION STARTS IN SUCH A WAY THAT X[L] INCREASES IF POS="TRUE" AND X[L] DECREASES IF POS="FALSE";
IF FI="FALSE" THEN POS IS OF NO SIGNIFICANCE.

DATA AND RESULTS: SEE REF[1].

PROCEDURES USED:

ZEROIN = CP34150.

REQUIRED CENTRAL MEMORY:

EXECUTION FIELD LENGTH: THREE ARRAYS OF ORDER N AND ONE OF ORDER 6 * N ARE USED.

RUNNING TIME:DEPENDS STRONGLY ON THE DIFFERENTIAL EQUATIONS TO SOLVE.

LANGUAGE:ALGOL 60.

METHOD AND PERFORMANCE : A 5-TH ORDER RUNGE-KUTTA METHOD IS USED; SEE REF [1] (RK4NA IS AN ADAPTED VERSION OF RK4N).

REFERENCES:

[1].J.A.ZONNEVELD.
AUTOMATIC NUMERICAL INTEGRATION.
MATHEMATICAL CENTRE TRACT 8 (1970).

EXAMPLE OF USE:

THE PERIOD OF THE SOLUTION OF THE VAN DER POL EQUATION
DX[1]/DT = X[2]
DX[2]/DT = 10*(1-X[1]**2)*X[2]-X[1], T>=0,
CAN BE OBTAINED BY THE FOLLOWING PROGRAM:

```
"BEGIN" "COMMENT" VAN DER POL;
    "PROCEDURE" RK4NA(X,XA,B,FXY,J,E,D,FI,N,L,POS);
    "CODE" 33017;
    "REAL" X0;

    "PROCEDURE" PRINT(X);"ARRAY" X;
    OUTPUT(61,"("/,4(+ZD.10D3B)")",X[0],X[1],X[2],X0);

    "INTEGER" J,K;"BOOLEAN" FIRST;
    "ARRAY" E[0:7],XA,X[0:2],D[0:5];
    "FOR" K:=0,1,2,3,4,5 "DO" E[K]:=.1"-6; E[6]:=E[7]:="-8 ;
    OUTPUT(61,"(""("VAN DER POL")",/BB"("EPS=")"D.10D,/BB
    "("THE VALUES OF X[0],X[1],X[2],P,RESPECTIVELY")"")",E[0]);
    X0:=XA[0]:=XA[2]:=0;XA[1]:=2;J:=0;PRINT(XA);
    FIRST:="TRUE";
    "FOR" J:=1,2,3,4 "DO"
    "BEGIN" RK4NA(X,XA,X[2],"IF" K=1 "THEN" X[2] "ELSE"
        10*(1-X[1]**2)*X[2]-X[1],K,E,D,FIRST,2,0,"TRUE");
        X0:=X[0]-X0;PRINT(X);FIRST:="FALSE"; X0:=X[0]
    "END"
"END"
    "EOP"
```

THE PROGRAM PRINTS THE FOLLOWING RESULTS:

```
VAN DER POL

EPS=0.0000001000

THE VALUES OF X[0],X[1],X[2],P,RESPECTIVELY:

+0.00000000  +2.00000000  +0.00000000  +0.00000000

+9.32386570  -2.01428560  +0.00000000  +9.32386570

+18.86305411 +2.01428557  +0.00000000  +9.53918840

+28.40224194  -2.01428858  -0.00000000  +9.53918783

+37.94143003  +2.01428558  +0.00000000  +9.53918809
```

SOURCE TEXT(S):

```
"CODE"" 33017 ;
"PROCEDURE" RK4NA(X, XA, B, FXJ, J, E, D, FI, N, L, POS);
"VALUE" FI, N, L, POS; "INTEGER" J, N, L; "BOOLEAN" FI, POS;
"REAL" B, FXJ; "ARRAY" X, XA, E, D;
"BEGIN" "INTEGER" I, IV, IV0;
   "BOOLEAN" FIR, FIRST, REJ;
   "REAL" H, COND0, COND1, FHM, ABSH, TOL, FH, MAX, X0,
   X1, S, HMIN, HL, MU, MU1;
   "ARRAY" XL, DISCR, Y[0:N], K[0:5,0:N], E1[1:2];

   "BOOLEAN" "PROCEDURE" ZEROIN(X,Y,FX,EPS) ; "REAL" X,Y,FX,EPS ;
   "CODE"                34150 ;

   "PROCEDURE" RKSTEP(H, D); "VALUE" H, D; "INTEGER" D; "REAL" H;
   "BEGIN" "INTEGER" I;


      "PROCEDURE" F(T); "VALUE" T; "INTEGER" T;
      "BEGIN" "INTEGER" I;
      "REAL" P;
      "FOR" J:= 1 "STEP" 1 "UNTIL" N "DO" Y[J]:= FXJ;
      P:= H / Y[IV];
      "FOR" I:= 0 "STEP" 1 "UNTIL" N "DO"
      "BEGIN" "IF" I ^= IV "THEN" K[T,I]:= Y[I] * P "END"
"END" F;

"IF" D = 2 "THEN" "GOTO" INTEGRATE; "IF" D = 3 "THEN"
"BEGIN" "FOR" I:= 0 "STEP" 1 "UNTIL" N "DO" X[I]:= XL[I];
   F(0)
"END"
"ELSE" "IF" D = 1 "THEN"
"BEGIN" "REAL" P;
   P:= H / Y[IV];
   "FOR" I:= 0 "STEP" 1 "UNTIL" N "DO"
   "BEGIN" "IF" I ^= IV "THEN" K[0,I]:= P * Y[I] "END"
"END"
"ELSE"
"FOR" I:= 0 "STEP" 1 "UNTIL" N "DO"
"BEGIN" "IF" I ^= IV "THEN" K[0,I]:= K[0,I] * MU "END";
"FOR" I:= 0 "STEP" 1 "UNTIL" N "DO" X[I]:= XL[I] + ("IF" I
= IV "THEN" H "ELSE" K[0,I]) / 4.5; F(1);
"FOR" I:= 0 "STEP" 1 "UNTIL" N "DO" X[I]:= XL[I] + ("IF" I
= IV "THEN" H * 4 "ELSE" (K[0,I] + K[1,I] * 3)) / 12;
F(2);
"FOR" I:= 0 "STEP" 1 "UNTIL" N "DO" X[I]:= XL[I] + ("IF" I
= IV "THEN" H * .5 "ELSE" (K[0,I] + K[2,I] * 3) / 8);
F(3);                                                    "COMMENT"
```

```
        "FOR" I:= 0 "STEP" 1 "UNTIL" N "DO" X[I]:= XL[I] + ("IF" I
        = IV "THEN" H * .8 "ELSE" (K[0,I] * 53 - K[1,I] * 135
        + K[2,I] * 126 + K[3,I] * 56) / 125); F(4);
        "IF" D <= 1 "THEN"
          "BEGIN" "FOR" I:= 0 "STEP" 1 "UNTIL" N "DO" X[I]:= XL[I] +
            ("IF" I = IV "THEN" H "ELSE" (K[0,I] * 133 -
            K[1,I] * 378 + K[2,I] * 276 + K[3,I] * 112 +
            K[4,I] * 25) / 168); F(5);
            "FOR" I:= 0 "STEP" 1 "UNTIL" N "DO"
            "BEGIN" "IF" I ^= IV "THEN" DISCR[I]:= ABS(K[0,I] * 21
              - K[2,I] * 162 + K[3,I] * 224 - K[4,I] *
              125 + K[5,I] * 42) / 14
            "END";
            "GOTO" END
          "END";
      INTEGRATE: "FOR" I:= 0 "STEP" 1 "UNTIL" N "DO" X[I]:= XL[I]
          + ("IF" I = IV "THEN" H "ELSE" ( - K[0,I] * 63 + K[1,I]
          * 189 - K[2,I] * 36 - K[3,I] * 112 + K[4,I] * 50)
          / 28); F(5);
          "FOR" I:= 0 "STEP" 1 "UNTIL" N "DO"
          "BEGIN" "IF" I ^= IV "THEN" X[I]:= XL[I] + (K[0,I] * 35
            + K[2,I] * 162 + K[4,I] * 125 + K[5,I] * 14) / 336
          "END" ;
          END ..
          "END" RKSTEP ;


"REAL" "PROCEDURE" FZERO;
"BEGIN" "IF" S = X0 "THEN" FZERO:= COND0 "ELSE" "IF" S = X1
    "THEN" FZERO:= COND1 "ELSE"
    "BEGIN" RKSTEP(S - XL[IV], 3); FZERO:= B "END"
"END" FZERO;

"IF" FI "THEN"
"BEGIN" "FOR" I:= 0 "STEP" 1 "UNTIL" N "DO" D[I + 3]:= XA[I];
   D[0]:= D[2]:= 0
"END";
D[1]:= 0;
"FOR" I:= 0 "STEP" 1 "UNTIL" N "DO" X[I]:= XL[I]:= D[I + 3];
   IV:= D[0]; H:= D[2]; FIRST:= FIR:= "TRUE"; Y[0]:= 1;
   "GOTO" CHANGE;
AGAIN: ABSH:= ABS(H); "IF" ABSH < HMIN "THEN"
   "BEGIN" H:= "IF" H > 0 "THEN" HMIN "ELSE" - HMIN;
      ABSH:= ABS(H)
   "END";
   RKSTEP(H, I); REJ:= "FALSE"; FHM:= 0;
   "FOR" I:= 0 "STEP" 1 "UNTIL" N "DO"
   "BEGIN" "IF" I ^= IV "THEN"
      "BEGIN" TOL:= E[2 * I] * ABS(K[0,I]) + E[2 * I + 1]
         * ABSH; REJ:= TOL < DISCR[I] "OR" REJ;
         FH:= DISCR[I] / TOL; "IF" FH > FHM "THEN" FHM:= FH
      "END"
   "END";                                                    "COMMENT"
```

```
      MU:= 1 / (1 + FHM) + .45; "IF" REJ "THEN"
      "BEGIN" "IF" ABSH <= HMIN "THEN"
         "BEGIN" "FOR" I:= 0 "STEP" 1 "UNTIL" N "DO"
            "BEGIN" "IF" I ^= IV "THEN" X[I]:= XL[I] + K[0,I]
               "ELSE" X[I]:= XL[I] + H
            "END";
            D[1]:= D[1] + 1; FIRST:= "TRUE"; "GOTO" NEXT
         "END";
         H:= H * MU; I:= 0; "GOTO" AGAIN
      "END";
      "IF" FIRST "THEN"
      "BEGIN" FIRST:= FIR; HL:= H; H:= MU * H; "GOTO" ACCEPT
      "END";
      FH:= MU * H / HL + MU - MU1; HL:= H; H:= FH * H;
ACCEPT: RKSTEP(HL, 2); MU1:= MU;
NEXT: "IF" FIR "THEN"
      "BEGIN" FIR:= "FALSE"; COND0:= B;
         "IF" "NOT"(FI "OR" REJ) "THEN" H:= D[2]
      "END"
      "ELSE"
      "BEGIN" D[2]:= H; COND1:= B;
         "IF" COND0 * COND1 <= 0 "THEN" "GOTO" ZERO;
         COND0:= COND1
      "END";
      "FOR" I:= 0 "STEP" 1 "UNTIL" N "DO" D[I + 3]:= XL[I]:= X[I];
CHANGE: IV0:= IV;
      "FOR" J:= 1 "STEP" 1 "UNTIL" N "DO" Y[J]:= FXJ;
      MAX:= ABS(Y[IV]);
      "FOR" I:= 0 "STEP" 1 "UNTIL" N "DO"
      "BEGIN" "IF" ABS(Y[I]) > MAX "THEN"
         "BEGIN" MAx:= ABS(Y[I]); IV:= I "END"
      "END";
      "IF" IV0 ^= IV "THEN"
      "BEGIN" FIRST:= "TRUE"; D[0]:= IV;
         D[2]:= H:= Y[IV] / Y[IV0] * H
      "END";
      X0:= XL[IV]; "IF" FIR "THEN"
      "BEGIN" HMIN:= E[0] + E[1];
         "FOR" I:= 1 "STEP" 1 "UNTIL" N "DO"
         "BEGIN" H:= E[2 * I] + E[2 * I + 1];
            "IF" H < HMIN "THEN" HMIN:= H
         "END";
         H:= E[2 * IV] + E[2 * IV + 1];
            "IF" (FI "AND" (Y[L]/Y[IV]*H<0 "EQUIV" POS)) "OR"
            ("NOT" FI "AND" D[2]*H<0) "THEN" H:= -H
      "END";
      I:= 1; "GOTO" AGAIN;
ZERO: E1[1]:= E[2 * N + 2]; E1[2]:= E[2 * N + 3];
      X1:=X[IV] ; S:=X0 ;
      ZEROIN(S,X1,FZERO,ABS(E1[1]*S) + ABS(E1[2])) ; X0:=S ; X1:=X[IV];
      RKSTEP(X0 - XL[IV], 3);
      "FOR" I:= 0 "STEP" 1 "UNTIL" N "DO" D[I + 3]:= X[I]
"END" RK4NA;
      "EOP"
```

AUTHOR: J.A.ZONNEVELD.

CONTRIBUTORS: M.BAKKER AND I.BRINK.

INSTITUTE: MATHEMATICAL CENTRE.

RECEIVED: 730715.

BRIEF DESCRIPTION:

RK5NA CAN BE USED TO INTEGRATE THE SYSTEM OF
FIRST ORDER DIFFERENTIAL EQUATIONS

```
DX[J] / DX[0] = F(J, X[0], ... ,X[N]) / F(0, X[0], ... ,X[N]),
                J = 1, ..., N,
```

WHERE F(J, X[0], ..., X[N]) AND F(0, X[0], ..., X[N]) REMAIN FINITE; THE ARC LENGTH S IS INTRODUCED AS INTEGRATION VARIABLE ;
THE SYSTEM SOLVED IS

```
DX[J] / DS = F(J, X[0], ..., X[N]) / SQRT(A), J = 0, ..., N,
         A = SUM(I, 0, N, F(I, X[0], ..., X[N]) ** 2) ;
```

THE INTEGRATION STOPS IF SOME CONDITION ON X[0], ..., X[N] ,
TO BE SUPPLIED BY THE USER, IS SATISFIED WITHIN CERTAIN TOLERANCES.

KEYWORDS:

RUNGE-KUTTA METHODS,
DIFFERENTIAL EQUATIONS,
INITIAL VALUE PROBLEMS.

CALLING SEQUENCE:

THE HEADING OF THE PROCEDURE READS:
"PROCEDURE" RK5NA (X, XA, B, FXJ, J, E, D, FI, N, L, POS);
"VALUE" FI, N, L, POS; "INTEGER" J, N, L; "BOOLEAN" FI, POS;
"REAL" B, FXJ; "ARRAY" X, XA, E, D;

THE MEANING OF THE FORMAL PARAMETERS IS:

X: <ARRAY IDENTIFIER>;
"ARRAY" X[0 : N];
THE DEPENDENT VARIABLES;
X[0], ..., X[N] CAN BE USED AS JENSEN PARAMETERS;

XA: <ARRAY IDENTIFIER>;
"ARRAY" XA[0 : N];
ENTRY: THE INITIAL VALUES OF X[J], J = 0, ..., N;

B: <ARITHMETIC EXPRESSION>;
B DEPENDS ON X[0], ..., X[N];
IF,WITHIN SOME TOLERANCE,B = 0 THEN THE INTEGRATION IS TERMINATED; SEE ALSO THE EXPLANATION OF THE PARAMETER E;

FXJ: <ARITHMETIC EXPRESSION>;
THE RIGHT HAND SIDE OF THE DIFFERENTIAL EQUATION;
FXJ DEPENDS ON X[0], ..., X[N], J, GIVING THE VALJE OF
DX[J] / DX[0];

J: <VARIABLE>;
J IS USED AS A JENSEN PARAMETER TO DENOTE,IN THE ACTUAL
PARAMETER CORRESPONDING TO FXJ,THE NUMBER OF THE FUNCTION
REQUIRED;

E: <ARRAY IDENTIFIER>;
"ARRAY" E[0 : 2 * N + 3];
ENTRY:
E[2 * J] AND E[2 * J + 1] ARE THE RELATIVE AND THE ABSOLUTE
TOLERANCE,RESPECTIVELY,ASSOCIATED WITH X[J],J = 0,..., N, WHILE
E[2 * N + 2] AND E[2 * N + 3] ARE THE ONES ASSOCIATED WITH B;

D: <ARRAY IDENTIFIER>;
"ARRAY" D[1 : N + 3];
AFTER COMPLETION OF EACH STEP WE HAVE:

ABS(D[1])	THE ARC LENGTH,
D[2]	THE STEP LENGTH,
D[I + 3]	THE LATEST VALUE OF X[I], I = 0, ..., N;

FI: <BOOLEAN EXPRESSION>;
IF FI = "TRUE" THEN THE INTEGRATION IS STARTED WITH INITIAL
CONDITIONS X[I] = XA[I], I = 0, ..., N;
IF FI = "FALSE" THEN THE INTEGRATION IS CONTINUED WITH
X[I] = D[I + 3];

N: <ARITHMETIC EXPRESSION>;
THE NUMBER OF EQUATIONS;

L: <ARITHMETIC EXPRESSION>;
1 <= L <= N; SEE THE EXPLANATION OF POS;
POS: <BOOLEAN EXPRESSION>;
IF FI = "TRUE" THEN THE INTEGRATION STARTS IN SUCH A WAY THAT

$$X[L] \quad \begin{cases} \text{INCREASES, IF POS = "TRUE"}, \\ \\ \text{DECREASES, IF POS = "FALSE"}; \end{cases}$$

IF FI = "FALSE" THEN POS IS IGNORED.

DATA AND RESULTS: SEE REF[1].

PROCEDURES USED:

ZEROIN = CP34150.

REQUIRED CENTRAL MEMORY:

EXECUTION FIELD LENGTH: THREE ARRAYS OF ORDER N AND ONE OF ORDER 6 * N ARE USED.

RUNNING TIME: DEPENDS ON THE SYSTEM TO SOLVE.

LANGUAGE: ALGOL 60.

METHOD AND PERFORMANCE: A 5-TH ORDER RUNGE-KUTTA METHOD IS USED; SEE REF [1] (RK5NA IS AN ADAPTED VERSION OF RK5N).

REFERENCES:

[1]. J.A.ZONNEVELD,
AUTOMATIC NUMERICAL INTEGRATION,
MATH.CENTRE TRACT 8 (1970).

EXAMPLE OF USE:

THE VAN DER POL EQUATION IN THE PHASE PLANE

DX[1] / DX[0] = (10*(1-X[0]**2)*X[1]-X[0])/X[1]

CAN BE INTEGRATED BY THE PROCEDURE RK5NA; THE STARTING VALUES ARE X[0] = 2, X[1] = 0. THE INTEGRATION PROCEEDS UNTIL THE NEXT ZERO OF X[1],THEN IT CONTINUES UNTIL THE NEXT ZERO AND SO ON UNTIL THE FOURTH ZERO IS ENCOUNTERED.IN THE EXAMPLE THE OUTPUT IS GIVEN FOR THE TOLERANCES E[0]=E[1]=E[2]=E[3]= "-6, E[4]=E[5]= "-10.
THE PROGRAM READS AS FOLLOWS:

```
"BEGIN" "COMMENT" VAN DER POL IN THE PHASE PLANE;
    "PROCEDURE" RK5NA(X,XA,B,FXJ,J,E,D,FI,N,L,POS); "CODE" 33018;
    "INTEGER" J,K; "BOOLEAN" FIRST;
    "ARRAY" E[0:5],XA,X[0:1],D[1:4];
    "PROCEDURE" PRINT(X); "ARRAY" X;
    "BEGIN" OUTPUT(61,"("/B"("X[0]=")"+D.10D,
        10B"("X[1]=")"+D.10D,10B"("S=")"3D.10D")",X[0],
        X[1],ABS(D[1]))
    "END";
    "FOR" K:=0,1,2,3 "DO" E[K]:="-6 ; E[4]:=E[5]:="-10 ;  D[1]:=0;
    XA[0]:=2; XA[1]:=0; J:=0; PRINT(XA); AA:
    FIRST:=J=0;
    RK5NA(X,XA,X[1],"IF" K=0 "THEN" X[1] "ELSE"
    10*(1-X[0]**2)*X[1]-X[0],K,E,D,FIRST,1,1,"FALSE");;J:=J+1;
    PRINT(X); "IF" J<4 "THEN" "GO TO" AA
"END"
    "EOP"
```

RESULTS:

```
X[0]=+2.0000000000    X[1]=+0.0000000000    S=000.0000000000

X[0]=-2.0142853657    X[1]=-0.0000000012    S=029.3873834087

X[0]=+2.0142853659    X[1]=+0.0000000001    S=058.7884331939

X[0]=-2.0142853659    X[1]=-0.0000000000    S=088.1894829781

X[0]=+2.0142853659    X[1]=+0.0000000000    S=117.5905327623
```

```
SOURCE TEXT(S):
"CODE"" 33018 ;
    "PROCEDURE" RKSNA(X, XA, B, FXJ, J, E, D, FI, N, L, POS);
    "VALUE" FI, N, L, POS; "INTEGER" J, N, L; "BOOLEAN" FI, POS;
    "REAL" B, FXJ; "ARRAY" X, XA, E, D;
    "BEGIN" "INTEGER" I;
"BOOLEAN" FIRST, FIR, REJ;
"REAL" FHM, S, S0, COND0, S1, COND1, H, ABSH, TOL, FH,
HL, MU, MU1;
"ARRAY" Y, XL, DISCR[0:N], K[0:5,0:N], E1[1:2];
"REAL" "PROCEDURE" SUM(J,A,B,XJ) ; "INTEGER" J,A,B ; "REAL" XJ ;
"BEGIN" "REAL" S ; S:= 0 ;
    "FOR" J:=A "STEP" 1 "UNTIL" B "DO" S:=S+XJ ; SUM:= S
"END" SUM ;
"BOOLEAN" "PROCEDURE" ZEROIN(X,Y,FX,EPS) ; "REAL" X,Y,FX,EPS ;
"CODE"                34150 ;
"PROCEDURE" RKSTEP(H, D); "VALUE" H, D; "INTEGER" D; "REAL" H;
"BEGIN" "INTEGER" I;
    "PROCEDURE" F(T); "VALUE" T; "INTEGER" T;
    "BEGIN" "INTEGER" I;
        "REAL" P;
        "FOR" J:= 0 "STEP" 1 "UNTIL" N "DO" Y[J]:= FXJ;
        P:= H / SQRT(SUM(I, 0, N, Y[I] ** 2));
        "FOR" I:= 0 "STEP" 1 "UNTIL" N "DO" K[T,I]:= Y[I] * P
    "END" F;
    "IF" D = 2 "THEN" "GOTO" INTEGRATE; "IF" D = 1 "THEN"
    "BEGIN" "FOR" I:= 0 "STEP" 1 "UNTIL" N "DO" K[0,I]:= K[0,I]
        * MU; "GOTO" A
    "END";
    "FOR" I:= 0 "STEP" 1 "UNTIL" N "DO" X[I]:= XL[I]; F(0);
A:  "FOR" I:= 0 "STEP" 1 "UNTIL" N "DO" X[I]:= XL[I] +
    K[0,I] / 4.5; F(1);
    "FOR" I:= 0 "STEP" 1 "UNTIL" N "DO" X[I]:= XL[I] + (K[0,I]
    + K[1,I] * 3) / 12; F(2);
    "FOR" I:= 0 "STEP" 1 "UNTIL" N "DO" X[I]:= XL[I] + (K[0,I]
    + K[2,I] * 3) / 8; F(3);
    "FOR" I:= 0 "STEP" 1 "UNTIL" N "DO" X[I]:= XL[I] + (K[0,I]
    * 53 - K[1,I] * 135 + K[2,I] * 126 + K[3,I] * 56)
    / 125; F(4); "IF" D <= 1 "THEN"
    "BEGIN" "FOR" I:= 0 "STEP" 1 "UNTIL" N "DO" X[I]:= XL[I] +
        (K[0,I] * 133 - K[1,I] * 378 + K[2,I] * 276 +
        K[3,I] * 112 + K[4,I] * 25) / 168; F(5);
        "FOR" I:= 0 "STEP" 1 "UNTIL" N "DO" DISCR[I]:=
        ABS(K[0,I] * 21 - K[2,I] * 162 + K[3,I] * 224
        - K[4,I] * 125 + K[5,I] * 42) / 14; "GOTO" END
    "END";
INTEGRATE: "FOR" I:= 0 "STEP" 1 "UNTIL" N "DO" X[I]:= XL[I]
    + ( - K[0,I] * 63 + K[1,I] * 189 - K[2,I] * 36 -
    K[3,I] * 112 + K[4,I] * 50) / 28; F(5);
    "FOR" I:= 0 "STEP" 1 "UNTIL" N "DO" X[I]:= XL[I] + (K[0,I]
    * 35 + K[2,I] * 162 + K[4,I] * 125 + K[5,I] * 14)
    / 336;
END;
"END" RKSTEP;                                          "COMMENT"
```

```
      "REAL" "PROCEDURE" FZERO;
      "BEGIN" "IF" S = S0 "THEN" FZERO:= COND0 "ELSE" "IF" S = S1
         "THEN" FZERO:= COND1 "ELSE"
         "BEGIN" RKSTEP(S - S0, 3); FZERO:= B "END"
         "END" FZERO;

         "IF" FI "THEN"
         "BEGIN" "FOR" I:= 0 "STEP" 1 "UNTIL" N "DO" D[I + 3]:= XA[I];
            D[1]:= D[2]:= 0
         "END";
         "FOR" I:= 0 "STEP" 1 "UNTIL" N "DO" X[I]:= XL[I]:= D[I + 3];
         S:= D[1]; FIRST:= FIR:= "TRUE"; H:= E[0] + E[1];
         "FOR" I:= 1 "STEP" 1 "UNTIL" N "DO"
         "BEGIN" ABSH:= E[2 * I] + E[2 * I + 1];
            "IF" H > ABSH "THEN" H:= ABSH
         "END";
         "IF" FI "THEN"
         "BEGIN" J:= L; "IF" FXJ * H < 0 "EQUIV" POS "THEN" H:= - H "END"
         "ELSE" "IF" D[2] * H < 0 "THEN" H:= - H; I:= 0;
      AGAIN: RKSTEP(H, I); REJ:= "FALSE"; FHM:= 0;
         ABSH:= ABS(H);
         "FOR" I:= 0 "STEP" 1 "UNTIL" N "DO"
         "BEGIN" TOL:= E[2 * I] * ABS(K[0,I]) + E[2 * I + 1] *
            ABSH; REJ:= TOL < DISCR[I] "OR" REJ;
            FH:= DISCR[I] / TOL; "IF" FH > FHM "THEN" FHM:= FH
         "END";
         MU:= 1 / (1 + FHM) + .45; "IF" REJ "THEN"
         "BEGIN" H:= H * MU; I:= 1; "GOTO" AGAIN "END";
         "IF" FIRST "THEN"
         "BEGIN" FIRST:= FIR; HL:= H; H:= MU * H "END"
         "ELSE"
         "BEGIN" FH:= MU * H / HL + MU - MU1; HL:= H; H:= FH * H
         "END";
      ACCEPT: RKSTEP(HL, 2); MU1:= MU; S:= S + HL;
         "IF" FIR "THEN"
         "BEGIN" COND0:= B; FIR:= "FALSE"; "IF" "NOT"FI "THEN" H:= D[2]
         "END"
         "ELSE"
         "BEGIN" D[2]:= H; COND1:= B;
            "IF" COND0 * COND1 <= 0 "THEN" "GOTO" ZERO;
            COND0:= COND1
         "END";
         "FOR" I:= 0 "STEP" 1 "UNTIL" N "DO" D[I + 3]:= XL[I]:= X[I];
         D[1]:= S0:= S; I:= 0; "GOTO" AGAIN;
      ZERO: E1[1]:= E[2 * N + 2]; E1[2]:= E[2 * N + 3];
         S1:=S ; S:=S0 ;
         ZEROIN(S,S1,FZERO,ABS(E1[1]*S)+ABS(E1[2])) ;
         RKSTEP(S - S0, 3);
         "FOR" I:= 0 "STEP" 1 "UNTIL" N "DO" D[I + 3]:= X[I]; D[1]:= S
      "END" RK5NA;
            "EOP"
```

AUTHOR: P.W.HEMKER.

INSTITUTE: MATHEMATICAL CENTRE.

RECEIVED: 730515.

BRIEF DESCRIPTION:

MULTISTEP SOLVES AN INITIAL VALUE PROBLEM, GIVEN AS A SYSTEM OF FIRST ORDER DIFFERENTIAL EQUATIONS, BY MEANS OF A MULTISTEP METHOD: GEARS METHOD,ADAMS-MOULTON OR ADAMS-BASHFORTH-METHOD.
IN PARTICULAR THIS METHOD IS SUITABLE FOR THE INTEGRATION OF STIFF DIFFERENTIAL EQUATIONS.

KEYWORDS:

DIFFERENTIAL EQUATIONS,
INITIAL VALUE PROBLEM,
STIFF EQUATIONS;
MULTISTEP METHOD.

CALLING SEQUENCE:

THE HEADING OF THE PROCEDURE READS:

```
"BOOLEAN" "PROCEDURE" MULTISTEP(X,XEND,Y,HMIN,HMAX,YMAX,EPS,
                    FIRST,SAVE,DERIV,AVAILABLE,JACOBIAN,STIFF,N,OUT);
"VALUE" HMIN,HMAX,EPS,XEND,N,STIFF;
"BOOLEAN" FIRST,AVAILABLE,STIFF;
"INTEGER" N;
"REAL" X,XEND,HMIN,HMAX,EPS;
"ARRAY" Y,YMAX,SAVE,JACOBIAN;
"PROCEDURE" DERIV,OUT;
```

MULTISTEP: DELIVERS THE FOLLOWING BOOLEAN VALUE:
IF DIFFICULTIES ARE ENCOUNTERED DURING THE INTEGRATION (I.E. SAVE[-1] ^= 0 "OR" SAVE[-2] ^= 0) MULTISTEP IS SET TO "FALSE", OTHERWISE MULTISTEP IS SET "TRUE".

THE MEANING OF THE FORMAL PARAMETERS IS:

X: <VARIABLE>;
THE INDEPENDENT VARIABLE X.
CAN BE USED IN DERIV, AVAILABLE ETC.;
ENTRY: THE INITIAL VALUE X0;
EXIT : THE FINAL VALUE 'XEND';

XEND: <ARITHMETIC EXPRESSION>;
THE FINAL VALUE OF X (XEND >= X);

Y: <ARRAY IDENTIFIER>;
"ARRAY" Y[1:6*N];
THE DEPENDENT VARIABLE;
ENTRY: THE INITIAL VALUES OF THE SOLUTION OF THE SYSTEM OF DIFFERENTIAL EQUATIONS: Y[I] AT X = X0;
EXIT : THE FINAL VALUES OF THE SOLUTION: Y[I] AT X = XEND;

HMIN,HMAX: <ARITHMETIC EXPRESSION>;
MINIMAL RESP. MAXIMAL STEPLENGTH BY WHICH THE INTEGRATION IS PERFORMED;

YMAX: <ARRAY IDENTIFIER>;
"ARRAY" YMAX[1:N];
ENTRY: THE ABSOLUTE LOCAL ERROR BOUND DIVIDED BY EPS
EXIT : YMAX[1] GIVES THE MAXIMAL VALUE OF THE ENTRY VALUE OF YMAX[I] AND THE VALUES OF ABS(Y[I]) DURING INTEGRATION;

EPS: <ARITHMETIC EXPRESSION>;
THE RELATIVE LOCAL ERROR BOUND;

FIRST: <IDENTIFIER>;
IF FIRST = "TRUE" THEN THE PROCEDURE STARTS ITS STRATEGY WITH A FIRST ORDER ADAMS METHOD AND A STEPLENGTH EQUAL TO HMIN. UPON COMPLETION OF A CALL FIRST:= "FALSE";
IF FIRST = "FALSE" THEN THE PROCEDURE CONTINUES INTEGRATION;

```
SAVE:     <ARRAY IDENTIFIER>;
          "ARRAY" SAVE[-38:6*N];
          IN THIS ARRAY THE PROCEDURE STORES INFORMATION WHICH CAN BE
          USED IN THE NEXT CALL WITH FIRST="FALSE";
          BESIDES THE FOLLOWING MESSAGES ARE DELIVERED:
          SAVE[ 0]=0 : AN ADAMS METHOD HAS BEEN USED;
                   1 : THE PROCEDURE SWITCHED TO GEARS METHOD;
          SAVE[-1]=0 : NO ERROR MESSAGE;
                   1 : WITH  THE HMIN SPECIFIED  THE PROCEDURE CANNOT
                       HANDLE  THE  NONLINEARITY  (DECREASE HMIN!  );
          SAVE[-2]   NUMBER OF TIMES  THAT THE  REQUESTED  LOCAL ERROR
                     BOUND WAS EXCEEDED;
          SAVE[-3]   IF SAVE[-2] IS NONZERO THEN SAVE[-3] GIVES AN
                     ESTIMATE OF THE MAXIMAL LOCAL ERROR BOUND,
                     OTHERWISE SAVE[-3]=0;
DERIV:    <PROCEDURE IDENTIFIER>;
          THE HEADING OF THIS PROCEDURE READS;
          "PROCEDURE" DERIV(DF); "ARRAY" DF;
          THIS PROCEDURE SHOULD DELIVER DY[I]/DX IN DF[I];
AVAILABLE: <BOOLEAN EXPRESSION>;
          IF AN  ANALYTICAL EXPRESSION  OF THE JACOBIAN MATRIX IS NOT
          AVAILABLE THIS EXPRESSION IS SET TO "FALSE";
          OTHERWISE THIS EXPRESSION IS SET TO "TRUE" AND THE
          EVALUATION  OF  THIS  BOOLEAN  EXPRESSION  MUST  EFFECT THE
          FOLLOWING SIDE-EFFECT;
          THE ENTRIES OF THE JACOBIAN MATRIX   D(DY[I]/DX)/DY[J]  ARE
          DELIVERED IN THE ARRAY ELEMENTS  JACOBIAN[I,J];
JACOBIAN: <ARRAY IDENTIFIER>;
          "ARRAY" JACOBIAN[1:N,1:N];
          AT EACH EVALUATION OF THE BOOLEAN EXPRESSION AVAILABLE WITH
          THE RESULT AVAILABLE:="TRUE", THE JACOBIAN MATRIX HAS TO BE
          ASSIGNED TO THIS ARRAY;
STIFF:    <BOOLEAN EXPRESSION>;
          IF STIFF = "TRUE"  THE PROCEDURE  SKIPS AN ATTEMPT TO SOLVE
          THE PROBLEM WITH ADAMS-BASHFORTH- OR ADAMS-MOULTON
          METHODS, DIRECTLY USING GEARS METHOD;
N:        <ARITHMETIC EXPRESSION>;
          THE NUMBER OF EQUATIONS;
OUT:      <PROCEDURE IDENTIFIER>;
          THE HEADING OF THIS PROCEDURE READS:
          "PROCEDURE" OUT(H,K);  "VALUE" H,K;  "REAL" H; "INTEGER" K;
          AT THE END OF EACH ACCEPTED STEP OF THE INTEGRATION PROCESS
          THIS PROCEDURE IS CALLED, THE LAST STEPLENGTH USED (H)  AND
          THE ORDER OF THE METHOD (K) ARE DELIVERED.
          AT EACH CALL OF THE  PROCEDURE OUT, THE  CURRENT VALUES  OF
          THE INDEPENDENT VARIABLE (X) AND OF THE SOLUTION (Y[I](X) )
          ARE AVAILABLE  FOR USE, MOREOVER, IN THE  NEIGHBOURHOOD  OF
          THE CURRENT VALUE OF X, ANY VALUE OF  Y[I](X SPECIFIED) CAN
          BE COMPUTED BY MEANS OF THE FOLLOWING INTERPOLATION FORMULA
          Y[I](X SPECIFIED) =
               SUM(J,0,K, Y[I+J*N] * ((X SPECIFIED - X)/H) ** J ).
```

DATA AND RESULTS: SEE REF[1].

PROCEDURES USED:

MATVEC = CP34011 ,
DET = CP34050 ,
SOL = CP34051 .

REQUIRED CENTRAL MEMORY:

EXECUTION FIELD LENGTH: CIRCA 60 + N*(4+N).

RUNNING TIME: DEPENDS STRONGLY ON THE DIFFERENTIAL EQUATION TO SOLVE.

LANGUAGE: ALGOL 60.

METHOD AND PERFORMANCE: SEE REF[1].

REFERENCES:

[1].P.W.HEMKER.
AN ALGOL 60 PROCEDURE FOR THE SOLUTION OF STIFF DIFFERENTIAL EQUATIONS.
MATH. CENTRE, AMSTERDAM. REPORT MR 128/71;

EXAMPLE OF USE:

THE SOLUTION AT X=1 AND AT X=10 OF THE DIFFERENTIAL EQUATIONS:
DY[1]/DX = 0.04 * (1-Y[1]-Y[2]) - Y[1] * ("4*Y[2] + 3"7*Y[2])
DY[2]/DX = 3"7 Y[1]**2
WITH THE INITIAL CONDITIONS AT X = 0 :
Y[1] = 0 AND Y[2] = 0
MAY BE OBTAINED BY THE FOLLOWING PROGRAM:

```
 "BEGIN"
        "BOOLEAN" "PROCEDURE" MULTISTEP(X,XEND,Y,HMIN,HMAX,YMAX,EPS,
                  FIRST,SAVE,DERIV,AVAILABLE,JACOBIAN,STIFF,N,OUT);
        "CODE" 33080;
        "BOOLEAN" FIRST;
        "INTEGER" I,J,CF,CJ,CA;
        "REAL" X,XEND,HMIN,EPS,R;
        "ARRAY" Y[1:12],YMAX[1:2],D[-40:12],JAC[1:2,1:2];

        "PROCEDURE" DER (F);"ARRAY" F;
        "BEGIN" "REAL" R; CF:=CF+1;
             F[2]:= R:= 3"7*Y[1]*Y[1];
             F[1]:= 0,04*(1-Y[1]-Y[2]) - "4*Y[1]*Y[2] - R;
        "END" F;

        "BOOLEAN" "PROCEDURE" AVAIL;
        "BEGIN" "REAL" R; CJ:= CJ+1;
            AVAIL:= "TRUE";
            JAC[2,1]:= R:= 6"7*Y[1];
            JAC[1,1]:= -0,04 - "4*Y[2] - R;
            JAC[1,2]:= -0,04 - "4*Y[1];
            JAC[2,2]:= 0
        "END" JAC AVAIL;

        "PROCEDURE" OUT(H,K); "REAL" H; "INTEGER" K; CA:= CA+1;

        LABEL:
        OUTPUT(61,"("/,"("HMIN,EPS?")",/")");
        INREAL(60,HMIN); INREAL(60,EPS);
        "IF" HMIN<0 "THEN" "GOTO" ESCAPE;

        FIRST:= "TRUE"; CA:=CF:=CJ:=0;
        X:=0; Y[1]:= Y[2]:= 0;
        YMAX[1]:= 0,0001; YMAX[2]:= 1;

        "FOR" XEND:= 1, 10 "DO"
        "BEGIN"
            MULTISTEP(X,XEND,Y,HMIN,5,YMAX,EPS,FIRST,D,DER,AVAIL,
                                                    JAC,"TRUE",2,OUT);
              OUTPUT(61,"("3(5ZD,2B),2(+,13D"+2D,2B),/")",
                                                    CA,CF,CJ,Y[1],Y[2]);
            "END";

            "GOTO" LABEL;
            ESCAPE:
        "END"

        IT DELIVERS WITH  HMIN = "-10  AND  EPS = "-9:
        240    648    2    +,3074626[602000]"-04    +,3350951[493111]"-01
        315    902    3    +,16233909[62091]"-04    +,15861383[92015]"+00
        (NON-SIGNIFICANT DIGITS ARE PLACED BETWEEN [ ] ),
```

SOURCE TEXT(S):

```
"CODE"" 33080;
"BOOLEAN" "PROCEDURE" MULTISTEP(X,XEND,Y,HMIN,HMAX,YMAX,EPS,
          FIRST,SAVE,DERIV,AVAILABLE,JACOBIAN,STIFF,N,OUT);
"VALUE" HMIN,HMAX,EPS,XEND,N,STIFF;
"BOOLEAN" FIRST,AVAILABLE,STIFF;
"INTEGER" N;
"REAL" X,XEND,HMIN,HMAX,EPS;
"ARRAY" Y,YMAX,SAVE,JACOBIAN;
"PROCEDURE" DERIV,OUT;
"BEGIN" "OWN" "BOOLEAN" ADAMS,WITH JACOBIAN;
        "OWN" "INTEGER" M,SAME,KOLD;
        "OWN" "REAL" XOLD,HOLD,A0,TOLUP,TOL,TOLDWN,TOLCONV;
        "BOOLEAN" EVALUATE,EVALUATED,DECOMPOSE,DECOMPOSED,CONV;
        "INTEGER" I,J,L,K,KNEW,FAILS;
        "REAL" H, CH, CHNEW,ERROR,DFI,C;
        "ARRAY" A[0:5],DELTA,LAST DELTA,DF[1:N], JAC[1:N, 1:N];
        "INTEGER" "ARRAY" P[1:N];

"REAL" "PROCEDURE" MATVEC(L,U,I,A,B);"CODE" 34011;
"REAL" "PROCEDURE" DET(A,N,P); "CODE" 34050;
"PROCEDURE" SOL(A,N,P,B); "CODE" 34051;

"REAL" "PROCEDURE" NORM2(AI); "REAL" AI;
"BEGIN" "REAL" S,A; S:= 0;
        "FOR" I:= 1 "STEP" 1 "UNTIL" N "DO"
        "BEGIN" A:= AI/YMAX[I]; S:= S + A * A "END";
        NORM2:= S
"END" NORM2;

"PROCEDURE" RESET;
"BEGIN" "IF" CH < HMIN/HOLD "THEN" CH:= HMIN/HOLD "ELSE"
        "IF" CH > HMAX/HOLD "THEN" CH:= HMAX/HOLD;
        X:= XOLD; H:= HOLD * CH; C:= 1;
        "FOR" J:= 0 "STEP" M "UNTIL" K*M "DO"
        "BEGIN" "FOR" I:= 1 "STEP" 1 "UNTIL" N "DO"
                Y[J+I]:= SAVE[J+I] * C;
                C:= C * CH
        "END";
        DECOMPOSED:= "FALSE"
"END" RESET;                                                    "COMMENT"
```

```
        "PROCEDURE" METHOD;
        "BEGIN" I:= -39;
                "IF" ADAMS "THEN"
                "BEGIN" "FOR" C:= 1,1,144,4,0,.5,1,.5,576,144,1,5/12,1,
                                  .75,1/6,1436,576,4,.375,1,11/12,1/3,1/24,
                                   2844,1436,1,251/720,1,25/24,35/72,
                                   5/48,1/120,0,2844,0.1
                          "DO" "BEGIN" I:= I+ 1; SAVE[I]:= C "END"
                "END" "ELSE"

                "BEGIN" "FOR" C:= 1,1,9,4,0,2/3,1,1/3,36,20.25,1,6/11,
                        1,6/11,1/11.84.028,53.778.0.25,.48,1,.7,.2,.02,
                         156.25, 108.51. .027778, 120/274, 1, 225/274,
                         85/274, 15/274, 1/274, 0, 187.69, .0047361
                         "DO" "BEGIN" I:= I + 1; SAVE[I]:= C "END"
                "END"
        "END" METHOD;

        "PROCEDURE" ORDER;
        "BEGIN" C:= EPS * EPS; J:= (K-1) * (K + 8)/2 - 38;
                "FOR" I:= 0 "STEP" 1 "UNTIL" K "DO" A[I]:= SAVE[I+J];
                TOLUP  := C * SAVE[J + K + 1];
                TOL    := C * SAVE[J + K + 2];
                TOLDWN := C * SAVE[J + K + 3];
                TOLCONV:= EPS/(2 * N * (K + 2));
                A0:= A[0];  DECOMPOSE:= "TRUE";
        "END" ORDER;

        "PROCEDURE" EVALUATE JACOBIAN;
        "BEGIN" EVALUATE:= "FALSE";
                DECOMPOSE:= EVALUATED:= "TRUE";
                "IF" AVAILABLE "THEN" "ELSE"
                "BEGIN" "REAL" D; "ARRAY" FIXY,FIXDY,DY[1:N];
                        "FOR" I:= 1 "STEP" 1 "UNTIL" N "DO"
                        FIXY[I]:= Y[I];
                        DERIV(FIXDY);
                        "FOR" J:= 1 "STEP" 1 "UNTIL" N "DO"
                        "BEGIN" D:= "IF" EPS > ABS(FIXY[J])
                        "THEN" EPS * EPS
                        "ELSE" EPS * ABS(FIXY[J]);
                        Y[J]:= Y[J] + D; DERIV(DY);
                        "FOR" I:= 1 "STEP" 1 "UNTIL" N "DO"
                        JACOBIAN[I,J]:= (DY[I]-FIXDY[I])/D;
                        Y[J]:= FIXY[J]
                "END"
        "END"
"END" EVALUATE JACOBIAN;                                        "COMMENT"
```

```
        "PROCEDURE" DECOMPOSE JACOBIAN;
        "BEGIN" DECOMPOSE:= "FALSE";
                DECOMPOSED:= "TRUE"; C:= -A0 * H;
                "FOR" J:= 1 "STEP" 1 "UNTIL" N "DO"
                "BEGIN" "FOR" I:= 1 "STEP" 1 "UNTIL" N "DO"
                        JAC[I,J]:= JACOBIAN[I,J] * C;
                        JAC[J,J]:= JAC[J,J] + 1
                "END";
                DET(JAC,N,P)
        "END" DECOMPOSE JACOBIAN;

        "PROCEDURE" CALCULATE STEP AND ORDER;
        "BEGIN" "REAL" A1,A2,A3;
                A1:= "IF" K <= 1 "THEN" 0 "ELSE"
                     0.75 * (TOLDWN/NORM2(Y[K*M+I])) ** (0.5/K);
                A2:= 0.80 * (TOL/ERROR) ** (0.5/(K + 1));
                A3:= "IF" K >= 5 "OR" FAILS ^= 0
                     "THEN" 0 "ELSE"
                     0.70 * (TOLUP/NORM2(DELTA[I] - LAST DELTA[I])) **
                     (0.5/(K+2));

                "IF" A1 > A2 "AND" A1 > A3 "THEN"
                "BEGIN" KNEW:= K-1; CHNEW:= A1 "END" "ELSE"
                "IF" A2 > A3 "THEN"
                "BEGIN" KNEW:= K  ; CHNEW:= A2 "END" "ELSE"
                "BEGIN" KNEW:= K+1; CHNEW:= A3 "END"
        "END" CALCULATE STEP AND ORDER;

        "IF" FIRST "THEN"
        "BEGIN" FIRST:= "FALSE"; M:= N;
                "FOR" I:= -1,-2,-3 "DO" SAVE[I]:= 0;
                OUT(0,0);
                ADAMS:= "NOT" STIFF; WITH JACOBIAN:= "NOT" ADAMS;
                "IF" WITH JACOBIAN "THEN" EVALUATE JACOBIAN;
                METHOD;
        NEW START; K:= 1; SAME:= 2; ORDER; DERIV(DF);
                H:= "IF" "NOT" WITH JACOBIAN "THEN" HMIN "ELSE"
                SQRT(2 * EPS/SQRT(NORM2 (MATVEC(1,N,I,JACOBIAN,DF))));
                "IF" H > HMAX "THEN" H:= HMAX "ELSE"
                "IF" H < HMIN "THEN" H:= HMIN;
                "FOR" I:= 1 "STEP" 1 "UNTIL" N "DO"
                "BEGIN" XOLD:= X; HOLD:= H; KOLD:= K; CH:= 1;
                        SAVE[I]:= Y[I]; SAVE[M+I]:= Y[M+I]:= DF[I] * H
                "END";
                OUT(0,0)
        "END" "ELSE"
        "BEGIN" WITH JACOBIAN:= "NOT" ADAMS; CH:= 1;
                K:=KOLD; RESET; ORDER;
                DECOMPOSE:= WITH JACOBIAN
        "END";
        FAILS:= 0;                                              "COMMENT"
```

```
                "FOR" L:= 0 "WHILE" X < XEND "DO"
                "BEGIN" "IF" X + H <= XEND "THEN" X:= X + H "ELSE"
                        "BEGIN" H:= XEND-X; X:= XEND; CH:= H/HOLD; C:= 1;
                                "FOR" J:= M "STEP" M "UNTIL" K*M "DO"
                                "BEGIN" C:= C* CH;
                                        "FOR" I:= J+1 "STEP" 1 "UNTIL" J+N "DO"
                                        Y[I]:= Y[I] * C
                                "END";
                                SAME:= "IF" SAME<3 "THEN" 3 "ELSE" SAME+1;
                        "END";

                        "COMMENT" PREDICTION;
                        "FOR" L:= 1 "STEP" 1 "UNTIL" N "DO"
                        "BEGIN" "FOR" I:= L "STEP" M "UNTIL" (K-1)*M+L "DO"
                                "FOR" J:= (K-1)*M+L "STEP" -M "UNTIL" I "DO"
                                Y[J]:= Y[J] + Y[J+M];
                                DELTA[L]:= 0
                        "END";  EVALUATED:= "FALSE";

        "COMMENT" CORRECTION AND ESTIMATION LOCAL ERROR;
        "FOR" L:= 1,2,3 "DO"
        "BEGIN" DERIV(DF);
                "FOR" I:= 1 "STEP" 1 "UNTIL" N "DO"
                DF[I]:= DF[I] * H - Y[M+I];
                "IF" WITH JACOBIAN "THEN"
                "BEGIN" "IF" EVALUATE "THEN" EVALUATE JACOBIAN;
                        "IF" DECOMPOSE "THEN" DECOMPOSE JACOBIAN;
                        SOL(JAC,N,P,DF)
                "END";

                CONV:= "TRUE";
                "FOR" I:= 1 "STEP" 1 "UNTIL" N "DO"
                "BEGIN" DFI:= DF[I];
                        Y[  I]:= Y[  I] + A0 * DFI;
                        Y[M+I]:= Y[M+I] +      DFI;
                        DELTA[I]:= DELTA[I] +  DFI;
                        CONV:= CONV "AND" ABS(DFI) < TOLCONV * YMAX[I]
                "END";
                "IF" CONV "THEN"
                "BEGIN" ERROR:= NORM2(DELTA[I]);
                        "GOTO" CONVERGENCE
                "END"
        "END";                                                  "COMMENT"
```

```
        "COMMENT" ACCEPTANCE OR REJECTION;
        "IF" "NOT" CONV "THEN"
        "BEGIN" "IF" "NOT" WITH JACOBIAN "THEN"
                    "BEGIN" EVALUATE:= WITH JACOBIAN:= SAME >= K
                            "OR" H<1.1 * HMIN;
                        "IF" "NOT" WITH JACOBIAN "THEN" CH:= CH/4;
                "END" "ELSE"
                "IF" "NOT" DECOMPOSED "THEN" DECOMPOSE:= "TRUE" "ELSE"
                "IF" "NOT" EVALUATED  "THEN" EVALUATE :="TRUE" "ELSE"
                "IF" H > 1.1 * HMIN   "THEN" CH:= CH/4 "ELSE"
                "IF" ADAMS            "THEN" "GOTO" TRY CURTISS "ELSE"
                "BEGIN" SAVE[-1]:= 1; "GOTO" RETURN "END";

            RESET
    "END" "ELSE" CONVERGENCE:

    "IF" ERROR > TOL "THEN"
    "BEGIN" FAILS:= FAILS + 1;
            "IF" H > 1.1 * HMIN "THEN"
            "BEGIN" "IF" FAILS > 2 "THEN"
                        "BEGIN" "IF" ADAMS "THEN"
                                "BEGIN" ADAMS:= "FALSE"; METHOD "END";
                                KOLD:= 0; RESET; "GOTO" NEW START
                    "END" "ELSE"
                    "BEGIN" CALCULATE STEP AND ORDER;
                                "IF" KNEW <> K "THEN"
                                "BEGIN" K:= KNEW; ORDER "END";
                                CH:= CH * CHNEW; RESET
                    "END"
    "END" "ELSE"
    "BEGIN" "IF" ADAMS "THEN" TRY CURTISS:
            "BEGIN" ADAMS:= "FALSE"; METHOD
            "END" "ELSE"
            "IF" K = 1 "THEN"
            "BEGIN" "COMMENT" VIOLATE EPS CRITERION;
                        C:= EPS * SQRT(ERROR/TOL);
                        "IF" C > SAVE[-3] "THEN" SAVE[-3]:= C;
                        SAVE[-2]:= SAVE[-2] + 1;
                        SAME:= 4; "GOTO" ERROR TEST OK
                        "END";
                        KOLD:= 1; RESET; ORDER; SAME:= 2
        "END"
    "END" "ELSE" ERROR TEST OK;
    "BEGIN"                                                    "COMMENT"
```

```
                FAILS:= 0;
                "FOR" I:= 1 "STEP" 1 "UNTIL" N "DO"
                "BEGIN" C:= DELTA[I];
                        "FOR" L:= 2 "STEP" 1 "UNTIL" K "DO"
                        Y[L*M+I]:= Y[L*M+I] + A[L] * C;
                        "IF" ABS(Y[I]) > YMAX[I] "THEN"
                             YMAX[I]:=  ABS(Y[I])
                "END";

                SAME:= SAME-1;
                "IF" SAME= 1 "THEN"
                "BEGIN" "FOR" I:= 1 "STEP" 1 "UNTIL" N "DO"
                        LAST DELTA[I]:= DELTA[I]
                "END" "ELSE"
                "IF" SAME= 0 "THEN"
                "BEGIN" CALCULATE STEP AND ORDER;
                        "IF" CHNEW > 1.1 "THEN"
                        "BEGIN" DECOMPOSED:= "FALSE";
                                "IF" K ^= KNEW "THEN"
                                "BEGIN" "IF" KNEW > K "THEN"
                                        "BEGIN" "FOR" I:= 1 "STEP" 1
                                                "UNTIL" N "DO" Y[KNEW*M+I]
                                                   := DELTA[I] * A[K]/KNEW
                                        "END";
                                        K:= KNEW; ORDER
                                "END";
                                SAME:= K+1;
                                "IF" CHNEW * H > HMAX
                                     "THEN" CHNEW:= HMAX/H;
                                H:= H * CHNEW; C:= 1;
                                "FOR" J:= M "STEP" M "UNTIL" K*M "DO"
                                "BEGIN" C:= C * CHNEW;
                                        "FOR" I:= J+1 "STEP" 1 "UNTIL"
                                        J+N "DO" Y[I]:= Y[I] * C
                                "END"
                        "END"
                        "ELSE" SAME:= 10
        "END";
        "IF" X ^= XEND "THEN"
                "BEGIN" XOLD:= X; HOLD:= H; KOLD:= K; CH:= 1;
                        "FOR" I:= K * M + N "STEP" -1 "UNTIL" 1 "DO"
                        SAVE[I]:= Y[I];
                        OUT(H,K)
                "END"
        "END" CORRECTION AND ESTIMATION LOCAL ERROR;
        "END" STEP;

        RETURN: SAVE[0]:= "IF" ADAMS "THEN" 0 "ELSE" 1;
        MULTISTEP:= SAVE[-1]= 0 "AND" SAVE[-2]=0
"END" MULTISTEP;
        "EOP"
```

AUTHORS : R.BULIRSCH AND J.STOER.

CONTRIBUTOR: K.DEKKER.

INSTITUTE: MATHEMATICAL CENTRE.

RECEIVED: 731231.

BRIEF DESCRIPTION:

DIFFSYS SOLVES INITIAL VALUE PROBLEMS , GIVEN AS A SYSTEM OF FIRST ORDER DIFFERENTIAL EQUATIONS. EXTRAPOLATION, APPLIED TO LOWER ORDER RESULTS,DELIVERS A HIGH ORDER SOLUTION. AUTOMATIC STEP SIZE CONTROL IS PROVIDED.
IN PARTICULAR THIS METHOD IS SUITABLE FOR HIGH ACCURACY PROBLEMS.

KEYWORDS:

DIFFERENTIAL EQUATIONS,
INITIAL VALUE PROBLEMS,
EXTRAPOLATION METHODS,
MODIFIED MIDPOINT RULE.

CALLING SEQUENCE:

THE HEADING OF THE PROCEDURE DIFFSYS READS:
"PROCEDURE" DIFFSYS(X,XE,N,Y,DERIVATIVE,AETA,RETA,S,H0,OUTPUT);
"VALUE" N;
"INTEGER" N;
"REAL" X,XE,AETA,RETA,H0;
"ARRAY" Y,S;
"PROCEDURE" DERIVATIVE,OUTPUT;

THE MEANING OF THE FORMAL PARAMETERS IS:

X: <VARIABLE>;
THE INDEPENDENT VARIABLE X;
ENTRY: THE INITIAL VALUE X0;
EXIT : THE FINAL VALUE XE;

XE: <ARITHMETIC EXPRESSION>;
THE FINAL VALUE OF X (XE>=X);

N: <ARITHMETIC EXPRESSION>;
THE NUMBER OF EQUATIONS;

Y: <ARRAY IDENTIFIER>;
"REAL" "ARRAY" Y[1:N];
THE DEPENDENT VARIABLE;
ENTRY: THE INITIAL VALUES OF THE SYSTEM OF DIFFERENTIAL EQUATIONS: Y[I] AT X=X0;
EXIT : THE FINAL VALUES OF THE SOLUTION: Y[I] AT X=XE;

DERIVATIVE: <PROCEDURE IDENTIFIER>;
THE HEADING OF THIS PROCEDURE READS:
"PROCEDURE" DERIVATIVE(X,Y,DY); "REAL" X; "ARRAY" Y,DY;
THIS PROCEDURE SHOULD DELIVER THE RIGHT HAND SIDE OF THE I-TH DIFFERENTIAL EQUATION AT THE POINT (X,Y) AS DY[I], I=1,...,N;

AETA: <ARITHMETIC EXPRESSION>;
REQUIRED ABSOLUTE PRECISION IN THE INTEGRATION PROCESS;
AETA HAS TO BE POSITIVE;

RETA: <ARITHMETIC EXPRESSION>;
REQUIRED RELATIVE PRECISION IN THE INTEGRATION PROCESS;
RETA HAS TO BE POSITIVE;

S: <ARRAY IDENTIFIER>;
"REAL" "ARRAY" S[1:N];
THE ARRAY S IS USED TO CONTROL THE ACCURACY OF THE COMPUTED VALUES OF Y;
ENTRY: IT IS ADVISABLE TO SET S[I]=0, I=1,...,N;
EXIT : THE MAXIMUM VALUE OF ABS(Y[I]), ENCOUNTERED DURING INTEGRATION, IF THIS VALUE EXCEEDS THE VALUE OF S[I] ON ENTRY;

H0: <VARIABLE>;
THE INITIAL STEP TO BE TAKEN;

OUTPUT: <PROCEDURE IDENTIFIER>;
THE HEADING OF THIS PROCEDURE READS;
"PROCEDURE" OUTPUT;
THIS PROCEDURE IS CALLED AT THE END OF EACH INTEGRATION STEP ; THE USER CAN ASK FOR OUTPUT OF SOME PARAMETERS , FOR EXAMPLE X, Y, S.

DATA AND RESULTS: SEE EXAMPLE OF USE, AND REF [1], [2] AND [3].

PROCEDURES USED: NONE.

REQUIRED CENTRAL MEMORY:
EXECUTION FIELD LENGTH: CIRCA 10 + 28 * M.

RUNNING TIME: DEPENDS STRONGLY ON THE DIFFERENTIAL EQUATION TO SOLVE.

LANGUAGE: ALGOL 60.

METHOD AND PERFORMANCE:

THE PROCEDURE DIFFSYS IS A SLIGHT MODIFICATION OF THE ALGORITHM PUBLISHED BY BULIRSCH AND STOER (SEE REF[1]). BY THIS MODIFICATION INTEGRATION FROM X0 UNTIL XE CAN BE PERFORMED BY ONE CALL OF DIFFSYS. A NUMBER OF INTEGRATION STEPS ARE TAKEN, STARTING WITH THE INITIAL STEP H0. IN EACH INTEGRATION STEP A NUMBER OF SOLUTIONS ARE COMPUTED BY MEANS OF THE MODIFIED MIDPOINT RULE. EXTRAPOLATION IS USED TO IMPROVE THESE SOLUTIONS,UNTIL THE REQUIRED ACCURACY IS MET. AN INTEGRATION STEP IS REJECTED, IF THE ACCURACY REQUIREMENTS ARE NOT FULFILLED AFTER NINE EXTRAPOLATION STEPS. IN THESE CASES THE INTEGRATION STEP IS REJECTED, AND INTEGRATION IS TRIED AGAIN WITH THE INTEGRATION STEP HALVED.
THE ALGORITHM IS FOR EACH STEP A VARIABLE ORDER METHOD (THE HIGHEST ORDER IS 14), AND USES A VARIABLE NUMBER OF FUNCTION EVALUATIONS, DEPENDING ON THE ORDER (MINIMUM IS 3, MAXIMUM IS 217).
THE ALGORITHM IS LESS SENSITIVE TO TOO SMALL VALUES OF THE INITIAL STEPSIZE THAN THE ORIGINAL ALGORITHM (SEE REF [2]); HOWEVER BAD GUESSES REQUIRE STILL SOME MORE COMPUTATIONS.

REFERENCES:

[1]. R.BULIRSCH AND J.STOER.
NUMERICAL TREATMENT OF ORDINARY DIFFERENTIAL EQUATIONS BY EXTRAPOLATION METHODS.
NUMERISCHE MATHEMATIK, VOLUME 8, PAGE 1-13, 1965.

[2]. PHYLLIS FOX.
A COMPARATIVE STUDY OF COMPUTER PROGRAMS FOR INTEGRATING DIFFERENTIAL EQUATIONS.
COMMUNICATIONS OF THE A.C.M., VOLUME 15, PAGE 941-948, 1972.

[3]. T.E.HULL, W.H.ENRIGHT, B.M.FELLEN AND A.E.SEDGWICK.
COMPARING NUMERICAL METHODS FOR ORDINARY DIFFERENTIAL EQUATIONS.
SIAM JOURNAL ON NUMERICAL ANALYSIS,VOLUME 9,PAGE 603-635,1972.

EXAMPLE OF USE:

THE FOLLOWING PROGRAM ILLUSTRATES THE COSTS AND THE ACCURACIES WHICH ARE OBTAINED WHEN SOLVING A SYSTEM OF DIFFERENTIAL EQUATIONS ARISING FROM THE RESTRICTED PROBLEM OF THREE BODIES (SEE REF[1]). THE SOLUTION IS A CLOSED ORBIT WITH PERIOD T=6.192169331396.

```
"BEGIN"
    "PROCEDURE" DIFFSYS(X,XE,N,Y,DERIVATIVE,AETA,RETA,S,H0,OUTPUT);
    "CODE" 33180;
    "INTEGER" PASSES,K;
    "REAL" X,XE,TIME,TOL,H0;
    "REAL" "ARRAY" Y,S[1:4];

    "PROCEDURE" DER(X,Y,DY); "REAL" X; "ARRAY" Y,DY;
    "BEGIN" "REAL" MU,MU1,Y1,Y2,Y3,Y4,S1,S2;
        MU:=1/82.45; MU1:=1-MU;
        PASSES:=PASSES+1;
        Y1:=Y[1]; Y2:=DY[1]:=Y[2]; Y3:=Y[3]; Y4:=DY[3]:=Y[4];
        S1:=(Y1+MU)**2+Y3**2; S2:=(Y1-MU1)**2+Y3**2;
        S1:=S1*SQRT(S1); S2:=S2*SQRT(S2);
        DY[2]:=Y1+2*Y4-MU1*(Y1+MU)/S1-MU*(Y1-MU1)/S2;
    DY[4]:=Y3-2*Y2-MU1*Y3/S1-MU*Y3/S2
"END";

"PROCEDURE" OUT;
"BEGIN" K:=K+1;
    "IF" X>=XE "THEN"
    OUTPUT(61,"("2(-5ZD),2(4B+Z,3DB3DB3DB3D),-5ZD,3D,/")",K,
    PASSES,Y[1],Y[3],CLOCK-TIME)
"END";

OUTPUT(61,"(""("    THIS LINE AND THE FOLLOWING TEXT IS ")"
"("PRINTED BY THIS PROGRAM")",//,
"("    THE RESULTS WITH DIFFSYS - H0=,2 - ARE: ")",/,
"("     K   DER,EV,          Y[1]                    Y[3]       ")",
"("       TIME")",/")");
"FOR" TOL:="-4,"-6,"-8,"-10,"-12 "DO"
"BEGIN" PASSES:=K:=0; X:=0; XE:=6,192169331396;
    Y[1]:=1,2; Y[2]:=Y[3]:=0; Y[4]:=-1,04935750983;
        S[1]:=S[2]:=S[3]:=S[4]:=0; H0:=,2; TIME:=CLOCK;
        DIFFSYS(X,XE,4,Y,DER,TOL,TOL,S,H0,OUT);
    "END"
"END"
```

THIS LINE AND THE FOLLOWING TEXT IS PRINTED BY THIS PROGRAM:

THE RESULTS WITH DIFFSYS - H0=,2 - ARE:

K	DER,EV,	Y[1]	Y[3]	TIME
30	2591	+1,320 357 347 741	-,032 645 454 836	5,686
33	3414	+1,200 078 037 878	-,000 053 906 067	7,455
37	4213	+1,200 003 282 801	-,000 002 363 741	9,267
44	4618	+1,199 999 999 711	-,000 000 000 095	10,242
56	6299	+1,200 000 000 003	-,000 000 000 090	13,827

SOURCE TEXT:

```
"CODE"" 33180;
"PROCEDURE" DIFFSYS(X,XE,N,Y,DERIVATIVE,AETA,RETA,S,H0,OUTPUT);
"VALUE" N;
"INTEGER" N;
"REAL" X,XE,AETA,RETA,H0;
"ARRAY" Y,S;
"PROCEDURE" DERIVATIVE,OUTPUT;
"BEGIN" "REAL" A,B,B1,C,G,H,U,V,TA,FC; "INTEGER" I,J,K,KK,JJ,L,M,R,SR;
     "ARRAY" YA,YL,YM,DY,DZ[1:N],DT[1:N,0:6],D[0:6],YG,YH[0:7,1:N];
     "BOOLEAN" KONV,B0,BH,LAST;
     LAST:="FALSE"; H:=H0;
NEXT: "IF" H*1.1>=XE-X "THEN"
     "BEGIN" LAST:="TRUE"; H0:=H; H:=XE-X+"-13 "END";
     DERIVATIVE(X,Y,DZ); BH:="FALSE";
     "FOR" I:=1 "STEP" 1 "UNTIL" N "DO" YA[I]:=Y[I];
ANF: A:=H+X; FC:=1.5; B0:="FALSE"; M:=1; R:=2; SR:=3; JJ:=-1;
     "FOR" J:=0 "STEP" 1 "UNTIL" 9 "DO"
     "BEGIN" "IF" B0 "THEN"
          "BEGIN" D[1]:=16/9; D[3]:=64/9; D[5]:=256/9 "END"
          "ELSE" "BEGIN" D[1]:=9/4; D[3]:=9; D[5]:=36 "END";
          KONV:="TRUE.";
          "IF" J>6 "THEN" "BEGIN" L:=6; D[6]:=64; FC:=.6*FC "END"
          "ELSE" "BEGIN" L:=J; D[L]:=M*M "END";
          M:=M*2; G:=H/M; B:=G*2;
          "IF" BH "AND" J<8 "THEN"
          "BEGIN" "FOR" I:=1 "STEP" 1 "UNTIL" N "DO"
               "BEGIN" YM[I]:=YH[J,I]; YL[I]:=YG[J,I] "END"
          "END"
          "ELSE"
          "BEGIN"                                                    "COMMENT"
```

```
;
        KK:=(M-2)/2; M:=M-1;
    "FOR" I:=1 "STEP" 1 "UNTIL" N "DO"
    "BEGIN" YL[I]:=YA[I]; YM[I]:=YA[I]+G*DZ[I] "END";
    "FOR" K:=1 "STEP" 1 "UNTIL" M "DO"
        "BEGIN" DERIVATIVE(X+K*G,YM,DY);
            "FOR" I:=1 "STEP" 1 "UNTIL" N "DO"
            "BEGIN" U:=YL[I]+B*DY[I]; YL[I]:=YM[I]; YM[I]:=U;
                U:=ABS(U); "IF" U>S[I] "THEN" S[I]:=U
            "END";
            "IF" K=KK "AND" K^=2 "THEN"
            "BEGIN" JJ:=JJ+1; "FOR" I:=1 "STEP" 1 "UNTIL" N "DO"
                "BEGIN" YH[JJ,I]:=YM[I]; YG[JJ,I]:=YL[I] "END"
            "END"
        "END"
"END";
DERIVATIVE(A,YM,DY);
"FOR" I:=1 "STEP" 1 "UNTIL" N "DO"
"BEGIN" V:=DT[I,0]; TA:=C:=DT[I,0]:=(YM[I]+YL[I]+G*DY[I])/2;
    "FOR" K:=1 "STEP" 1 "UNTIL" L "DO"
    "BEGIN" B1:=D[K]*V; B:=B1-C; U:=V;
        "IF" B^=0 "THEN"
        "BEGIN" B:=(C-V)/B; U:=C*B; C:=B1+B "END";
            V:=DT[I,K]; DT[I,K]:=U; TA:=U+TA
        "END";
        "IF" ABS(Y[I]-TA)>RETA*S[I]+AETA "THEN" KONV:="FALSE";
        Y[I]:=TA
    "END";
    "IF" KONV "THEN" "GOTO" END;
    D[2]:=4; D[4]:=16; B0:=^B0; M:=R; R:=SR; SR:=M*2
"END";
BH:=^BH; LAST:="FALSE"; H:=H/2; "GOTO" ANF;
END: H:=FC*H; X:=A; OUTPUT; "IF" "NOT" LAST "THEN" "GOTO" NEXT;
"END" DIFFSYS;
    "EOP"
```

AUTHOR: P.A. BEENTJES.

INSTITUTE: MATHEMATICAL CENTRE.

RECEIVED: 740510.

BRIEF DESCRIPTION:

ARK SOLVES AN INITIAL VALUE PROBLEM, GIVEN AS A SYSTEM OF FIRST ORDER (NON-LINEAR) DIFFERENTIAL EQUATIONS BY MEANS OF A STABILIZED RUNGE KUTTA METHOD WITH LIMITED STORAGE REQUIREMENTS.

KEYWORDS:
DIFFERENTIAL EQUATIONS,
INITIAL VALUE PROBLEM,
EXPLICIT ONE-STEP METHOD,
STABILIZED RUNGE KUTTA METHOD.

CALLING SEQUENCE:

THE HEADING OF THE PROCEDURE READS:
"PROCEDURE" ARK (T, TE, M0, M, U, DERIVATIVE, DATA, OJT);
"INTEGER" M0, M; "REAL" T, TE; "ARRAY" U, DATA;
"PROCEDURE" DERIVATIVE, OUT;

ARK : INTEGRATES THE SYSTEM OF ORDINARY DIFFERENTIAL EQUATIONS
DU / DT = H(U, T), U = U0 AT T = T0.

THE MEANING OF THE FORMAL PARAMETERS IS:
T: <VARIABLE>;
THE INDEPENDENT VARIABLE T; CAN BE USED IN DERIVATIVE;
ENTRY: THE INITIAL VALUE T0;
EXIT : THE FINAL VALUE TE;
TE: <ARITHMETIC EXPRESSION>;
THE FINAL VALUE OF T (TE >= T);
M0,M: <ARITHMETIC EXPRESSION>;
INDICES OF THE FIRST AND LAST EQUATION OF THE SYSTEM;
U: <ARRAY IDENTIFIER>;
"ARRAY" U[M0 : M];
ENTRY: THE INITIAL VALUES OF THE SOLUTION OF THE SYSTEM OF
DIFFERENTIAL EQUATIONS AT T = T0;
EXIT : THE VALUES OF THE SOLUTION AT T = TE;
DERIVATIVE: <PROCEDURE IDENTIFIER>;
THE HEADING OF THIS PROCEDURE READS:
"PROCEDURE" DERIVATIVE(T, V); "REAL" T; "ARRAY" V;
THIS PROCEDURE PERFORMS AN EVALUATION OF THE RIGHT H
SIDE OF THE SYSTEM WITH DEPENDENT VARIABLES V[M0 : M]
INDEPENDENT VARIABLE T; UPON COMPLETION OF DERIVATIVE
RIGHT HAND SIDE SHOULD BE OVERWRITTEN ON V[

DATA: <ARRAY IDENTIFIER>;
"ARRAY" DATA[1 : 10 + DATA[1]];
IN ARRAY DATA ONE SHOULD GIVE:
DATA[1]: THE NUMBER OF EVALUATIONS OF H(U, T) PER INTEGRATION STEP(DATA[1] >= DATA[2]);
DATA[2]: THE ORDER OF ACCURACY OF THE METHOD (DATA[2]<=3);
DATA[3]: STABILITY BOUND(SEE REFERENCE [3]);
DATA[4]: THE SPECTRAL RADIUS OF THE JACOBIAN MATRIX WITH RESPECT TO THOSE EIGENVALUES, WHICH ARE LOCATED IN THE NON-POSITIVE HALF PLANE;
DATA[5]: THE MINIMAL STEPSIZE;
DATA[6]: THE ABSOLUTE TOLERANCE;
DATA[7]: THE RELATIVE TOLERANCE;
IF BOTH DATA[6] AND DATA[7] ARE NEGATIVE, THE INTEGRATION IS PERFORMED WITH A CONSTANT STEP DATA[5];
DATA[8]: DATA[8] SHOULD BE 0 IF ARK IS CALLED FOR A FIRST TIME; FOR CONTINUED INTEGRATION (E.G. FROM TE TO TE-NEW) DATA[8] SHOULD NOT BE CHANGED;
DATA[11], ..., DATA[10 + DATA[1]]: POLYNOMIAL COEFFICIENTS (SEE REFERENCE [3]);
AFTER EACH STEP THE FOLLOWING BY-PRODUCTS ARE DELIVERED:
DATA[8]: THE NUMBER OF INTEGRATION STEPS PERFORMED;
DATA[9]: AN ESTIMATION OF THE LOCAL ERROR LAST MADE;
DATA[10]: INFORMATIVE MESSAGES:
DATA[10] = 0: NO DIFFICULTIES;
DATA[10] = 1: MINIMAL STEPLENGTH EXCEEDS THE STEPLENGTH PRESCRIBED BY STABILITY THEORY, I.E. DATA[5] > DATA[3] / DATA[4]; (TERMINATION OF ARK); DECREASE MINIMAL STEPLENGTH;
IF NECESSARY, DATA[I],I = 4(1)7, CAN BE UPDATED(AFTER EACH STEP) BY MEANS OF PROCEDURE OUT;

OUT: <PROCEDURE IDENTIFIER>;
THE HEADING OF THIS PROCEDURE READS:
"PROCEDURE" OUT;
AFTER EACH INTEGRATION STEP PERFORMED INFORMATION CAN BE OBTAINED OR UPDATED BY THIS PROCEDURE, E.G. THE VALUES OF T, U[M0 : M] AND DATA[I], I = 4(1)10.

DATA AND RESULTS:

FOR THE INDICES M0 AND M THE FOLLOWING REMARKS CAN BE MADE: WHEN THE METHOD OF LINES IS APPLIED TO HYPERBOLIC DIFFERENTIAL EQUATIONS THE NUMBER OF RELEVANT ORDINARY DIFFERENTIAL EQUATIONS DECREASES DURING THE INTEGRATION PROCESS; IN PROCEDURE ARK THIS MAY BE REALIZED BY INTEGERS M0 AND M, WHICH ARE DEFINED AS FUNCTIONS OF THE NUMBER OF RIGHT HAND SIDE EVALUATIONS. A SELECTION OF POSSIBLE ENTRIES FOR ARRAY DATA (DEPENDENT ON THE KIND OF INITIAL VALUE PROBLEM) IS GIVEN IN REFERENCE [4],SECTION 8.

PROCEDURES USED:

INIVEC = CP31010,
MULVEC = CP31020,
DUPVEC = CP31030,
VECVEC = CP34010,
ELMVEC = CP34020,
DECSOL = CP34301.

REQUIRED CENTRAL MEMORY:

EXECUTION FIELD LENGTH: CIRCA 75 + 2 * (M - M0).

RUNNING TIME: DEPENDS STRONGLY ON THE PROBLEM TO BE SOLVED.

LANGUAGE: ALGOL60.

METHOD AND PERFORMANCE:

ARK IS AN IMPLEMENTATION OF LOW ORDER STABILIZED RUNGE KUTTA METHODS (SEE REFERENCE [1]);
AUTOMATIC STEPSIZE CONTROL IS PROVIDED BUT STEP-REJECTION HAS BEEN EXCLUDED IN ORDER TO SAVE STORAGE;
BECAUSE OF ITS LIMITED STORAGE REQUIREMENTS AND ADAPTIVE STABILITY FACILITIES THE METHOD IS WELL SUITED FOR THE SOLUTION OF INITIAL BOUNDARY VALUE PROBLEMS FOR PARTIAL DIFFERENTIAL EQUATIONS;
NUMERICAL RESULTS, OBTAINED WITH A SLIGHTLY DIFFERENT IMPLEMENTATION CAN BE FOUND IN REFERENCE [2].

REFERENCES:

[1]. P.J. VAN DER HOUWEN.
STABILIZED RUNGE KUTTA METHOD WITH LIMITED STORAGE REQUIREMENTS.
MATH. CENTR. REPORT TW 124/71;

[2]. P.A. BEENTJES.
AN ALGOL 60 VERSION OF STABILIZED RUNGE KUTTA METHODS (DUTCH).
MATH. CENTR. REPORT NR 23/72;

[3]. P.J. VAN DER HOUWEN, J. KOK.
NUMERICAL SOLUTION OF A MINIMAX PROBLEM.
MATH. CENTR. REPORT TW 123/71;

[4]. P.J. VAN DER HOUWEN ET AL.
ONE STEP METHODS FOR LINEAR INITIAL VALUE PROBLEMS, I.I.I.
NUMERICAL EXAMPLES,
MATH. CENTR. REPORT TW 130/71.

EXAMPLE OF USE:

THE VALUES OF

1. Y(1) AND Y(2) OF THE INITIAL VALUE PROBLEM
   DY / DX = Y - 2 * X / Y, Y(0) = 1

AND

2. U(.6, 0) OF THE CAUCHY PROBLEM (SEE REFERENCE [2]):
   DU / DT = .5 * DU / DX, U(0, X) = EXP(-X * X)

MAY BE OBTAINED BY THE FOLLOWING PROGRAM:

```
"BEGIN" "INTEGER" M0, M, I; "REAL" T, TE, DAT;
    "ARRAY" Y[1 : 1], U[-150 : 150], DATA[1 : 14];

    "PROCEDURE" ARK
    (T, TE, M0, M, U, DERIVATIVE, DATA, OUT); "CODE" 33061;

    "PROCEDURE" DER1(T, V); "REAL" T; "ARRAY" V;
    V[1]:= V[1] - 2 * T / V[1];

    "PROCEDURE" DER2(T, V); "REAL" T; "ARRAY" V;
    "BEGIN" "INTEGER" J; "REAL" V1, V2, V3;
       V2:= V[M0]; M0:= M0 + 1; M:= M - 1; V3:= V[M0];
       "FOR" J:= M0 "STEP" 1 "UNTIL" M "DO"
       "BEGIN" V1:= V2; V2:= V3; V3:= V[J + 1];
           V[J]:= 250 * (V3 - V1) / 3
       "END"
    "END" DER2;
```

```
"PROCEDURE" OUT1;
"IF" T = TE "THEN"
"BEGIN" "IF" T = 1 "THEN" OUTPUT(61, "("/, "(" PROBLEM 1")", //,
    "(" X  NUMBER OF INTEGRATION STEPS Y(COMPUTED)  Y(EXACT)")",
    //")");
    OUTPUT(61, "("ZD, 13ZD,12B,2(-3ZD.7D),"("..."), /")",
    T, DATA[8], Y[1], SQRT(2 * T + 1));
    TE:= 2
"END" OUT1;

"PROCEDURE" OUT2;
"IF" T = .6 "THEN"
OUTPUT(61, "("//, "(" PROBLEM 2")", //,
   "(" NUMBER OF DERIVATIVE CALLS")",
   "("  U(.6, 0)COMPUTED   U(.6, 0)EXACT")", //, 13ZD,
   2(-10Z.7D), "("...")"")", DATA[1] * DATA[8], U[0], EXP(-.09));

I:= 1;
"FOR" DAT:= 3, 3, 1, 1, "-3, "-6, "-6, 0, 0, 0, 1, .5, 1 / 6 "DO"
"BEGIN" DATA[I]:= DAT; I:= I + 1 "END";
T:= 0; Y[1]:= 1; TE:= 1;
   ARK(T, TE, 1, 1, Y, DER1, DATA, OUT1);
   I:= 1;
   "FOR" DAT:= 4, 3, SQRT(8), 500 / 3, DATA[3] / DATA[4], -1, -1,
      0, 0, 0, 1, .5, 1 / 6, 1 / 24 "DO"
   "BEGIN" DATA[I]:= DAT; I:= I + 1 "END";
   M0:= -150; M:= 150; T:= 0; U[0]:= 1;
   "FOR" I:= 1 "STEP" 1 "UNTIL" M "DO"
   U[I]:= U[-I]:= EXP(-(.003 * I) ** 2);
   ARK(T, .6, M0, M, U, DER2, DATA, OUT2)
"END"
```

THIS PROGRAM DELIVERS:

PROBLEM 1

X	NUMBER OF INTEGRATION STEPS	Y(COMPUTED)	Y(EXACT)
1	38	1.7320535	1.7320508...
2	56	2.2360928	2.2360680...

PROBLEM 2

NUMBER OF DERIVATIVE CALLS	U(.6, 0)COMPUTED	U(.6, 0)EXACT
144	.9139326	.9139312...

SOURCE TEXT(S):

```
"CODE" 33061;
"PROCEDURE" ARK (T, TE, M0, M, U, DERIVATIVE, DATA, OUT);
"INTEGER" M0, M;
"REAL" T, TE;
"ARRAY" U, DATA;
"PROCEDURE" DERIVATIVE, OUT;

"BEGIN" "INTEGER" P, N, Q;
    "OWN" "REAL" EC0, EC1, EC2, TAU0, TAU1, TAU2, TAUS, T2;
    "REAL" THETANM1, TAU, BETAN, QINV, ETA;
    "ARRAY" MU, LAMBDA[1:DATA[1]], THETHA[0:DATA[1]], R0, R[M0:M];
    "BOOLEAN" START, STEP1, LAST;

    "PROCEDURE" INIVEC(L, U, A, X);                    "CODE" 31010;
    "PROCEDURE" MULVEC(L, U, SHIFT, A, B, X);          "CODE" 31020;
    "PROCEDURE" DUPVEC(L, U, SHIFT, A, B);             "CODE" 31030;
    "REAL" "PROCEDURE" VECVEC(L, U, SHIFT, A, B);      "CODE" 34010;
    "PROCEDURE" ELMVEC(L, U, SHIFT, A, B, X);          "CODE" 34020;
    "PROCEDURE" DECSOL(A, N, AUX, B);                  "CODE" 34301;

"PROCEDURE" INITIALIZE;
"BEGIN" "INTEGER" I, J, K, L, N1; "REAL" S, THETA0;
    "ARRAY" ALFA[1:8, 1:DATA[1]+1], TH[1:8], AUX[1:3];

    "REAL" "PROCEDURE" LABDA(I, J); "VALUE" I, J; "INTEGER" I, J;
    LABDA:= "IF" P < 3 "THEN" ("IF" J =I-1 "THEN" MUI(I) "ELSE" 0)
            "ELSE" "IF" P =3 "THEN" ("IF" I =N "THEN" ("IF" J=0
            "THEN" .25 "ELSE" "IF" J =N - 1 "THEN" .75
            "ELSE" 0) "ELSE" "IF" J =0 "THEN" ("IF" I =1
            "THEN" MUI(1) "ELSE" .25) "ELSE" "IF" J =I - 1
            "THEN" LAMBDA[I] "ELSE" 0) "ELSE" 0;

    "REAL" "PROCEDURE" MUI(I); "VALUE" I; "INTEGER" I;
    MUI:= "IF" I =N "THEN" 1 "ELSE"
          "IF" I < 1 ! I > N "THEN" 0 "ELSE"
          "IF" P < 3 "THEN" LAMBDA[I] "ELSE"
          "IF" P =3 "THEN" LAMBDA[I] + .25 "ELSE" 0;

"REAL" "PROCEDURE" SUM(I, A, B, X);
"VALUE" B; "INTEGER" I, A, B; "REAL" X;
"BEGIN" "REAL" S; S:= 0;
    "FOR" I:= A "STEP" 1 "UNTIL" B "DO" S:= S + X;
    SUM:= S
"END" SUM;
                                                    "COMMENT"
```

```
        N:= DATA[1]; P:= DATA[2];
        BETAN:= DATA[3];
        THETANM1:= "IF" P=3 "THEN" .75 "ELSE" 1;
        THETA0:= 1 - THETANM1; S:= 1;
        "FOR" J:= N - 1 "STEP" - 1 "UNTIL" 1 "DO"
        "BEGIN" S:= - S * THETA0 + DATA[N + 10 - J];
            MU[J]:= DATA[N + 11 - J] / S;
            LAMBDA[J]:= MU[J] - THETA0
        "END";
        "FOR" I:= 1 "STEP" 1 "UNTIL" 8 "DO"
        "FOR" J:= 0 "STEP" 1 "UNTIL" N "DO"
        ALFA[I, J + 1]:= "IF" I = 1 "THEN" 1 "ELSE"
          "IF" J = 0 "THEN" 0 "ELSE" "IF" I = 2 ! I = 4 ! I = 8 "THEN"
          MUI(J) ** ENTIER((I + 2) / 3) "ELSE"
          "IF" (I = 3 ! I = 6) & J > 1 "THEN" SUM(L, 1, J-1,
          LABDA(J, L) * MUI(L) ** ENTIER(I / 3)) "ELSE"
          "IF" I = 5 & J > 2 "THEN" SUM(L, 2, J - 1, LABDA(J, L) *
          SUM(K, 1, L - 1, LABDA(L, K) * MUI(K))) "ELSE"
          "IF" I = 7 & J > 1 "THEN" SUM(L, 1, J - 1, LABDA(J, L) *
          MUI(L)) * MUI(J) "ELSE" 0;
        N1:="IF" N < 4 "THEN" N + 1 "ELSE" "IF" N < 7 "THEN" 4
          "ELSE" 8;
        I:= 1;
        "FOR" S:= 1, .5, 1 / 6, 1 / 3, 1 / 24, 1 / 12, .125, .25 "DO"
        "BEGIN" TH[I]:= S; I:= I + 1 "END";
        "IF" P = 3 & N < 7 "THEN" TH[1]:= TH[2]:= 0;
        AUX[2]:= " - 14; DECSOL(ALFA, N1, AUX, TH);
        INIVEC(0, N, THETHA, 0);
        DUPVEC(0, N1 - 1, 1, THETHA, TH);
        "IF" ^ (P = 3 & N < 7) "THEN"
        "BEGIN" THETHA[0]:= THETHA[0] - THETA0;
            THETHA[N - 1]:= THETHA[N - 1] - THETANM1; Q:= P + 1
        "END" "ELSE" Q:= 3;
        QINV:= 1 / Q;
        START:= DATA[8] = 0; DATA[10]:= 0; LAST:= "FALSE";
        DUPVEC(M0, M, 0, R, U); DERIVATIVE(T, R)
    "END" INITIALIZE
```

```
        "PROCEDURE" LOCAL ERROR CONSTRUCTION(I); "VALUE" I; "INTEGER" I;
        "BEGIN" "IF" THETHA[I] ^= 0 "THEN"
            ELMVEC(M0, M, 0, R0, R, THETHA[I]);
            "IF" I = N "THEN"
            "BEGIN" DATA[9]:= SQRT(VECVEC(M0, M, 0, R0, R0))* TAU;
                EC0:= EC1; EC1:= EC2; EC2:= DATA[9] / TAU ** Q
            "END"
    "END" LEC;

    "PROCEDURE" STEPSIZE;
    "BEGIN" "REAL" TAUACC, TAUSTAB, AA, BB, CC, EC;
        ETA:= SQRT(VECVEC(M0, M, 0, U, U)) * DATA[7] + DATA[6];
        "IF" ETA > 0 "THEN"
        "BEGIN" "IF" START "THEN"
            "BEGIN" "IF" DATA[8] = 0 "THEN"
                "BEGIN" TAUACC:= DATA[5];
                    STEP1:= "TRUE"
                "END" "ELSE" "IF" STEP1 "THEN"
                "BEGIN" TAUACC:= (ETA / EC2) ** QINV;
                    "IF" TAUACC > 10 * TAU2 "THEN"
                    TAUACC:= 10 * TAU2 "ELSE" STEP1:= "FALSE"
            "END" "ELSE"
            "BEGIN" BB:= (EC2 - EC1) / TAU1; CC:= - BB * T2 + EC2;
                EC:= BB * T + CC;
                TAUACC:= "IF" EC < 0 "THEN" TAU2 "ELSE"
                (ETA / EC) ** QINV;
                START:= "FALSE"
            "END"
        "END" "ELSE"
        "BEGIN" AA:= ((EC0 - EC1) / TAU0 + (EC2 - EC1) / TAU1)
                    / (TAU1 + TAU0);
            BB:= (EC2 - EC1) / TAU1 - (2 * T2 - TAU1) * AA;
            CC:= - (AA * T2 + BB) * T2 + EC2;
            EC:= (AA * T + BB) * T + CC;
            TAUACC:= "IF" EC < 0 "THEN"
                    TAUS "ELSE" (ETA / EC) ** QINV;
            "IF" TAUACC > 2 * TAUS "THEN" TAUACC:= 2 * TAUS;
            "IF" TAUACC < TAUS / 2 "THEN" TAUACC:= TAUS / 2
            "END"
        "END" "ELSE" TAUACC:= DATA[5];
         "IF" TAUACC < DATA[5] "THEN" TAUACC:= DATA[5];
        TAUSTAB:= BETAN / DATA[4]; "IF" TAUSTAB < DATA[5] "THEN"
        "BEGIN" DATA[10]:= 1; "GOTO" ENDARK "END";
        TAU:= "IF" TAUACC > TAUSTAB "THEN" TAUSTAB "ELSE" TAUACC;
        TAUS:= TAU; "IF" TAU >= TE - T "THEN"
        "BEGIN" TAU:= TE - T; LAST:= "TRUE" "END";
        TAU0:= TAU1; TAU1:= TAU2; TAU2:= TAU
    "END" STEPSIZE
```

```
    "PROCEDURE" DIFFERENCE SCHEME;
"BEGIN" "INTEGER" I, J;
    "REAL" MT, LT;
    MULVEC(M0, M, 0, R0, R, THETHA[0]);
    "IF" P = 3 "THEN" ELMVEC(M0, M, 0, U, R, .25 * TAU);
    "FOR" I:= 1 "STEP" 1 "UNTIL" N - 1 "DO"
    "BEGIN" MT:= MU[I] * TAU; LT:= LAMBDA[I] * TAU;
        "FOR" J:= M0 "STEP" 1 "UNTIL" M "DO"
        R[J]:= LT * R[J] + U[J];
        DERIVATIVE(T + MT, R); LOCAL ERROR CONSTRUCTION(I)
    "END";
    ELMVEC(M0, M, 0, U, R, THETANM1 * TAU);
    DUPVEC(M0, M, 0, R, U); DERIVATIVE(T + TAU, R);
    LOCAL ERROR CONSTRUCTION(N); T2:= T;
    "IF" LAST "THEN"
    "BEGIN" LAST:= "FALSE"; T:= TE "END" "ELSE" T:= T + TAU;
    DATA[8]:= DATA[8]+1
"END" DIFSCH;
    INITIALIZE;

  NEXT STEP:
    STEPSIZE; DIFFERENCE SCHEME; OUT;
    "IF" T ^= TE "THEN" "GOTO" NEXT STEP;

  ENDARK:
"END" ARK;
        "EOP"
```

AUTHOR: K.DEKKER.

INSTITUTE: MATHEMATICAL CENTRE.

RECEIVED: 740710.

BRIEF DESCRIPTION:

EFRK SOLVES AN INITIAL VALUE PROBLEM , GIVEN AS A SYSTEM OF FIRST ORDER DIFFERENTIAL EQUATIONS , BY MEANS OF AN EXPONENTIALLY FITTED, EXPLICIT RUNGE-KUTTA METHOD OF FIRST , SECOND OR THIRD ORDER , AUTOMATIC STEPSIZE CONTROL IS NOT PROVIDED; HOWEVER, FOR REASONS OF THE USER PRESCRIBED STEPSIZE CAN BE ADJUSTED.
IN PARTICULAR THIS METHOD IS SUITABLE FOR THE INTEGRATION OF STIFF DIFFERENTIAL EQUATIONS.

KEYWORDS:

DIFFERENTIAL EQUATIONS,
INITIAL VALUE PROBLEMS,
STIFF EQUATIONS,
EXPONENTIAL FITTING,
TWO AND THREE CLUSTER METHODS,
EXPLICIT RUNGE-KUTTA METHODS.

CALLING SEQUENCE:

THE HEADING OF THE PROCEDURE EFRK READS:

```
"PROCEDURE" EFRK ( T,TE, M0,M, U, SIGMA, PHI, DIAMETER, DERIVATIVE,
                   K, STEP, R, L, BETA,  THIRDORDER, TOL, OUTPUT );
"VALUE" R,L;
"INTEGER" M0,M,K,R,L;
"REAL" T,TE,SIGMA,PHI,DIAMETER,STEP,TOL;
"ARRAY" U,BETA;
"BOOLEAN" THIRDORDER;
"PROCEDURE" DERIVATIVE,OUTPUT;
```

THE MEANING OF THE FORMAL PARAMETERS IS:

T: <VARIABLE>;
THE INDEPENDENT VARIABLE T;
MAY BE USED IN DERIVATIVE, SIGMA, OUTPUT, ETC.;
ENTRY: THE INITIAL VALUE T0;
EXIT ; THE FINAL VALUE TE;

TE: <ARITHMETIC ETPRESSION>;
THE FINAL VALUE OF T (TE>=T);

M0: <ARITHMETIC EXPRESSION>;
THE INDEX OF THE FIRST EQUATION;

M: <ARITHMETIC EXPRESSION>;
THE INDEX OF THE LAST EQUATION;

U: <ARRAY IDENTIFIER>;
"REAL" "ARRAY" U[M0:M];
THE DEPENDENT VARIABLE;
ENTRY: THE INITIAL VALUES OF THE SOLUTION OF THE SYSTEM OF DIFFERENTIAL EQUATIONS AT T = T0;
EXIT : THE VALUES OF THE SOLUTION AT T = TE;

SIGMA: <ARITHMETIC EXPRESSION>;
THE MODULUS OF THE POINT AT WHICH EXPONENTIAL FITTING IS DESIRED , FOR EXAMPLE AN APPROXIMATION OF THE CENTRE OF THE LEFT HAND CLUSTER;

PHI: <ARITHMETIC EXPRESSION>;
THE ARGUMENT OF THE CENTRE OF THE LEFT HAND CLUSTER; IN THE CASE OF TWO COMPLEX CONJUGATE) CLUSTERS , THE ARGUMENT OF THE CENTRE IN THE SECOND QUADRANT SHOULD BE TAKEN;

DIAMETER: <ARITHMETIC EXPRESSION>;
THE DIAMETER OF THE LEFT HAND CLUSTER OF EIGENVALUES OF THE JACOBIAN MATRIX OF THE SYSTEM OF DIFFERENTIAL EQUATIONS; IN CASE OF NON-LINEAR EQUATIONS DIAMETER SHOULD HAVE SUCH A VALUE THAT THE VARIATION OF THE EIGENVALUES IN THIS CLUSTER IN THE PERIOD (T ,T+STEP) IS LESS THAN HALF THE DIAMETER;

DERIVATIVE: <PROCEDURE IDENTIFIER>;
THE HEADING OF THIS PROCEDURE READS:
"PROCEDURE" DERIVATIVE(T,U); "REAL" T; "ARRAY" U;
THIS PROCEDURE SHOULD DELIVER THE VALUE(S) OF H(T,U) IN THE POINT (T,U) IN THE ARRAY U;

K: <VARIABLE>;
COUNTS THE NUMBER OF INTEGRATION STEPS TAKEN;
FOR EXAMPLE, K MAY BE USED IN THE EXPRESSION FOR TE;
ENTRY: AN (ARBIRARY) CHOSEN VALUE K0, E.G. K0=0;
EXIT : K0 + THE NUMBER OF INTEGRATION STEPS PERFORMED;

STEP: <ARITHMETIC EXPRESSION>;
THE STEPSIZE CHOSEN WILL BE AT MOST EQUAL TO STEP ; THIS STEPSIZE MAY BE REDUCED BY STABILITY CONSTRAINTS, IMPOSED BY A POSITIVE DIAMETER , OR BY CONSIDERATIONS OF INTERNAL STABILITY (SEE REF[1], PAGE 11);

R: <ARITHMETIC EXPRESSION>;
R + L: THE NUMBER OF EVALUATIONS OF H(T, U) ON WHICH THE RUNGE-KUTTA SCHEME IS BASED;
FOR R=1,2,>=3 FIRST, SECOND AND THIRD ORDER ACCURACY MAY BE OBTAINED BY AN APPROPRIATE CHOICE OF THE ARRAY BETA;

L: <ARITHMETIC EXPRESSION>;
ENTRY;
IF PHI = 4*ARCTAN(1): THE ORDER OF THE EXPONENTIAL FITTING, ELSE TWICE THE ORDER OF THE EXPONENTIAL FITTING;
NOTE THAT L SHOULD BE EVEN IN THE LATTER CASE;

BETA: <ARRAY IDENTIFIER>;
"REAL" "ARRAY" BETA[0:R+L];
ENTRY: THE ELEMENTS BETA[I] , I=0, ... ,R SHOULD HAVE THE VALUE OF THE R+1 FIRST COEFFICIENTS OF THE STABILITY POLYNOMIAL;

THIRDORDER: <BOOLEAN EXPRESSION>;
IF THIRD ORDER ACCURACY IS DESIRED , THIRDORDER SHOULD HAVE THE VALUE "TRUE" , IN COMBINATION WITH APPROPRIATE CHOICES OF R (R>=3) AND THE ARRAY BETA (BETA[I]=1/I!, I=0,1,2,3); IN ALL OTHER CASES THIRDORDER MUST HAVE THE VALUE "FALSE";

TOL: <ARITHMETIC EXPRESSION>;
AN UPPERBOUND FOR THE ROUNDING ERRORS IN THE COMPUTATIONS IN ONE RUNGE-KUTTA STEP ; IN SOME CASES (E.G. LARGE VALUES OF SIGMA AND R) TOL WILL CAUSE A DECREASE OF THE STEPSIZE;

OUTPUT: <PROCEDURE IDENTIFIER>;
THE HEADING OF THIS PROCEDURE READS;
"PROCEDURE" OUTPUT;
THIS PROCEDURE IS CALLED AT THE END OF EACH INTEGRATION STEP ; THE USER CAN ASK FOR OUTPUT OF SOME PARAMETERS , FOR EXAMPLE T, K, U, AND COMPUTE NEW VALUES FOR SIGMA, PHI, AND DIAMETER.

DATA AND RESULTS: SEE EXAMPLE OF USE , AND THE REFERENCES [1] AND [4].

PROCEDURES USED:

ELMVEC= CP34020,
DEC = CP34300,
SOL = CP34051.

REQUIRED CENTRAL MEMORY:

EXECUTION FIELD LENGTH: CIRCA 30 + (M-M0) + L * (5+L).

RUNNING TIME: DEPENDS STRONGLY ON THE DIFFERENTIAL EQUATION TO SOLVE.

LANGUAGE: ALGOL 60.

METHOD AND PERFORMANCE:

A DETAILED DESCRIPTION OF THE METHOD AND SOME NUMERICAL EXAMPLES ARE GIVEN IN REF[1], REF [3], PAGE 170 REPRESENTS A BRIEF SURVEY. A COMPARATIVE TEST OVER A LARGE CLASS OF DIFFERENTIAL EQUATIONS IS GIVEN IN REF [4].
FROM THESE RESULTS IT APPEARS THAT CALLS WITH THIRDORDER = "TRUE" ARE LESS ADVISABLE.

REFERENCES:

[1]. K. DEKKER.
AN ALGOL 60 VERSION OF EXPONENTIALLY FITTED RUNGE-KUTTA METHODS (DUTCH).
NR 25 (1972), MATHEMATICAL CENTRE.

[2]. T. J. DEKKER, P. W. HEMKER AND P. J. VAN DER HOUWEN.
COLLOQUIUM STIFF DIFFERENTIAL EQUATIONS 1 (DUTCH).
MC SYLLABUS 15.1, (1972) MATHEMATICAL CENTRE.

[3]. P. A. BEENTJES, K. DEKKER, H. C. HEMKER, S.P.N. VAN KAMPEN AND G. M. WILLEMS.
COLLOQUIUM STIFF DIFFERENTIAL EQUATIONS 2 (DUTCH).
MC SYLLABUS 15.2, (1973) MATHEMATICAL CENTRE.

[4]. (TO APPEAR).
COLLOQUIUM STIFF DIFFERENTIAL EQUATIONS 3 (DUTCH).
MC SYLLABUS 15.3, (1974) MATHEMATICAL CENTRE.

EXAMPLE OF USE:

CONSIDER THE SYSTEM OF DIFFERENTIAL EQUATIONS:
DY[1]/DX = -Y[1] + Y[1] * Y[2] + .99 * Y[2]
DY[2]/DX = -1000 * (-Y[1] + Y[1] * Y[2] + Y[2])
WITH THE INITIAL CONDITIONS AT X = 0:
Y[1] = 1 AND Y[2] = 0. (SEE REF[2], PAGE 11).
THE SOLUTION AT X = 50 IS APPROXIMATELY:
Y[1] = .765 878 320 487 AND Y[2] = .433 710 353 5768.
THE FOLLOWING PROGRAM SHOWS SOME DIFFERENT CALLS OF THE PROCEDURE EFRK, AND ILLUSTRATES THE ACCURACIES WHICH MAY BE OBTAINED BY THEM:

```
"BEGIN"
    "PROCEDURE" EFRK(X,XE,M0,M,Y,SIGMA,PHI,DIAMETER,DERIVATIVE,K,
                                    STEP,R,L,BETA,THIRDORDER,TOL,OUTPUT);
    "CODE" 33070;
    "INTEGER" K,R,L,PASSES;
    "REAL" X,SIGMA,PHI,TIME,STEP,DIAMETER;
    "REAL" "ARRAY" Y[1:2],BETA[0:6];

    "PROCEDURE" DER(X,Y); "REAL" X; "ARRAY" Y;
    "BEGIN" "REAL" Y1,Y2; Y1:=Y[1]; Y2:=Y[2];
        Y[1]:=(Y1+.99)*(Y2-1)+.99;
        Y[2]:=1000*((1+Y1)*(1-Y2)-1);
        PASSES:=PASSES+1
    "END";

    "PROCEDURE" OUT;
    "BEGIN" "REAL" S;
        S:=(-1000*Y[1]-1001+Y[2])/2;
        SIGMA:=ABS(S-SQRT(S*S+10*(Y[2]-1)));
        DIAMETER:=2*STEP*ABS(1000*(1.99*Y[2]-2*Y[1]*(1-Y[2])));
        "IF" X=50 "THEN"
        OUTPUT(61,"("4BD,2BD.2(-5ZD),2(4B+.3DB3DB3D),-5ZD.3D,/")",
        R,L,K,PASSES,Y[1],Y[2],CLOCK-TIME)
    "END";

    OUTPUT(61,"(""("     THIS LINE AND THE FOLLOWING TEXT IS ")"
    "("PRINTED BY THIS PROGRAM")",///,
    "("     THE RESULTS WITH EFRK ARE:")",/,
    "("     R  L      K    DER.EV.          Y[1]                    Y[2]")"
    "("           TIME")",/")");
    PHI:=4*ARCTAN(1); BETA[0]:=BETA[1]:=1;
    "FOR" R:=1,2,3 "DO" "FOR" L:=1,2,3 "DO"
    "BEGIN" "FOR" K:=2 "STEP" 1 "UNTIL" R "DO"
        BETA[K]:=BETA[K-1]/K;
        "FOR" STEP:=1,.1 "DO"
        "BEGIN" PASSES:=K:=0; X:=Y[2]:=0; Y[1]:=1; TIME:=CLOCK;
            OUT;
            EFRK(X,50,1,2,Y,SIGMA,PHI,DIAMETER,DER,K,STEP,R,L,BETA,
                    R>=3,"-4,OUT);
            "END"; OUTPUT(61,"("/")");
        "END";
    "END";
```

THIS LINE AND THE FOLLOWING TEXT IS PRINTED BY THIS PROGRAM:

THE RESULTS WITH EFRK ARE:

R	L	K	DER.EV.	Y[1]	Y[2]	TIME
1	1	237	474	+.765 812 555	+.433 689 306	1.395
1	1	501	1002	+.765 847 870	+.433 700 619	3.381
1	2	52	156	+.765 570 874	+.433 615 119	0.465
1	2	501	1503	+.765 848 220	+.433 700 709	4.200
1	3	52	208	+.765 571 278	+.433 615 202	0.531
1	3	500	2000	+.765 848 512	+.433 700 827	4.879
2	1	3317	9951	+.765 878 320	+.433 710 353	21.808
2	1	1050	3150	+.765 878 321	+.433 710 330	7.153
2	2	174	696	+.765 878 335	+.433 710 335	1.385
2	2	501	2004	+.765 878 323	+.433 709 211	4.915
2	3	57	285	+.765 881 339	+.433 817 185	0.642
2	3	501	2505	+.765 878 323	+.433 709 725	5.756
3	1	7010	28.040	+.765 878 320	+.433 710 354	55.298
3	1	3255	13020	+.765 878 320	+.433 710 374	25.772
3	2	949	4745	+.765 878 319	+.433 711 893	8.499
3	2	1384	6920	+.765 862 498	+.449 724 830	13.452
3	3	917	5502	+.765 878 018	+.434 105 184	9.143
3	3	1166	6996	+.765 861 696	+.433 705 641	15.512

SOURCE TEXT(S):

```
"CODE"" 33070;
"PROCEDURE" EFRK(T,TE,M0,M,U,SIGMA,PHI,DIAMETER,DERIVATIVE,K,STEP,R,L,
        BETA,THIRDORDER,TOL,OUTPUT);
"VALUE" R,L;
"INTEGER" M0,M,K,R,L;
"REAL" T,TE,SIGMA,PHI,DIAMETER,STEP,TOL;
"ARRAY" U,BETA;
"BOOLEAN" THIRDORDER;
"PROCEDURE" DERIVATIVE,OUTPUT;
"BEGIN" "INTEGER" N;
    "REAL" THETA0,THETANM1,H,B,B0,PHI0,PHIL,PI,COSPHI,SINPHI,EPS,BETAR;
    "BOOLEAN" FIRST,LAST,COMPLEX,CHANGE;
    "INTEGER" "ARRAY" P[1:L];
    "REAL" "ARRAY" MU,LABDA[0:R+L-1],PT[0:R],FAC,BETAC[0:L-1],RL[M0:M],
                   A[1:L,1:L],AUX[0:3];
    "PROCEDURE" ELMVEC(L,U,SHIFT,A,B,X); "CODE" 34020;
    "PROCEDURE" SOL(A,N,P,B); "CODE" 34051;
    "PROCEDURE" DEC(A,N,AUX,P); "CODE" 34300;                    "COMMENT"
```

```
;

    "PROCEDURE" FORM CONSTANTS;
    "BEGIN" "INTEGER" I;
        FIRST:="FALSE";
        FAC[0]:=1;
        "FOR" I:=1 "STEP" 1 "UNTIL" L-1 "DO" FAC[I]:=I*FAC[I-1];
        PT[R]:=L*FAC[L-1];
        "FOR" I:=1 "STEP" 1 "UNTIL" R "DO"
        PT[R-I]:=PT[R-I+1]*(L+I)/I
"END" FORM CONSTANTS;

"PROCEDURE" FORM BETA;
"BEGIN" "INTEGER" I,J; "REAL" BB,C,D;
    "IF" FIRST "THEN" FORM CONSTANTS;
    "IF" L=1 "THEN"
    "BEGIN" C:=1-EXP(-B);
        "FOR" J:=1 "STEP" 1 "UNTIL" R "DO" C:=BETA[J]-C/B;
        BETA[R+1]:=C/B
    "END" "ELSE"
    "IF" B>40 "THEN"
    "BEGIN" "FOR" I:=R+1 "STEP" 1 "UNTIL" R+L "DO"
        "BEGIN" C:=0;
            "FOR" J:=0 "STEP" 1 "UNTIL" R "DO"
            C:=BETA[J]*PT[J]/(I-J)-C/B;
            BETA[I]:=C/B/FAC[L+R-I]/FAC[I-R-1]
    "END";
"END" "ELSE"
"BEGIN" D:=C:=EXP(-B); BETAC[L-1]:=D/FAC[L-1];
    "FOR" I:=1 "STEP" 1 "UNTIL" L-1 "DO"
    "BEGIN" C:=B*C/I; D:=D+C; BETAC[L-1-I]:=D/FAC[L-1-I] "END";
    BB:=1;
    "FOR" I:=R+1 "STEP" 1 "UNTIL" R+L "DO"
    "BEGIN" C:=0;
        "FOR" J:=0 "STEP" 1 "UNTIL" R "DO"
        C:=(BETA[J]-("IF" J<L "THEN" BETAC[J] "ELSE" 0))*
            PT[J]/(I-J)-C/B;
        BETA[I]:=C/B/FAC[L+R-I]/FAC[I-R-1]+
                ("IF" I<L "THEN" BB*BETAC[I] "ELSE" 0);
        BB:=BB*B
        "END"
    "END"
"END" FORM BETA;

"PROCEDURE" SOLUTION OF COMPLEX EQUATIONS;
"BEGIN" "INTEGER" I,J,C1,C3;
    "REAL" C2,E,B1,ZI,COSIPHI,SINIPHI,COSPHIL;                    "COMMENT"
    "REAL" "ARRAY" D[1:L];
```

```
"PROCEDURE" ELEMENTS OF MATRIX;
"BEGIN" PHIL:=PHI0;
    COSPHI:=COS(PHIL); SINPHI:=SIN(PHIL);
    COSIPHI:=1; SINIPHI:=0;
    "FOR" I:=0 "STEP" 1 "UNTIL" L-1 "DO"
    "BEGIN" C1:=R+1+I; C2:=1;
        "FOR" J:=L-1 "STEP" -2 "UNTIL" 1 "DO"
        "BEGIN" A[J,L-I]:=C2*COSIPHI;
            A[J+1,L-I]:=C2*SINIPHI;
            C2:=C1*C2; C1:=C1-1
        "END";
        COSPHIL:=COSIPHI*COSPHI-SINIPHI*SINPHI;
        SINIPHI:=COSIPHI*SINPHI+SINIPHI*COSPHI;
        COSIPHI:=COSPHIL
    "END";
    AUX[2]:=0; DEC(A,L,AUX,P)
"END" EL OF MAT;

"PROCEDURE" RIGHTHANDSIDE;
"BEGIN" E:=EXP(B*COSPHI);
    B1:=B*SINPHI-(R+1)*PHIL;
    COSIPHI:=E*COS(B1); SINIPHI:=E*SIN(B1);
    B1:=1/B; ZI:=B1**R;
    "FOR" J:=L "STEP" -2 "UNTIL" 2 "DO"
    "BEGIN" D[J]:=ZI*SINIPHI;
        D[J-1]:=ZI*COSIPHI;
        COSPHIL :=COSIPHI*COSPHI-SINIPHI*SINPHI;
        SINIPHI:=COSIPHI*SINPHI+SINIPHI*COSPHI;
        COSIPHI:=COSPHIL;
        ZI:=ZI*B
    "END";
    COSIPHI:=ZI:=1; SINIPHI:=0;
    "FOR" I:=R "STEP" -1 "UNTIL" 0 "DO"
    "BEGIN" C1:=I; C2:=BETA[I];
        C3:="IF" 2*I>L-2 "THEN" 2 "ELSE" L-2*I;
        COSPHIL :=COSIPHI*COSPHI-SINIPHI*SINPHI;
        SINIPHI:=COSIPHI*SINPHI+SINIPHI*COSPHI;
        COSIPHI:=COSPHIL;
        "FOR" J:=L "STEP" -2 "UNTIL" C3 "DO"
        "BEGIN" D[J]:=D[J]+ZI*C2*SINIPHI;
            D[J-1]:=D[J-1]-ZI*C2*COSIPHI;
            C2:=C2*C1; C1:=C1-1
        "END";
        ZI:=ZI*B1
    "END"
"END" RIGHT HAND SIDE;

"IF" PHI0^=PHIL "THEN" ELEMENTS OF MATRIX;
RIGHTHANDSIDE;
SOL(A,L,P,D);
"FOR" I:=1 "STEP" 1 "UNTIL" L "DO" BETA[R+I]:=D[L+1-I]*B1
"END" SOLOFCOMEQ;                                    "COMMENT"
```

```
;

    "PROCEDURE" COEFFICIENT;
    "BEGIN" "INTEGER" J,K; "REAL" C;
        B0:=B; PHI0:=PHI;
        "IF" B>=1 "THEN"
        "BEGIN" "IF" COMPLEX "THEN" SULUTION OF COMPLEX EQUATIONS
        "ELSE" FORM BETA
    "END";
    LABDA[0]:=MU[0]:=0;
    "IF" THIRDORDER "THEN"
    "BEGIN" THETA0:=.25; THETANM1:=.75;
        "IF" B<1 "THEN"
        "BEGIN" C:=MU[N-1]:=2/3; LABDA[N-1]:=5/12;
            "FOR" J:=N-2 "STEP" -1 "UNTIL" 1 "DO"
            "BEGIN" C:=MU[J]:=C/(C-.25)/(N-J+1);
                LABDA[J]:=C-.25
            "END"
        "END" "ELSE"
        "BEGIN" C:=MU[N-1]:=BETA[2]*4/3; LABDA[N-1]:=C-.25;
            "FOR" J:=N-2 "STEP" -1 "UNTIL" 1 "DO"
            "BEGIN" C:=MU[J]:=C/(C-.25)*BETA[N-J+1]/BETA[N-J]/
                        ("IF" J<L "THEN" B "ELSE" 1);
                LABDA[J]:=C-.25
            "END"
        "END" "ELSE"
        "BEGIN" THETA0:=0; THETANM1:=1;
            "IF" B<1 "THEN"
            "BEGIN" "FOR" J:=N-1 "STEP" -1 "UNTIL" 1 "DO"
                MU[J]:=LABDA[J]:=1/(N-J+1)
            "END" "ELSE"
            "BEGIN" LABDA[N-1]:=MU[N-1]:=BETA[2];
                "FOR" J:=N-2 "STEP" -1 "UNTIL" 1 "DO"
                MU[J]:=LABDA[J]:=BETA[N-J+1]/BETA[N-J]/
                        ("IF" J<L "THEN" B "ELSE" 1)
            "END"
        "END"
    "END" COEFFICIENT;                                          "COMMENT"
```

```
;


      "PROCEDURE" STEPSIZE;
      "BEGIN" "REAL" D,HSTAB,HSTABINT;
           H:=STEP;
           D:=ABS(SIGMA*SIN(PHI));
           COMPLEX:=L//2*2=L "AND" 2*D>DIAMETER;
           "IF" DIAMETER>0 "THEN"
           HSTAB:=(SIGMA**2/(DIAMETER*(DIAMETER*.25+D)))**(L*.5/R)/
                  BETAR/SIGMA
           "ELSE" HSTAB:=H;
           D:= "IF" THIRDORDER "THEN" (2*TOL/EPS/BETA[R])**(1/(N-1))*
                4**((L-1)/(N-1)) "ELSE" (TOL/EPS)**(1/R)/BETAR;
           HSTABINT:= ABS(D/SIGMA);
           "IF" H>HSTAB "THEN" H:=HSTAB;
           "IF" H>HSTABINT "THEN" H:=HSTABINT;
           "IF" T+H>TE*(1-K*EPS) "THEN"
           "BEGIN" LAST:="TRUE"; H:=TE-T "END";
           B:=H*SIGMA; D:=DIAMETER*.1*H; D:=D*D;
           "IF" H<T*EPS  "THEN" "GOTO" ENDOFEFRK;
           CHANGE:=B0=-1 "OR" ((B-B0)*(B-B0)+B*B0*(PHI-PHI0)*(PHI-PHI0)>D)
      "END" STEPSIZE;

      "PROCEDURE" DIFFERENCESCHEME ;
           "BEGIN" "INTEGER" I,J; "REAL" MT,LT,THT;
                I:=-1;
                NEXTTERM;
                I:=I+1; MT:=MU[I]*H; LT:=LABDA[I]*H;
                "FOR" J:=M0 "STEP" 1 "UNTIL" M "DO" RL[J]:=U[J]+LT*RL[J];
                DERIVATIVE(T+MT,RL);
                "IF" I=0 "OR" I=N-1 "THEN"
                "BEGIN" THT:="IF" I=0 "THEN" THETA0*H "ELSE" THETANM1*H;
                     ELMVEC(M0,M,0,U,RL,THT)
                "END";
                "IF" I<N-1 "THEN" "GOTO" NEXTTERM;
                T:=T+H
           "END" DIFFERENCE SCHEME;

           N:=R+L; FIRST:="TRUE"; B0:=-1; BETAR:=BETA[R]**(1/R);
           LAST:="FALSE"; EPS:=2**(-48); PI:=PHI0:=PHIL:=4*ARCTAN(1);
      NEXTLEVEL:
           STEPSIZE;
           "IF" CHANGE "THEN" COEFFICIENT;
           K:=K+1;
           DIFFERENCE SCHEME;
           OUTPUT;
           "IF" "NOT" LAST "THEN" "GOTO" NEXTLEVEL;
      ENDOFEFRK;
      "END" EXPONENTIALLY FITTED RUNGE KUTTA;
                "EOP"
```

SECTION 5.2.1.1.1.2 CONTAINS SIX ALTERNATIVE PROCEDURES FOR SOLVING FIRST-ORDER INITIAL VALUE PROBLEMS WITH THE JACOBIAN MATRIX AVAILABLE.

A. EFSIRK SOLVES AN INITIAL VALUE PROBLEM, GIVEN AS AN AUTONOMOUS SYSTEM OF FIRST ORDER DIFFERENTIAL EQUATIONS DY/DX = F(Y), BY MEANS OF AN EXPONENTIALLY FITTED, SEMI-IMPLICIT RUNGE-KUTTA METHOD; IN PARTICULAR THIS PROCEDURE IS SUITABLE FOR THE INTEGRATION OF STIFF EQUATIONS.

B. EFERK SOLVES INITIAL VALUE PROBLEMS, GIVEN AS AN AUTONOMOUS SYSTEM OF FIRST ORDER DIFFERENTIAL EQUATIONS, BY MEANS OF AN EXPONENTIALLY FITTED, EXPLICIT RUNGE KUTTA METHOD OF THIRD ORDER, WHICH INVOLVES THE USE OF THE JACOBIAN MATRIX. AUTOMATIC STEP CONTROL IS PROVIDED. IN PARTICULAR THIS METHOD IS SUITABLE FOR THE INTEGRATION OF STIFF DIFFERENTIAL EQUATIONS.

C. LINIGER1VS SOLVES INITIAL VALUE PROBLEMS, GIVEN AS AN AUTONOMOUS SYSTEM OF FIRST ORDER DIFFERENTIAL EQUATIONS, BY MEANS OF AN IMPLICIT, FIRST ORDER ACCURATE, EXPONENTIALLY FITTED ONESTEP METHOD.
AUTOMATIC STEPSIZE CONTROL IS PROVIDED.

D. LINIGER2 SOLVES INITIAL VALUE PROBLEMS, GIVEN AS AN AUTONOMOUS SYSTEM OF FIRST ORDER DIFFERENTIAL EQUATIONS, BY MEANS OF AN EXPONENTIALLY FITTED ONESTEP METHOD.
NO AUTOMATIC STEPSIZE CONTROL IS PROVIDED.

E. GMS SOLVES AN INITIAL VALUE PROBLEM, GIVEN AS AN AUTONOMOUS SYSTEM OF FIRST ORDER DIFFERENTIAL EQUATIONS DY / DX = F(Y), BY MEANS OF A THIRD ORDER GENERALIZED LINEAR MULTISTEP METHOD.

F. IMPEX SOLVES AN INITIAL VALUE PROBLEM, GIVEN AS AN AUTONOMOUS SYSTEM OF FIRST ORDER DIFFERENTIAL EQUATIONS, BY MEANS OF THE IMPLICIT MID-POINT RULE WITH SMOOTHING AND EXTRAPOLATION.
AUTOMATIC STEPSIZE CONTROL IS PROVIDED.

IN PARTICULAR ALL THESE METHODS ARE SUITABLE FOR THE INTEGRATION OF STIFF DIFFERENTIAL EQUATIONS.

AUTHOR: S.P.N. VAN KAMPEN.

INSTITUTE: MATHEMATICAL CENTRE.

RECEIVED: 730529.

BRIEF DESCRIPTION:

EFSIRK SOLVES AN INITIAL VALUE PROBLEM, GIVEN AS AN AUTONOMOUS SYSTEM OF FIRST ORDER DIFFERENTIAL EQUATIONS DY/DX = F(Y), BY MEANS OF AN EXPONENTIALLY FITTED, SEMI-IMPLICIT RUNGE-KUTTA METHOD; IN PARTICULAR THIS PROCEDURE IS SUITABLE FOR THE INTEGRATION OF STIFF EQUATIONS.

KEYWORDS:

DIFFERENTIAL EQUATIONS,
INITIAL VALUE PROBLEM,
AUTONOMOUS SYSTEM,
STIFF EQUATIONS,
SEMI-IMPLICIT RUNGE-KUTTA METHOD,
EXPONENTIAL FITTING,

CALLING SEQUENCE:

HEADING:
"PROCEDURE" EFSIRK(X, XE, M, Y, DELTA, DERIVATIVE, JACOBIAN, J,
N, AETA, RETA, HMIN, HMAX, LINEAR, OUTPUT);
"VALUE" M; "INTEGER" M, N;
"REAL" X, XE, DELTA, AETA, RETA, HMIN, HMAX;
"PROCEDURE" DERIVATIVE, JACOBIAN, OUTPUT;
"ARRAY" Y, J;
"BOOLEAN" LINEAR;

THE MEANING OF THE FORMAL PARAMETERS IS:
X: <VARIABLE>;
THE INDEPENDENT VARIABLE X;
ENTRY: THE INITIAL VALUE X0;
EXIT : THE END VALUE XE;
XE: <ARITHMETIC EXPRESSION>;
THE END VALUE OF X;
M: <ARITHMETIC EXPRESSION>;
THE NUMBER OF DIFFERENTIAL EQUATIONS;
Y: <ARRAY IDENTIFIER>;
"ARRAY" Y[1 : M];
THE DEPENDENT VARIABLE;
DURING THE INTEGRATION PROCESS THE COMPUTED SOLUTION
AT THE POINT X IS ASSIGNED TO THE ARRAY Y;
ENTRY: THE INITIAL VALUES OF THE SOLUTION OF THE SYSTEM;

DELTA: <ARITHMETIC EXPRESSION>;
DELTA DENOTES THE REAL PART OF THE POINT AT WHICH EXPONENTIAL FITTING IS DESIRED;
ALTERNATIVES:
DELTA = (AN ESTIMATE OF) THE REAL PART OF THE, IN ABSOLUTE VALUE, LARGEST EIGENVALUE OF THE JACOBIAN MATRIX OF THE SYSTEM;
DELTA < -10**14, IN ORDER TO OBTAIN ASYMPTOTIC STABILITY;
DELTA = 0, IN ORDER TO OBTAIN A HIGHER ORDER OF ACCURACY IN CASE OF LINEAR OR ALMOST LINEAR EQUATIONS;

DERIVATIVE: <PROCEDURE IDENTIFIER>;
"PROCEDURE" DERIVATIVE(A); "ARRAY" A;
WHEN IN EFSIRK DERIVATIVE IS CALLED, A[I] CONTAINS THE VALUES OF Y[I];
UPON COMPLETION OF A CALL OF DERIVATIVE, THE ARRAY A SHOULD CONTAIN THE VALUES OF F(Y);
NOTE THAT THE VARIABLE X SHOULD NOT BE USED IN DERIVATIVE, BECAUSE THE DIFFERENTIAL EQUATION IS SUPPOSED TO BE AUTONOMOUS;

JACOBIAN: <PROCEDURE IDENTIFIER>;
"PROCEDURE" JACOBIAN(J, Y); "ARRAY" J, Y;
WHEN IN EFSIRK JACOBIAN IS CALLED THE ARRAY Y CONTAINS THE VALUES OF THE DEPENDENT VARIABLE;
UPON COMPLETION OF A CALL OF JACOBIAN THE ARRAY J SHOULD CONTAIN THE VALUES OF THE JACOBIAN MATRIX OF F(Y);

J: <ARRAY IDENTIFIER>;
J[1 : M, 1 : M];
J IS AN AUXILLIARY ARRAY WHICH IS USED IN THE PROCEDURE JACOBIAN;

N: <VARIABLE>;
AN INTEGER WHICH COUNTS THE INTEGRATION STEPS;

AETA, RETA:
<ARITHMETIC EXPRESSION>;
REQUIRED ABSOLUTE AND RELATIVE LOCAL ACCURACY;

HMIN, HMAX:
<ARITHMETIC EXPRESSION>;
MINIMAL AND MAXIMAL STEPSIZE BY WHICH THE INTEGRATION IS PERFORMED;

LINEAR: <BOOLEAN EXPRESSION>;
IF LINEAR = "TRUE" THE PROCEDURE JACOBIAN WILL ONLY BE CALLED IF N = 1; THE INTEGRATION WILL THEN BE PERFORMED WITH A STEPSIZE HMAX; THE CORRESPONDING REDUCTION OF COMPUTING TIME CAN BE EXPLOITED IN CASE OF LINEAR OR ALMOST LINEAR EQUATIONS;

OUTPUT: <PROCEDURE IDENTIFIER>;
"PROCEDURE" OUTPUT;
IN OUTPUT ONE MAY PRINT THE VALUES OF E,G, X, Y[I], J[K, L] AND N.

DATA AND RESULTS: SEE REF[2] AND [3].

PROCEDURES USED:

```
VECVEC = CP34010,
MATVEC = CP34011,
MATMAT = CP34013,
GSSELM = CP34231,
SOLELM = CP34061.
```

REQUIRED CENTRAL MEMORY:

EXECUTION FIELD LENGTH: CIRCA M * M + 5 * M.

RUNNING TIME:

DEPENDS STRONGLY ON THE DIFFERENTIAL EQUATION TO BE SOLVED

LANGUAGE: ALGOL 60.

METHOD AND PERFORMANCE:

THE PROCEDURE EFSIRK IS AN EXPONENTIALLY FITTED, A-STABLE, SEMI-IMPLICIT RUNGE-KUTTA METHOD OF THIRD ORDER (SEE REF[1] AND [2]). THE ALGORITHM USES FOR EACH STEP TWO FUNCTION EVALUATIONS AND IF LINEAR = "FALSE" ONE EVALUATION OF THE JACOBIAN MATRIX. THE STEPSIZE IS NOT DETERMINED BY THE ACCURACY OF THE NUMERICAL SOLUTION, BUT BY THE AMOUNT BY WHICH THE GIVEN DIFFERENTIAL EQUATION DIFFERS FROM A LINEAR EQUATION (SEE REF[2]). THE PROCEDURE DOES NOT REJECT INTEGRATION STEPS.

REFERENCES:

[1].P.J. VAN DER HOUWEN.
ONE-STEP METHODS WITH ADAPTIVE STABILITY FUCNTIONS FOR THE INTEGRATION OF DIFFERENTIAL EQUATIONS.
LECTURE NOTES OF THE CONFERENCE ON NUMERISCHE, INSBESONDERE APPROXIMATIONSTHEORETISCHE BEHANDLUNG VON FUNKTIONALGLEICHUNGEN,
OBERWOLFACH, DECEMBER, 3 - 12, 1972.

[2].SYLLABUS COLLOQUIUM STIFF DIFFERENTIAL EQUATIONS 2 (DUTCH).
MATH.CENTR. SYLLABUS 15.2/73.

[3].SYLLABUS COLLOQUIUM STIFF DIFFERENTIAL EQUATIONS 3 (DUTCH).
MATH.CENTR. SYLLABUS 15.3/73.
TO APPEAR IN 1973.

EXAMPLE OF USE:

```
     WE CONSIDER THE DIFFERENTIAL EQUATION
     DY / DX = -EXP(X) * (Y - LN(X)) + 1 / X,
ON THE INTERVAL  [0.01, 8], WITH  INITIAL VALUE  Y(0.01) = LN(0.01)
AND ANALYTICAL SOLUTION Y(X) = LN(X);
FOR THE FIT POINT WE USE THE  EIGENVALJE  OF THE JACOBIAN MATRIX,
I.E. DELTA = -EXP(X);

"BEGIN"
    "PROCEDURE" EFSIRK(X, XE, M, Y, DELTA, DERIVATIVE, JACOBIAN, J,
                       N, AETA, RETA, HMIN, HMAX, LINEAR, OUTPUT);
    "CODE" 33160;
    "PROCEDURE" DER(Y); "ARRAY" Y;
    "BEGIN" "REAL" Y2; Y2:= Y[2];
        DELTA:= -EXP(Y2); LNX:= LN(Y2);
        Y[1]:= (Y[1] - LNX) * DELTA + 1 / Y2;
        Y[2]:= 1
    "END" DER;
    "PROCEDURE" JAC(J, Y); "ARRAY" J, Y;
    "BEGIN" "REAL" Y2; Y2:= Y[2];
        J[1, 1]:= DELTA;
        J[1, 2]:= (Y[1] - LNX - 1 / Y2) * DELTA - 1 / (Y2 * Y2);
        J[2, 1]:= J[2, 2]:= 0
"END" JAC;
"PROCEDURE" OUTP;
"IF" X = XE "THEN"
"BEGIN" "REAL" Y1; Y1:= Y[1]; LNX:= LN(X);
    OUTPUT(61, "(""("N = ")", 2ZD,
                "("    X = ")", +D.D,
                "("  Y(X) = ")", +D.5D,
                "("  DELTA = ")", +3ZD.2D, /,
                "("ABS. ERR. = ")", .2D"+2D,
                "("  REL. ERR. = ")", .2D"+2D, //")",
                N, X, Y1, DELTA,
                ABS(Y1 - LNX), ABS((Y1 - LNX) / LNX));
    "IF" X = 0.4 "THEN" XE:= 8
"END" OUTP;
"INTEGER" N;
"REAL" X, XE, DELTA, LNX;
    "ARRAY" Y[1 : 2], J[1 : 2, 1 : 2];

    XE:= 0.4; X:= 0.01; Y[1]:= LN(0.01); Y[2]:= X;
    EFSIRK(X, XE, 2, Y, DELTA, DER, JAC, J,
           N, "-2, "-2, 0.005, 1.5, "FALSE", OUTP)
"END"
```

THIS PROGRAM DELIVERS:

```
    N =  10    X = +0.4  Y(X) = -0.91099  DELTA =     -1.44
    ABS. ERR. = .53"-02  REL. ERR. = .58"-02

    N = 98     X = +8.0  Y(X) = +2.07911  DELTA = -2980.02
    ABS. ERR. = .33"-03  REL. ERR. = .16"-03
```

SOURCE TEXT(S):

```
"CODE"" 33160;
    "PROCEDURE" EFSIRK(X, XE, M, Y, DELTA, DERIVATIVE, JACOBIAN, J,
                       N, AETA, RETA, HMIN, HMAX, LINEAR, OUTPUT);
    "VALUE" M; "INTEGER" M, N;
    "REAL" X, XE, DELTA, AETA, RETA, HMIN, HMAX;
    "PROCEDURE" DERIVATIVE, JACOBIAN, OUTPUT;
    "BOOLEAN" LINEAR;
    "ARRAY" Y, J;

    "BEGIN" "INTEGER" K, L;
        "REAL" STEP, H, MU0, MU1, MU2, THETA0, THETA1, NU1, NU2,
        NU3, YK, FK, C1, C2, D;
        "ARRAY" F, K0, LABDA[1 : M], J1[1 : M,1 : M], AUX[1 : 7];
        "INTEGER" "ARRAY" RI, CI[1 : M];
        "BOOLEAN" LIN;
        "REAL" "PROCEDURE" VECVEC(L, U, SHIFT, A, B); "CODE" 34010;
        "REAL" "PROCEDURE" MATMAT(L, U, I, J, A, B); "CODE" 34013;
        "REAL" "PROCEDURE" MATVEC(L, U, I, A, B); "CODE" 34011;
        "PROCEDURE" GSSELM(A, N, AUX, RI, CI); "CODE" 34231;
        "PROCEDURE" SOLELM(A, N, RI, CI, B); "CODE" 34061;

        "REAL" "PROCEDURE" STEPSIZE;
        "BEGIN" "REAL" DISCR, ETA, S;
            "IF" LINEAR "THEN" S:= H:= HMAX "ELSE"
            "IF" N = 1 "OR" HMIN = HMAX "THEN" S:= H:= HMIN "ELSE"
            "BEGIN" ETA:= AETA + RETA * SQRT(VECVEC(1, M, 0, Y, Y));
                C1:= NU3 * STEP; "FOR" K:= 1 "STEP" 1 "UNTIL" M "DO"
                LABDA[K]:= LABDA[K] + C1 * F[K] - Y[K];
                DISCR:= SQRT(VECVEC(1, M, 0, LABDA, LABDA));
                S:= H:= (ETA / (0,75 * (ETA + DISCR)) + 0,33) * H;
                "IF" H < HMIN "THEN" S:= H:= HMIN "ELSE"
                "IF" H > HMAX "THEN" S:= H:= HMAX
            "END";
            "IF" X + S > XE "THEN" S:= XE - X;
            LIN:= STEP = S "AND" LINEAR; STEPSIZE:= S
        "END" STEPSIZE;

        "PROCEDURE" COEFFICIENT;
        "BEGIN" "REAL" Z1, E, ALPHA1, A, B;
            "OWN" "REAL" Z2;
            Z1:= STEP * DELTA; "IF" N = 1 "THEN" Z2:= Z1 + Z1;
            "IF" ABS(Z2 - Z1) > " - 6 * ABS(Z1) "OR" Z2 > - 1 "THEN"
            "BEGIN" A:= Z1 * Z1 + 12; B:= 6 * Z1;
                "IF" ABS(Z1) < 0,1 "THEN"
                ALPHA1:= (Z1 * Z1 / 140 - 1) * Z1 / 30 "ELSE"
                "IF" Z1 < - "14 "THEN" ALPHA1:= 1 / 3 "ELSE"
                "IF" Z1 < - 33 "THEN"
                ALPHA1:= (A + B) / (3 * Z1 * (2 + Z1)) "ELSE"
                "BEGIN" E:= "IF" Z1 < 230 "THEN" EXP(Z1) "ELSE" "100;
                    ALPHA1:= ((A - B) * E - A - B) /
                             (((2 - Z1) * E - 2 - Z1) * 3 * Z1)
                "END";                                          "COMMENT"
```

```
                MU2:= (1 / 3 + ALPHA1) * 0.25;
                MU1:= - (1 + ALPHA1) * 0.5;
                MU0:= (6 * MU1 + 2) / 9; THETA0:= 0.25;
                THETA1:= 0.75; A:= 3 * ALPHA1;
                NU3:= (1 + A) / (5 - A) * 0.5; A:= NU3 + NU3;
                NU1:= 0.5 - A; NU2:= (1 + A) * 0.75;
                Z2:= Z1
            "END"
        "END" COEFFICIENT;

        "PROCEDURE" DIFFERENCE SCHEME;
        "BEGIN" DERIVATIVE(F); STEP:= STEPSIZE;
            "IF" "NOT" LINEAR "OR" N = 1 "THEN" JACOBIAN(J, Y);
            "IF" "NOT" LIN "THEN"
            "BEGIN" COEFFICIENT;
                C1:= STEP * MU1; D:= STEP * STEP * MU2;
                "FOR" K:= 1 "STEP" 1 "UNTIL" M "DO"
                "BEGIN" "FOR" L:= 1 "STEP" 1 "UNTIL" M "DO"
                    J1[K,L]:= D * MATMAT(1, M, K, L, J, J) +
                              C1 * J[K,L];
                    J1[K,K]:= J1[K,K] + 1
                "END";
                GSSELM(J1, M, AUX, RI, CI)
            "END";
            C1:= STEP * STEP * MU0; D:= STEP * 2 / 3;
            "FOR" K:= 1 "STEP" 1 "UNTIL" M "DO"
            "BEGIN" K0[K]:= FK:= F[K];
                LABDA[K]:= D * FK + C1 * MATVEC(1, M, K, J, F)
            "END";
            SOLELM(J1, M, RI, CI, LABDA);
            "FOR" K:= 1 "STEP" 1 "UNTIL" M "DO" F[K]:= Y[K] + LABDA[K];
            DERIVATIVE(F);
            C1:= THETA0 * STEP; C2:= THETA1 * STEP; D:= NU1 * STEP;
            "FOR" K:= 1 "STEP" 1 "UNTIL" M "DO"
            "BEGIN" YK:= Y[K]; FK:= F[K];
                LABDA[K]:= YK + D * FK + NU2 * LABDA[K];
                Y[K]:= F[K]:= YK + C1 * K0[K] + C2 * FK
            "END"
        "END" DIFFERENCE SCHEME;

            AUX[2]:= "-14; AUX[4]:= 8;
            "FOR" K:= 1 "STEP" 1 "UNTIL" M "DO" F[K]:= Y[K];
            N:= 0; OUTPUT; STEP:= 0;
        NEXT STEP; N:= N + 1;
            DIFFERENCE SCHEME; X:= X + STEP; OUTPUT;
            "IF" X < XE "THEN" "GOTO" NEXT STEP
        "END" EFSIRK;
            "EOP"
```

AUTHOR: K.DEKKER.

INSTITUTE: MATHEMATICAL CENTRE.

RECEIVED: 1973/07/31.

BRIEF DESCRIPTION:

EFERK SOLVES INITIAL VALUE PROBLEMS , GIVEN AS AN AUTONOMOUS SYSTEM OF FIRST ORDER DIFFERENTIAL EQUATIONS, BY MEANS OF AN EXPONENTIALLY FITTED, EXPLICIT RUNGE KUTTA METHOD OF THIRD ORDER, WHICH INVOLVES THE USE OF THE JACOBIAN MATRIX. AUTOMATIC STEP CONTROL IS PROVIDED. IN PARTICULAR THIS METHOD IS SUITABLE FOR THE INTEGRATION OF STIFF DIFFERENTIAL EQUATIONS.

KEYWORDS:
DIFFERENTIAL EQUATIONS,
INITIAL VALUE PROBLEMS,
STIFF EQUATIONS,
EXPONENTIAL FITTING,
EXPLICIT RUNGE KUTTA METHODS.

CALLING SEQUENCE:

```
    THE HEADING OF THE PROCEDURE EFERK READS:
    "PROCEDURE" EFERK(X,XE,M,Y,SIGMA,PHI,DERIVATIVE,J,JACOBIAN,
                      K,L,AUT,AETA,RETA,HMIN,HMAX,LINEAR,OUTPUT);
    "VALUE" L;
    "INTEGER" M,K,L;
    "REAL" X,XE,SIGMA,PHI,AETA,RETA,HMIN,HMAX;
    "ARRAY" Y,J;
    "BOOLEAN" AUT,LINEAR;
    "PROCEDURE" DERIVATIVE,JACOBIAN,OUTPUT;

    THE MEANING OF THE FORMAL PARAMETERS IS:
    X:          <VARIABLE>;
                THE INDEPENDENT VARIABLE X;
                CAN BE USED IN DERIVATIVE, JACOBIAN, OUTPUT, ETC.;
                ENTRY: THE INITIAL VALUE X0;
                EXIT : THE FINAL VALUE XE;
    XE:         <ARITHMETIC EXPRESSION>;
                THE FINAL VALUE OF X (XE>=X);
    M:          <ARITHMETIC EXPRESSION>;
                THE NUMBER OF EQUATIONS;
    Y:          <ARRAY IDENTIFIER>;
                "REAL" "ARRAY" Y[1:M];
                THE DEPENDENT VARIABLE;
                ENTRY: THE INITIAL VALUES OF THE SYSTEM OF DIFFERENTIAL
                       EQUATIONS: Y[I] AT X=X0;
                EXIT : THE FINAL VALUES OF THE SOLUTION: Y[I] AT X=XE;
    SIGMA:      <ARITHMETIC EXPRESSION>;
                THE MODULUS OF THE POINT AT WHICH EXPONENTIAL FITTING IS
                DESIRED, FOR EXAMPLE THE LARGEST NEGATIVE EIGENVALUE OF THE
                JACOBIAN MATRIX OF THE SYSTEM OF DIFFERENTIAL EQUATIONS;
    PHI:        <ARITHMETIC EXPRESSION>;
                THE ARGUMENT OF THE COMPLEX POINT AT WHICH EXPONENTIAL
                FITTING IS DESIRED;
    DERIVATIVE: <PROCEDURE IDENTIFIER>;
                THE HEADING OF THIS PROCEDURE READS:
                "PROCEDURE" DERIVATIVE(Y); "ARRAY" Y;
                THIS PROCEDURE SHOULD DELIVER THE RIGHT HAND SIDE OF THE
                I-TH DIFFERENTIAL EQUATION AT THE POINT (Y) AS Y[I];
    J:          <ARRAY IDENTIFIER>;
                "REAL" "ARRAY" J[1:M,1:M];
                THE JACOBIAN MATRIX OF THE SYSTEM;
                THE ARRAY J SHOULD BE UPDATED IN THE PROCEDURE JACOBIAN;
```

JACUBIAN: <PROCEDURE IDENTIFIER>;
THE HEADING OF THIS PROCEDURE READS;
"PROCEDURE" JACOBIAN(J,Y); "ARRAY" J,Y;
IN THIS PROCEDURE THE JACOBIAN AT THE POINT (Y) HAS TO BE
ASSIGNED TO THE ARRAY J;

K: <VARIABLE>;
COUNTS THE NUMBER OF INTEGRATION STEPS TAKEN;
FOR EXAMPLE, MAY BE USED IN THE EXPRESSION FOR XE;

L.: <ARITHMETIC EXPRESSION>;
ENTRY;
IF PHI = 4*ARCTAN(1): THE ORDER OF THE EXPONENTIAL FITTING,
ELSE TWICE THE ORDER OF THE EXPONENTIAL FITTING;

AUT: <BOOLEAN EXPRESSION>;
IF THE SYSTEM HAS BEEN WRITTEN IN AUTONOMOUS FORM BY ADDING
THE EQUATION DY[M]/DX = 1 TO THE SYSTEM , THEN AUT MAY HAVE
THE VALUE "FALSE", ELSE AUT SHOULD HAVE THE VALUE "TRUE";

AETA: <ARITHMETIC EXPRESSION>;
REQUIRED ABSOLUTE PRECISION IN THE INTEGRATION PROCESS;
AETA HAS TO BE POSITIVE;

RETA: <ARITHMETIC EXPRESSION>;
REQUIRED RELATIVE PRECISION IN THE INTEGRATION PROCESS;
RETA HAS TO BE POSITIVE;

HMIN: <ARITHMETIC EXPRESSION>;
THE STEPLENGTH CHOSEN WILL BE AT LEAST EQUAL TO HMIN;

HMAX: <ARITHMETIC EXPRESSION>;
THE STEPLENGTH CHOSEN WILL BE AT MOST EQUAL TO HMAX;

LINEAR: <ARITHMETIC EXPRESSION>;
THE PROCEDURE JACOBIAN IS CALLED ONLY IF LINEAR="FALSE" OR
K=0; SO IF THE SYSTEM IS LINEAR , LINEAR MAY HAVE THE VALUE
"TRUE";

OUTPUT: <PROCEDURE IDENTIFIER>;
THE HEADING OF THIS PROCEDURE READS;
"PROCEDURE" OUTPUT;
THIS PROCEDURE IS CALLED AT THE END OF EACH INTEGRATION
STEP , THE USER CAN ASK FOR OUTPUT OF SOME PARAMETERS , FOR
EXAMPLE X, K, Y.

DATA AND RESULTS: SEE EXAMPLE OF USE, AND REF[4].

PROCEDURES USED:

VECVEC= CP34010,
MATVEC= CP34011,
DEC = CP34300,
SOL = CP34051.

REQUIRED CENTRAL MEMORY:
EXECUTION FIELD LENGTH: CIRCA 30 + 4 * M + L * (5+L).

RUNNING TIME: DEPENDS STRONGLY ON THE DIFFERENTIAL EQUATION TO SOLVE.

LANGUAGE: ALGOL 60.

METHOD AND PERFORMANCE:

THE PROCEDURE EFERK IS AN EXPONENTIALLY FITTED, SEMI-EXPLICIT RUNGE KUTTA METHOD OF THIRD ORDER (SEE REF [1] AND [3]) . THE ALGORITHM USES FOR EACH STEP TWO FUNCTION EVALUATIONS AND IF LINEAR = "FALSE" ONE EVALUATION OF THE JACOBIAN MATRIX.THE STEPSIZE IS DETERMINED BY AN ESTIMATION OF THE LOCAL TRUNCATION ERROR BASED ON THE RESIDUAL FUNCTION (SEE REF[3]). INTEGRATION STEPS ARE NOT REJECTED.

REFERENCES:

[1]. P.J.VAN DER HOUWEN.
ONE-STEP METHODS WITH ADAPTIVE STABILITY FUNCTIONS FOR THE INTEGRATION OF DIFFERENTIAL EQUATIONS.
LECTURES NOTES OF THE CONFERENCE ON NUMERISCHE, INSBESONDERE APPROXIMATIONSTHEORETISCHE BEHANDLUNG VON FUNKTIONAL-GLEICHUNGEN.
OBERWOLFACH, DECEMBER, 3 -12, 1972.

[2]. T.J.DEKKER, P.W.HEMKER AND P.J.VAN DER HOUWEN.
COLLOQUIUM STIFF DIFFERENTIAL EQUATIONS 1 (DUTCH).
MC SYLLABUS 15.1, (1972) MATHEMATICAL CENTRE.

[3]. P.A.BEENTJES, K.DEKKER, H.C.HEMKER, S.P.N.VAN KAMPEN AND G.M.WILLEMS.
COLLOQUIUM STIFF DIFFERENTIAL EQUATIONS 2 (DUTCH).
MC SYLLABUS 15.2, (1973) MATHEMATICAL CENTRE.

[4]. (TO APPEAR).
COLLOQUIUM STIFF DIFFERENTIAL EQUATIONS 3 (DUTCH).
MC SYLLABUS 15.3, (1973) MATHEMATICAL CENTRE.

EXAMPLE OF USE:

CONSIDER THE SYSTEM OF DIFFERENTIAL EQJATIONS:
DY[1]/DX = -Y[1] + Y[1] * Y[2] + .99 * Y[2]
DY[2]/DX = -1000 * (-Y[1] + Y[1] * Y[2] + Y[2])
WITH THE INITIAL CONDITIONS AT X = 0:
Y[1] = 1 AND Y[2] = 0. (SEE REF[2], PAGE 11).
THE SOLUTION AT X = 50 IS APPROXIMATELY:
Y[1] = .765 878 320 487 AND Y[2] = .433 710 353 5768.
THE FOLLOWING PROGRAM SHOWS SOME DIFFERENT CALLS OF THE PROCEDURE EFERK,AND ILLUSTRATES THE ACCURACIES WHICH MAY BE OBTAINED BY THEM:

```
"BEGIN"
    "PROCEDURE" EFERK(X,XE,M,Y,SIGMA,PHI,DERIVATIVE,J,JACOBIAN,
                      K,L,AUT,AETA,RETA,HMIN,HMAX,LINEAR,OUTPUT);
    "CODE" 33120;

 "INTEGER" K,PASSES,PASJAC;
 "REAL" X,SIGMA,PHI,TIME,TOL;
 "REAL" "ARRAY" Y[1:2],J[1:2,1:2];

 "PROCEDURE" DER(Y); "ARRAY" Y;
 "BEGIN" "REAL" Y1,Y2; Y1:=Y[1]; Y2:=Y[2];
     Y[1]:=(Y1+.99)*(Y2-1)+.99;
     Y[2]:=1000*((1+Y1)*(1-Y2)-1);
     PASSES:=PASSES+1
"END";

 "PROCEDURE" JACOBIAN(J,Y); "ARRAY" J,Y;
 "BEGIN" J[1,1]:=Y[2]-1; J[1,2]:=.99+Y[1];
     J[2,1]:=1000*(1-Y[2]); J[2,2]:=-1000*(1+Y[1]);
     SIGMA:=ABS(J[2,2]+J[1,1]-SQRT((J[2,2]-J[1,1])**2+
            4*J[2,1]*J[1,2]))/2;
     PASJAC:=PASJAC+1
 "END" JACOBIAN;

 "PROCEDURE" OUT;
 "IF" X=50 "THEN"
OUTPUT(61,"("3(-5ZD),2(4B+.3DB3DB3D),-5ZD.3D,/")",K,PASSES,
 PASJAC,Y[1],Y[2],CLOCK-TIME);

 OUTPUT(61,"(""("     THIS LINE AND THE FOLLOWING TEXT IS ")"
 "("PRINTED BY THIS PROGRAM")",//,
 "("     THE RESULTS WITH EFERK -FIRST ORDER FIT- ARE:")",/,
 "("      K    DER.EV. JAC.EV.        Y[1]                  Y[2]")"
 "("           TIME")",/")");
 PHI:=4*ARCTAN(1);
 "FOR" TOL:=1,"-1,"-2,"-3 "DO"
 "BEGIN" PASSES:=PASJAC:=0; X:=Y[2]:=0; Y[1]:=1; TIME:=CLOCK;
     EFERK(X,50,2,Y,SIGMA,PHI,DER,J,JACOBIAN,K,1,"TRUE",TOL,
                                                 TOL,"-6,50,"FALSE",OUT);
     "END";
 "END";
```

THIS LINE AND THE FOLLOWING TEXT IS PRINTED BY THIS PROGRAM:

THE RESULTS WITH EFERK -FIRST ORDER FIT- ARE:

K	DER.EV.	JAC.EV.	Y[1]	Y[2]	TIME
93	186	93	+.765 883 211	+.428 752 781	1.170
105	210	105	+.765 878 445	+.433 569 561	1.296
147	294	147	+.765 878 317	+.433 708 489	1.834
266	532	266	+.765 878 320	+.433 710 229	3.297

SOURCE TEXT(S):

```
"CODE"" 33120;
"PROCEDURE" EFERK(X,XE,M,Y,SIGMA,PHI,DERIVATIVE,J,JACOBIAN,
                  K,L,AUT,AETA,RETA,HMIN,HMAX,LINEAR,OUTPUT);
"VALUE" L; "INTEGER" M,K,L;
"REAL" X,XE,SIGMA,PHI,AETA,RETA,HMIN,HMAX; "ARRAY" Y,J;
"BOOLEAN" AUT,LINEAR; "PROCEDURE" DERIVATIVE,JACOBIAN,OUTPUT;
"BEGIN" "INTEGER" M1,I;
    "REAL" H,B,B0,PHI0,COSPHI,SINPHI,ETA,DISCR,FAC,PI;
    "BOOLEAN" CHANGE,LAST;
    "INTEGER" "ARRAY" P[1:L];
    "REAL" "ARRAY" BETA,BETHA[0:L],BETAC[0:L+3],K0,D,D1,D2[1:M],
        A[1:L,1:L],AUX[1:3];
    "REAL" "PROCEDURE" VECVEC(L,U,SHIFT,A,B); "CODE" 34010;
    "REAL" "PROCEDURE" MATVEC(L,U,I,A,B); "CODE" 34011;
    "PROCEDURE" DEC(A,N,AUX,P); "CODE" 34300;
    "PROCEDURE" SOL(A,N,P,B); "CODE" 34051;
    "REAL" "PROCEDURE" SUM(I,L,U,T); "VALUE" L,U; "INTEGER" I,L,U;
    "REAL" T;
    "BEGIN" "REAL" S; S:=0;
        "FOR" I:=L "STEP" 1 "UNTIL" U "DO" S:=S+T;
    SUM:=S
"END";
"PROCEDURE" FORMBETA;
"IF" L=1 "THEN"
"BEGIN" BETHA[1]:=(.5-(1-(1-EXP(-B))/B)/B)/B;
    BETA[1]:=(1/6-BETHA[1])/B
"END" "ELSE"
"IF" L=2 "THEN"
"BEGIN" "REAL" E,EMIN1; E:=EXP(-B); EMIN1:=E-1;
    BETHA[1]:=(1-(3+E+4*EMIN1/B)/B)/B;
    BETHA[2]:=(.5-(2+E+3*EMIN1/B)/B)/B/B;
    BETA[2]:=(1/6-BETHA[1])/B/B;
    BETA[1]:=(1/3-(1.5-(4+E+5*EMIN1/B)/B)/B)/B
"END" "ELSE"
"BEGIN" "REAL" B0,B1,B2,A0,A1,A2,A3,C,D;
    BETAC[L-1]:=C:=D:=EXP(-B)/FAC;
    "FOR" I:=L-1 "STEP" -1 "UNTIL" 1 "DO"
    "BEGIN" C:=I*B*C/(L-I); BETAC[I-1]:=D:=D*I+C "END";
    B2:=.5-BETAC[2];
    B1:=(1-BETAC[1])*(L+1)/B;
    B0:=(1-BETAC[0])*(L+2)*(L+1)*.5/B/B;
    A3:=1/6-BETAC[3];
    A2:=B2*(L+1)/B;
    A1:=B1*(L+2)*.5/B;
    A0:=B0*(L+3)/3/B;
    D:=L/B;
    "FOR" I:=1 "STEP" 1 "UNTIL" L "DO"
    "BEGIN"BETA[I]:=(A3/I-A2/(I+1)+A1/(I+2)-A0/(I+3))*D+BETAC[I+3];
        BETHA[I]:=(B2/I-B1/(I+1)+B0/(I+2))*D+BETAC[I+2];
        D:=D*(L-I)/I/B;
    "END"
"END" FORMBETA;                                          "COMMENT"
```

```
    "PROCEDURE" SOLUTIONOFCOMPLEXEQUATIONS;
    "IF" L=2 "THEN"
    "BEGIN" "REAL" COS2PHI,COSA,SINA,E,ZI;
        PHI0:=PHI; COSPHI:=COS(PHI0); SINPHI:=SIN(PHI0);
        E:=EXP(B*COSPHI); ZI:=B*SINPHI-3*PHI0;
        SINA:=("IF" ABS(SINPHI)<"-6 "THEN" -E*(B+3)
              "ELSE" E*SIN(ZI)/SINPHI);
        COS2PHI:=2*COSPHI*COSPHI-1;
        BETHA[2]:=(.5+(2*COSPHI+(1+2*COS2PHI+SINA)/B)/B)/B/B;
        SINA:=("IF" ABS(SINPHI)<"-6 "THEN" E*(B+4)
              "ELSE" SINA*COSPHI-E*COS(ZI));
        BETHA[1]:=-(COSPHI+(1+2*COS2PHI+(4*COSPHI*COS2PHI+SINA)
                  /B)/B)/B;
        BETA[1]:=BETHA[2]+2*COSPHI*(BETHA[1]-1/6)/B;
        BETA[2]:=(1/6-BETHA[1])/B/B
    "END" "ELSE"

    "BEGIN" "INTEGER" J,C1;
        "REAL" C2,E,ZI,COSIPHI,SINIPHI,COSPHIL;
        "REAL" "ARRAY" D[1:L];
        "PROCEDURE" ELEMENTS OF MATRIX;
        "BEGIN" PHI0:=PHI;
            COSPHI:=COS(PHI0); SINPHI:=SIN(PHI0);
            COSIPHI:=1; SINIPHI:=0;
            "FOR" I:=0 "STEP" 1 "UNTIL" L-1 "DO"
            "BEGIN" C1:=4+I; C2:=1;
                "FOR" J:=L-1 "STEP" -2 "UNTIL" 1 "DO"
                "BEGIN"  A[J,L-I]:=C2*COSIPHI;
                    A[J+1,L-I]:=C2*SINIPHI;
                    C2:=C2*C1; C1:=C1-1
                "END";
                COSPHIL:=COSIPHI*COSPHI-SINIPHI*SINPHI;
                SINIPHI:=COSIPHI*SINPHI+SINIPHI*COSPHI;
                COSIPHI:=COSPHIL
            "END";
            AUX[2]:=0; DEC(A,L,AUX,P)
        "END" EL OF MAT;
        "PROCEDURE" RIGHT HAND SIDE;
        "BEGIN" E:=EXP(B*COSPHI);
            ZI:=B*SINPHI-4*PHI0;
            COSIPHI:=E*COS(ZI); SINIPHI:=E*SIN(ZI);
            ZI:=1/B/B/B;
            "FOR" J:=L "STEP" -2 "UNTIL" 2 "DO"
            "BEGIN" D[J]:=ZI*SINIPHI;
                D[J-1]:=ZI*COSIPHI;
                COSPHIL:=COSIPHI*COSPHI-SINIPHI*SINPHI;
                SINIPHI:=COSIPHI*SINPHI+SINIPHI*COSPHI;
                COSIPHI:=COSPHIL; ZI:=ZI*B
            "END";                                          "COMMENT"
```

```
                SINIPHI:=2*SINPHI*COSPHI;
                COSIPHI:=2*COSPHI*COSPHI-1;
                COSPHIL:=COSPHI*(2*COSIPHI-1);
                D[L]:=D[L]+SINPHI*(1/6+(COSPHI+(1+2*COSIPHI*(1+2*COSPHI/B))
                                /B)/B);
                D[L-1]:=D[L-1]-COSPHI/6-(.5*COSIPHI+(COSPHIL+(2*COSIPHI*
                                COSIPHI-1)/B)/B)/B;
                D[L-2]:=D[L-2]+SINPHI*(.5+(2*COSPHI+(2*COSIPHI+1)/B)/B);
                D[L-3]:=D[L-3]-.5*COSPHI-(COSIPHI+COSPHIL/B)/B;
                "IF" L<5 "THEN" "GOTO" END;
                D[L-4]:=D[L-4]+SINPHI+SINIPHI/B;
                D[L-5]:=D[L-5]-COSPHI-COSIPHI/B;
                "IF" L<7 "THEN" "GOTO" END;
                D[L-6]:=D[L-6]+SINPHI;
                D[L-7]:=D[L-7]-COSPHI;
        END;
        "END" RHS;
        "IF" PHI0"=PHI "THEN" ELEMENTS OF MATRIX;
        RIGHT HAND SIDE;
        SOL(A,L,P,D);
        ZI:=1/B;
        "FOR" I:=1 "STEP" 1 "UNTIL" L "DO"
                "BEGIN" BETA[I]:=D[L+1-I]*ZI;
                        BETHA[I]:=(I+3)*BETA[I];
                        ZI:=ZI/B
                "END"
        "END" SOLOFEQCOM;

        "PROCEDURE" COEFFICIENT;
        "BEGIN" B0:=B:=ABS(H*SIGMA);
                "IF" B>=.1 "THEN"
                "BEGIN" "IF" PHI"=PI "AND" L=2 "OR" ABS(PHI-PI)>.01 "THEN"
                        SOLUTION OF COMPLEX EQUATIONS "ELSE" FORMBETA
                "END" "ELSE"
                "BEGIN" "FOR" I:=1 "STEP" 1 "UNTIL" L "DO"
                        "BEGIN" BETHA[I]:=BETA[I-1];
                                BETA[I]:=BETA[I-1]/(I+3);
                        "END"
                "END"
        "END" COEFFICIENT;

        "PROCEDURE" LOCAL ERROR BOUND;
        ETA:=AETA+RETA*SQRT(VECVEC(1,M1,0,Y,Y));

        "PROCEDURE" STEPSIZE;
        "BEGIN" LOCAL ERROR BOUND;
                "IF" K=0 "THEN"
                "BEGIN" DISCR:=SQRT(VECVEC(1,M1,0,D,D)); H:=ETA/DISCR
                "END" "ELSE"
                "BEGIN" DISCR:=H*SQRT(SUM(I,1,M1,(D[I]-D2[I])**2))/ETA;
                        H:=H*("IF" LINEAR "THEN" 4/(4+DISCR)+.5
                                        "ELSE" 4/(3+DISCR)+1/3)
                "END";                                                  "COMMENT"
```

```
        "IF" H<HMIN "THEN" H:=HMIN;
        "IF" H>HMAX "THEN" H:=HMAX;
        B:=ABS(H*SIGMA);
        CHANGE:=ABS(1-B/B0)>.05 "OR" PHI"=PHI0;
        "IF" 1.1*H>=XE-X "THEN"
            "BEGIN" CHANGE:=LAST:="TRUE"; H:=XE-X "END";
            "IF" "NOT" CHANGE "THEN" H:=H*B0/B
    "END" STEPSIZE;

    "PROCEDURE" DIFFERENCE SCHEME;
    "BEGIN" "INTEGER" K;
        "REAL" BETAI,BETHAI;
        "IF" M1<M "THEN"
        "BEGIN" D2[M]:=1; K0[M]:=Y[M]+2*H/3; Y[M]:=Y[M]+.25*H "END";
        "FOR" K:=1 "STEP" 1 "UNTIL" M1 "DO"
        "BEGIN" K0[K]:=Y[K]+2*H/3*D[K];
            Y[K]:=Y[K]+.25*H*D[K];
            D1[K]:=H*MATVEC(1,M,K,J,D);
            D2[K]:=D1[K]+D[K]
        "END";
        "FOR" I:=0 "STEP" 1 "UNTIL" L "DO"
        "BEGIN" BETAI:=4*BETA[I]/3; BETHAI:=BETHA[I];
            "FOR" K:=1 "STEP" 1 "UNTIL" M1 "DO" D[K]:=H*D1[K];
            "FOR" K:=1 "STEP" 1 "UNTIL" M1 "DO"
            "BEGIN" K0[K]:=K0[K]+BETAI*D[K];
                D1[K]:=MATVEC(1,M1,K,J,D);
                D2[K]:=D2[K]+BETHAI*D1[K]
            "END"
        "END";
        DERIVATIVE(K0);
        "FOR" K:=1 "STEP" 1 "UNTIL" M "DO" Y[K]:=Y[K]+.75*H*K0[K]
    "END" DIFF SCHEME;

    B0:=PHI0:=-1; PI:=4*ARCTAN(1);
    BETAC[L]:=BETAC[L+1]:=BETAC[L+2]:=BETAC[L+3]:=0;
    BETA[0]:=1/6; BETHA[0]:=.5;
    FAC:=1; "FOR" I:=2 "STEP" 1 "UNTIL" L-1 "DO" FAC:=I*FAC;
    M1:= "IF" AUT "THEN" M "ELSE" M-1;
    K:=0; LAST:="FALSE";
NEXT LEVEL:
    "FOR" I:=1 "STEP" 1 "UNTIL" M "DO" D[I]:=Y[I];
    DERIVATIVE(D);
    "IF" "NOT" LINEAR "OR" K=0 "THEN" JACOBIAN(J,Y);
    STEPSIZE;
    "IF" CHANGE "THEN" COEFFICIENT;
    OUTPUT;
    DIFFERENCE SCHEME;
    K:=K+1;
    X:=X+H;
    "IF" "NOT" LAST "THEN" "GOTO" NEXT LEVEL;
END OF EFERK: OUTPUT;
"END" EFERK;
        "EOP"
```

AUTHOR: K.DEKKER.

INSTITUTE: MATHEMATICAL CENTRE.

RECEIVED: 1973/09/01.

BRIEF DESCRIPTION:

LINIGER1VS SOLVES INITIAL VALUE PROBLEMS , GIVEN AS AN AUTONOMOUS SYSTEM OF FIRST ORDER DIFFERENTIAL EQUATIONS , BY MEANS OF AN IMPLICIT,FIRST ORDER ACCURATE, EXPONENTIALLY FITTED ONESTEP METHOD.
AUTOMATIC STEPSIZE CONTROL IS PROVIDED.
IN PARTICULAR THIS METHOD IS SUITABLE FOR THE INTEGRATION OF STIFF DIFFERENTIAL EQUATIONS.

KEYWORDS:

DIFFERENTIAL EQUATIONS,
INITIAL VALUE PROBLEMS,
STIFF EQUATIONS,
EXPONENTIAL FITTING,
IMPLICIT ONESTEP METHODS.

CALLING SEQUENCE:

THE HEADING OF THE PROCEDURE LINIGER1VS READS:

```
"PROCEDURE" LINIGER1VS (X,XE,M,Y,SIGMA,DERIVATIVE,J,JACOBIAN,
                        ITMAX,HMIN,HMAX,AETA,RETA,INFO,OUTPUT);
"VALUE" M;
"INTEGER" M,ITMAX;
"REAL" X,XE,SIGMA,HMIN,HMAX,AETA,RETA;
"ARRAY" Y,J,INFO;
"PROCEDURE" DERIVATIVE,JACOBIAN,OUTPUT;
```

THE MEANING OF THE FORMAL PARAMETERS IS:

```
X:      <VARIABLE>;
        THE INDEPENDENT VARIABLE X;
        ENTRY: THE INITIAL VALUE X0;
        EXIT : THE FINAL VALUE XE;
XE:     <ARITHMETIC EXPRESSION>;
        THE FINAL VALUE OF X (XE>=X);
M:      <ARITHMETIC EXPRESSION>;
        THE NUMBER OF EQUATIONS;
Y:      <ARRAY IDENTIFIER>;
        "ARRAY" Y[1:M];
        THE DEPENDENT VARIABLE;
        ENTRY: THE  INITIAL  VALUES  OF  THE SYSTEM OF DIFFERENTIAL
               EQUATIONS: Y[I] AT X=X0;
        EXIT : THE FINAL VALUES OF THE SOLUTION: Y[I] AT X=XE;
```

SIGMA: <ARITHMETIC EXPRESSION>;
THE MODULUS OF THE POINT AT WHICH EXPONENTIAL FITTING IS DESIRED, FOR EXAMPLE THE LARGEST NEGATIVE EIGENVALUE OF THE JACOBIAN OF THE SYSTEM OF DIFFERENTIAL EQUATIONS;

DERIVATIVE: <PROCEDURE IDENTIFIER>;
THE HEADING OF THIS PROCEDURE READS:
"PROCEDURE" DERIVATIVE(Y); "ARRAY" Y;
THIS PROCEDURE SHOULD DELIVER THE RIGHT HAND SIDE OF THE I-TH DIFFERENTIAL EQUATION AT THE POINT (Y) AS Y[I];

J: <ARRAY IDENTIFIER>;
"ARRAY" J[1:M,1:M];
THE JACOBIAN MATRIX OF THE SYSTEM;
THE ARRAY J SHOULD BE UPDATED IN THE PROCEDURE JACOBIAN;

JACOBIAN: <PROCEDURE IDENTIFIER>;
THE HEADING OF THIS PROCEDURE READS:
"PROCEDURE" JACOBIAN(J,Y); "ARRAY" J,Y;
IN THIS PROCEDURE (AN APPROXIMATION OF) THE JACOBIAN HAS TO BE ASSIGNED TO THE ARRAY J;

ITMAX: <ARITHMETIC EXPRESSION>;
AN UPPERBOUND FOR THE NUMBER OF ITERATIONS IN NEWTON'S PROCESS, USED TO SOLVE THE IMPLICIT EQUATIONS;

HMIN: <ARITHMETIC EXPRESSION>;
MINIMAL STEPSIZE BY WHICH THE INTEGRATION IS PERFORMED;

HMAX: <ARITHMETIC EXPRESSION>;
MAXIMAL STEPSIZE BY WHICH THE INTEGRATION IS PERFORMED;

AETA: <ARITHMETIC EXPRESSION>;
REQUIRED ABSOLUTE PRECISION IN THE INTEGRATION PROCESS;

RETA: <ARITHMETIC EXPRESSION>;
REQUIRED RELATIVE PRECISION IN THE INTEGRATION PROCESS;
IF BOTH AETA AND RETA HAVE NEGATIVE VALUES , INTEGRATION WILL BE PERFORMED WITH A STEPSIZE EQUAL TO HMAX , WHICH MAY BE VARIATED BY USER ; IN THIS CASE THE ABSOLUTE VALUES OF AETA AND RETA WILL CONTROL THE NEWTON ITERATION;

INFO: <ARRAY IDENTIFIER>;
"ARRAY" INFO[1:9];
DURING INTEGRATION AND UPON EXIT THIS ARRAY CONTAINS THE FOLLOWING INFORMATION;
INFO[1]: NUMBER OF INTEGRATION STEPS TAKEN;
INFO[2]: NUMBER OF DERIVATIVE EVALUATIONS USED;
INFO[3]: NUMBER OF JACOBIAN EVALUATIONS USED;
INFO[4]: NUMBER OF INTEGRATION STEPS EQUAL TO HMIN TAKEN ;
INFO[5]: NUMBER OF INTEGRATION STEPS EQUAL TO HMAX TAKEN ;
INFO[6]: MAXIMAL NUMBER OF ITERATIONS TAKEN IN THE NEWTON PROCESS;
INFO[7]: LOCAL ERROR TOLERANCE;
INFO[8]: ESTIMATED LOCAL ERROR;
INFO[9]: MAXIMUM VALUE OF THE ESTIMATED LOCAL ERROR;

OUTPUT: <PROCEDURE IDENTIFIER>;
THE HEADING OF THIS PROCEDURE READS,
"PROCEDURE" OUTPUT;
THIS PROCEDURE IS CALLED AT THE END OF EACH INTEGRATION STEP , THE USER CAN ASK FOR OUTPUT OF THE PARAMETERS , FOR EXAMPLE X, Y, INFO.

DATA AND RESULTS: SEE EXAMPLE OF USE, AND REF[2].

PROCEDURES USED:

```
INIVEC= CP31010,
MULVEC= CP31020,
MULROW= CP31021,
DUPVEC= CP31030,
MATVEC= CP34011,
ELMVEC= CP34020,
VECVEC= CP34010,
DEC   = CP34300,
SOL   = CP34051.
```

REQUIRED CENTRAL MEMORY:

EXECUTION FIELD LENGTH : CIRCA 20 + M * (5 + M).

RUNNING TIME: DEPENDS STRONGLY ON THE DIFFERENTIAL EQUATION TO SOLVE.

LANGUAGE: ALGOL 60.

METHOD AND PERFORMANCE:

LINIGER1VS: INTEGRATES THE SYSTEM OF DIFFERENTIAL EQUATIONS FROM X0 UNTIL XE, BY MEANS OF A FIRST ORDER FORMULA.

THE INTEGRATION METHOD IS BASED ON THE F(1) FORMULA DESCRIBED BY LINIGER AND WILLOUGHBY (SEE REF[1]). ERROR ESTIMATES AND A STEPSIZE STRATEGY FOR THIS METHOD ARE DESCRIBED IN [2] , AND A VARIABLE STEP METHOD IS CONSTRUCTED FOR THE CONVENIENCE OF THE USER, HOWEVER, THE STEPSIZE STRATEGY REQUIRES MANY EXTRA ARRAY OPERATIONS. THE USER MAY AVOID THIS EXTRA WORK BY GIVING AETA AND RETA A NEGATIVE VALUE AND PRESCRIBING A STEPSIZE (HMAX) HIMSELF.

REFERENCES:

[1]. W.LINIGER AND R.A.WILLOUGHBY.
EFFICIENT INTEGRATION METHODS FOR STIFF SYSTEMS OF ORDINARY DIFFERENTIAL EQUATIONS.
SIAM J. NUM. ANAL. 7 (1970) PAGE 47.

[2]. K.DEKKER.
ERROR ESTIMATES AND STEPSIZE STRATEGIES FOR THE LINIGER-WILLOUGHBY FORMULAE.
(TO APPEAR IN 1974).

EXAMPLE OF USE:

CONSIDER THE SYSTEM OF DIFFERENTIAL EQUATIONS:
DY[1]/DX = -Y[1] + Y[1] * Y[2] + .99 * Y[2]
DY[2]/DX = -1000 * (-Y[1] + Y[1] * Y[2] + Y[2])
WITH THE INITIAL CONDITIONS AT X = 0:
Y[1] = 1 AND Y[2] = 0.
THE SOLUTION AT X = 50 IS APPROXIMATELY:
Y[1] = .765 878 320 2487 AND Y[2] = .433 710 353 5768.
THE FOLLOWING PROGRAM SHOWS INTEGRATION OF THIS PROBLEM WITH VARIABLE AND CONSTANT STEPSIZES:

```
"BEGIN" "COMMENT" TEST LINIGER1VS;
    "PROCEDURE" LINIGER1VS(X,XE,M,Y,SIGMA,F,J,JACOBIAN,
        ITMAX,HMIN,HMAX,AETA,RETA,INFO,OUTPUT);
    "CODE" 33132;
    "INTEGER" ITMAX;
    "REAL" X,SIGMA,RETA,TIME;
    "REAL" "ARRAY" Y[1:2],J[1:2,1:2],INFO[1:9];
    "PROCEDURE" F(A); "ARRAY" A;
    "BEGIN" "REAL" A1,A2; A1:=A[1]; A2:=A[2];
        A[1]:=(A1+.99)*(A2-1)+.99;
        A[2]:=1000*((1+A1)*(1-A2)-1);
    "END";
    "PROCEDURE" JACOBIAN(J,Y); "ARRAY" J,Y;
    "BEGIN" J[1,1]:=Y[2]-1; J[1,2]:=.99+Y[1];
        J[2,1]:=1000*(1-Y[2]); J[2,2]:=-1000*(1+Y[1]);
        SIGMA:=ABS(J[2,2]+J[1,1]-SQRT((J[2,2]-J[1,1])**2+
            4*J[2,1]*J[1,2]))/2;
    "END" JACOBIAN;
    "PROCEDURE" OUT;
    "IF" X=50 "THEN"
    OUTPUT(61,"("6(3ZDB),2BD"-ZD,2(2B+.3DB3D),-3ZD,3D,/")",
    INFO[1],INFO[2],INFO[3],INFO[4],INFO[5],INFO[6],INFO[9],Y[1],
    Y[2],CLOCK-TIME);
    "FOR" RETA:="-2,"-4,"-6 "DO"
    "BEGIN" X:=Y[2]:=0; Y[1]:=1; TIME:=CLOCK;
        LINIGER1VS(X,50,2,Y,SIGMA,F,J,JACOBIAN,10,.1,50,RETA,
            RETA,INFO,OUT);
    "END"; OUTPUT(61,"("//")");
    "FOR" RETA:=-"-2,-"-4,-"-6 "DO"
        "BEGIN" X:=Y[2]:=0; Y[1]:=1; TIME:=CLOCK;
            LINIGER1VS(X,50,2,Y,SIGMA,F,J,JACOBIAN,10,.1,1,RETA,
                RETA,INFO,OUT);
        "END";
"END"
```

17	21	8	2	0	2	2" -2	+.772 017	+.435 672	0.525
13	25	23	2	0	2	2" -2	+.767 414	+.434 202	0.717
105	210	105	2	0	2	2" -2	+.766 027	+.433 758	4.741
50	52	1	0	50	2	0" 0	+.766 670	+.433 081	0.549
50	104	3	0	50	3	0" 0	+.766 183	+.433 811	1.158
50	152	12	0	50	4	0" 0	+.766 185	+.433 809	1.653

SOURCE TEXT(S):

```
"CODE" 33132;
"PROCEDURE" LINIGER1VS(X,XE,M,Y,SIGMA,DERIVATIVE,J,
   JACOBIAN,ITMAX,HMIN,HMAX,AETA,RETA,INFO,OUTPUT);
"VALUE" M;
"INTEGER" M,ITMAX;
"REAL" X,XE,SIGMA,AETA,RETA,HMIN,HMAX;
"ARRAY" Y,J,INFO;
"PROCEDURE" DERIVATIVE,JACOBIAN,OUTPUT;

"BEGIN" "INTEGER" I,ST,LASTJAC;
     "REAL" H,HNEW,MU,MU1,BETA,P,E,E1,ETA,ETA1,DISCR;
     "BOOLEAN" LAST,FIRST,EVALJAC,EVALCOEF;
     "INTEGER" "ARRAY" PI[1:M];
     "REAL" "ARRAY" DY,YL,YR,F[1:M],A[1:M,1:M],AUX[1:3];

     "PROCEDURE" INIVEC(L,U,A,X);                    "CODE" 31010;
     "PROCEDURE" MULVEC(AL,U,SHIFT,A,B,X);           "CODE" 31020;
     "PROCEDURE" MULROW(L,U,I,J,A,B,X);              "CODE" 31021;
     "PROCEDURE" DUPVEC(L,U,SHIFT,A,B);              "CODE" 31030;
     "REAL" "PROCEDURE" VECVEC(L,U,SHIFT,A,B);       "CODE" 34010;
     "REAL" "PROCEDURE" MATVEC(A,B,C,D,E);           "CODE" 34011;
     "PROCEDURE" ELMVEC(L,U,SHIFT,A,B,X);            "CODE" 34020;
     "PROCEDURE" DEC(A,N,AUX,P);                     "CODE" 34300;
     "PROCEDURE" SOL(A,N,P,B);                       "CODE" 34051;

     "REAL" "PROCEDURE" NORM(A); "ARRAY" A;
     NORM:=SQRT(VECVEC(1,M,0,A,A));

     "PROCEDURE" COEFFICIENT;
     "BEGIN" "REAL" B,E; B:=ABS(H*SIGMA);
          "IF" B>40 "THEN"
          "BEGIN" MU:=1/B; BETA:=1; P:=2+2/(B-2)
          "END" "ELSE"
          "IF" B<.04 "THEN"
          "BEGIN" E:=B*B/30; P:=3-E;
               MU:=.5-B/12*(1-E/2);
               BETA:=.5+B/6*(1-E)
          "END" "ELSE"
          "BEGIN" E:=EXP(B)-1;
               MU:=1/B-1/E;
               BETA:=(1-B/E)*(1+1/E);
               P:=(BETA-MU)/(.5-MU)
          "END";
          MU1:=H*(1-MU);
          LUDECOMP
     "END" COEFFICIENT;                                   "COMMENT"
```

```
"PROCEDURE" LUDECOMP;
"BEGIN" "INTEGER" I;
    "FOR" I:=1 "STEP" 1 "UNTIL" M "DO"
    "BEGIN" MULROW(1,M,I,I,A,J,-MU1);
        A[I,I]:=A[I,I]+1
    "END";
    AUX[2]:=0; DEC(A,M,AUX,PI)
"END" LUDECOMP;

"PROCEDURE" STEPSIZE;
"IF" ETA<0 "THEN"
"BEGIN" "REAL" HL; HL:=H;
    H:=HNEW:=HMAX; INFO[5]:=INFO[5]+1;
    "IF" 1.1*HNEW>XE-X "THEN"
    "BEGIN" LAST:="TRUE"; H:=HNEW:=XE-X
    "END";
    EVALCOEF:=H^=HL;
"END" "ELSE"
"IF" FIRST "THEN"
"BEGIN" H:=HNEW:=HMIN; FIRST:="FALSE"; INFO[4]:=INFO[4]+1
"END" "ELSE"
"BEGIN" "REAL" B,HL;
    B:=DISCR/ETA; HL:=H; "IF" B<.01 "THEN" B:=.01;
    HNEW:= "IF" B>0 "THEN" H*B**(-1/P) "ELSE" HMAX;
    "IF" HNEW<HMIN "THEN"
    "BEGIN" HNEW:=HMIN; INFO[4]:=INFO[4]+1
    "END" "ELSE"
    "IF" HNEW>HMAX "THEN"
    "BEGIN" HNEW:=HMAX; INFO[5]:=INFO[5]+1 "END";
    "IF" 1.1*HNEW>=XE-X "THEN"
    "BEGIN" LAST:="TRUE"; H:=HNEW:=XE-X
    "END" "ELSE"
    "IF" ABS(H/HNEW-1)>.1 "THEN" H:=HNEW;
    EVALCOEF:=H^=HL
"END" STEPSIZE;

"PROCEDURE" LINEARITY(ERROR); "REAL" ERROR;
"BEGIN" "INTEGER" K;
    "FOR" K:=1 "STEP" 1 "UNTIL" M "DO"
    DY[K]:=Y[K]-MU1*F[K];
    SOL(A,M,PI,DY);
    ELMVEC(1,M,0,DY,Y,-1);
    ERROR:=NORM(DY)
"END" LINEARITY;                                                    "COMMENT"
```

```
"PROCEDURE" ITERATION(I); "INTEGER" I;
"IF" RETA<0 "THEN"
"BEGIN" "INTEGER" K;
    "IF" I=1 "THEN"
    "BEGIN" MULVEC(1,M,0,DY,F,H);
        "FOR" K:=1 "STEP" 1 "UNTIL" M "DO" YL[K]:=Y[K]+MU*DY[K];
        SOL(A,M,PI,DY); E:=1;
    "END" "ELSE"
    "BEGIN" "FOR" K:=1 "STEP" 1 "UNTIL" M "DO"
        DY[K]:=YL[K]-Y[K]+MU1*F[K];
        "IF" E*NORM(Y)>E1*E1 "THEN"
        "BEGIN" EVALJAC:=I>=3;
            "IF" I>3 "THEN"
            "BEGIN" INFO[3]:=INFO[3]+1; JACOBIAN(J,Y);
                LUDECOMP
            "END";
        "END";
        SOL(A,M,PI,DY)
    "END";
    E1:=E; E:=NORM(DY);
    ELMVEC(1,M,0,Y,DY,1);
    ETA:=NORM(Y)*RETA+AETA;
    DISCR:=0;
    DUPVEC(1,M,0,F,Y);
    DERIVATIVE(F);
    INFO[2]:=INFO[2]+1;
"END" "ELSE"
"BEGIN" "INTEGER" K;
    "IF" I=1 "THEN"
    "BEGIN" LINEARITY(E);
        "IF" E*(ST-LASTJAC)>ETA "THEN"
        "BEGIN" JACOBIAN(J,Y); LASTJAC:=ST;
            INFO[3]:=INFO[3]+1;
            H:=HNEW; COEFFICIENT;
            LINEARITY(E)
        "END";
        EVALJAC:= E*(ST+1-LASTJAC)>ETA;
MULVEC(1,M,0,DY,F,H);
"FOR" K:=1 "STEP" 1 "UNTIL" M "DO" YL[K]:=Y[K]+MU*DY[K];
SOL(A,M,PI,DY);
"FOR" K:=1 "STEP" 1 "UNTIL" M "DO"
YR[K]:=H*BETA*MATVEC(1,M,K,J,DY);
SOL(A,M,PI,YR);
ELMVEC(1,M,0,YR,DY,1);                                  "COMMENT"
```

```
        "END" "ELSE"
        "BEGIN" "FOR" K:=1 "STEP" 1 "UNTIL" M "DO"
            DY[K]:=YL[K]-Y[K]+MU1*F[K];
            "IF" E>ETA1 "AND" DISCR>ETA1 "THEN"
            "BEGIN" INFO[3]:=INFO[3]+1; JACOBIAN(J,Y);
                LUDECOMP
            "END";
            SOL(A,M,PI,DY);
            E:=NORM(DY)
        "END";
        ELMVEC(1,M,0,Y,DY,1);
        ETA:=NORM(Y)*RETA+AETA;
        ETA1:=ETA/SQRT(RETA);
        DUPVEC(1,M,0,F,Y);
        DERIVATIVE(F);
        INFO[2]:=INFO[2]+1;
        "FOR" K:=1 "STEP" 1 "UNTIL" M "DO" DY[K]:=YR[K]-H*F[K];
        DISCR:=NORM(DY)/2
    "END" ITERATION;

    FIRST:=EVALJAC:="TRUE"; LAST:=EVALCOEF:="FALSE";
    INIVEC(1,9,INFO,0);
    ETA:=RETA*NORM(Y)+AETA;
    ETA1:=ETA/SQRT(ABS(RETA));
    DUPVEC(1,M,0,F,Y);
    DERIVATIVE(F);
    INFO[2]:=1;
    "FOR" ST:=1,ST+1 "WHILE" ¬LAST "DO"
    "BEGIN" STEPSIZE; INFO[1]:=INFO[1]+1;
        "IF" EVALJAC "THEN"
        "BEGIN" JACOBIAN(J,Y);
            INFO[3]:=INFO[3]+1;
            H:=HNEW;
            COEFFICIENT;
            EVALJAC:="FALSE"; LASTJAC:=ST
        "END" "ELSE"
        "IF" EVALCOEF "THEN" COEFFICIENT;
        "FOR" I:=1,I+1 "WHILE" E>ABS(ETA) "AND" DISCR>1.3*ETA
            "AND" I<=ITMAX "DO"
            "BEGIN" ITERATION(I); "IF" I>INFO[6] "THEN" INFO[6]:=I;
            "END"; INFO[7]:=ETA; INFO[8]:=DISCR;
            X:=X+H;
            "IF" DISCR>INFO[9] "THEN" INFO[9]:=DISCR;
            OUTPUT;
        "END";
"END" LINIGER1VS;
            "EOP"
```

AUTHOR: K.DEKKER.

INSTITUTE: MATHEMATICAL CENTRE.

RECEIVED: 1973/07/16.

BRIEF DESCRIPTION:

LINIGER2 SOLVES INITIAL VALUE PROBLEMS , GIVEN AS AN AUTONOMOUS SYSTEM OF FIRST ORDER DIFFERENTIAL EQUATIONS , BY MEANS OF AN EXPONENTIALLY FITTED ONESTEP METHOD.
NO AUTOMATIC STEPSIZE CONTROL IS PROVIDED.
IN PARTICULAR THIS METHOD IS SUITABLE FOR THE INTEGRATION OF STIFF DIFFERENTIAL EQUATIONS.

KEYWORDS:
DIFFERENTIAL EQUATIONS,
INITIAL VALUE PROBLEMS,
STIFF EQUATIONS,
EXPONENTIAL FITTING,
IMPLICIT ONESTEP METHODS.

CALLING SEQUENCES:

THE HEADING OF THE PROCEDURE LINIGER2 READS:

```
"PROCEDURE" LINIGER2(X,XE,M,Y,SIGMA1,SIGMA2,F,EVALUATE,J,
                JACOBIAN,K,ITMAX,STEP,AETA,RETA,OUTPUT);
"INTEGER" M,K,ITMAX;
"REAL" X,XE,SIGMA1,SIGMA2,STEP,AETA,RETA;
"ARRAY" Y,J;
"BOOLEAN" "PROCEDURE" EVALUATE;
"REAL" "PROCEDURE" F;
"PROCEDURE" JACOBIAN,OUTPUT;
```

THE MEANING OF THE FORMAL PARAMETERS IS:

X: <VARIABLE>;
THE INDEPENDENT VARIABLE X;
CAN BE USED IN F, JACOBIAN, OUTPUT, ETC.;
ENTRY: THE INITIAL VALUE X0;
EXIT : THE FINAL VALUE XE;

XE: <ARITHMETIC EXPRESSION>;
THE FINAL VALUE OF X (XE>=X);

M: <ARITHMETIC EXPRESSION>;
THE NUMBER OF EQUATIONS;

Y: <ARRAY IDENTIFIER>;
"ARRAY" Y[1:M];
THE DEPENDENT VARIABLE;
ENTRY: THE INITIAL VALUES OF THE SYSTEM OF DIFFERENTIAL EQUATIONS: Y[I] AT X=X0;
EXIT : THE FINAL VALUES OF THE SOLUTION: Y[I] AT X=XE;

SIGMA1: <ARITHMETIC EXPRESSION>;
THE MODULUS OF THE POINT AT WHICH EXPONENTIAL FITTING IS DESIRED; THIS POINT MAY BE COMPLEX OR REAL AND NEGATIVE;

SIGMA2: <ARITHMETIC EXPRESSION>;
SIGMA2 MAY DEFINE THREE DIFFERENT TYPES OF EXPONENTIAL FITTING: FITTING IN TWO COMPLEX CONJUGATED POINTS , FITTING IN TWO REAL NEGATIVE POINTS , OR FITTING IN ONE POINT COMBINED WITH THIRD ORDER ACCURACY;
IF THIRD ORDER ACCURACY IS DESIRED , SIGMA2 SHOULD HAVE THE VALUE 0;
IF FITTING IN A SECOND NEGATIVE POINT IS DESIRED , SIGMA2 SHOULD HAVE THE VALUE OF THE MODULUS OF THIS POINT;
IF FITTING IN TWO COMPLEX CONJUGATED POINTS IS DESIRED , THEN SIGMA2 SHOULD BE MINUS THE VALUE OF THE ARGUMENT OF THE POINT IN THE SECOND QUADRANT (THUS A VALUE BETWEEN - PI AND - PI/2);

F: <PROCEDURE IDENTIFIER>;
THE HEADING OF THIS PROCEDURE READS:
"REAL" "PROCEDURE" F(I); "INTEGER" I;
THIS PROCEDURE SHOULD DELIVER THE RIGHT HAND SIDE OF THE I-TH DIFFERENTIAL EQUATION AS F;

EVALUATE: <PROCEDURE IDENTIFIER>;
THE HEADING OF THIS PROCEDURE READS;
"BOOLEAN" "PROCEDURE" EVALUATE(ITNUM); "INTEGER" ITNUM;
EVALUATE SHOULD HAVE THE VALUE "TRUE", IF IT IS DESIRED
THAT THE JACOBIAN OF THE SYSTEM IS UPDATED IN THE ITNUM-TH
ITERATION STEP OF THE NEWTON PROCESS;

J: <ARRAY IDENTIFIER>;
"ARRAY" J[1:M,1:M];
THE JACOBIAN MATRIX OF THE SYSTEM;
THE ARRAY J SHOULD BE UPDATED IN THE PROCEDURE JACOBIAN;

JACOBIAN: <PROCEDURE IDENTIFIER>;
THE HEADING OF THIS PROCEDURE READS:
"PROCEDURE" JACOBIAN(J,Y); "ARRAY" J,Y;
IN THIS PROCEDURE THE JACOBIAN HAS TO BE ASSIGNED TO THE
THE ARRAY J , OR AN APPROXIMATION OF THE JACOBIAN , IF ONLY
SECOND ORDER ACCURACY IS REQUIRED;

K: <VARIABLE>;
COUNTS THE NUMBER OF INTEGRATION STEPS TAKEN;
FOR EXAMPLE, CAN BE USED IN EVALUATE;

ITMAX: <ARITHMETIC EXPRESSION>;
AN UPPERBOUND FOR THE NUMBER OF ITERATIONS IN NEWTON'S
PROCESS, USED TO SOLVE THE IMPLICIT EQUATIONS;

STEP: <ARITHMETIC EXPRESSION>;
THE LENGTH OF THE INTEGRATION STEP, TO BE PRESCRIBED BY THE
THE USER;

AETA: <ARITHMETIC EXPRESSION>;
REQUIRED ABSOLUTE PRECISION IN THE NEWTON PROCESS, USED TO
SOLVE THE IMPLICIT EQUATIONS;

RETA: <ARITHMETIC EXPRESSION>;
REQUIRED RELATIVE PRECISION IN THE NEWTON PROCESS, USED TO
SOLVE THE IMPLICIT EQUATIONS;

OUTPUT: <PROCEDURE IDENTIFIER>;
THE HEADING OF THIS PROCEDURE READS;
"PROCEDURE" OUTPUT;
. THIS PROCEDURE IS CALLED AT THE END OF EACH INTEGRATION
STEP ; THE USER CAN ASK FOR OUTPUT OF THE PARAMETERS , FOR
EXAMPLE X, K, Y.

DATA AND RESULTS: SEE EXAMPLE OF USE, AND REF[3].

PROCEDURES USED:

VECvEC= CP34010,
MATVEC= CP34011,
MATMAT= CP34013,
DEC = CP34300,
SOL = CP34051.

REQUIRED CENTRAL MEMORY:

EXECUTION FIELD LENGTH: CIRCA 20 + M * (4+M).

RUNNING TIME: DEPENDS STRONGLY ON THE DIFFERENTIAL EQUATION TO SOLVE.

LANGUAGE: ALGOL 60.

METHOD AND PERFORMANCE:

LINIGER2: INTEGRATES THE SYSTEM OF DIFFERENTIAL EQUATIONS FROM X0 UNTIL XE, BY MEANS OF A SECOND ORDER FORMULA (IF SIGMA2=0 AND EVALUATE="TRUE" EVEN THIRD ORDER).
SEE ALSO REF[1] AND REF[2].

REFERENCES:

[1]. W.LINIGER AND R.A.WILLOUGHBY.
EFFICIENT INTEGRATION METHODS FOR STIFF SYSTEMS OF ORDINARY DIFFERENTIAL EQUATIONS.
SIAM J. NUM. ANAL. 7 (1970) PAGE 47.

[2]. T.J.DEKKER, P.W.HEMKER AND P.W.VAN DER HOUWEN.
COLLOQUIUM STIFF DIFFERENTIAL EQUATIONS 1 (DUTCH).
MC SYLLABUS 15.1, (1972) MATHEMATICAL CENTRE.

[3]. P.A.BEENTJES, K.DEKKER, H.C.HEMKER, S.P.N.VAN KAMPEN AND G.M.WILLEMS.
COLLOQUIUM STIFF DIFFERENTIAL EQUATIONS 2 (DUTCH).
MC SYLLABUS 15.2, (1973) MATHEMATICAL CENTRE.

EXAMPLE OF USE:

CONSIDER THE SYSTEM OF DIFFERENTIAL EQUATIONS:
DY[1]/DX = -Y[1] + Y[1] * Y[2] + .99 * Y[2]
DY[2]/DX = -1000 * (-Y[1] + Y[1] * Y[2] + Y[2])
WITH THE INITIAL CONDITIONS AT X = 0:
Y[1] = 1 AND Y[2] = 0. (SEE REF[2], PAGE 11).
THE SOLUTION AT X = 50 IS APPROXIMATELY:
Y[1] = .765 878 320 2487 AND Y[2] = .433 710 353 5768.
THE FOLLOWING PROGRAM SHOWS SOME DIFFERENT CALLS OF THE PROCEDURE LINIGER2, AND ILLUSTRATES THE ACCURACY WHICH MAY BE OBTAINED BY IT:

```
"BEGIN"
    "PROCEDURE" LINIGER2(X,XE,M,Y,SIGMA1,SIGMA2,F,EVALUATE,J,
                         JACOBIAN,K,ITMAX,STEP,AETA,RETA,OUTPUT);
    "CODE" 33131;

    "INTEGER" K,ITMAX,PASSES,PASJAC;
    "REAL" X,SIGMA,STEP,TIME;
    "REAL" "ARRAY" Y[1:2],J[1:2,1:2];

    "REAL" "PROCEDURE" F(I); "INTEGER" I;
    "IF" I=1 "THEN" F:=(Y[1]+.99)*(Y[2]-1)+.99 "ELSE"
    "BEGIN" PASSES:=PASSES+1; F:=1000*((1+Y[1])*(1-Y[2])-1) "END";
```

```
"PROCEDURE" JACOBIAN(J,Y); "ARRAY" J,Y;
"BEGIN" J[1,1]:=Y[2]-1; J[1,2]:=.99+Y[1];
    J[2,1]:=1000*(1-Y[2]); J[2,2]:=-1000*(1+Y[1]);
    SIGMA:=ABS(J[2,2]+J[1,1]-SQRT((J[2,2]-J[1,1])**2+
           4*J[2,1]*J[1,2]))/2;
    PASJAC:=PASJAC+1
"END" JACOBIAN;
"BOOLEAN" "PROCEDURE" EVALUATE1(I); "INTEGER" I;
EVALUATE1:= I=1;
"BOOLEAN" "PROCEDURE" EVALUATE2(I); "INTEGER" I;
EVALUATE2:= "TRUE";
"PROCEDURE" OUT;
"IF" X=50 "THEN"
OUTPUT(61,"("3(-4ZDB),2(4B+.3DB3DB3D),-5ZD.3D,/")",K,PASSES,
PASJAC,Y[1],Y[2],CLOCK-TIME);
OUTPUT(61,"(""("    THIS LINE AND THE FOLLOWING TEXT IS ")"
"("PRINTED BY THIS PROGRAM")",//,
"("    THE RESULTS WITH LINIGER2 -SECOND ORDER- ARE:")",/,
"("     K   DER.EV. JAC.EV.       Y[1]                Y[2]")"
"("         TIME")",/")");
"FOR" STEP:=10,1 "DO"
"FOR" ITMAX:=1,3 "DO"
"BEGIN" PASSES:=PASJAC:=0; X:=Y[2]:=0; Y[1]:=1; TIME:=CLOCK;
    LINIGER2(X,50,2,Y,SIGMA,0,F,EVALUATE1,J,JACOBIAN,K,ITMAX,
                                                STEP,"-4,"-4,OUT);
    "END";
    OUTPUT(61,"("//,
    "("    THE RESULTS WITH LINIGER2 -THIRD ORDER- ARE:")",/,
    "("     K   DER.EV. JAC.EV.       Y[1]                Y[2]")"
    "("         TIME")",/")");
    "FOR" STEP:=10,1 "DO"
    "FOR" ITMAX:=1,3 "DO"
    "BEGIN" PASSES:=PASJAC:=0; X:=Y[2]:=0; Y[1]:=1; TIME:=CLOCK;
        LINIGER2(X,50,2,Y,SIGMA,0,F,EVALUATE2,J,JACOBIAN,K,ITMAX,
                                                STEP,"-4,"-4,OUT);
    "END";
"END";
```

THIS LINE AND THE FOLLOWING TEXT IS PRINTED BY THIS PROGRAM:

THE RESULTS WITH LINIGER2 -SECOND ORDER- ARE:

K	DER.EV.	JAC.EV.	Y[1]	Y[2]	TIME
5	5	5	+.766 392 210	+.434 218 863	0.092
5	15	5	+.765 755 853	+.433 671 223	0.175
50	50	50	+.765 884 310	+.433 715 687	0.949
50	101	50	+.765 877 388	+.433 710 059	1.494

THE RESULTS WITH LINIGER2 -THIRD ORDER- ARE:

K	DER.EV.	JAC.EV.	Y[1]	Y[2]	TIME
5	5	5	+.766 392 210	+.434 218 863	0.092
5	15	15	+.765 882 250	+.433 711 614	0.300
50	50	50	+.765 884 310	+.433 715 687	0.949
50	101	101	+.765 878 873	+.433 710 531	2.080

SOURCE TEXT(S):

```
"CODE"" 33131;
"PROCEDURE" LINIGER2(X,XE,M,Y,SIGMA1,SIGMA2,F,EVALUATE,J,
                     JACOBIAN,K,ITMAX,STEP,AETA,RETA,OUTPUT);
"INTEGER" M,K,ITMAX;
"REAL" X,XE,SIGMA1,SIGMA2,STEP,AETA,RETA;
"ARRAY" Y,J;
"BOOLEAN" "PROCEDURE" EVALUATE;
"REAL" "PROCEDURE" F;
"PROCEDURE" JACOBIAN,OUTPUT;

"BEGIN" "INTEGER" I;
     "REAL" H,HL,B1,B2,P,Q,C0,C1,C2,C3,C4;
     "BOOLEAN" LAST;
     "INTEGER" "ARRAY" PI[1:M];
     "REAL" "ARRAY" DY,YL,FL[1:M],A[1:M,1:M],AUX[1:3];
     "REAL" "PROCEDURE" VECVEC(L,U,SHIFT,A,B); "CODE" 34010;
     "REAL" "PROCEDURE" MATVEC(L,U,I,A,B); "CODE" 34011;
     "REAL" "PROCEDURE" MATMAT(L,U,I,J,A,B); "CODE" 34013;
     "PROCEDURE" DEC(A,N,AUX,P); "CODE" 34300;
"PROCEDURE" SOL(A,N,P,B); "CODE" 34051;

"PROCEDURE" STEPSIZE;
"BEGIN" H:=STEP;
     "IF" 1.1*H>=XE-X "THEN"
     "BEGIN" LAST:="TRUE"; H:=XE-X; X:=XE
     "END" "ELSE" X:=X+H
"END" STEPSIZE;

"PROCEDURE" COEFFICIENT;
"BEGIN" "REAL" R1,R2,EX,ZETA,ETA,SINL,COSL,SINH,COSH,D;
     "REAL" "PROCEDURE" R(X); "VALUE" X; "REAL" X;
   . "IF" X>40 "THEN" R:=X/(X-2) "ELSE"
     "BEGIN" EX:=EXP(-X); R:=X*(1-EX)/(X-2+(X+2)*EX) "END";

     B1:=H*SIGMA1;
     B2:=H*SIGMA2;
     "IF" B1<.1 "THEN" "BEGIN" P:=0; Q:=1/3; "GOTO" OUT "END";
     "IF" B2<0 "THEN" "GOTO" COMPLEX;
     "IF" B1<1 "OR" B2<.1 "THEN" "GOTO" THIRDORDER;
     "IF" ABS(B1-B2)<B1*B1*"-6 "THEN" "GOTO" DOUBLEFIT;

     R1:=R(B1)*B1; R2:=R(B2)*B2;
     D:=B2*R1-B1*R2;
     P:=2*(R2-R1)/D;
     Q:=2*(B2-B1)/D;
     "GOTO" OUT;                                        "COMMENT'
```

```
                                                                    ;


THIRDORDER: Q:=1/3;
            P:=R(B1)/3-2/B1;
            "GOTO" OUT;
DOUBLEFIT:  B1:=.5*(B1+B2);
            R1:=R(B1);
            "IF" B1>40 "THEN" EX:=0;
            R2:=B1/(1-EX); R2:=1-EX*R2*R2;
            Q:=1/(R1*R1*R2);
            P:=R1*Q-2/B1;
            "GOTO" OUT;
COMPLEX:    ETA:=ABS(B1*SIN(SIGMA2));
            ZETA:=ABS(B1*COS(SIGMA2));
            "IF" ETA<B1*B1*"-6 "THEN"
            "BEGIN" B1:=B2:=ZETA; "GOTO" DOUBLEFIT "END";
            "IF" ZETA>40 "THEN"
            "BEGIN" P:=1-4*ZETA/B1/B1; Q:=4*(1-ZETA)/B1/B1+1 "END" "ELSE"
            "BEGIN" EX:=EXP(ZETA);
                SINL:=SIN(ETA); COSL:=COS(ETA);
                SINH:=.5*(EX-1/EX); COSH:=.5*(EX+1/EX);
                D:=ETA*(COSH-COSL)-.5*B1*B1*SINL;
                P:=(ZETA*SINL+ETA*SINH-4*ZETA*ETA/B1/B1*(COSH-COSL))/D;
                Q:=ETA*((COSH-COSL-ZETA*SINH-ETA*SINL)*4/B1/B1+COSH+COSL)/D
            "END";
OUT:        C0:=.25*H*H*(P+Q);
            C1:=.5*H*(1+P);
            C2:=H-C1;
            C3:=.25*H*H*(Q-P);
            C4:=.5*H*P;
            ELEMENTS OF MATRIX
        "END" COEFFICIENT;

        "PROCEDURE" ELEMENTS OF MATRIX;
        "BEGIN" "INTEGER" K;
            "FOR" I:=1 "STEP" 1 "UNTIL" M "DO"
          . "BEGIN" "FOR" K:=1 "STEP" 1 "UNTIL" M "DO"
                A[I,K]:=C0*MATMAT(1,M,I,K,J,J)-C1*J[I,K];
                A[I,I]:=A[I,I]+1
            "END";
            AUX[2]:=0; DEC(A,M,AUX,PI)
        "END" ELOFMAT;
```

```
;

     "PROCEDURE" NEWTON ITERATION;
     "BEGIN" "INTEGER" ITNUM; "REAL" JFL,ETA,DISCR;
          ITNUM:=0;
NEXT:     ITNUM:=ITNUM+1;
"IF" EVALUATE(ITNUM) "THEN"
"BEGIN" JACOBIAN(J,Y); COEFFICIENT "END"
"ELSE" "IF" ITNUM=1 "AND" H=HL "THEN" COEFFICIENT;
"FOR" I:=1 "STEP" 1 "UNTIL" M "DO" FL[I]:=F(I);
"IF" ITNUM=1 "THEN"
"BEGIN" "FOR" I:=1 "STEP" 1 "UNTIL" M "DO"
     "BEGIN" JFL:=MATVEC(1,M,I,J,FL);
          DY[I]:=H*(FL[I]-C4*JFL);
          YL[I]:=Y[I]+C2*FL[I]+C3*JFL
     "END"
"END" "ELSE"
"FOR" I:=1 "STEP" 1 "UNTIL" M "DO"
DY[I]:=YL[I]-Y[I]+C1*FL[I]-C0*MATVEC(1,M,I,J,FL);
SOL(A,M,PI,DY);
"FOR" I:=1 "STEP" 1 "UNTIL" M "DO" Y[I]:=Y[I]+DY[I];
"IF" ITNUM<ITMAX "THEN"
"BEGIN" ETA:=SQRT(VECVEC(1,M,0,Y,Y))*RETA+AETA;
          DISCR:=SQRT(VECVEC(1,M,0,DY,DY));
          "IF" ETA<DISCR "THEN" "GOTO" NEXT
     "END"
"END" NEWTON;

     LAST:="FALSE"; K:=0; HL:=0;
NEXT LEVEL:
     K:=K+1;
     STEPSIZE;
     NEWTON ITERATION;
     HL:=H;
     OUTPUT;
     "IF" "NOT" LAST "THEN" "GOTO" NEXT LEVEL
"END" 'LINIGER2;
          "EOP"
```

2-nd REVISION, 1977

MC

AUTHOR: J.G. VERWER.

INSTITUTE: MATHEMATICAL CENTRE.

RECEIVED: 740809.

BRIEF DESCRIPTION:

GMS SOLVES AN INITIAL VALUE PROBLEM, GIVEN AS AN AUTONOMOUS SYSTEM OF FIRST ORDER DIFFERENTIAL EQUATIONS DY/DX = F(Y), BY MEANS OF A THIRD ORDER GENERALIZED LINEAR MULTISTEP METHOD; IN PARTICULAR THIS PROCEDURE IS SUITABLE FOR THE INTEGRATION OF STIFF EQUATIONS.

KEYWORDS:

DIFFERENTIAL EQUATIONS,
INITIAL VALUE PROBLEM,
AUTONOMOUS SYSTEM,
STIFF EQUATIONS,
GENERALIZED LINEAR MULTISTEP METHOD.

CALLING SEQUENCE:

THE HEADING OF THE PROCEDURE READS:

```
"PROCEDURE" GMS(X, XE, R, Y, H, HMIN, HMAX, DELTA, DERIVATIVE,
                JACOBIAN, AETA, RETA, N, JEV, LU, NSJEV,
                LINEAR, OUT);
"VALUE" R;
"REAL" X, XE, H, HMIN, HMAX, DELTA, AETA, RETA;
"INTEGER" R, N, JEV, NSJEV, LU;
"BOOLEAN" LINEAR;
"ARRAY" Y;
"PROCEDURE" DERIVATIVE, JACOBIAN, OUT;
```

GMS INTEGRATES THE SYSTEM OF DIFFERENTIAL EQUATIONS DY/DX = F(Y) FROM X = X0 TO X = XE;

THE MEANING OF THE FORMAL PARAMETERS IS:

X: <VARIABLE>;
THE INDEPENDENT VARIABLE X;
ENTRY: THE INITIAL VALUE OF X;
EXIT : THE END VALUE OF X;

XE: <ARITHMETIC EXPRESSION>;
ENTRY: THE END VALUE OF X;

R: <ARITHMETIC EXPRESSION>;
ENTRY: THE NUMBER OF DIFFERENTIAL EQUATIONS;

```
Y:          «ARRAY IDENTIFIER»;
            "ARRAY" Y[1:R];
            THE DEPENDENT VARIABLE;
            ENTRY: THE INITIAL VALUE OF Y;
            EXIT : THE SOLUTION Y AT THE POINT X AFTER EACH
                   INTEGRATION STEP;
H:          «ARITHMETIC EXPRESSION»;
            ENTRY: THE STEPLENGTH WHEN THE INTEGRATION HAS TO BE
                   PERFORMED WITHOUT THE STEPSIZE MECHANISM, OTHER-
                   WISE THE INITIAL STEPLENGTH (SEE THE PARAMETERS
                   HMIN AND HMAX);
HMIN, HMAX: «ARITHMETIC EXPRESSION»;
            ENTRY: MINIMAL AND MAXIMAL STEPLENGTH BY WHICH THE INTE-
                   GRATION IS ALLOWED TO BE PERFORMED;
            BY PUTTING HMIN = HMAX THE STEPSIZE MECHANISM IS
            ELIMINATED; IN THIS CASE THE GIVEN VALUES FOR HMIN AND
            HMAX ARE IRRELEVANT, WHILE THE INTEGRATION IS PERFORMED
            WITH THE STEPLENGTH GIVEN BY H;
DELTA:      «ARITHMETIC EXPRESSION»;
            ENTRY: THE REAL PART OF THE POINT AT WHICH EXPONENTIAL
                   FITTING IS DESIRED;
                   (SEE METHOD AND PERFORMANCE);
            ALTERNATIVES:
            DELTA = (AN ESTIMATE OF) THE REAL PART OF THE LARGEST
            EIGENVALUE IN MODULUS OF THE JACOBIAN MATRIX OF THE
            SYSTEM;
            DELTA <= -10**15, IN ORDER TO OBTAIN ASYMPTOTIC STABILITY;
            DELTA = 0, IN ORDER TO OBTAIN A HIGHER ORDER OF ACCURACY
            IN CASE OF LINEAR EQUATIONS;
DERIVATIVE: «PROCEDURE IDENTIFIER»;
            "PROCEDURE" DERIVATIVE(Y); "ARRAY" Y;
            «REPLACEMENT OF THE I-TH COMPONENT OF THE SOLUTION Y BY
            THE I-TH COMPONENT OF THE DERIVATIVE F(Y), I = 1,..., R»;
JACOBIAN: «PROCEDURE IDENTIFIER»;
            "PROCEDURE" JACOBIAN(J, Y); "ARRAY" J, Y;
            WHEN IN GMS JACOBIAN IS CALLED THE ARRAY Y CONTAINS
            THE VALUES OF THE DEPENDENT VARIABLE;
            UPON COMPLETION OF A CALL OF JACOBIAN THE ARRAY J SHOULD
            CONTAIN THE VALUES OF THE JACOBIAN MATRIX OF F(Y);
AETA, RETA: «ARITHMETIC EXPRESSION»;
            ENTRY: MEASURE OF THE ABSOLUTE AND RELATIVE LOCAL
                   ACCURACY REQUIRED;
            THESE VALUES ARE IRRELEVANT WHEN THE INTEGRATION IS PER-
            FORMED WITHOUT THE STEPSIZE MECHANISM;
N:          «VARIABLE»;
            EXIT : THE NUMBER OF INTEGRATION STEPS;
JEV:        «VARIABLE»;
            EXIT: THE NUMBER OF JACOBIAN EVALUATIONS;
LU:         «VARIABLE»;
            EXIT: THE NUMBER OF LU-DECOMPOSITIONS;
```

NSJEV: <VARIABLE>;
ENTRY: NUMBER OF INTEGRATION STEPS PER
JACOBIAN EVALUATION;
THE VALUE OF NSJEV IS RELEVANT ONLY WHEN THE INTEGRATION
IS PERFORMED WITHOUT THE STEPSIZE MECHANISM AND THE
SYSTEM TO BE SOLVED IS NON-LINEAR;

LINEAR: <BOOLEAN EXPRESSION>;
ENTRY: TRUE WHEN THE SYSTEM TO BE INTEGRATED IS LINEAR,
OTHERWISE FALSE;
IF LINEAR IS TRUE THE STEPSIZE MECHANISM IS AUTOMATICALLY
ELIMINATED;

OUT: <PROCEDURE IDENTIFIER>;
"PROCEDURE" OUT;
<BY MEANS OF OUT ONE MAY PRINT THE VALUES OF THE RELEVANT
PARAMETERS OCCURRING IN THE PARAMETERLIST; OUT IS CALLED
AFTER EACH INTEGRATION STEP>;

DATA AND RESULTS: SEE REF[2].

PROCEDURES USED:

VECVEC = CP34010,
MATVEC = CP34011,
MATMAT = CP34013,
ELMROW = CP34024,
ELMVEC = CP34020,
DUPVEC = CP31030,
GSSELM = CP34231,
SOLELM = CP34061,
COLCST = CP31131,
MULVEC = CP31020.

REQUIRED CENTRAL MEMORY:

EXECUTION FIELD LENGTH: 8 * R + 3 * R * R;

RUNNING TIME:

DEPENDS STRONGLY ON THE DIFFERENTIAL EQUATION TO BE SOLVED.

LANGUAGE: ALGOL 60.

METHOD AND PERFORMANCE:

THE PROCEDURE GMS DESCRIBES AN IMPLEMENTATION OF A THIRD ORDER THREE-STEP GENERALIZED LINEAR MULTISTEP METHOD WITH QUASI-ZERO PARASITIC ROOTS AND QUASI-ADAPTIVE STABILITY FUNCTION. IN PARTICULAR THE ALGORITHM IS DEVELOPED FOR THE INTEGRATION OF STIFF SYSTEMS OF ORDINARY DIFFERENTIAL EQUATIONS. THE PROCEDURE SUPPLIES THE ADDITIONAL STARTING VALUES AND PERFORMS A STEPSIZE CONTROL WHICH IS BASED ON THE NON-LINEARITY OF THE DIFFERENTIAL EQUATION. BY THIS CONTROL THE JACOBIAN MATRIX IS INCIDENTALLY EVALUATED. IT IS POSSIBLE TO ELIMINATE THE STEPSIZE CONTROL. THEN, ONE HAS TO GIVE THE NUMBER OF INTEGRATION STEPS PER JACOBIAN EVALUATION. FOR LINEAR EQUATIONS THE STEPSIZE CONTROL IS AUTOMATICALLY ELIMINATED. WHILE THE PROCEDURE PERFORMS ONE EVALUATION OF THE JACOBIAN. MOREOVER, IN THIS CASE THE THREE-STEP SCHEME IS REDUCED TO A ONE-STEP SCHEME. THE PROCEDURE USES ONE FUNCTION EVALUATION PER INTEGRATION STEP AND IT DOES NOT REJECT INTEGRATION STEPS. EACH CHANGE IN THE STEPLENGTH OR EACH REEVALUATION OF THE JACOBIAN COSTS ONE LU-DECOMPOSITION. IT IS POSSIBLE TO FIT EXPONENTIALLY, THIS FITTING IS EQUIVALENT TO FITTING IN THE SENSE OF LINIGER AND WILLOUGHBY, ONLY WHEN THE JACOBIAN MATRIX IS EVALUATED AT EACH INTEGRATION STEP. WHEN THE SYSTEM TO BE INTEGRATED IS NON-LINEAR AND THE JACOBIAN MATRIX IS NOT EVALUATED AT EACH INTEGRATION STEP, IT IS RECOMMENDED TO FIT AT INFINITY (DELTA <= -10**15). DETAILS ARE GIVEN IN REFERENCE 2.

REFERENCES:

[1] HOUWEN, P. J. VAN DER AND VERWER, J. G.,
GENERALIZED LINEAR MULTISTEP METHODS 1, DEVELOPMENT OF ALGORITHMS WITH ZERO-PARASITIC ROOTS,
REPORT NW 10/74, MATHEMATISCH CENTRUM, AMSTERDAM 1974.

[2] VERWER, J. G.,
GENERALIZED LINEAR MULTISTEP METHODS 2, NUMERICAL APPLICATIONS, REPORT NW 12/74, MATHEMATISCH CENTRUM, AMSTERDAM, 1974.

EXAMPLE OF USE:

WE CONSIDER THE DIFFERENTIAL EQUATION

DY1/DX = -1000 * Y1 * (Y1 + Y2 -1.999987),
DY2/DX = -2500 * Y2 * (Y1 + Y2 - 2),

ON THE INTERVAL [0,50], WITH INITIAL VALUE Y1(0) = Y2(0) = 1.
THE REFERENCE SOLUTION AT X = 50 IS GIVEN BY:
Y1(50) = .5976546988,
Y2(50) = 1.4023434075.

```
"BEGIN"
    "PROCEDURE" DER(Y); "ARRAY" Y;
    "BEGIN" "REAL" Y1, Y2;
    Y1:= Y[1]; Y2:= Y[2];
    Y[1]:= -1000 * Y1 * (Y1 + Y2 - 1.999987);
    Y[2]:= -2500 * Y2 * (Y1 + Y2 - 2)
"END" DER;

"PROCEDURE" JAC(J, Y); "ARRAY" J, Y;
"BEGIN" "REAL" Y1, Y2; Y1:= Y[1]; Y2:= Y[2];
    J[1,1]:= 1999.987 - 1000 * (2 * Y1 + Y2);
    J[1,2]:= -1000 * Y1;
    J[2,1]:= -2500 * Y2;
    J[2,2]:= 2500 * (2 - Y1 - 2 * Y2)
"END" JAC;

"PROCEDURE" OUTP;
"IF" X = 50 "THEN"
"BEGIN"  "REAL"  YE1, YE2;
    YE1:= .5976546988; YE2:= 1.4023434075;
    OUTPUT(61, "("
    "("X = ")", 2D2B,
    "("N = ")", 3ZD2B,
    "("JEV = ")", 3ZD2B,
    "("LU = ")", 3ZD, 2/,
    "("Y1 = ")", +.13D"+2D2B,
    "("REL. ERR. = ")", .2D"+2D, /,
    "("Y2 = ")", +.13D"+2D2B,
    "("REL. ERR. = ")", .2D"+2D")",
    X, N, JEV, LU, Y[1], ABS((Y[1] - YE1) / YE1),
    Y[2], ABS((Y[2] - YE2) / YE2))
"END" OUTP;

"INTEGER" N, JEV, LU; "REAL" X;
"ARRAY" Y[1:2];
"PROCEDURE" GMS(X, XE, R, Y, H, HMIN, HMAX, DELTA,
                DERIVATIVE, JACOBIAN, AETA, RETA, N,
                JEV, LU, NBJEV, LINEAR, OUT); "CODE" 33191;
GMS(X, 50, 2, Y, .01, .001, .5, -"15,
    DER, JAC, "-5, "-5, N, JEV,
    LU, 0, "FALSE", OUTP)
"END"

THIS PROGRA" DELIVERS:

X = 50   N =  109   JEV =    3   LU =   12

Y1 = +.597654795800 4"+00   REL. ERR. = .16"-06
Y2 = +.140234331081 3"+01   REL. ERR. = .69"-07
```

```
SOURCE TEXT;
"CODE" 33191;
"PROCEDURE" GMS(X, XE, R, Y, H, HMIN, HMAX, DELTA, DERIVATIVE,
                JACOBIAN, AETA, RETA, N, JEV, LU, NSJEV,
                LINEAR, OUT);
"VALUE" R;
"REAL" X, XE, H, HMIN, HMAX, DELTA, AETA, RETA;
"INTEGER" R, N, JEV, NSJEV, LU;
"BOOLEAN" LINEAR;
"ARRAY" Y;
"PROCEDURE" DERIVATIVE, JACOBIAN, OUT;
"BEGIN"
    "INTEGER" I, J, K, L, NSJEV1, COUNT, COUNT1, KCHANGE;
    "REAL" A, A1, ALFA, E, S1, S2, Z1, X0, XL0, XL1,
    XL2, ETA, H0, H1, Q, Q1, Q2, Q12, Q22, Q1Q2, DISCR;
    "BOOLEAN" UPDATE, CHANGE, REEVAL, STRATEGY;
    "INTEGER" "ARRAY" RI, CI[1:R];
    "ARRAY" AUX[1:9], BD1, BD2[1:3,1:3], Y1,
    Y0[1:R], HJAC, H2JAC2, RQZ[1:R,1:R], YL, FL[1:3 * R];

"REAL" "PROCEDURE" VECVEC(L, U, SHIFT, A, B); "CODE" 34010;
"REAL" "PROCEDURE" MATVEC(L, U, I, A, B);     "CODE" 34011;
"REAL" "PROCEDURE" MATMAT(L, U, I, J, A, B);  "CODE" 34013;
"PROCEDURE" ELMROW(L, U, I, J, A, B, X);      "CODE" 34024;
"PROCEDURE" ELMVEC(L, U, SHIFT, A, B, X);     "CODE" 34020;
"PROCEDURE" DUPVEC(L, U, SHIFT, A, B);        "CODE" 31030;
"PROCEDURE" GSSELM(A, N, AUX, RI, CI);        "CODE" 34231;
"PROCEDURE" SOLELM(A, N, RI, CI, B);          "CODE" 34061;
"PROCEDURE" COLCST(L, U, J, A, X);            "CODE" 31131;
"PROCEDURE" MULVEC(L, U, SHIFT, A, B, X);     "CODE" 31020;

"PROCEDURE" INITIALIZATION;
"BEGIN" LU:= JEV:= N:= NSJEV1:= KCHANGE:= 0; X0:= X; DISCR:= 0;
    K:=1; H1:= H0:= H; COUNT:= -2; AUX[2]:= "-14; AUX[4]:= B;
    DUPVEC(1, R, 0, YL, Y); REEVAL:= CHANGE:= "TRUE";
    STRATEGY:= HMIN "=" HMAX "AND" "LINEAR; Q1:= -1; Q2:= -2;
    COUNT1:= 0; XL0:= XL1:= XL2:= 0
"END" INITIALIZATION;                         "COMMENT"
```

```
        "PROCEDURE" COEFFICIENT;
        "BEGIN" XL2:= XL1; XL1:= XL0; XL0:= X0;
            "IF" CHANGE "THEN"
            "BEGIN" "IF" N > 2 "THEN"
                "BEGIN" Q1:= (XL1 - XL0) / H1;
                    Q2:= (XL2 - XL0) / H1
                "END";
                Q12:= Q1 * Q1; Q22:= Q2 * Q2; Q1Q2:= Q1 * Q2;
                A:= -(3 * ALFA + 1) / 12;
                BD1[1,3]:= 1 + (1 / 3 - (Q1 + Q2) * .5) / Q1Q2;
                BD1[2,3]:= (1 / 3 - Q2 * .5) / (Q12 - Q1Q2);
                BD1[3,3]:= (1 / 3 - Q1 * .5) / (Q22 - Q1Q2);
                BD2[1,3]:= -ALFA * .5 + A * (1 - Q1 - Q2) / Q1Q2;
                BD2[2,3]:= A * (1 - Q2) / (Q12 - Q1Q2);
                BD2[3,3]:= A * (1 - Q1) / (Q22 - Q1Q2);
                "IF" STRATEGY "OR" N <= 2 "THEN"
                "BEGIN" BD1[2,2]:= 1 / (2 * Q1);
                    BD1[1,2]:= 1 - BD1[2,2];
                    BD2[2,2]:= -(3 * ALFA + 1) / (12 * Q1);
                    BD2[1,2]:= -BD2[2,2] - ALFA * .5
                "END"
        "END"
    "END" COEFFICIENT;

    "PROCEDURE" OPERATOR CONSTRUCTION;
    "BEGIN" "IF" REEVAL "THEN"
        "BEGIN" JACOBIAN(HJAC, Y);
            JEV:= JEV + 1; NSJEV1:= 0;
            "IF" DELTA <= -"15 "THEN" ALFA:= 1 / 3 "ELSE"
            "BEGIN" Z1:= H1 * DELTA;
                A:= Z1 * Z1 + 12; A1:= 6 * Z1;
                "IF" ABS(Z1) < .1 "THEN"
                ALFA:= (Z1 * Z1 / 140 - 1) * Z1 / 30 "ELSE"
                "IF" Z1 < -33 "THEN"
                ALFA:= (A + A1) / (3 * Z1 * (2 + Z1)) "ELSE"
                "BEGIN" E:= EXP(Z1); ALFA:= ((A - A1) *
                    E - A - A1) / (((2 - Z1) * E - 2 - Z1) *
                    Z1 * 3)
                "END"
        "END";
        S1:= -(1 + ALFA) * .5; S2:= (ALFA * 3 + 1) / 12
    "END"; "COMMENT"
```

```
        A:= H1 / H0; A1:= A * A;
        "IF" REEVAL "THEN" A:= H1;
        "IF" A "=" 1 "THEN"
        "FOR" J:= 1 "STEP" 1 "UNTIL" R "DO"
        COLCST(1, R, J, HJAC, A);
        "FOR" I:= 1 "STEP" 1 "UNTIL" R "DO"
        "BEGIN" "FOR" J:= 1 "STEP" 1 "UNTIL" R "DO"
            "BEGIN" Q:= H2JAC2[I,J]:= "IF" REEVAL "THEN"
                MATMAT(1, R, I, J, HJAC, HJAC)
                "ELSE" H2JAC2[I,J] * A1;
                RQZ[I,J]:= S2 * Q
            "END";
            RQZ[I,I]:= RQZ[I,I] + 1;
            ELMROW(1, R, I, I, RQZ, HJAC, S1)
        "END";
        GSSELM(RQZ, R, AUX, RI, CI); LU:= LU + 1;
        REEVAL:= UPDATE:= "FALSE"
    "END" OPERATOR CONSTRUCTION;

    "PROCEDURE" DIFFERENCE SCHEME;
    "BEGIN" "IF" COUNT "=" 1 "THEN"
        "BEGIN" DUPVEC(1, R, 0, FL, YL);
            DERIVATIVE(FL); N:= N + 1; NSJEV1:= NSJEV1 + 1
        "END";
        MULVEC(1, R, 0, Y0, YL, (1 - ALFA) / 2 - BD1[1,K]);
        "FOR" L:= 2 "STEP" 1 "UNTIL" K "DO"
        ELMVEC(1, R, R * (L - 1), Y0, YL, -BD1[L,K]);
        "FOR" L:= 1 "STEP" 1 "UNTIL" K "DO"
        ELMVEC(1, R, R * (L - 1), Y0, FL, H1 * BD2[L,K]);
        "FOR" I:= 1 "STEP" 1 "UNTIL" R "DO"
        Y[I]:= MATVEC(1, R, I, HJAC, Y0);
        MULVEC(1, R, 0, Y0, YL, (1 - 3 * ALFA) / 12 - BD2[1,K]);
        "FOR" L:= 2 "STEP" 1 "UNTIL" K "DO"
        ELMVEC(1, R, R * (L - 1), Y0, YL, -BD2[L,K]);
        "FOR" I:= 1 "STEP" 1 "UNTIL" R "DO"
        Y[I]:= Y[I] + MATVEC(1, R, I, H2JAC2, Y0);
        DUPVEC(1, R, 0, Y0, YL);
        "FOR" L:= 1 "STEP" 1 "UNTIL" K "DO"
        ELMVEC(1, R, R * (L - 1), Y0, FL, H1 * BD1[L,K]);
        ELMVEC(1, R, 0, Y, Y0, 1); SOLELM(RQZ, R, RI, CI, Y)
    "END" DIFFERENCE SCHEME;

    "PROCEDURE" NEXT INTEGRATION STEP;
    "BEGIN" "FOR" L:= 2, 1 "DO"
        "BEGIN" DUPVEC(L * R + 1, (L + 1) * R, -R, YL, YL);
            DUPVEC(L * R + 1, (L + 1) * R, -R, FL, FL)
        "END";
        DUPVEC(1, R, 0, YL, Y)
    "END" NEXT INTEGRATION STEP;                                    "COMMENT"
```

```
        "PROCEDURE" TEST ACCURACY;
        "BEGIN" K:= 2;
            DUPVEC(1, R, 0, Y1, Y); DIFFERENCE SCHEME; K:= 3;
            ETA:= AETA + RETA * SQRT(VECVEC(1, R, 0, Y1, Y1));
            ELMVEC(1, R, 0, Y, Y1, -1);
            DISCR:= SQRT(VECVEC(1, R, 0, Y, Y));
            DUPVEC(1, R, 0, Y, Y1)
        "END" TEST ACCURACY;

        "PROCEDURE" STEPSIZE;
        "BEGIN" X0:= X; H0:= H1;
            "IF" N <= 2 "AND" ¬LINEAR "THEN" K:= K + 1;
            "IF" COUNT = 1 "THEN"
            "BEGIN" A:= ETA / (.75 * (ETA + DISCR)) + .33;
                H1:= "IF" A <= .9 "OR" A >= 1.1 "THEN" A * H0
                "ELSE" H0; COUNT:= 0;
                REEVAL:= A <= .9 "AND" NSJEV1 ¬= 1;
                COUNT1:= "IF" A >= 1 "OR" REEVAL "THEN" 0 "ELSE"
                COUNT1 + 1; "IF" COUNT1 = 10 "THEN"
                "BEGIN" COUNT1:= 0; REEVAL:= "TRUE";
                    H1:= A * H0
                "END"
            "END" "ELSE"
            "BEGIN" H1:= H; REEVAL:= NSJEV = NSJEV1 "AND"
                ¬STRATEGY "AND" ¬LINEAR
            "END";
            "IF" STRATEGY "THEN" H1:= "IF" H1 > HMAX
            "THEN" HMAX "ELSE" "IF" H1 < HMIN "THEN" HMIN "ELSE" H1;
            X:= X + H1; "IF" X >= XE "THEN"
            "BEGIN" H1:= XE - X0; X:= XE "END";
            "IF" N <= 2 "AND" ¬LINEAR "THEN" REEVAL:= "TRUE";
            "IF" H1 ¬= H0 "THEN"
            "BEGIN" UPDATE:= "TRUE"; KCHANGE:= 3 "END";
            "IF" REEVAL "THEN" UPDATE:= "TRUE";
            CHANGE:= KCHANGE > 0 "AND" ¬LINEAR;
            KCHANGE:= KCHANGE - 1
        "END" STEPSIZE;

            INITIALIZATION; OUT; X:= X + H1;
            OPERATOR CONSTRUCTION;
            BD1[1,1]:= 1; BD2[1,1]:= -ALFA * .5;
            "IF" ¬LINEAR "THEN" COEFFICIENT;
         NEXT STEP; DIFFERENCE SCHEME;
            "IF" STRATEGY "THEN" COUNT:= COUNT + 1;
            "IF" COUNT = 1 "THEN" TEST ACCURACY;
            OUT; "IF" X >= XE "THEN" "GOTO" END;
            STEPSIZE; "IF" UPDATE "THEN" OPERATOR CONSTRUCTION;
            "IF" ¬LINEAR "THEN" COEFFICIENT;
            NEXT INTEGRATION STEP; "GOTO" NEXT STEP;
        END;
    "END" GMS;
          "EOP"
```

AUTHOR: B.LINDBERG.

CONTRIBUTOR: K.DEKKER.

INSTITUTE: MATHEMATICAL CENTRE.

RECEIVED: 741101.

BRIEF DESCRIPTION:

IMPEX SOLVES AN INITIAL VALUE PROBLEM,GIVEN AS AN AUTONOMOUS SYSTEM OF FIRST ORDER DIFFERENTIAL EQUATIONS , BY MEANS OF THE IMPLICIT MIDPOINT RULE WITH SMOOTHING AND EXTRAPOLATION.
AUTOMATIC STEPSIZE CONTROL IS PROVIDED.
IN PARTICULAR THIS METHOD IS SUITABLE FOR THE INTEGRATION OF STIFF DIFFERENTIAL EQUATIONS.

KEYWORDS:

DIFFERENTIAL EQUATIONS,
INITIAL VALUE PROBLEMS,
STIFF EQUATIONS,
IMPLICIT MIDPOINT RULE,
SMOOTHING,
EXTRAPOLATION.

CALLING SEQUENCE:

THE HEADING OF THE PROCEDURE IMPEX READS:

```
"PROCEDURE" IMPEX (N, TO, TEND, YO, DERIV, AVAILABLE, HO, HMAX,
                   PRESCH, EPS, WEIGHTS, UPDATE, FAIL, CONTROL);
"VALJE" N;
"INTEGER" N;
"REAL" TO,TEND,HO,HMAX,EPS;
"BOOLEAN" PRESCH,FAIL;
"ARRAY" YO,WEIGHTS;
"BOOLEAN" "PROCEDURE" AVAILABLE;
"PROCEDURE" DERIV,UPDATE,CONTROL;
```

IMPEX: INTEGRATES THE SYSTEM OF DIFFERENTIAL EQUATIONS , GIVEN AS DY/DT=F(T,Y), FROM TO UNTIL TEND.

THE MEANING OF THE FORMAL PARAMETERS IS:

N: <ARITHMETIC EXPRESSION>;
THE NUMBER OF EQUATIONS;

TO: <ARITHMETIC EXPRESSION>;
THE INITIAL VALUE OF THE INDEPENDENT VARIABLE;

TEND: <ARITHMETIC EXPRESSION>;
THE FINAL VALUE OF THE INDEPENDENT VARIABLE;

Y0: <ARRAY IDENTIFIER>;
"REAL" "ARRAY" Y0[1:N];
ENTRY: THE INITIAL VALUES OF THE SYSTEM OF DIFFERENTIAL EQUATIONS: Y0[I] AT T=T0;

DERIV: <PROCEDURE" IDENTIFIER>;
THE HEADING OF THIS PROCEDURE READS:
"PROCEDURE" DERIV(T,Y,F,N);
"INTEGER" N; "REAL" T; "ARRAY" Y,F;
THIS PROCEDURE SHOULD DELIVER THE VALUE OF F(T,Y) IN THE ARRAY F[1:N];

AVAILABLE: <PROCEDURE IDENTIFIER>;
THE HEADING OF THIS PROCEDURE READS:
"BOOLEAN" "PROCEDURE" AVAILABLE(T,Y,A,N);
"INTEGER" N; "REAL" T; "ARRAY" Y,A;
IF AN ANALYTIC EXPRESSION OF THE JACOBIAN MATRIX AT THE POINT (T,Y) IS NOT AVAILABLE, THIS PROCEDURE SHOULD DELIVER THE VALUE "FALSE";
OTHERWISE THE PROCEDURE SHOULD DELIVER THE VALUE "TRUE", AND THE JACOBIAN MATRIX SHOULD BE ASSIGNED TO THE ARRAY A[1:N,1:N];

H0: <ARITHMETIC EXPRESSION>;
THE INITIAL STEPSIZE;

HMAX: <ARITHMETIC EXPRESSION>;
MAXIMAL STEPSIZE BY WHICH THE INTEGRATION IS PERFORMED;

PRESCH: <BOOLEAN EXPRESSION>;
INDICATOR FOR CHOICE OF STEPSIZE;
THE STEPSIZE IS AUTOMATICALLY CONTROLLED IF PRESCH="FALSE";
OTHERWISE THE STEPSIZE HAS TO BE PRESCRIBED , EITHER ONLY INITIALLY OR ALSO BY THE PROCEDURE CONTROL;

EPS: <ARITHMETIC EXPRESSION>;
BOUND FOR THE ESTIMATE OF THE LOCAL ERROR;

WEIGHTS: <ARRAY IDENTIFIER>;
"REAL" "ARRAY" WEIGHTS[1:N];
WEIGHTS FOR THE COMPUTATION OF THE WEIGHTED EUCLIDEAN NORM OF THE ERRORS;
ENTRY: INITIAL WEIGHTS;
NOTE THAT THE CHOICE WEIGHTS[I] = 1 IMPLIES AN ESTIMATION OF THE ABSOLUTE ERROR , WHEREAS WEIGTHS[I] = Y[I] DEFINES A RELATIVE ERROR;

UPDATE: <PROCEDURE IDENTIFIER>;
THE HEADING OF THIS PROCEDURE READS:
"PROCEDURE" UPDATE(WEIGHTS,Y2,N);
"INTEGER" N; "ARRAY" WEIGHTS,Y2;
THIS PROCEDURE MAY CHANGE THE ARRAY WEIGHTS , ACCORDING TO THE VALUE OF AN APPROXIMATION FOR Y(T) , GIVEN IN THE ARRAY Y2[1:N];

```
     FAIL:    <BOOLEAN EXPRESSION>;
              EXIT :
              IF  THE PROCEDURE  FAILS TO SOLVE  THE SYSTEM OF EQUATIONS,
              FAIL WILL HAVE THE VALUE "TRUE" UPON EXIT;
              THIS MAY OCCUR  UPON DIVERGENCE  OF THE ITERATION  PROCESS,
              USED IN THE MIDPOINT RULE , WHILE INTEGRATION  IS PERFORMED
              WITH A USER DEFINED PRESCRIBED STEPSIZE;
   CONTROL:   <PROCEDURE IDENTIFIER>;
              THE HEADING OF THIS PROCEDURE READS:
              "PROCEDURE" CONTROL(TPRINT,T,H,HNEW,Y,ERROR,N);
              "INTEGER" N; "REAL" TPRINT,T,H,HNEW; "ARRAY" Y,ERROR;
              CONTROL IS CALLED ON ENTRY OF IMPEX, AND FURTHER AS SOON AS
              THE INEQUALITY  TPRINT <= T  HOLDS;
              DURING A CALL OF CONTROL  PRINTING OF RESULTS AND
              CHANGE OF STEPSIZE (IF PRESCH = "TRUE")  IS THEN  POSSIBLE;
              THE MEANING OF THE FORMAL PARAMETERS IS:
              TPRINT: <VARIABLE>;
                      ENTRY: THE  VALUE  OF THE  INDEPENDENT  VARIABLE AT
                             WHICH A CALL OF CONTROL WAS DESIRED;
                      EXIT:  A NEW VALUE  (TPRINT>T)  AT WHICH  A CALL OF
                             CONTROL IS DESIRED;
              T:      <VARIABLE>;
                      THE ACTUAL VALUE OF THE INDEPENDENT VARIABLE, UP TO
                      WHICH INTEGRATION HAS BEEN PERFORMED;
               H:     <VARIABLE>;
                      HALVE THE ACTUAL STEPSIZE;
               HNEW:  <VARIABLE>;
                      THE NEW STEPSIZE;
                      IF PRESCH="TRUE", THEN THE USER MAY PRESCRIBE A NEW
                      STEPSIZE BY CHANGING HNEW;
               Y:     <ARRAY IDENTIFIER>;
                      "REAL" "ARRAY" Y[1:5,1:N];
                      THE VALUE  OF THE DEPENDENT VARIABLE  AND ITS FIRST
                      FOUR DIVIDED DIFFERENCES  AT THE POINT T  ARE GIVEN
                      IN THIS ARRAY;
              ERROR:  <ARRAY IDENTIFIER>;
                      "REAL" "ARRAY" ERROR[1:3];
                      THE ELEMENTS  OF THIS ARRAY  CONTAIN  THE FOLLOWING
                      ERRORS:
                         ERROR[1]: THE LOCAL ERROR;
                         ERROR[2]: THE GLOBAL ERROR OF SECOND ORDER IN H;
                         ERROR[3]: THE GLOBAL ERROR OF FOURTH ORDER IN H;
                   N:    <VARIABLE>;
                         THE NUMBER OF EQUATIONS;
              EXAMPLE OF USE: SEE EXAMPLE OF USE OF THE PROCEDURE IMPEX;


DATA AND RESULTS:
    FOR DATA, SEE REF[1].
    THE RESULTS OF THE INTEGRATION ARE ATTAINABLE THROUGH THE PROCEDURE
    CONTROL , WHICH IS CALLED AT SPECIFIED, USER DEFINED, VALUES OF THE
    INDEPENDENT  VARIABLE . IN PARTICULAR , THE VALUES OF THE DEPENDENT
    VARIABLE  AT THE ENDPOINT OF INTEGRATION  ARE OBTAINED BY A CALL OF
    CONTROL WITH TPRINT=TEND.
```

PROCEDURES USED:

INIVEC = CP31010,
INIMAT = CP31011,
MULVEC = CP31020,
MULROW = CP31021,
DUPVEC = CP31030,
DUPROWVEC= CP31032,
DUPMAT = CP31035,
VECVEC = CP34010,
MATVEC = CP34011,
MATMAT = CP34013,
ELMVEC = CP34020,
ELMROW = CP34024,
DEC = CP34300,
SOL = CP34051.

REQUIRED CENTRAL MEMORY:

EXECUTION FIELD LENGTH: CIRCA N * (23 + 2 * N) (DECIMAL).

RUNNING TIME: DEPENDS STRONGLY ON THE DIFFERENTIAL EQUATION TO SOLVE.

LANGUAGE: ALGOL 60.

METHOD AND PERFORMANCE:

THE INTEGRATION METHOD (REF[1]) IS BASED ON THE COMPUTATION OF TWO INDEPENDENT SOLUTIONS Y(T,H) AND Y(T,H/2) BY THE IMPLICIT MIDPOINT RULE. PASSIVE SMOOTHING AND PASSIVE EXTRAPOLATION IS PERFORMED TO OBTAIN STABILITY AND HIGH ACCURACY . THE ALGORITHM USES FOR EACH STEP AT LEAST THREE FUNCTION EVALUATIONS, AND ON CHANGE OF STEPSIZE OR AT SLOW CONVERGENCE IN THE ITERATION PROCESS AN APPROXIMATION OF THE JACOBIAN MATRIX (COMPUTED BY DIVIDED DIFFERENCES OR EXPLICITLY SPECIFIED BY THE USER) . IF THE COMPUTED LOCAL ERROR EXCEEDS THE TOLERANCE , THE LAST STEP IS REJECTED. MOREOVER , TWO GLOBAL ERRORS ARE COMPUTED.

REFERENCES:

[1]. B.LINDBERG.
IMPEX 2 , A PROCEDURE FOR THE SOLUTION OF SYSTEMS OF STIFF DIFFERENTIAL EQUATIONS.
ROYAL INSTITUTE OF TECHNOLOGY,STOCKHOLM. TRITA-NA-7303 (1973).

[2]. (TO APPEAR)
COLLOQUIUM STIFF DIFFERENTIAL EQUATIONS 3 (DUTCH).
M.C. SYLLABUS 15.3 (1974), MATHEMATICAL CENTRE.

EXAMPLE OF USE:
CONSIDER THE AUTONOMOUS SYSTEM OF DIFFERNETIAL EQUATIONS:
DY[1]/DX = .2 * (Y[2] - Y[1]),
DY[2]/DX = 10 * Y[1] - (60 - Y[3]/8) * Y[2] + Y[3]/8,
DY[3]/DX = 1,
WITH INITIAL CONDITIONS AT X=0 : Y[1]=Y[2]=Y[3]=0 (SEE REF[2]).
THE SOLUTION AT SEVERAL POINTS IN THE INTERVAL [0, 400] MAY BE
OBTAINED BY THE FOLLOWING PROGRAM:
(THE SOLUTION AT X=400 IS: Y[1]=22.24222011, Y[2]=27.11071335)

```
"BEGIN" "INTEGER" N,NFE,NJE,POINT;
    "REAL" T,TEND,EPS,HMAX,L,H2,TIME;
    "ARRAY" Y,SW[1:3],PRINT[1:5];
    "BOOLEAN" FAIL;
    "PROCEDURE" IMPEX(N,TO,TEND,YO,DERIV,AVAIL,HO,HMAX,PRESCH,EPS,
    WEIGHTS,UPDATE,FAIL,CONTROL); "CODE" 33135;

    "PROCEDURE" LIPEST(L,Y,EPS,T,F,N);
    "REAL" T,L,EPS; "ARRAY" Y; "INTEGER" N; "PROCEDURE" F;
    "BEGIN" "REAL" N1,N2; "INTEGER" I,IT; "ARRAY" F1,F2,Z,X[1:N];
        "PROCEDURE" DUPVEC(L,U,SHIFT,A,B); "CODE" 31030;
        "REAL" "PROCEDURE" VECVEC(L,U,SHIFT,A,B); "CODE" 34010;
        "PROCEDURE" ELMVEC(L,U,SHIFT,A,B,X); "CODE" 34020;
        "REAL" "PROCEDURE" NORM(Y); "ARRAY" Y;
        NORM:=SQRT(VECVEC(1,N,0,Y,Y));
        DUPVEC(1,N,0,Z,Y);
        "FOR" I:=1 "STEP" 1 "UNTIL" N "DO"
        X[I]:="IF" Y[I]=0 "THEN" EPS "ELSE" (1+EPS)*Y[I];
        N1:=NORM(X)*EPS; F(T,X,F1,N);
        "FOR" IT:=1 "STEP" 1 "UNTIL" 5 "DO"
        "BEGIN" F(T,Z,F2,N);
            ELMVEC(1,N,0,F2,F1,-1);
            N2:=N1/NORM(F2);
            DUPVEC(1,N,0,Z,X); ELMVEC(1,N,0,Z,F2,N2)
        "END";
        F(T,Z,F2,N);
        ELMVEC(1,N,0,F2,F1,-1);
        L:=NORM(F2)/N1
"END" LIPEST;

"PROCEDURE" F(T,Y,F1,N); "VALUE" T; "REAL" T; "ARRAY" Y,F1;
"INTEGER" N;
"BEGIN" NFE:=NFE+1;
    F1[1]:=0.2*(Y[2]-Y[1]);
    F1[2]:=10*Y[1]-(60-.125*Y[3])*Y[2]+.125*Y[3];
    F1[3]:=1
"END";

"BOOLEAN" "PROCEDURE" AVAILABLE(T,Y,A,N);
"INTEGER" N; "REAL" T; "ARRAY" Y,A;
"BEGIN" NJE:=NJE+1; AVAILABLE:="TRUE";
    A[1,1]:=-.2; A[1,2]:=.2; A[1,3]:=A[3,1]:=A[3,2]:=A[3,3]:=0;
    A[2,1]:=10; A[2,2]:=.125*Y[3]-60; A[2,3]:=.125*(1+Y[2])
"END"
```

```
      "PROCEDURE" UPDATE(SW,R1,N); "INTEGER" N; "ARRAY" SW,R1;
        "BEGIN" "REAL" S1,S2; "INTEGER" I;
            "FOR" I:=1 "STEP" 1 "UNTIL" N "DO"
            "BEGIN" S1:=1/SW[I]; S2:=ABS(R1[I]);
                "IF" S1<S2 "THEN" SW[I]:=1/S2
            "END"
        "END";

        "PROCEDURE" CONTROL(TP,T,H,HNEW,Y,ERR,N);
        "REAL" TP,T,H,HNEW; "ARRAY" Y,ERR; "INTEGER" N;
        "BEGIN" "INTEGER" I;
            "ARRAY" C[3:5],X[1:N];
            "REAL" S,S2,S3,S4,C1;
NEXT:       S:=(T-TP)/H;
            S2:=S*S; S3:=S2*S; S4:=S3*S;
            C[3]:=(S2-S)/2;
            C[4]:=-S3/6+S2/2-S/3;
            C[5]:=S4/24-S3/4+11*S2/24-S/4;
            "FOR" I:=1 "STEP" 1 "UNTIL" N "DO"
            X[I]:=Y[1,I]-S*Y[2,I]+C[3]*Y[3,I]+C[4]*Y[4,I]+C[5]*Y[5,I];
            OUTPUT(61,"("3ZD.2D2B,D.D"-D2B,2(+D.8D"-D2B),2(4ZD),3ZD.2D,
                  /")",TP,ERR[3],X[1],X[2],NFE,NJE,CLOCK-TIME);
            "IF" TP<TEND "THEN"
            "BEGIN" POINT:=POINT+1; TP:=PRINT[POINT];
                  "IF" TP<=T "THEN" "GOTO" NEXT
            "END"
      "END" CONTROL;

      N:=3; NJE:=NFE:=0; T:=0; TEND:=400; EPS:="-5; HMAX:=400;
      Y[1]:=Y[2]:=Y[3]:=0; SW[1]:=SW[2]:=SW[3]:=1;
      PRINT[1]:=.1; PRINT[2]:=1; PRINT[3]:=10; PRINT[4]:=100;
      PRINT[5]:=400;
      LIPEST(L,Y,"-5,T,F,N);
      H2:=(EPS*320)**(1/5)/(4*L);
      OUTPUT(61,"(""("EPS=")",D.2D"-D,/,"("INTERVAL OF INTEGRATION=(")",
      3ZD,"(",")",3ZD,"(")")",/,"("MAXIMALLY ALLOWED STEPSIZE=")",
      D.2D"-D,//")",EPS,T,TEND,HMAX);
      OUTPUT(61,"(""("LIPSCHCONST=")",BD.3D"+D,/,"("STARTING STEPSIZE")"
      "("=")",BD.2D"+D,/,"("FUNCTIONAL EVAL=")",4ZD,//")",L,H2,NFE);
      TIME:=CLOCK;
      OUTPUT(61,"(""("    X     ERROR        Y[1]                Y[2]")"
      "("          NFE  NJE   TIME")",/")");
      IMPEX(N,T,TEND,Y,F,AVAILABLE,H2,HMAX,"FALSE",EPS,SW,UPDATE,FAIL,
      CONTROL);
      OUTPUT(61,"("/"("NO OF FUNCTIONAL EVALUATIONS= ")",3ZD,/,
      "("NO OF JACOBEAN EVALUATIONS= ")",3ZD,/")",NFE,NJE)
"END"
```

THIS PROGRAM DELIVERS:

EPS=1.00"-5
INTERVAL OF INTEGRATION=(0, 400)
MAXIMALLY ALLOWED STEPSIZE=4.00" 2

LIPSCHCONST= 6.003"+1
STARTING STEPSIZE= 1.32"-3
FUNCTIONAL EVAL= 7

X	ERROR	Y[1]	Y[2]	NFE	NJE	TIME
0.00	0.0" 0	+0.00000000" 0	+0.00000000" 0	7	0	0.01
0.10	6.3"-7	+1.49614151"-6	+1.74013792"-4	46	4	0.72
1.00	1.5"-6	+1.91041887"-4	+2.08361269"-3	85	8	1.48
10.00	8.7"-7	+1.30147663"-2	+2.34487800"-2	119	9	1.99
100.00	1.3"-5	+3.06302487"-1	+3.27552180"-1	225	13	3.47
400.00	1.4"-5	+2.22406546" 1	+2.71090507" 1	556	30	7.51

NO OF FUNCTIONAL EVALUATIONS= 556
NO OF JACOBEAN EVALUATIONS= 30

SOURCE TEXT(S):

```
"CODE" 33135;
"PROCEDURE" IMPEX (N, TO, TEND, YO, DERIV, AVAILABLE, HO, HMAX,
                   PRESCH, EPS, WEIGHTS, UPDATE, FAIL, CONTROL);
"VALUE" N;
"INTEGER" N;
"REAL" TO,TEND,HO,HMAX,EPS;
"BOOLEAN" PRESCH,FAIL;
"ARRAY" YO,WEIGHTS;
"BOOLEAN" "PROCEDURE" AVAILABLE;
"PROCEDURE" DERIV,UPDATE,CONTROL;
"BEGIN" "INTEGER" I,K,ECI;
    "REAL" T,T1,T2,T3,TP,H,H2,HNEW,ALF,LQ;
    "ARRAY" Y,Z,S1,S2,S3,U1,U3,W1,W2,W3,EHR[1:N],R,RF[1:5,1:N],
            ERR[1:3],A1,A2[1:N,1:N];
    "INTEGER" "ARRAY" PS1,PS2[1:N];
    "BOOLEAN" START,TWO,HALV;
    "PROCEDURE" INIVEC(L,U,A,X);                "CODE" 31010;
    "PROCEDURE" INIMAT(LR,UR,LC,UC,A,X);        "CODE" 31011;
    "PROCEDURE" MULVEC(L,U,SHIFT,A,B,X);        "CODE" 31020;
    "PROCEDURE" MULROW(L,U,I,J,A,B,X);          "CODE" 31021;
    "PROCEDURE" DUPVEC(L,U,SHIFT,A,B);          "CODE" 31030;
    "PROCEDURE" DUPROWVEC(L,U,I,A,B);           "CODE" 31032;
    "PROCEDURE" DUPMAT(L,U,I,J,A,B);            "CODE" 31035;
    "REAL" "PROCEDURE" VECVEC(L,U,SHIFT,A,B);   "CODE" 34010;
    "REAL" "PROCEDURE" MATVEC(L,U,I,A,B);       "CODE" 34011;
    "REAL" "PROCEDURE" MATMAT(L,U,I,J,A,B);     "CODE" 34013;
    "PROCEDURE" ELMVEC(L,U,SHIFT,A,B,X);        "CODE" 34020;
    "PROCEDURE" ELMROW(L,U,I,J,A,B,X);          "CODE" 34024;
    "PROCEDURE" DEC(A,N,AUX,P);                 "CODE" 34300;
    "PROCEDURE" SOL(A,N,P,B);                   "CODE" 34051;

    "PROCEDURE" DFDY(T,Y,A); "REAL" T; "ARRAY" Y,A;
    "BEGIN" "INTEGER" I,J; "REAL" SL; "ARRAY" F1,F2[1:N];
        DERIV(T,Y,F1,N);
        "FOR" I:=1 "STEP" 1 "UNTIL" N "DO"
        "BEGIN"
            SL:="-6*Y[I]; "IF" ABS(SL)<"-6 "THEN" SL:="-6;
            Y[I]:=Y[I]+SL; DERIV(T,Y,F2,N);
            "FOR" J:=1 "STEP" 1 "UNTIL" N "DO"
            A[J,I]:=(F2[J]-F1[J])/SL;
            Y[I]:=Y[I]-SL;
        "END"
    "END" DFDY;

    "PROCEDURE" STARTV(Y,T); "VALUE" T; "REAL" T; "ARRAY" Y;
    "BEGIN" "REAL" A,B,C;
        A:=(T-T1)/(T1-T2); B:=(T-T2)/(T1-T3);
        C:=(T-T1)/(T2-T3)*B; B:=A*B;
        A:=1+A+B; B:=A+C-1;
        MULVEC(1,N,0,Y,S1,A); ELMVEC(1,N,0,Y,S2,-B);
        ELMVEC(1,N,0,Y,S3,C)
    "END" STARTV
```

```
;

    "PROCEDURE" ITERATE(Z,Y,A,H,T,WEIGHTS,FAIL,PS);
    "ARRAY" Z,Y,A,WEIGHTS; "REAL" H,T; "LABEL" FAIL;
    "INTEGER" "ARRAY" PS;
    "BEGIN" "INTEGER" IT,LIT; "REAL" MAX,MAX1,CONV; "ARRAY" DZ,F1[1:N];
        "FOR" I:=1 "STEP" 1 "UNTIL" N "DO" Z[I]:=(Z[I]+Y[I])/2;
        IT:=LIT:=1; CONV:=1;
ATER:   DERIV(T,Z,F1,N);
        "FOR" I:=1 "STEP" 1 "UNTIL" N "DO"
        F1[I]:=DZ[I]:=Z[I]-H*F1[I]/2-Y[I];
        SOL(A,N,PS,DZ);
        ELMVEC(1,N,0,Z,DZ,-1);
        MAX:=0;
        "FOR" I:=1 "STEP" 1 "UNTIL" N "DO"
        MAX:=MAX+(WEIGHTS[I]*DZ[I])**2;
        MAX:=SQRT(MAX);
        "IF" MAX*CONV<EPS/10 "THEN" "GOTO" OUT;
        IT:=IT+1; "IF" IT=2 "THEN" "GOTO" ASS;
        CONV:=MAX/MAX1;
        "IF" CONV>.2 "THEN"
        "BEGIN" "IF" LIT=0 "THEN" "GOTO" FAIL;
            LIT:=0; CONV:=1; IT:=1;
            RECOMP(A,H,T,Z,FAIL,PS);
        "END";
ASS:    MAX1:=MAX;
        "GOTO" ATER;
OUT:    "FOR" I:=1 "STEP" 1 "UNTIL" N "DO" Z[I]:=2*Z[I]-Y[I];
"END" ITERATE;

"PROCEDURE" RECOMP(A,H,T,Y,FAIL,PS);
"REAL" H,T; "ARRAY" A,Y; "LABEL" FAIL; "INTEGER" "ARRAY" PS;
"BEGIN" "REAL" SL; "ARRAY" AUX[1:3];
        SL:=H/2;
        "IF" "NOT" AVAILABLE(T,Y,A,N) "THEN" DFDY(T,Y,A);
        "FOR" I:=1 "STEP" 1 "UNTIL" N "DO"
        "BEGIN" MULROW(1,N,I,I,A,A,-SL); A[I,I]:=1+A[I,I]
        "END";
        AUX[2]:="-14;
        DEC(A,N,AUX,PS);
        "IF" AUX[3]<N "THEN" "GOTO" FAIL
"END" RECOMP;

"PROCEDURE" INITIALIZATION;
"BEGIN" H2:=HNEW; H:=H2/2;
        DUPVEC(1,N,0,S1,Y0); DUPVEC(1,N,0,S2,Y0); DUPVEC(1,N,0,S3,Y0);
        DUPVEC(1,N,0,W1,Y0); DUPROWVEC(1,N,1,R,Y0);
        INIVEC(1,N,U1,0); INIVEC(1,N,W2,0);
        INIMAT(2,5,1,N,R,0); INIMAT(1,5,1,N,RF,0);
        T:=T1:=T0; T2:=T0-2*H-"6; T3:=2*T2+1;
        RECOMP(A1,H,T,S1,MISS,PS1);RECOMP(A2,H2,T,W1,MISS,PS2);
"END"
```

```
;

    "PROCEDURE" ONE LARGE STEP;
    "BEGIN" STARTV(Z,T+H);
        ITERATE(Z,S1,A1,H,T+H/2,WEIGHTS,MISS,PS1);
        DUPVEC(1,N,0,Y,Z);
        STARTV(Z,T+H2);
        ITERATE(Z,Y,A1,H,T+3*H/2,WEIGHTS,MISS,PS1);
        DUPVEC(1,N,0,U3,U1); DUPVEC(1,N,0,U1,Y);
        DUPVEC(1,N,0,S3,S2); DUPVEC(1,N,0,S2,S1);
        DUPVEC(1,N,0,S1,Z);
        ELMVEC(1,N,0,Z,W1,1); ELMVEC(1,N,0,Z,S2,-1);
        ITERATE(Z,W1,A2,H2,T+H,WEIGHTS,MISS,PS2);
        T3:=T2; T2:=T1; T1:=T+H2;
        DUPVEC(1,N,0,W3,W2); DUPVEC(1,N,0,W2,W1); DUPVEC(1,N,0,W1,Z);
    "END";

    "PROCEDURE" CHANGE OF INFORMATION;
    "BEGIN" "REAL" ALF1,C1,C2,C3; "ARRAY" KOF[2:4,2:4],E,D[1:4];
        C1:=HNEW/H2; C2:=C1*C1; C3:=C2*C1;
        KOF[2,2]:=C1; KOF[2,3]:=(C1-C2)/2; KOF[2,4]:=C3/6-C2/2+C1/3;
        KOF[3,3]:=C2; KOF[3,4]:=C2-C3; KOF[4,4]:=C3;
        "FOR" I:=1 "STEP" 1 "UNTIL" N "DO"
        U1[I]:=R[2,I]+R[3,I]/2+R[4,I]/3;
        ALF1:=MATVEC(1,N,1,RF,U1)/VECVEC(1,N,0,U1,U1);
        ALF:=(ALF+ALF1)*C1;
        "FOR" I:=1 "STEP" 1 "UNTIL" N "DO"
        "BEGIN"
            E[1]:=RF[1,I]-ALF1*U1[I];
            E[2]:=RF[2,I]-ALF1*2*R[3,I];
            E[3]:=RF[3,I]-ALF1*4*R[4,I];
            E[4]:=RF[4,I];
            D[1]:=R[1,I]; RF[1,I]:=E[1]:=E[1]*C2;
            "FOR" K:=2 "STEP" 1 "UNTIL" 4 "DO"
            "BEGIN" R[K,I]:=D[K]:=MATMAT(K,4,K,I,KOF,R);
                RF[K,I]:=E[K]:=C2*MATVEC(K,4,K,KOF,E)
            "END" K;
            S1[I]:=D[1]+E[1];W1[I]:=D[1]+4*E[1];
            S2[I]:=S1[I]-(D[2]+E[2]/2);
            S3[I]:=S2[I]-(D[2]+E[2])+(D[3]+E[3]/2);
        "END" I;
        T3:=T-HNEW; T2:=T-HNEW/2; T1:=T;
        H2:=HNEW; H:=H2/2; ERR[1]:=0;
        "IF" HALV "THEN"
        "BEGIN" DUPVEC(1,N,0,PS2,PS1); DUPMAT(1,N,1,N,A2,A1) "END";
        "IF" TWO "THEN"
        "BEGIN" DUPVEC(1,N,0,PS1,PS2); DUPMAT(1,N,1,N,A1,A2)
        "END" "ELSE" RECOMP(A1,HNEW/2,T,S1,MISS,PS1);
        "IF" ^HALV "THEN" RECOMP(A2,HNEW,T,W1,MISS,PS2);
    "END" HNEW^=H2
```

```
;

    "PROCEDURE" BACKWARD DIFFERENCES;
    "FOR"I:=1 "STEP" 1 "UNTIL" N "DO"
    "BEGIN" "REAL" B0,B1,B2,B3;
        B1:=(U1[I]+2*S2[I]+U3[I])/4;
        B2:=(W1[I]+2*W2[I]+W3[I])/4;
        B3:=(S3[I]+2*U3[I]+S2[I])/4;
        B2:=(B2-B1)/3; B0:=B1-B2;
        B2:=B2-(S1[I]-2*S2[I]+S3[I])/16;
        B1:=2*B3-(B2+RF[1,I])-(B0+R[1,I])/2;
        B3:=0;
        "FOR" K:=1 "STEP" 1 "UNTIL" 4 "DO"
        "BEGIN" B1:=B1-B3; B3:=R[K,I]; R[K,I]:=B0; B0:=B0-B1
        "END"; R[5,I]:=B0;
        "FOR" K:=1 "STEP" 1 "UNTIL" 4 "DO"
        "BEGIN" B3:=RF[K,I]; RF[K,I]:=B2; B2:=B2-B3 "END";
        RF[5,I]:=B2;
    "END";

    "PROCEDURE" ERROR ESTIMATES;
    "BEGIN" "REAL" C0,C1,C2,C3,B0,B1,B2,B3,W,SL1,SN,LR;
        C0:=C1:=C2:=C3:=0;
        "FOR" I:=1 "STEP" 1 "UNTIL" N "DO"
        "BEGIN" W:=WEIGHTS[I]**2;
            B0:=RF[4,I]/36; C0:=C0+B0*B0*W; LR:=ABS(B0);
            B1:=RF[1,I]+ALF*R[2,I]; C1:=C1+B1*B1*W;
            B2:=RF[3,I]; C2:=C2+B2*B2*W;
            SL1:=ABS(RF[1,I]-RF[2,I]);
            SN:="IF" SL1<"-10 "THEN"1"ELSE"ABS(RF[1,I]-R[4,I]/6)/SL1;
            "IF" SN>1 "THEN" SN:=1;
            "IF" START "THEN" "BEGIN" SN:=SN**4; LR:=LR*4 "END";
            EHR[I]:=B3:=SN*EHR[I]+LR; C3:=C3+B3*B3*W;
        "END" I;
        B0:=ERR[1];
        ERR[1]:=B1:=SQRT(C0); ERR[2]:=SQRT(C1);
        ERR[3]:=SQRT(C3)+SQRT(C2)/2;
        LQ:=EPS/("IF" B0<B1 "THEN" B1"ELSE" B0);
        "IF" B0<B1 "AND" LQ>=80 "THEN" LQ:=10;
    "END";

    "PROCEDURE" REJECT;
    "IF" START "THEN"
    "BEGIN" HNEW:=LQ**(1/5)*H/2; "GOTO" INIT
    "END" "ELSE"
    "BEGIN" "FOR" K:=1,2,3,4,1,2,3 "DO" ELMROW(1,N,K,K+1,R,R,-1);
        "FOR" K:=1,2,3,4 "DO" ELMROW(1,N,K,K+1,RF,RF,-1);
        T:=T-H2; HALV:="TRUE"; HNEW:=H; "GOTO" MSTP
    "END"
```

```
    "PROCEDURE" STEPSIZE;
    "IF" LQ<2 "THEN"
    "BEGIN" HALV:="TRUE"; HNEW:=H "END" "ELSE"
    "BEGIN" "IF" LQ>80 "THEN"
        HNEW:=("IF" LQ>5120 "THEN" (LQ/5)**(1/5) "ELSE" 2)*H2;
        "IF" HNEW>HMAX "THEN" HNEW:=HMAX;
        "IF" TEND>T "AND" TEND-T<HNEW "THEN" HNEW:=TEND-T;
        TWO:=HNEW=2*H2;
    "END";

    "IF" PRESCH "THEN" H:=H0 "ELSE"
    "BEGIN" "IF" H0>HMAX "THEN" H:=HMAX "ELSE" H:=H0;
        "IF" H>(TEND-T0)/4 "THEN" H:=(TEND-T0)/4;
    "END";
    HNEW:=H;
    ALF:=0; T:=TP:=T0;
    INIVEC(1,3,ERR,0); INIVEC(1,N,EHR,0);
    DUPROWVEC(1,N,1,R,Y0);
    CONTROL(TP,T,H,HNEW,R,ERR,N);
INIT: INITIALIZATION; START:="TRUE";
    "FOR" ECI:=0,1,2,3 "DO"
    "BEGIN" ONE LARGE STEP; T:=T+H2;
        "IF" ECI>0 "THEN"
        "BEGIN" BACKWARD DIFFERENCES; UPDATE(WEIGHTS,S2,N) "END"
    "END";
    ECI:=4;
MSTP: "IF" HNEW^=H2 "THEN"
    "BEGIN" ECI:=1; CHANGE OF INFORMATION;
        ONE LARGE STEP; T:=T+H2; ECI:=2;
    "END";
    ONE LARGE STEP;
    BACKWARD DIFFERENCES;
    UPDATE(WEIGHTS,S2,N);
    ERROR ESTIMATES;
    "IF" ECI<4 "AND" LQ>80 "THEN" LQ:=20;
    HALV:=TWO:="FALSE";
    "IF" PRESCH "THEN" "GOTO" TRYCK;
    "IF" LQ<1 "THEN" REJECT "ELSE" STEPSIZE;
TRYCK: "IF" TP<=T "THEN" CONTROL(TP,T,H,HNEW,R,ERR,N);
    "IF" START "THEN" START:="FALSE";
    "IF" HNEW=H2 "THEN" T:=T+H2; ECI:=ECI+1;
    "IF" T<TEND+H2 "THEN" "GOTO" MSTP "ELSE" "GOTO" END;
MISS: FAIL:=PRESCH;
    "IF" ^ FAIL "THEN"
    "BEGIN" "IF" ECI>1 "THEN" T:=T-H2;
        HALV:=TWO:="FALSE"; HNEW:=H2/2;
        "IF" START "THEN" "GOTO" INIT "ELSE" "GOTO" TRYCK
    "END";
END:
"END" IMPEX;
        "EOP"
```

SECTION 5.2.1.1.1.3 CONTAINS TWO ALTERNATIVE PROCEDURES FOR SOLVING FIRST-ORDER INITIAL VALUE PROBLEMS WITH SEVERAL DERIVATIVES AVAILABLE.

A. MODIFIED TAYLOR SOLVES AN INITIAL (BOUNDARY) VALUE PROBLEM, GIVEN AS A SYSTEM OF FIRST ORDER DIFFERENTIAL EQUATIONS , BY MEANS OF A ONE-STEP TAYLOR-METHOD.
IN PARTICULAR THIS METHOD IS SUITABLE FOR THE INTEGRATION OF LARGE SYSTEMS ARISING FROM PARTIAL DIFFERENTIAL EQUATIONS, PROVIDED THAT HIGHER ORDER DERIVATIVES CAN BE EASILY OBTAINED.

B. EXPONENTIALLY FITTED TAYLOR SOLVES AN INITIAL VALUE PROBLEM, GIVEN AS A SYSTEM OF FIRST ORDER DIFFERENTIAL EQUATIONS , BY MEANS OF A ONE-STEP TAYLOR-METHOD . AUTOMATIC STEPSIZE CONTROL IS PROVIDED.
IN PARTICULAR THIS METHOD IS SUITABLE FOR THE INTEGRATION OF STIFF DIFFERENTIAL EQUATIONS , PROVIDED THAT HIGHER ORDER DERIVATIVES CAN BE EASILY OBTAINED.

AUTHORS: P.J. VAN DER HOUWEN AND P.A.BEENTJES.

INSTITUTE: MATHEMATICAL CENTRE.

RECEIVED: 730616.

BRIEF DESCRIPTION:

MODIFIED TAYLOR SOLVES AN INITIAL (BOUNDARY) VALUE PROBLEM, GIVEN AS A SYSTEM OF FIRST ORDER DIFFERENTIAL EQUATIONS , BY MEANS OF A ONE-STEP TAYLOR-METHOD.
IN PARTICULAR THIS METHOD IS SUITABLE FOR THE INTEGRATION OF LARGE SYSTEMS ARISING FROM PARTIAL DIFFERENTIAL EQUATIONS , PROVIDED THAT HIGHER ORDER DERIVATIVES CAN BE EASILY OBTAINED.

KEYWORDS:

DIFFERENTIAL EQUATIONS,
INITIAL (BOUNDARY) VALUE PROBLEMS,
ONE-STEP TAYLOR-METHOD.

CALLING SEQUENCE:

THE HEADING OF THE PROCEDURE READS:
"PROCEDURE" MODIFIED TAYLOR (T,TE,M0,M,U,SIGMA,TAUMIN,I,DERIVATIVE,
K,DATA,ALFA,NORM,AETA,RETA,ETA,RHO,OUT);
"INTEGER" M0,M,I,K,NORM;
"REAL" T,TE,SIGMA,TAUMIN,ALFA,AETA,RETA,RHO;
"ARRAY" U,DATA;
"PROCEDURE" DERIVATIVE,OUT;

THE MEANING OF THE FORMAL PARAMETERS IS:

T: <VARIABLE>;
THE INDEPENDENT VARIABLE T;
MAY BE USED IN DERIVATIVE, SIGMA ETC.;
ENTRY: THE INITIAL VALUE T0;
EXIT : THE FINAL VALUE TE;

TE: <ARITHMETIC EXPRESSION>;
THE FINAL VALUE OF T (TE >= T);

M0,M: <ARITHMETIC EXPRESSION>;
INDICES OF THE FIRST AND LAST EQUATION OF THE SYSTEM TO BE SOLVED;

U: <ARRAY IDENTIFIER>;
"ARRAY" U[M0:M];
THE DEPENDENT VARIABLE;
ENTRY: THE INITIAL VALUES OF THE SOLUTION OF THE SYSTEM OF DIFFERENTIAL EQUATIONS AT T = T0;
EXIT : THE VALUES OF THE SOLUTION AT T = TE;

SIGMA: <ARITHMETIC EXPRESSION>;
THE SPECTRAL RADIUS OF THE JACOBIAN MATRIX WITH RESPECT TO THOSE EIGENVALUES WHICH ARE LOCATED IN THE LEFT HALFPLANE;
IF SIGMA TENDS TO INFINITY , PROCEDURE MODIFIED TAYLOR TERMINATES;

TAUMIN: <ARITHMETIC EXPRESSION>;
MINIMAL STEP LENGTH BY WHICH THE INTEGRATION IS PERFORMED; HOWEVER,ACTUAL STEPSIZES WILL ALWAYS BE WITHIN THE INTERVAL [MIN(HMIN,HSTAB),HSTAB],WHERE HSTAB(= DATA[0]/SIGMA) IS THE STEPLENGTH PRESCRIBED BY STABILITY CONSIDERATIONS;

I: <VARIABLE>;
A JENSEN PARAMETER FOR PROCEDURE DERIVATIVE;
MAY BE USED IN M0 AND M;

DERIVATIVE: <PROCEDURE IDENTIFIER>;
THE HEADING OF THIS PROCEDURE READS:
"PROCEDURE" DERIVATIVE(I,A); "INTEGER" I; "ARRAY" A;
WHEN THIS PROCEDURE IS CALLED, ARRAY A[M0 : M] CONTAINS THE COMPONENTS OF THE (I-1)-ST DERIVATIVE OF U AT THE POINT T;
UPON COMPLETION OF DERIVATIVE, ARRAY A SHOULD CONTAIN THE COMPONENTS OF THE I-TH DERIVATIVE OF U AT THE POINT T;

K: <VARIABLE>;
INDICATES THE NUMBER OF INTEGRATION STEPS PERFORMED;
ENTRY: K = 0;

DATA: <ARRAY IDENTIFIER>;
"ARRAY" DATA[-2 : DATA[-2]];
ENTRY;
DATA[-2]: THE ORDER OF THE HIGHEST DERIVATIVE UPON WHICH THE TAYLOR METHOD IS BASED;
DATA[-1]: ORDER OF ACCURACY OF THE METHOD;
DATA[0] : STABILITY PARAMETER;
DATA[1] , ... , DATA[DATA[-2]] : POLYNOMIAL COEFFICIENTS;
FOR FURTHER EXPLANATION AND POSSIBLE VALUES OF THE ELEMENTS OF ARRAY DATA SEE REFERENCES [2] AND [3];

ALFA: <ARITHMETIC EXPRESSION>;
GROWTH FACTOR FOR THE INTEGRATION STEP LENGTH;

NORM: <ARITHMETIC EXPRESSION>;
IF NORM = 1 DISCREPANCY AND TOLERANCE ARE ESTIMATED IN THE MAXIMUM NORM, OTHERWISE IN THE EUCLIDIAN NORM;

AETA,RETA: <ARITHMETIC EXPRESSION>;
DESIRED ABSOLUTE AND RELATIVE ACCURACY;
IF BOTH AETA AND RETA ARE NEGATIVE , ACCURACY CONDITIONS WILL BE IGNORED;

ETA,RHO: <VARIABLE>;
COMPUTED TOLERANCE AND DISCREPANCY;

OUT; <PROCEDURE IDENTIFIER>;
THE HEADING OF THIS PROCEDURE READS : "PROCEDURE" OUT;
THROUGH THIS PROCEDURE THE VALUES AFTER EACH INTEGRATION STEP OF FOR INSTANCE T, U, ETA AND RHO ARE ACCESSIBLE.

DATA AND RESULTS:

FOR FURTHER EXPLANATION OF THE PARAMETERS AETA, RETA, ETA, RHO, M0, M AND THE ARRAY DATA SEE REFERENCES [2] AND [3].
AS FOR THE INDICES M0 AND M THE FOLLOWING MAY BE REMARKED: WHEN THE METHOD OF LINES IS APPLIED TO HYPERBOLIC DIFFERENTIAL EQUATIONS THE NUMBER OF RELEVANT ORDINARY DIFFERENTIAL EQUATIONS DECREASES DURING THE INTEGRATION PROCESS.
IN PROCEDURE MODIFIED TAYLOR , THIS MAY BE REALIZED BY INTEGER PROCEDURES M0 AND M WHICH ARE DEFINED AS FUNCTIONS OF I, K AND DATA[-2].

PROCEDURES USED: VECVEC = CP34010.

REQUIRED CENTRAL MEMORY:

EXECUTION FIELD LENGTH: CIRCA 75 + M - M0.

RUNNING TIME:

DEPENDS STRONGLY ON THE DIFFERENTIAL EQUATION TO BE SOLVED.

LANGUAGE: ALGOL 60.

METHOD AND PERFORMANCE: SEE REFERENCES.

REFERENCES:

[1].P.J. VAN DER HOUWEN.
ONE-STEP METHODS FOR LINEAR INITIAL VALUE PROBLEMS I,
POLYNOMIAL METHODS,TW REPORT 119,
MATHEMATICAL CENTRE, AMSTERDAM (1970).

[2].P.J. VAN DER HOUWEN, P.BEENTJES, K.DEKKER AND E.SLAGT
ONE-STEP METHODS FOR LINEAR INITIAL VALUE PROBLEMS III,
NUMERICAL EXAMPLES, TW REPORT 130/71,
MATHEMATICAL CENTRE, AMSTERDAM (1971).

[3].P.J. VAN DER HOUWEN, J.KOK.
NUMERICAL SOLUTION OF A MINIMAX PROBLEM, TW REPORT 123/71,
MATHEMATICAL CENTRE, AMSTERDAM (1971).

EXAMPLE OF USE:

THE SOLUTION AT T=EXP(1) AND T=EXP(2) OF THE DIFFERENTIAL EQUATION
DU/DT=-EXP(T)*(U-LN(T)) + 1/T WITH INITIAL CONDITION U(.01)=LN(.01)
AND ANALYTICAL SOLUTION U(T) = LN(T), MAY BE OBTAINED AS FOLLOWS:

```
"BEGIN" "INTEGER" I,K;"REAL" T,TE,ETA,RHO,EXPT,LNT,C0,C1,C2,C3;
    "ARRAY" U[0:0],DATA[-2:4];
    "PROCEDURE" OP;"IF" T=TE "THEN"
    OUTPUT(61,"("("("NUMBER OF STEPS:")",3ZD,/,
                  "("SOLUTION:  T= ")",+D.5D,
                  "("    U(T) = ")",+D.7D,//")",K,T,U[0]);
    "PROCEDURE" DER(I,A);"INTEGER" I;"ARRAY" A;
    "BEGIN" "IF" I=1 "THEN"
        "BEGIN" EXPT:=EXP(T);LNT:=LN(T);C0:=A[0];
            C1:=A[0]:=-EXPT*C0+1/T+EXPT*LNT
        "END";
        "IF" I=2 "THEN" C2:=A[0]:=EXPT*(LNT+1/T-C0-C1)-1/T/T;
        "IF" I=3 "THEN" C3:=A[0]:=
        EXPT*(LNT+2/T-C0-2*C1-C2-1/T/T)+2/T/T/T;
        "IF" I=4 "THEN" A[0]:=C3-2*(1+3/T)/T/T/T+
        EXPT*((1-(2-2/T)/T)/T-C1-C2*2-C3)
        "END";
    "PROCEDURE" MODIFIED TAYLOR(T,TE,M0,M,U,SIGMA,TAUMIN,I,
                DERIVATIVE,K,DATA,ALFA,NORM,AETA,RETA,ETA,RHO,OUT);
    "CODE" 33040;
    I:=-2;"FOR" T:=4,3,6.025,1,.5,1/6,.018455702 "DO"
    "BEGIN" DATA[I]:=T;I:=I+1 "END";
    T:=U[0]:="-2;K:=0;"FOR" TE:=EXP(1),TE*TE "DO"
    MODIFIED TAYLOR(T,TE,0,0,U,EXP(T),"-4,I,DER,K,DATA,1.5,1,"-5,
                    "-4,ETA,RHO,OP)
"END"
```

```
     THIS PROGRAM DELIVERS:
     NUMBER OF STEPS:  46
        SOLUTION:  T= +2.71828     U(T) = +1.0000285

 OF STEPS: 424                                                    051510
        SOLUTION:  T= +7.38906     U(T) = +1.9999967


SOURCE TEXT(S):

"CODE"" 33040;
"PROCEDURE" MODIFIED TAYLOR(T,TE,M0,M,U,SIGMA,TAUMIN,I,DERIVATIVE,K,
                    DATA,ALFA,NORM,AETA,RETA,ETA,RHO,OUT);
"INTEGER" M0,M,I,K,NORM;
"REAL" T,TE,SIGMA,TAUMIN,ALFA,AETA,RETA,ETA,RHO;
"ARRAY" U,DATA;
"PROCEDURE" DERIVATIVE,OUT;

"BEGIN" I:=0;
"BEGIN" "INTEGER" N,P,Q;
    "OWN" "REAL" EC0,EC1,EC2,TAU0,TAU1,TAU2,TAUS,T2;
    "REAL" T0,TAU,TAUI,TAUEC,ECL,BETAN,GAMMA;
    "REAL" "ARRAY" C[M0:M],BETA,BETHA[1:DATA[-2]];
    "BOOLEAN" START,STEP1,LAST;
    "REAL" "PROCEDURE" VECVEC(L,U,SHIFT,A,B); "CODE" 34010;

    "PROCEDURE" COEFFICIENT;
    "BEGIN" "INTEGER" J;"REAL" IFAC;
        IFAC:=1; GAMMA:=.5; N:=DATA[-2]; P:=DATA[-1];
        BETAN:=DATA[0]; Q:= "IF" P<N "THEN" P+1 "ELSE" N;
        "FOR" J:=1 "STEP" 1 "UNTIL" N "DO"
        "BEGIN" BETA[J]:=DATA[J]; IFAC:=IFAC/J;
            BETHA[J]:=IFAC-BETA[J]
        "END";
        "IF" P=N "THEN" BETHA[N]:=IFAC
    "END";                                                    "COMMENT"
```

```
"REAL" "PROCEDURE" NORMFUNCTION(NORM,W);
"INTEGER" NORM; "ARRAY" W;
"BEGIN" "INTEGER" J; "REAL" S,X;
    S:=0;
    "IF" NORM=1 "THEN"
    "BEGIN" "FOR" J:=M0 "STEP" 1 "UNTIL" M "DO"
        "BEGIN"  X:=ABS(W[J]); "IF" X>S "THEN" S:=X "END"
    "END" "ELSE"
    S:=SQRT(VECVEC(M0,M,0,W,W));
    NORMFUNCTION:=S
"END";

"PROCEDURE" LOCAL ERROR BOUND;
ETA:=AETA+RETA * NORMFUNCTION(NORM,U);

"PROCEDURE" LOCAL ERROR CONSTRUCTION(I);"INTEGER" I;
"BEGIN" "IF" I=P "THEN" "BEGIN" ECL:=0;TAUEC:=1 "END";
    "IF" I>P+1 "THEN" TAUEC:=TAUEC*TAU;
    ECL:=ECL+ABS(BETHA[I])*TAUEC*NORMFUNCTION(NORM,C);
    "IF" I=N "THEN"
    "BEGIN" EC0:=EC1;EC1:=EC2;EC2:=ECL;
        RHO:=ECL*TAU**Q
    "END"
"END";

"PROCEDURE" STEPSIZE;
"BEGIN" "REAL" TAUACC,TAUSTAB,AA,BB,CC,EC;
    LOCAL ERROR BOUND;
    "IF" ETA>0 "THEN"
    "BEGIN" "IF" START "THEN"
        "BEGIN" "IF" K=0 "THEN"
            "BEGIN" "INTEGER" J;
                "FOR" J:=M0 "STEP" 1 "UNTIL" M "DO" C[J]:=U[J];
                I:=1; DERIVATIVE(I,C);
                TAUACC:=ETA/NORMFUNCTION(NORM,C);
                STEP1:="TRUE"
            "END" "ELSE"
            "IF" STEP1 "THEN"
            "BEGIN" TAUACC:=(ETA/RHO)**(1/Q)*TAU2;
    "IF" TAUACC>10*TAU2 "THEN"
    TAUACC:=10*TAU2 "ELSE" STEP1:="FALSE"
"END" "ELSE"
"BEGIN" BB:=(EC2-EC1)/TAU1; CC:=EC2-BB*T2;
    EC:=BB*T+CC;
    TAUACC:="IF" EC<0 "THEN" TAU2 "ELSE"
    (ETA/EC)**(1/Q);
    START:="FALSE"
"END"
```

```
                "END" "ELSE"
                "BEGIN" AA:=((EC0-EC1)/TAU0+(EC2-EC1)/TAU1)/
                                (TAU1+TAU0);
                        BB:=(EC2-EC1)/TAU1-AA*(2*T2-TAU1);
                        CC:=EC2-T2*(BB+AA*T2); EC:=CC+T*(BB+T*AA);
                        TAUACC:="IF" EC<0 "THEN" TAUS
                                "ELSE" (ETA/EC)**(1/Q);
                        "IF" TAUACC>ALFA*TAUS "THEN" TAUACC:=ALFA*TAUS;
                        "IF" TAUACC<GAMMA*TAUS "THEN" TAUACC:=GAMMA*TAUS;
                    "END"
                "END" "ELSE" TAUACC:=TE-T;
                "IF" TAUACC<TAUMIN "THEN" TAUACC:=TAUMIN;
                TAUSTAB:=BETAN/SIGMA;
                "IF" TAUSTAB<"-12 * (T-T0) "THEN"
                "BEGIN" OUT;"GOTO" END OF MODIFIED TAYLOR "END";
                TAU:="IF" TAUACC>TAUSTAB "THEN" TAUSTAB "ELSE" TAUACC;
                TAUS:=TAU; "IF" TAU>=TE-T "THEN"
                "BEGIN" TAU:=TE-T;LAST:= "TRUE" "END";
                TAU0:=TAU1;TAU1:=TAU2;TAU2:=TAU
            "END";
              "PROCEDURE" DIFFERENCE SCHEME;
              "BEGIN" "INTEGER" J; "REAL" B;
                  "FOR" J:=M0 "STEP" 1 "UNTIL" M "DO" C[J]:=U[J]; TAUI:=1;
                NEXT TERM:
                  I:=I+1; DERIVATIVE(I,C); TAUI:=TAUI*TAU;
                  B:=BETA[I]*TAUI;
                  "IF" ETA>0 "AND" I>=P "THEN" LOCAL ERROR CONSTRUCTION(I);
                  "FOR" J:=M0 "STEP" 1 "UNTIL" M "DO" U[J]:=U[J]+B*C[J];
                  "IF" I<N "THEN" "GOTO" NEXT TERM;
                  T2:=T; "IF" LAST "THEN"
                  "BEGIN" LAST:= "FALSE"; T:= TE "END"
                  "ELSE" T:= T + TAU
              "END";

              START:= K=0; T0:=T;
              COEFFICIENT; LAST:= "FALSE";
            NEXT LEVEL:
                  STEPSIZE; K:=K+1; I:=0; DIFFERENCE SCHEME; OUT;
                  "IF" T "~=" TE "THEN" "GOTO" NEXT LEVEL
              "END";
              END OF MODIFIED TAYLOR:
        "END" MODIFIED TAYLOR;
                  "EOP"
```

AUTHORS: P.J. VAN DER HOUWEN AND K.DEKKER.

INSTITUTE: MATHEMATICAL CENTRE.

RECEIVED: 740416.

BRIEF DESCRIPTION:

EXPONENTIALLY FITTED TAYLOR SOLVES AN INITIAL VALUE PROBLEM, GIVEN AS A SYSTEM OF FIRST ORDER DIFFERENTIAL EQUATIONS , BY MEANS OF A ONE-STEP TAYLOR-METHOD . AUTOMATIC STEPSIZE CONTROL IS PROVIDED. IN PARTICULAR THIS METHOD IS SUITABLE FOR THE INTEGRATION OF STIFF DIFFERENTIAL EQUATIONS , PROVIDED THAT HIGHER ORDER DERIVATIVES CAN BE EASILY OBTAINED.

KEYWORDS:

DIFFERENTIAL EQUATIONS,
INITIAL VALUE PROBLEMS,
EXPONENTIAL FITTING,
STIFF EQUATIONS,
THREE-CLUSTER METHOD,
ONE-STEP TAYLOR-METHOD,

CALLING SEQUENCE:

```
     THE HEADING OF THE PROCEDURE READS:
     "PROCEDURE"  EXPONENTIALLY  FITTED  TAYLOR (T, TE, MO, M, U, SIGMA,
                                PHI, DIAMETER, DERIVATIVE, I, K, ALFA, NORM,
                                AETA, RETA, ETA, RHO, HMIN, HSTART, OUTPUT);
"INTEGER" MO,M,I,K,NORM;
"REAL"  T,TE,SIGMA,PHI,DIAMETER,ALFA,AETA,RETA,ETA,RHO,HMIN,HSTART;
"ARRAY" U;
"PROCEDURE" DERIVATIVE,OUTPUT;

THE MEANING OF THE FORMAL PARAMETERS IS:
T:          <VARIABLE>;
            THE INDEPENDENT VARIABLE T;
            MAY BE USED IN DERIVATIVE, SIGMA ETC.;
            ENTRY: THE INITIAL VALUE T0;
            EXIT : THE FINAL VALUE TE;
TE:         <ARITHMETIC EXPRESSION>;
            THE FINAL VALUE OF T (TE >= T);
MO:         <ARITHMETIC EXPRESSION>;
            INDEX OF THE FIRST EQUATION OF THE SYSTEM TO BE SOLVED;
M:          <ARITHMETIC EXPRESSION>;
            INDEX OF THE LAST EQUATION OF THE SYSTEM TO BE SOLVED;
U:          <ARRAY IDENTIFIER>;
            "ARRAY" U[MO:M];
            THE DEPENDENT VARIABLE;
            ENTRY: THE INITIAL VALUES OF THE SOLUTION OF THE SYSTEM  OF
                   DIFFERENTIAL EQUATIONS AT T = T0;
            EXIT : THE VALUES OF THE SOLUTION AT T = TE;
SIGMA:      <ARITHMETIC EXPRESSION>;
            THE  MODULUS  OF THE (COMPLEX) POINT  AT WHICH  EXPONENTIAL
            FITTING  IS DESIRED , FOR EXAMPLE  AN APPROXIMATION  OF THE
            MODULUS OF THE CENTRE OF THE LEFT HAND CLUSTER;
PHI:        <ARITHMETIC EXPRESSION>;
            THE ARGUMENT  OF THE (COMPLEX) POINT  AT WHICH  EXPONENTIAL
            FITTING IS DESIRED;
            PHI SHOULD HAVE A VALUE FROM THE RANGE [PI/2,PI];
DIAMETER: <ARITHMETIC EXPRESSION>;
            THE DIAMETER OF THE LEFT HAND CLUSTER;
DERIVATIVE: <PROCEDURE IDENTIFIER>;
            THE HEADING OF THIS PROCEDURE READS;
            "PROCEDURE" DERIVATIVE(I,A): "INTEGER" I; "ARRAY" A;
            I ASSUMES THE VALUES 1,2,3 AND A IS A ONE-DIMENSIONAL ARRAY
            A[MO:M];
            WHEN  THIS  PROCEDURE  IS  CALLED , ARRAY  A  CONTAINS  THE
            COMPONENTS OF THE (I-1)-ST DERIVATIVE OF U  AT THE POINT T;
            UPON COMPLETION OF DERIVATIVE, ARRAY A SHOULD  CONTAIN  THE
            COMPONENTS  OF THE I-TH  DERIVATIVE  OF U AT  THE POINT  T;
I:          <VARIABLE>;
            A JENSEN PARAMETER FOR PROCEDURE DERIVATIVE;
            MAY BE USED IN MO AND M;
```

K: <VARIABLE>;
INDICATES THE NUMBER OF INTEGRATION STEPS PERFORMED;
ENTRY: K = 0;
EXIT : THE NUMBER OF INTEGRATION STEPS PERFORMED;

ALFA: <ARITHMETIC EXPRESSION>;
MAXIMAL GROWTH FACTOR FOR THE INTEGRATION STEP LENGTH;

NORM: <ARITHMETIC EXPRESSION>;
IF NORM = 1 DISCREPANCY AND TOLERANCE ARE ESTIMATED IN THE MAXIMUM NORM, OTHERWISE IN THE EUCLIDIAN NORM;

AETA: <ARITHMETIC EXPRESSION>;
DESIRED ABSOLUTE LOCAL ACCURACY ; AETA SHOULD BE POSITIVE;

RETA: <ARITHMETIC EXPRESSION>;
DESIRED RELATIVE LOCAL ACCURACY ; RETA SHOULD BE POSITIVE;

ETA: <VARIA2LE>;
COMPUTED TOLERANCE;

RHO: <VARIABLE>;
COMPUTED DISCREPANCY;

HMIN: <ARITHMETIC EXPRESSION>;
MINIMAL STEPSIZE BY WHICH THE INTEGRATION IS PERFORMED;
HOWEVER, A SMALLER STEP WILL BE TAKEN IF HMIN EXCEEDS THE STEPSIZE HSTAB , PRESCRIBED BY THE STABILITY CONDITIONS (SEE REF[2], FORMULA 6.12);
IF HSTAB TENDS TO ZERO, THE PROCEDURE TERMINATES;

HSTART: <VARIABLE>;
ENTRY: THE INTITIAL STEPSIZE ; HOWEVER, IF K = 0 ON ENTRY, THE VALUE OF HSTART IS NOT TAKEN INTO CONSIDERATION;
EXIT: A SUGGESTION FOR THE STEPSIZE , IF THE INTEGRATION SHOULD BE CONTINUED FOR T>TE;
HSTART MAY BE USED IN SUCCESSIVE CALLS OF THE PROCEDURE, IN ORDER TO OBTAIN THE SOLUTION IN SEVERAL POINTS TE1,TE2,ETC;

OUTPUT: <PROCEDURE IDENTIFIER>;
THE HEADING OF THIS PROCEDURE READS;
"PROCEDURE" OUTPUT;
THROUGH THIS PROCEDURE THE VALUES AFTER EACH INTEGRATION STEP OF FOR INSTANCE T, U, ETA AND RHO ARE ACCESSIBLE;

DATA AND RESULTS:

FOR FURTHER EXPLANATION OF THE PARAMETERS SIGMA,PHI,DIAMETER,AETA, RETA,ETA,RHO,M0,M SEE REF[2];
FOR RESULTS; SEE EXAMPLE OF USE AND REF[2];

PROCEDURES USED:

INIVEC = CP 31010;
DUPVEC = CP 31030;
VECVEC = CP 34010;
ELMVEC = CP 34020;
ZEROIN = CP 34150.

REQUIRED CENTRAL MEMORY:

EXECUTION FIELD LENGTH: CIRCA 40 + 2 * (M - M0).

RUNNING TIME:

DEPENDS STRONGLY ON THE DIFFERENTIAL EQUATION TO BE SOLVED.

LANGUAGE: ALGOL 60.

METHOD AND PERFORMANCE: SEE REFERENCES.

REFERENCES:

[1]. P.J. VAN DER HOUWEN.
ONE-STEP METHODS FOR LINEAR INITIAL VALUE PROBLEMS II,
POLYNOMIAL METHODS.
TW REPORT 122, (1970) MATHEMATICAL CENTRE.

[2]. P.J. VAN DER HOUWEN, P.BEENTJES, K.DEKKER AND E.SLAGT.
ONE-STEP METHODS FOR LINEAR INITIAL VALUE PROBLEMS III,
NUMERICAL EXAMPLES.
TW REPORT 130. (1971) MATHEMATICAL CENTRE.

EXAMPLE OF USE:

THE SOLUTION AT T=EXP(1) AND T=EXP(2) OF THE DIFFERENTIAL EQUATION
DU/DT=-EXP(T)*(U-LN(T)) + 1/T WITH INITIAL CONDITION U(.01)=LN(.01)
AND ANALYTICAL SOLUTION U(T) = LN(T), MAY BE OBTAINED AS FOLLOWS:

```
"BEGIN" "INTEGER" I,K;
    "REAL" T,TE,TE1,TE2,RETA,ETA,RHO,PI,HS,EXPT,LNT,TIME,U0,U1,U2;
    "REAL" "ARRAY" U[0:0];

"PROCEDURE" EFT (T,TE,M0,M,U,SIGMA,PHI,DIAMETER,DERIVATIVE,I,K,
ALFA,NORM,AETA,RETA,ETA,RHO,HMIN,HSTART,OUTPUT) ; "CODE" 33050;

"PROCEDURE" DERIVATIVE(I,U); "INTEGER" I; "ARRAY" U;
"IF" I=1 "THEN" "BEGIN" EXPT:=EXP(T); LNT:=LN(T); U0:=U[0];
                         U1:=U[0]:=EXPT*(LNT-U0)+1/T
                 "END" "ELSE"
"IF" I=2 "THEN" U2:=U[0]:=EXPT*(LNT-U0-U1+1/T)-1/T/T
         "ELSE" U[0]:=EXPT*(LNT-U0-2*U1-U2+2/T-1/T/T)+2/T/T/T;

"PROCEDURE" OUT;
"IF" T=TE "THEN" OUTPUT(61,"("6ZD,+3ZD.3DB3DB3D")",K,U[0]);
```

```
"PROCEDURE" OUT1;
OUTPUT(61,"("4BD"-D,3Z,3D,/")",RETA,CLOCK-TIME);
OUTPUT(61,"(""("    THIS LINE AND THE FOLLOWING TEXT IS ")"
"("PRINTED BY THIS PROGRAM")",//,
"("    THE RESULTS WITH EFT ARE -CONFER REF[2]- :")",/,
"("     K          U(TE1)          K          U(TE2)")"
"("          RETA   TIME")",/")");
PI:=4*ARCTAN(1); TE1:=EXP(1); TE2:=EXP(2);
"FOR" RETA:="-1,"-2,"-3,"-4 "DO"
"BEGIN" T:=.01; U[0]:=LN(T); K:=0; HS:=0; TIME:=CLOCK;
    "FOR" TE:=TE1,TE2 "DO"
    EFT(T,TE,0,0,U,EXP(T),PI,2*EXP(2*T/3),DERIVATIVE,I,K,1.5,2,
    RETA/10,RETA,ETA,RHO,"-4,HS,OUT); OUT1
"END";

OUTPUT(61,"("//,"("    WITH RELAXED ACCURACY CONDITIONS FOR ")"
"("T>3:")",/,"("     K          U(TE1)          K          U(TE2)")"
"("          RETA   TIME")",/")");
"FOR" RETA:="-1,"-2,"-3,"-4 "DO"
"BEGIN" T:=.01; U[0]:=LN(T); K:=0; HS:=0; TIME:=CLOCK;
    "FOR" TE:=TE1,TE2 "DO"
        EFT(T,TE,0,0,U,EXP(T),PI,2*EXP(2*T/3),DERIVATIVE,I,K,1.5,2,
            RETA/10*("IF" T<3 "THEN" 1 "ELSE" EXP(2*(T-3))),
            RETA*("IF" T<3 "THEN" 1 "ELSE" EXP(2*(T-3))),
            ETA,RHO,"-4,HS,OUT); OUT1
    "END"
"END"
```

```
THIS LINE AND THE FOLLOWING TEXT IS PRINTED BY THIS PROGRAM

THE RESULTS WITH EFT ARE -CONFER REF[2]- :
 K          U(TE1)          K          U(TE2)          RETA    TIME
 15   +1.003 845 001       42   +2.000 076 417      1"-1    .938
 22   +1.001 211 286       52   +2.000 066 067      1"-2   1.121
 36   +1.000 108 738       92   +2.000 020 495      1"-3   1.872
 56   +1.000 045 271      171   +2.000 000 925      1"-4   3.493

WITH RELAXED ACCURACY CONDITIONS FOR T>3:
 K          U(TE1)          K          U(TE2)          RETA    TIME
 15   +1.003 845 001       42   +2.000 076 417      1"-1   1.037
 22   +1.001 211 286       50   +2.000 049 978      1"-2   1.154
 36   +1.000 108 738       68   +2.000 023 330      1"-3   1.419
 56   +1.000 045 271       98   +2.000 065 056      1"-4   2.008
```

SOURCE TEXT(S):

```
"CODE"" 33050;
"PROCEDURE" EXPONENTIALLY FITTED TAYLOR(T,TE,M0,M,U,SIGMA,PHI,DIAMETER,
        DERIVATIVE,I,K,ALFA,NORM,AETA,RETA,ETA,RHO,HMIN,HSTART,OUTPUT);
"INTEGER" M0,M,I,K,NORM;
"REAL" T,TE,SIGMA,PHI,DIAMETER,ALFA,AETA,RETA,ETA,RHO,HMIN,HSTART;
"ARRAY" U;
"PROCEDURE" DERIVATIVE,OUTPUT;
"BEGIN" "INTEGER" KL;
    "REAL" Q,EC0,EC1,EC2,H,HI,H0,H1,H2,BETAN,T2,SIGMAL,PHIL;
    "REAL" "ARRAY" C,R0[M0:M],BETA,BETHA[1:3];
    "BOOLEAN" LAST,START;
    "PROCEDURE" INIVEC(L,U,A,X); "CODE" 31010;
    "PROCEDURE" DUPVEC(L,U,SHIFT,A,B); "CODE" 31030;
    "REAL" "PROCEDURE" VECVEC(L, U, SHIFT, A, B); "CODE" 34010;
    "PROCEDURE" ELMVEC(L,U,SHIFT,A,B,X); "CODE" 34020;
    "BOOLEAN" "PROCEDURE" ZEROIN(X, Y, FX, EPS); "CODE" 34150;

    "PROCEDURE" COEFFICIENT;
    "BEGIN" "REAL" B,B1,B2,BB,E,BETA2,BETA3;
        B:=H*SIGMAL; B1:=B*COS(PHIL); BB:=B*B;
    "IF" ABS(B)<"-3 "THEN"
    "BEGIN" BETA2:=.5-BB/24;
        BETA3:=1/6+B1/12;
        BETHA[3]:=.5+B1/3
    "END" "ELSE"
    "IF" B1<-40 "THEN"
    "BEGIN" BETA2:=(-2*B1-4*B1*B1/BB+1)/BB;
        BETA3:=(1+2*B1/BB)/BB;
        BETHA[3]:=1/BB
    "END" "ELSE"
    "BEGIN" E:=EXP(B1)/BB; B2:=B*SIN(PHIL);
        BETA2:=(-2*B1-4*B1*B1/BB+1)/BB;
        BETA3:=(1+2*B1/BB)/BB;
        "IF" ABS(B2/B)<"-5 "THEN"
        "BEGIN" BETA2:=BETA2-E*(B1-3);
            BETA3:=BETA3+E*(B1-2)/B1;
            BETHA[3]:=1/BB+E*(B1-1)
        "END" "ELSE"
        "BEGIN" BETA2:=BETA2-E*SIN(B2-3*PHIL)/B2*B;
            BETA3:=BETA3+E*SIN(B2-2*PHIL)/B2;
            BETHA[3]:=1/BB+E*SIN(B2-PHIL)/B2*B;
        "END"
    "END";
    BETA[1]:=BETHA[1]:=1;
    BETA[2]:=BETA2; BETA[3]:=BETA3;
    BETHA[2]:=1-BB*BETA3; B:=ABS(B);
    Q:="IF" B<1.5 "THEN" 4-2*B/3 "ELSE" "IF" B<6 "THEN" (30-2*B)/9
    "ELSE" 2;
"END";                                                    "COMMENT"
```

```
;

    "REAL" "PROCEDURE" NORMFUNCTION(NORM,W);
    "INTEGER" NORM; "ARRAY" W;
    "BEGIN" "INTEGER" J; "REAL" S,X;
        S:=0;
        "IF" NORM=1 "THEN"
        "BEGIN" "FOR" J:=M0 "STEP" 1 "UNTIL" M "DO"
            "BEGIN" X:=ABS(W[J]); "IF" X>S "THEN" S:=X "END"
        "END" "ELSE"
        S:=SQRT(VECVEC(M0,M,0,W,W));
        NORMFUNCTION:=S;
"END";

"PROCEDURE" LOCAL ERROR BOUND;
ETA:=AETA+RETA * NORMFUNCTION(NORM,U);

"PROCEDURE" LOCAL ERROR CONSTRUCTION(I); "INTEGER" I;
"BEGIN" "IF" I=1 "THEN" INIVEC(M0,M,R0,0);
        "IF" I<4 "THEN" ELMVEC(M0,M,0,R0,C,BETHA[I]*HI);
        "IF" I=4 "THEN"
        "BEGIN" ELMVEC(M0,M,0,R0,C,-H);
            RHO:=NORMFUNCTION(NORM,R0);
            EC0:=EC1;EC1:=EC2;EC2:=RHO/H**Q;
        "END"
"END";

"PROCEDURE" STEPSIZE;
"BEGIN" "REAL" HACC,HSTAB,HCR,HMAX,A,B,C;
        "IF" "NOT" START "THEN" LOCAL ERROR BOUND;
        "IF" START "THEN"
        "BEGIN" H1:=H2:=HACC:=HSTART;
            EC2:=EC1:=1; KL:=1; START:="FALSE"
        "END" "ELSE"
        "IF" KL<3 "THEN"
        "BEGIN" HACC:=(ETA/RHO)**(1/Q)*H2;
            "IF" HACC>10*H2 "THEN" HACC:=10*H2 "ELSE" KL:=KL+1
        "END" "ELSE"
        "BEGIN" A:=(H0*(EC2-EC1)-H1*(EC1-EC0))/(H2*H0-H1*H1);
            H:=H2*("IF" ETA<RHO "THEN" (ETA/RHO)**(1/Q) "ELSE" ALFA);
            "IF" A>0 "THEN"
        "BEGIN" B:=(EC2-EC1-A*(H2-H1))/H1;
            C:=EC2-A*H2-B*T2; HACC:=0; HMAX:=H;
            "IF" ^ZEROIN(HACC,H,HACC**Q*(A*HACC+B*T+C)-ETA,
            "-3*H2) "THEN" HACC:=HMAX
        "END" "ELSE" HACC:=H;
        "IF" HACC<.5*H2 "THEN" HACC:=.5*H2;
"END";
"IF" HACC<HMIN "THEN" HACC:=HMIN; H:=HACC;                    "COMMENT"
```

```
                "IF" H*SIGMAL>1 "THEN"
                "BEGIN" A:=ABS(DIAMETER/SIGMAL+"-14)/2; B:=2*ABS(SIN(PHIL));
                    BETAN:=("IF" A>B "THEN" 1/A "ELSE" 1/B)/A;
                    HSTAB:=ABS(BETAN/SIGMAL);
                    "IF" HSTAB<"-14*T "THEN" "GOTO" ENDOFEFT;
                    "IF" H>HSTAB "THEN" H:=HSTAB
                "END";
                HCR:=H2*H2/H1;
                "IF" KL>2 "AND" ABS(H-HCR)<"-6*HCR "THEN"
                H:="IF" H<HCR "THEN" HCR*(1-"-7) "ELSE" HCR*(1+"-7);
                "IF" T+H>TE "THEN"
                "BEGIN" LAST:="TRUE"; HSTART:=H; H:=TE-T "END";
                H0:=H1;H1:=H2;H2:=H;
            "END";

            "PROCEDURE" DIFFERENCE SCHEME;
            "BEGIN" HI:=1; SIGMAL:=SIGMA; PHIL:=PHI;
                STEPSIZE;
                COEFFICIENT;
                "FOR" I:=1,2,3 "DO"
                "BEGIN" HI:=HI*H;
                    "IF" I>1 "THEN" DERIVATIVE(I,C);
                    LOCALERRORCONSTRUCTION(I);
                    ELMVEC(M0,M,0,U,C,BETA[I]*HI)
                "END";
                T2:=T; K:=K+1;
                "IF" LAST "THEN"
                "BEGIN" LAST:="FALSE"; T:=TE; START:="TRUE"
                "END" "ELSE" T:=T+H;
                DUPVEC(M0,M,0,C,U);
                DERIVATIVE(1,C);
                LOCALERRORCONSTRUCTION(4);
                OUTPUT;
            "END";

            START:="TRUE"; LAST:="FALSE";
            DUPVEC(M0,M,0,C,U);
            DERIVATIVE(1,C);
            "IF" K=0 "THEN"
                "BEGIN" LOCAL ERROR BOUND; HSTART:=ETA/NORMFUNCTION(NORM,C)
                "END";
            NEXT LEVEL:
                DIFFERENCE SCHEME;
                "IF" T"=TE "THEN" "GOTO" NEXT LEVEL;
            ENDOFEFT:
             "END" EXPONENTIAL FITTED TAYLOR;
                    "EOP"
```

SECTION 5.2.1.1.2.1 CONTAINS FOUR PROCEDURES FOR SOLVING SECOND ORDER DIFFERENTIAL EQUATIONS.

A. RK2 INTEGRATES THE INITIAL VALUE PROBLEM
(D/DX) (D/DX) Y = F(X, Y, (D/DX)Y), A<= X <=B OR B <= X <= A,
Y(A) = YA, (D/DX) Y(A) = ZA.

B. RK2N INTEGRATES THE VECTOR INITIAL PROBLEM
(D/DX) (D/DX) Y = F(X, Y, (D/DX) Y), A<= X <= B OR B <= X <= A,
Y[J] (A) = YA[J], J = 1, .., N,
(D/DX) Y[J] (A) = ZA[J], J = 1, .. N.

C. RK3 INTEGRATES THE INITIAL VALUE PROBLEM (D/DX) (D/DX) Y = F(X,Y)
(WITHOUT THE DERIVATIVE (D/DX) Y IN F), A <= X <= B OR B <= X <=A,
Y(A) = YA, (D/DX) Y(A) = ZA.

D. RK3N INTEGRATES THE VECTOR INITIAL PROBLEM
(D/DX) (D/DX) Y = F(X,Y), A <= X <= B OR B <= X <= A,
Y[J] (A) = YA[J], J = 1, .. ,N,
(D/DX) Y[J] (A) = ZA[J], J = 1, .. ,N.

AUTHOR: J.A.ZONNEVELD.

CONTRIBUTORS: M.BAKKER AND I.BRINK.

INSTITUTE: MATHEMATICAL CENTRE.

RECEIVED: 730715.

BRIEF DESCRIPTION:

RK2 INTEGRATES THE INITIAL VALUE PROBLEM
(D/DX) (D/DX) Y = F(X, Y, (D/DX)Y), A<= X <=B OR B <= X <= A,
Y(A) = YA, (D/DX) Y(A) = ZA.
A 5-TH ORDER RUNGE-KUTTA METHOD IS USED.

KEYWORDS:

RUNGE-KUTTA METHODS,
SECOND ORDER DIFFERENTIAL EQUATION,
INITIAL VALUE PROBLEM.

CALLING SEQUENCE:

```
    THE HEADING OF THE PROCEDURE READS:
    "PROCEDURE" RK2(X, A, B, Y, YA, Z, ZA, FXYZ, E, D, FI);
    "VALUE" B, FI; "REAL" X, A, Y, YA, Z, ZA, FXYZ;
    "BOOLEAN" FI; "ARRAY" E, D;

    THE MEANING OF THE FORMAL PARAMETERS IS:
X:  <VARIABLE>;
    THE INDEPENDENT VARIABLE;
    X CAN BE USED AS A JENSEN PARAMETER;
A:  <ARITHMETIC EXPRESSION>;
    THE INITIAL VALUE OF X;
B:  <ARITHMETIC EXPRESSION>;
    THE END VALUE OF X, (B <= A IS ALLOWED);
Y:  <VARIABLE>;
    THE DEPENDENT VARIABLE;
    Y CAN BE USED AS A JENSEN PARAMETER;
    EXIT :  THE VALUE OF Y(X) AT X = B;
YA:  <ARITHMETIC EXPRESSION>;
    ENTRY : THE INITIAL VALUE OF Y AT X = A,
Z:  <VARIABLE>;
    THE DERIVATIVE DY / DX;
    Z CAN BE USED AS A JENSEN PARAMETER;
    EXIT : THE VALUE OF Z(X) AT X = B;
ZA:  <ARITHMETIC EXPRESSION>;
    ENTRY : THE INITIAL VALUE OF (D/DX) Y AT X = A;
FXYZ:  <ARITHMETIC EXPRESSION>;
    THE RIGHT HAND SIDE OF THE DIFFERENTIAL EQUATION;
    FXYZ DEPENDS ON X, Y, Z, GIVING THE VALUE OF (D/DX) (D/DX) Y;
E:  <ARRAY IDENTIFIER>;
    "ARRAY" E[1 : 4];
    E[1] AND E[3] ARE USED AS RELATIVE , E[2] AND E[4] ARE USED
    AS ABSOLUTE TOLERANCES FOR Y AND DY / DX, RESPECTIVELY;
D:  <ARRAY IDENTIFIER>;
    "ARRAY" D[1 : 5];
    EXIT:
        ENTIER(D[1] + .5)      =    THE NUMBER OF STEPS SKIPPED,
              D[2]             =    THE LAST STEP LENGTH USED,
              D[3]             =    B,
              D[4]             =    Y(B),
              D[5]             =    (D/DX) Y, FOR X = B;
FI:  <BOOLEAN EXPRESSION>;
    IF FI = "TRUE" THEN THE INTEGRATION STARTS AT X=A  WITH A TRIAL
    STEP B - A ; IF FI = "FALSE" THEN THE INTEGRATION IS CONTINUED
    WITH,AS INITIAL CONDITIONS, X = D[3], Y = D[4], Z = D[5];
    A, YA AND ZA ARE IGNORED.
```

DATA AND RESULTS: SEE REF[1].

PROCEDURES USED: NONE.

RUNNING TIME: DEPENDS STRONGLY ON THE DIFFERENTIAL EQUATION TO BE SOLVED.

LANGUAGE: ALGOL 60.

METHOD AND PERFORMANCE: A 5-TH ORDER RUNGE-KUTTA METHOD IS USED, SEE REF[1].

REFERENCES:

[1]. J.A.ZONNEVELD.
AUTOMATIC NUMERICAL INTEGRATION.
MATH. CENTRE TRACT 8 (1970).

EXAMPLE OF USE:

THE VAN DER POL EQUATION

```
(D/DX) (D/DX) Y = 10*(1-Y**2)*(DY/DX) - Y, X >= 0,
Y = 2, DY/DX = 0                        , X=0
```

CAN BE INTEGRATED BY THE PROCEDURE RK2; AT THE POINTS
X = 9.32386578, 18.86305405, 28.40224162, 37.94142918
THE DERIVATIVE DY / DX VANISHES; THE PROGRAM WHICH SOLVES THE VAN DER POL EQUATION READS AS FOLLOWS (WITH E[I] = "-8, I = 1,...,4):

```
"BEGIN" "COMMENT" VAN DER POL;
    "PROCEDURE" RK2(X,A,B,Y,YA,Z,ZA,FXYZ,E,D,FI); "CODE" 33012;
    "REAL" X,Y,Z,B; "BOOLEAN" FI; "ARRAY" E[1:4],D[1:5];
    E[1]:=E[2]:=E[3]:=E[4]:="-8;
    "FOR" B:=9.32386578,18.86305405,28.40224162,37.94142918 "DO"
    "BEGIN" FI:= B<10;
        RK2(X,0,B,Y,2,Z,0,10*(1-Y**2)*Z-Y,E,D,FI);
        OUTPUT(61,"("//10B"("X=")"2D.10D,10B"("Y=")"+2D.10D    ,
        10B"("DY / DX =")"+.5D"2D")",X,Y,Z)
    "END"
"END"
```

RESULTS:

```
  X=09.3238657800      Y=-02.0142853609      DY/DX=+.00000"00
  X=18.8630540500      Y=+02.0142853609      DY/DX=-.00001"00
  X=28.4022416200      Y=-02.0142853609      DY/DX=+.00001"00
  X=37.9414291800      Y=+02.0142853608      DY/DX=-.00002"00
```

SOURCE TEXT(S):

```
"CODE"" 33012 ;
"PROCEDURE" RK2(X, A, B, Y, YA, Z, ZA, FXYZ, E, D, FI);
"VALUE" B, FI; "REAL" X, A, B, Y, YA, Z, ZA, FXYZ; "BOOLEAN" FI;
"ARRAY" E, D;
"BEGIN" "REAL" E1, E2, E3, E4, XL, YL, ZL, H, INT, HMIN, HL,
   ABSH, K0, K1, K2, K3, K4, K5, DISCRY, DISCRZ, TOLY,
   TOLZ, MU, MU1, FHY, FHZ;
   "BOOLEAN" LAST, FIRST, REJECT;
   "IF" FI "THEN"
   "BEGIN" D[3]:= A; D[4]:= YA; D[5]:= ZA "END";
   D[1]:= 0; XL:= D[3]; YL:= D[4]; ZL:= D[5];
   "IF" FI "THEN" D[2]:= B - D[3]; ABSH:= H:= ABS(D[2]);
   "IF" B - XL < 0 "THEN" H:= - H; INT:= ABS(B - XL);
   HMIN:= INT * E[1] + E[2]; HL:= INT * E[3] + E[4];
   "IF" HL < HMIN "THEN" HMIN:= HL; E1:= E[1] / INT;
   E2:= E[2] / INT; E3:= E[3] / INT; E4:= E[4] / INT;
   FIRST:= "TRUE"; "IF" FI "THEN"
   "BEGIN" LAST:= "TRUE"; "GOTO" STEP "END";
TEST: ABSH:= ABS(H); "IF" ABSH < HMIN "THEN"
   "BEGIN" H:= "IF" H > 0 "THEN" HMIN "ELSE" - HMIN; ABSH:= HMIN
   "END";
   "IF" H >= B - XL "EQUIV" H >= 0 "THEN"
   "BEGIN" D[2]:= H; LAST:= "TRUE"; H:= B - XL;
      ABSH:= ABS(H)
   "END"
   "ELSE" LAST:= "FALSE";
STEP: X:= XL; Y:= YL; Z:= ZL; K0:= FXYZ * H;
   X:= XL + H / 4.5;
   Y:= YL + (ZL * 18 + K0 * 2) / 81 * H;
   Z:= ZL + K0 / 4.5  ; K1:= FXYZ * H; X:= XL + H / 3;
   Y:= YL + (ZL * 6 + K0) / 18 * H;
   Z:= ZL + (K0 + K1 * 3) / 12; K2:= FXYZ * H;
   X:= XL + H * .5;
   Y:= YL + (ZL * 8 + K0 + K2) / 16 * H;
   Z:= ZL + (K0 + K2 * 3) / 8; K3:= FXYZ * H;
   X:= XL + H * .8;
   Y:= YL + (ZL * 100 + K0 * 12 + K3 * 28) / 125 * H;      "COMMENT"
```

```
Z:= ZL + (K0 * 53 - K1 * 135 + K2 * 126 + K3 * 56)
/ 125; K4:= FXYZ * H; X:= "IF" LAST "THEN" B "ELSE" XL + H;
Y:= YL + (ZL * 336 + K0 * 21 + K2 * 92 + K4 * 55) /
336 * H;
Z:= ZL + (K0 * 133 - K1 * 378 + K2 * 276 + K3 * 112
+ K4 * 25) / 168; K5:= FXYZ * H;
DISCRY:= ABS(( - K0 * 21 + K2 * 108 - K3 * 112 + K4
* 25) / 56 * H);
DISCRZ:= ABS(K0 * 21 - K2 * 162 + K3 * 224 - K4 *
125 + K5 * 42) / 14;
TOLY:= ABSH * (ABS(ZL) * E1 + E2);
TOLZ:= ABS(K0) * E3 + ABSH * E4;
REJECT:= DISCRY > TOLY "OR" DISCRZ > TOLZ;
FHY:= DISCRY / TOLY; FHZ:= DISCRZ / TOLZ;
"IF" FHZ > FHY "THEN" FHY:= FHZ;
MU:= 1 / (1 + FHY) + .45; "IF" REJECT "THEN"
"BEGIN" "IF" ABSH <= HMIN "THEN"
   "BEGIN" D[1]:= D[1] + 1; Y:= YL; Z:= ZL;
      FIRST:= "TRUE"; "GOTO" NEXT
   "END";
   H:= MU * H; "GOTO" TEST
"END";
   "IF" FIRST "THEN"
   "BEGIN" FIRST:= "FALSE"; HL:= H; H:= MU * H; "GOTO" ACC
   "END";
   FHY:= MU * H / HL + MU - MU1; HL:= H; H:= FHY * H;
ACC: MU1:= MU;
   Y:= YL + (ZL * 56 + K0 * 7 + K2 * 36 - K4 * 15) / 56
   * HL;
   Z:= ZL + ( - K0 * 63 + K1 * 189 - K2 * 36 - K3 * 112
   + K4 * 50) / 28; K5:= FXYZ * HL;
   Y:= YL + (ZL * 336 + K0 * 35 + K2 * 108 + K4 * 25)
   / 336 * HL;
   Z:= ZL + (K0 * 35 + K2 * 162 + K4 * 125 + K5 * 14)
   / 336;
NEXT: "IF" B ^= X "THEN"
   "BEGIN" XL:= X; YL:= Y; ZL:= Z; "GOTO" TEST "END";
   "IF" "NOT"LAST "THEN" D[2]:= H; D[3]:= X; D[4]:= Y; D[5]:= Z
"END" RK2;
"EOP"
```

AUTHOR: J. A. ZONNEVELD.

CONTRIBUTORS: M. BAKKER AND I. BRINK.

INSTITUTE : MATHEMATICAL CENTRE.

RECEIVED: 730715.

BRIEF DESCRIPTION:

RK2N INTEGRATES THE VECTOR INITIAL PROBLEM
(D/DX) (D/DX) Y = F(X, Y, (D/DX) Y), A<= X <= B OR B <= X <= A,
Y[J] (A) = YA[J], J = 1, .., N,
(D/DX) Y[J] (A) = ZA[J], J = 1, .. N;
A 5-TH ORDER RUNGE-KUTTA METHOD IS USED.

KEYWORDS :

RUNGE-KUTTA METHODS.
SECOND ORDER DIFFERENTIAL EQUATION.
INITIAL VALUE PROBLEM.

CALLING SEQUENCE:

```
    THE HEADING OF THE PROCEDURE READS:
    "PROCEDURE" RK2N(X,A,B,Y,YA,Z,ZA,FXYZJ,J,E,D,FI,N);
    "VALUE" B,FI,N;
    "INTEGER" J,N;
    "REAL" X,A,B,FXYZJ;
    "BOOLEAN" FI;
    "ARRAY" Y,YA,Z,ZA,E,D;

    THE MEANING OF THE FORMAL PARAMETERS IS:
X:      <VARIABLE>;
        THE INDEPENDENT VARIABLE.
        UPON COMPLETION OF A CALL OF RK2N,
        IT IS EQUAL TO B;
A:      <ARITHMETIC EXPRESSION>;
        THE STARTING VALUE OF X;
B:      <ARITHMETIC EXPRESSION>;
        A VALUE PARAMETER,GIVING THE END VALUE OF X;
Y:      <ARRAY IDENTIFIER>;
        "ARRAY" Y[1:N];
        THE VECTOR OF DEPENDENT VARIABLES;
        EXIT: THE VALUE OF Y[J] (B), (J = 1, .. ,N);
YA:     <ARRAY IDENTIFIER>;
        "ARRAY" YA[1:N];
        ENTRY : THE STARTING VALUES OF Y[J],I.E. THE VALUES AT X=A;
Z:      <ARRAY IDENTIFIER>;
        "ARRAY" Z[1:N];
        THE FIRST DERIVATIVES OF THE DEPENDENT VARIABLES;
        EXIT : THE VALUE OF (D/DX)Y[J](B) (J = 1, .. ,N);
ZA:     <ARRAY IDENTIFIER>;
        "ARRAY" ZA[1:N];
        ENTRY : THE STARTING VALUES OF Z[J],I.E. THE VALUES AT X=A;
FXYZJ:<ARITHMETIC EXPRESSION>;
        AN EXPRESSION DEPENDING ON X,J,Y[I],Z[I]  (I=1,...,N),
        GIVING THE VALUE OF (D/DX)(D/DX)Y[J];
J:      <VARIABLE>;
        A VARIABLE OF TYPE INTEGER,USED IN THE ACTUAL PARAMETER
        CORRESPONDING TO FXYZJ,TO DENOTE THE NUMBER OF THE
        EQUATION REQUIRED (JENSEN'S DEVICE);
E:      <ARRAY IDENTIFIER>;
        "ARRAY" E[1:4*N];
        THE ELEMENT E[2*J-1] IS A RELATIVE AND E[2*J] IS AN ABSOLUTE
        TOLERANCE ASSOCIATED WITH Y[J];
        E[2*(N+J)-1] IS A RELATIVE AND E[2*(N+J)] IS AN ABSOLUTE
        TOLERANCE ASSOCIATED WITH Z[J];
```

```
D:      <ARRAY IDENTIFIER>;
        "ARRAY" D[1:2*N+3];
        EXIT:
        ENTIER(D[1]+.5) IS THE NUMBER OF STEPS SKIPPED;
        D[2] IS THE LAST  STEP LENGTH USED;
        D[3] IS EQUAL TO B;
        D[4],...,D[N+3] ARE EQUAL TO Y[1],...,Y[N] FOR X=B,
        D[N+4],...,D[2*N+3] ARE EQUAL TO THE DERIVATIVES
        Z[1],...,Z[N] FOR X=B;
FI:     <BOOLEAN EXPRESSION>;
        IF FI="TRUE" THEN THE INTEGRATION STARTS AT A,WITH A TRIAL
        STEP B-A;IF FI="FALSE" THEN THE INTEGRATION IS CONTINUED
        VIZ. WITH INITIAL CONDITIONS:X=D[3],Y[J]=D[J+3],Z[J]=
        D[N+3+J] AND STEP LENGTH H=D[2]*SIGN(B-D[3]);
        A, YA, ZA ARE IGNORED;
N:      <ARITHMETIC EXPRESSION>;
        THE NUMBER OF EQUATIONS.
```

DATA AND RESULTS:

```
RK2N INTEGRATES (D/DX)(D/DX)Y = F(X,Y,Z) FROM X TO B,WITH, EITHER
(IF FI = "TRUE") X=A, Y[J]=YA[J], Z[J]=ZA[J], OR  (IF FI="FALSE")
X = D[3], Y[J]=D[J+3], Z[J]=D[N+J+3], J=1,...,N, USING A 5-TH ORDER
RUNGE-KUTTA METHOD.
UPON COMPLETION OF A CALL OF RK2N WE HAVE:X=D[3]=B, Y[J]=D[J+3]
THE VALUE OF THE DEPENDENT VARIABLES FOR X=B, Z[J]=D[N+J+3], THE
VALUE OF THE DERIVATIVES OF Y[J] AT X=B, J=1,...,N.
RK2N USES AS ITS MININMAL ABSOLUTE STEP LENGTH
HMIN=MIN (E[2*J-1]*INT+E[2 *J]) WITH 1<=J<=2*N AND INT=
ABS(B-("IF" FI "THEN" A "ELSE" D[3])).
IF A STEP OF LENGTH ABS(H)<=HMIN IS REJECTED, A STEP SIGN(H)*HMIN
IS SKIPPED.  A STEP IS REJECTED IF THE ABSOLUTE VALUE OF THE
COMPUTED DISCRETIZATION ERROR IS GREATER THAN
( ABS(Z[J]) * E[2 * J - 1] + E[2 * J] ) * ABS(H) / INT
OR IF THAT TERM IS GREATER THEN (ABS(FXYZJ)*E[2*(J+N)-1
+E[2*(J+N)])ABS(H)/INT, FOR ANY VALUE OF J ,1<=J<=N (INT=ABS(B-A)).
SEE REF[1].
```

PROCEDURES USED: NONE.

REQUIRED CENTRAL MEMORY:
EIGHT ARRAYS OF ORDER N AND ONE OF ORDER 4 * N ARE USED.

LANGUAGE:ALGOL 60.

METHOD AND PERFORMANCE:SEE REF[1];

EXAMPLE OF USE:

THE SECOND ORDER (VECTOR) DIFFERENTIAL EQUATION

(D/DX)(D/DX)Y[1] = -5*(Y[1] + (D/DX)Y[2]) + Y[2],
(D/DX)(D/DX)Y[2] = -5*(Y[2] + (D/DX)Y[1]) + Y[1], X>=0,

Y[1] = (D/DX)Y[2] = 1, Y[2] = (D/DX)Y[1] = 0, X=0

WITH ANALYTIC SOLUTION

Y[1] = -EXP(-X)*(EXP(-X)*(EXP(-X)*(EXP(-X)/3+.5)-1)-5/6),
Y[2] = -EXP(-X)*(EXP(-X)*(EXP(-X)*(EXP(-X)/3-.5)+1)-5/6)

CAN BE INTEGRATED BY RK2N FROM 0 TO 5 WITH 1,2,3,4 AS REFERENCE POINTS. THE PROGRAM READS AS FOLLOWS:

```
"BEGIN" "REAL" B, X, EXPX; "INTEGER" K; "BOOLEAN" FI;
    "ARRAY" Y,YA,Z,ZA[0:2],E[1:8],D[0:7];
    "PROCEDURE" RK2N(X,A,B,Y,YA,Z,ZA,FXYZJ,J,E,D,FI,N);"CODE"33013;
    "FOR"  K:=1,2,3,4,5,6,7,8 "DO" E[K]:="-7;
    YA[1]:=ZA[2]:=1; YA[2]:=ZA[1]:=0; B:=1; AA: FI:=B=1;
    RK2N(X,0,8,Y,YA,Z,ZA,-5*(Y[K]+Z[K])+("IF"K=1"THEN"Y[2]"ELSE"
    Y[1]),K,E,D,FI,2);
    "COMMENT" COMPUTATION  OF THE EXACT VALUES OF Y AND DY/DX;
    EXPX:=EXP(-X);
    YA[1]:=-EXPX*(EXPX*(EXPX*(EXPX/3+.5)-1)-5/6);
    YA[2]:=-EXPX*(EXPX*(EXPX*(EXPX/3-.5)+1)-5/6);
    ZA[1]:=+EXPX*(EXPX*(EXPX*(EXPX/.75+1.5)-2)-5/6);
    ZA[2]:=+EXPX*(EXPX*(EXPX*(EXPX/.75-1.5)+2)-5/6);
    OUTPUT(61,"("/20B"("X=")"D.4D/,
    10B"("Y[1]-YEXACT[1]=")"+.14D ,10B"("Y[2]-YEXACT[2]=")"+.14D4/,
    10B"("Z[1]-ZEXACT[1]=")"+.14D ,10B"("Z[2]-ZEXACT[2]=")"+.14D
    5/")",X,Y[1]-YA[1],Y[2]-YA[2],Z[1]-ZA[1],Z[2]-ZA[2]);
    B:=B+1; "IF" B<5 "THEN" "GO TO" AA
"END"
```

RESULTS:

```
      X=1.0000
Y[1]-YEXACT[1]=+.000000000002955    Y[2]-YEXACT[2]=+.000000000567
Z[1]-ZEXACT[1]=-.000000000013770    Z[2]-ZEXACT[2]=-.000000002422

      X=2.0000
Y[1]-YEXACT[1]=-.000000000085294    Y[2]-YEXACT[2]=+.000000001486
Z[1]-ZEXACT[1]=+.000000000378800    Z[2]-ZEXACT[2]=-.000000006509

      X=3.0000
Y[1]-YEXACT[1]=-.000000000162707    Y[2]-YEXACT[2]=-.000000004796
Z[1]-ZEXACT[1]=+.000000000803265    Z[2]-ZEXACT[2]=+.000000019380

      X=4.0000
Y[1]-YEXACT[1]=-.000000000117993    Y[2]-YEXACT[2]=-.000000008505
Z[1]-ZEXACT[1]=+.000000000633393    Z[2]-ZEXACT[2]=+.000000039114
```

SOURCE TEXT(S):

```
"CODE"" 33013 ;
"PROCEDURE" RK2N(X, A, B, Y, YA, Z, ZA, FXYZJ, J, E, D,
FI, N); "VALUE" B, FI, N; "INTEGER" J, N; "REAL" X, A, B, FXYZJ;
"BOOLEAN" FI; "ARRAY" Y, YA, Z, ZA, E, D;
"BEGIN" "INTEGER" JJ;
   "REAL" XL, H, INT, HMIN, HL, ABSH, FHM, DISCRY, DISCRZ,
   TOLY, TOLZ, MU, MU1, FHY, FHZ;
   "BOOLEAN" LAST, FIRST, REJECT;
   "ARRAY" YL, ZL, K0, K1, K2, K3, K4, K5[1:N], EE[1:4 *
   N];
   "IF" FI "THEN"
   "BEGIN" D[3]:= A;
      "FOR" JJ:= 1 "STEP" 1 "UNTIL" N "DO"
      "BEGIN" D[JJ + 3]:= YA[JJ]; D[N + JJ + 3]:= ZA[JJ]
      "END"
   "END";
   D[1]:= 0; XL:= D[3];
   "FOR" JJ:= 1 "STEP" 1 "UNTIL" N "DO"
   "BEGIN" YL[JJ]:= D[JJ + 3]; ZL[JJ]:= D[N + JJ + 3] "END";
   "IF" FI "THEN" D[2]:= B - D[3]; ABSH:= H:= ABS(D[2]);
   "IF" B - XL < 0 "THEN" H:= - H; INT:= ABS(B - XL);
   HMIN:= INT * E[1] + E[2];
   "FOR" JJ:= 2 "STEP" 1 "UNTIL" 2 * N "DO"
   "BEGIN" HL:= INT * E[2 * JJ - 1] + E[2 * JJ];
      "IF" HL < HMIN "THEN" HMIN:= HL
   "END";
   "FOR" JJ:= 1 "STEP" 1 "UNTIL" 4 * N "DO" EE[JJ]:= E[JJ] / INT;
   FIRST:= "TRUE"; "IF" FI "THEN"
   "BEGIN" LAST:= "TRUE"; "GOTO" STEP "END";
TEST; ABSH:= ABS(H); "IF" ABSH < HMIN "THEN"
   "BEGIN" H:= "IF" H > 0 "THEN" HMIN "ELSE" - HMIN;
      ABSH:= ABS(H)
   "END";
   "IF" H >= B - XL "EQUIV" H >= 0 "THEN"
   "BEGIN" D[2]:= H; LAST:= "TRUE"; H:= B - XL;
      ABSH:= ABS(H)
   "END"
   "ELSE" LAST:= "FALSE";
STEP; X:= XL;
   "FOR" JJ:= 1 "STEP" 1 "UNTIL" N "DO"
   "BEGIN" Y[JJ]:= YL[JJ]; Z[JJ]:= ZL[JJ] "END";
   "FOR" J:= 1 "STEP" 1 "UNTIL" N "DO" K0[J]:= FXYZJ * H;
   X:= XL + H / 4.5;
   "FOR" JJ:= 1 "STEP" 1 "UNTIL" N "DO"
   "BEGIN" Y[JJ]:= YL[JJ] + (ZL[JJ] * 18 + K0[JJ] * 2) /
      81 * H; Z[JJ]:= ZL[JJ] + K0[JJ] / 4.5;
   "END";                                                    "COMMENT"
```

```
;
      "FOR" J:= 1 "STEP" 1 "UNTIL" N "DO" K1[J]:= FXYZJ * H;
      X:= XL + H / 3;
      "FOR" JJ:= 1 "STEP" 1 "UNTIL" N "DO"
      "BEGIN" Y[JJ]:= YL[JJ] + (ZL[JJ] * 6 + K0[JJ]) / 18 * H;
         Z[JJ]:= ZL[JJ] + (K0[JJ] + K1[JJ] * 3) / 12
      "END";
      "FOR" J:= 1 "STEP" 1 "UNTIL" N "DO" K2[J]:= FXYZJ * H;
      X:= XL + H * .5;
      "FOR" JJ:= 1 "STEP" 1 "UNTIL" N "DO"
      "BEGIN" Y[JJ]:= YL[JJ] + (ZL[JJ] * 8 + K0[JJ] + K2[JJ])
         / 16 * H;
         Z[JJ]:= ZL[JJ] + (K0[JJ] + K2[JJ] * 3) / 8
      "END";
      "FOR" J:= 1 "STEP" 1 "UNTIL" N "DO" K3[J]:= FXYZJ * H;
      X:= XL + H * .8;
      "FOR" JJ:= 1 "STEP" 1 "UNTIL" N "DO"
      "BEGIN" Y[JJ]:= YL[JJ] + (ZL[JJ] * 100 + K0[JJ] * 12 +
         K3[JJ] * 28) / 125 * H;
         Z[JJ]:= ZL[JJ] + (K0[JJ] * 53 - K1[JJ] * 135 +
         K2[JJ] * 126 + K3[JJ] * 56) / 125
      "END";
      "FOR" J:= 1 "STEP" 1 "UNTIL" N "DO" K4[J]:= FXYZJ * H;
      X:= "IF" LAST "THEN" B "ELSE" XL + H;
      "FOR" JJ:= 1 "STEP" 1 "UNTIL" N "DO"
      "BEGIN" Y[JJ]:= YL[JJ] + (ZL[JJ] * 336 + K0[JJ] * 21 +
         K2[JJ] * 92 + K4[JJ] * 55) / 336 * H;
         Z[JJ]:= ZL[JJ] + (K0[JJ] * 133 - K1[JJ] * 378 +
         K2[JJ] * 276 + K3[JJ] * 112 + K4[JJ] * 25) / 168
      "END";
      "FOR" J:= 1 "STEP" 1 "UNTIL" N "DO" K5[J]:= FXYZJ * H;
      REJECT:= "FALSE"; FHM:= 0;
      "FOR" JJ:= 1 "STEP" 1 "UNTIL" N "DO"
      "BEGIN" DISCRY:= ABS(( - K0[JJ] * 21 + K2[JJ] * 108 -
         K3[JJ] * 112 + K4[JJ] * 25) / 56 * H);
         DISCRZ:= ABS(K0[JJ] * 21 - K2[JJ] * 162 + K3[JJ]
         * 224 - K4[JJ] * 125 + K5[JJ] * 42) / 14;
         TOLY:= ABSH * (ABS(ZL[JJ]) * EE[2 * JJ - 1] +
         EE[2 * JJ]);
         TOLZ:= ABS(K0[JJ]) * EE[2 * (JJ + N) - 1] + ABSH
         * EE[2 * (JJ + N)];
         REJECT:= DISCRY > TOLY "OR" DISCRZ > TOLZ "OR" REJECT;
         FHY:= DISCRY / TOLY; FHZ:= DISCRZ / TOLZ;
         "IF" FHZ > FHY "THEN" FHY:= FHZ;
         "IF" FHY > FHM "THEN" FHM:= FHY
      "END";                                                    "COMMENT"
```

```
      MU:= 1 / (1 + FHM) + .45; "IF" REJECT "THEN"
   "BEGIN" "IF" ABSH <= HMIN "THEN"
      "BEGIN" D[1]:= D[1] + 1;
         "FOR" JJ:= 1 "STEP" 1 "UNTIL" N "DO"
         "BEGIN" Y[JJ]:= YL[JJ]; Z[JJ]:= ZL[JJ] "END";
         FIRST:= "TRUE"; "GOTO" NEXT
      "END";
      H:= MU * H; "GOTO" TEST
   "END";
   "IF" FIRST "THEN"
   "BEGIN" FIRST:= "FALSE"; HL:= H; H:= MU * H; "GOTO" ACC
   "END";
   FHM:= MU * H / HL + MU - MU1; HL:= H; H:= FHM * H;
ACC: MU1:= MU;
   "FOR" JJ:= 1 "STEP" 1 "UNTIL" N "DO"
   "BEGIN" Y[JJ]:= YL[JJ] + (ZL[JJ] * 56 + K0[JJ] * 7 +
      K2[JJ] * 36 - K4[JJ] * 15) / 56 * HL;
      Z[JJ]:= ZL[JJ] + ( - K0[JJ] * 63 + K1[JJ] * 189
      - K2[JJ] * 36 - K3[JJ] * 112 + K4[JJ] * 50) / 28
   "END";
   "FOR" J:= 1 "STEP" 1 "UNTIL" N "DO" K5[J]:= FXYZJ * HL;
   "FOR" JJ:= 1 "STEP" 1 "UNTIL" N "DO"
   "BEGIN" Y[JJ]:= YL[JJ] + (ZL[JJ] * 336 + K0[JJ] * 35 +
      K2[JJ] * 108 + K4[JJ] * 25) / 336 * HL;
      Z[JJ]:= ZL[JJ] + (K0[JJ] * 35 + K2[JJ] * 162 +
      K4[JJ] * 125 + K5[JJ] * 14) / 336
   "END";
NEXT: "IF" B ≠ X "THEN"
   "BEGIN" XL:= X;
      "FOR" JJ:= 1 "STEP" 1 "UNTIL" N "DO"
      "BEGIN" YL[JJ]:= Y[JJ]; ZL[JJ]:= Z[JJ] "END";
      "GOTO" TEST
   "END";
   "IF" "NOT"LAST "THEN" D[2]:= H; D[3]:= X;
   "FOR" JJ:= 1 "STEP" 1 "UNTIL" N "DO"
   "BEGIN" D[JJ + 3]:= Y[JJ]; D[N + JJ + 3]:= Z[JJ] "END"
"END" RK2N;
      "EOP"
```

AUTHOR: J.A.ZONNEVELD.

CONTRIBUTORS: M.BAKKER AND I.BRINK.

INSTITUTE: MATHEMATICAL CENTRE.

RECEIVED: 730715.

BRIEF DESCRIPTION:

RK3 INTEGRATES THE INITIAL VALUE PROBLEM (D/DX) (D/DX) Y = F(X,Y)
(WITHOUT THE DERIVATIVE (D/DX) Y IN F), A <= X <= B OR B <= X <=A,
Y(A) = YA, (D/DX) Y(A) = ZA.
A 5-TH ORDER RUNGE-KUTTA METHOD IS USED.

KEYWORDS:

RUNGE-KUTTA METHODS,
SECOND ORDER DIFFERENTIAL EQUATION,
INITIAL VALUE PROBLEM.

CALLING SEQUENCE:

THE HEADING OF THE PROCEDURE READS:

```
"PROCEDURE" RK3(X,A,B,Y,YA,Z,ZA,FXY,E,D,FI);
"VALUE" B,FI;
"REAL" X,A,B,Y,YA,Z,ZA,FXY;
"BOOLEAN" FI;
"ARRAY" E,D;
```

THE MEANING OF THE FORMAL PARAMETERS IS:

```
X:      <VARIABLE>;
        THE INDEPENDENT VARIABLE.
        UPON COMPLETION OF A CALL OF RK3 ,
        IT IS EQUAL TO B;
A:      <ARITHMETIC EXPRESSION>;
        THE STARTING VALUE OF X;
B:      <ARITHMETIC EXPRESSION>;
        A VALUE PARAMETER, GIVING THE END VALUE OF X;
        B <= A IS ALLOWED;
Y:      <VARIABLE>;
        THE DEPENDENT VARIABLE;
        EXIT : THE VALUE OF Y(X) AT X = B;
YA:     <ARITHMETIC EXPRESSION>;
        ENTRY : THE VALUE OF Y AT X=A;
Z:      <VARIABLE>;
        THE DERIVATIVE  DY/DX;
        EXIT : THE VALUE OF DY/DX AT X = B;
ZA:     <ARITHMETIC EXPRESSION>;
        ENTRY : THE VALUE OF DY/DX AT X=A;
FXY:    <ARITHMETIC EXPRESSION>;
        AN EXPRESSION,DEPENDING ON X AND Y ,GIVING THE VALUE OF
        (D/DX)(D/DX)Y;
E:     <ARRAY IDENTIFIER>;
        "ARRAY" E[1:4];
        E[1] AND E[3] ARE USED AS RELATIVE TOLERANCES,
        E[2] AND E[4] ARE USED AS ABSOLUTE TOLERANCES
        FOR Y AND DY/DX, RESPECTIVELY;
D:      <ARRAY IDENTIFIER>;
        "ARRAY" D[1:5];
        EXIT:
        ENTIER(D[1]+.5) IS THE NUMBER OF STEPS SKIPPED;
        D[2] IS THE LAST STEP LENGTH USED;
        D[3] IS EQUAL TO B;
        D[4] IS EQUAL TO Y(B);
        D[5] IS EQUAL TO DY/DX FOR X=B;
FI:     <BOOLEAN EXPRESSION>;
        IF FI="TRUE" THEN THE INTEGRATION STARTS AT X=A WITH A TRIAL
        STEP B-A;IF FI="FALSE" THEN THE INTEGRATION IS CONTINUED
        VIZ.  WITH THE INITIAL CONDITIONS X=D[3], Y=D[4], Z=D[5] AND
        STEP LENGTH  H=D[2]*SIGN(B-D[3]); A,YA,ZA ARE IGNORED.
```

DATA AND RESULTS:

RK3 INTEGRATES (D/DX)(D/DX)Y = F(X,Y) FROM X TO B,WITH IF FI="TRUE"
THEN X=A, Y=YA,DY/DX=ZA ELSE X=D[3], Y=D[4], Z=D[5].
A 5-TH ORDER RUNGE-KUTTA METHOD IS USED.
UPON COMPLETION OF A CALL OF RK3 WE HAVE X=D[3]=B, Y=D[4]=Y[B],
Z=D[5], I.E. THE VALUE OF DY/DX FOR X=B.
RK3 USES AS ITS MINIMAL ABSOLUTE STEP LENGTH
HMIN=MIN (E[2*J-1]*INT+E[2*J]) WITH 1<=J<=2 AND INT=
ABS(B-("IF" FI "THEN" A "ELSE" D[3])).
IF A STEP OF LENGTH ABS(H)<=HMIN IS REJECTED ,A STEP SIGN(H)*HMIN
IS SKIPPED. A STEP IS REJECTED IF THE ABSOLUTE VALUE OF THE LAST
TERM TAKEN INTO ACCOUNT IS GREATER THEN (ABS(DY/DX)*E[1]+E[2])*
ABS(H)/INT OR IF THAT TERM IS GREATER THEN (ABS(FXY)*E[3]+E[4])*
ABS(H)/INT (INT = ABS(B - A)).
SEE REF[1].

PROCEDURES USED: NONE.

RUNNING TIME:DEPENDS STRONGLY ON THE DIFFERENTIAL EQUATION TO SOLVE.

LANGUAGE:ALGOL 60.

METHOD AND PERFORMANCE:SEE REF[1].

REFERENCES:

[1]J. A.ZONNEVELD.
AUTOMATIC NUMERICAL INTEGRATION.
MATHEMATICAL CENTRE TRACT 8 (1970).

EXAMPLE OF USE:

```
"BEGIN" "COMMENT" SOLUTION OF Y"=X*Y,Y(0)=0,Y'(0)=1;
"PROCEDURE" RK3(X,A,B,Y,YA,Z,ZA,FXY,E,D,FI); "CODE" 33014;

"REAL" "PROCEDURE" YEXACT(X);"VALUE" X;"REAL" X;
"BEGIN" "INTEGER" N;"REAL" X3,S,TERM;
    X3:=X**3;TERM:=X;S:=0;
    "FOR" N:=3,N+3 "WHILE" ABS(TERM)>"-14 "DO"
    "BEGIN" S:=S+TERM;TERM:=TERM*X3/N/(N+1)
    "END";
    YEXACT:=S
"END";

"REAL" X,B,Y,Z;"BOOLEAN" FI;"ARRAY" D,E[1:5];
E[1]:=E[3]:="-8;E[2]:=E[4]:="-12;
"FOR" B:=.25,.50,.75,1.00 "DO"
"BEGIN" FI:=B<.30;
    RK3(X,0,B,Y,0,Z,1,X*Y,E,D,FI);
    OUTPUT(61,"("10B"("Y-YEXACT=")".10D"2D,5B"("X=")"Z.2D,
    5B"("Y=")"2D.10D//")",Y-YEXACT(X),X,Y)
"END"
"END"
```

DELIVERS:

Y-YEXACT=0.0000000000 X= .25 Y=00.2503256420

Y-YEXACT=0.0000000000 X= .50 Y=00.5052238559

Y-YEXACT=0.0000000000 X= .75 Y=00.7766332813

Y-YEXACT=0.0000000000 X=1.00 Y=01.0853396481

SOURCE TEXT(S):

```
"CODE""" 33014 ;
   "PROCEDURE" RK3(X, A, B, Y, YA, Z, ZA, FXY, E, D, FI);
   "VALUE" B, FI; "REAL" X, A, B, Y, YA, Z, ZA, FXY; "BOOLEAN" FI;
   "ARRAY" E, D;
   "BEGIN" "REAL" E1, E2, E3, E4, XL, YL, ZL, H, INT, HMIN, HL,
      ABSH, K0, K1, K2, K3, K4, K5, DISCRY, DISCRZ, TOLY,
   TOLZ, MU, MU1, FHY, FHZ;
   "BOOLEAN" LAST, FIRST, REJECT;
   "IF" FI "THEN"
   "BEGIN" D[3]:= A; D[4]:= YA; D[5]:= ZA "END";
   D[1]:= 0; XL:= D[3]; YL:= D[4]; ZL:= D[5];
   "IF" FI "THEN" D[2]:= B - D[3]; ABSH:= H:= ABS(D[2]);
   "IF" B - XL < 0 "THEN" H:= - H; INT:= ABS(B - XL);
   HMIN:= INT * E[1] + E[2]; HL:= INT * E[3] + E[4];
   "IF" HL < HMIN "THEN" HMIN:= HL; E1:= E[1] / INT;
   E2: = E[2] / INT; E3:= E[3] / INT; E4:= E[4] / INT;
   FIRST:= REJECT:= "TRUE"; "IF" FI "THEN"
   "BEGIN" LAST:= "TRUE"; "GOTO" STEP "END";
TEST: ABSH:= ABS(H); "IF" ABSH < HMIN "THEN"
   "BEGIN" H:= "IF" H > 0 "THEN" HMIN "ELSE" - HMIN; ABSH:= HMIN
   "END";
   "IF" H >= B - XL "EQUIV" H >= 0 "THEN"
   "BEGIN" D[2]:= H; LAST:= "TRUE"; H:= B - XL;
      ABSH:= ABS(H)
   "END"
   "ELSE" LAST:= "FALSE";                                    "COMMENT"
```

```
;

    STEP; "IF" REJECT "THEN"
       "BEGIN" X:= XL; Y:= YL; K0:= FXY * H "END"
    "ELSE" K0:= K5 * H / HL; X:= XL + .27639320225002l * H;
    Y:= YL + (ZL * .2763932022 50021 + K0 *
    .038196601125011) * H; K1:= FXY * H;
    X:= XL + .72360 6797749979 * H;
    Y:= YL + (ZL * .723606797749979 + K1 * .26180
    3398874989) * H; K2:= FXY * H; X:= XL + H * .5;
    Y:= YL + (ZL * .5 + K0 * .046875 + K1 *
    .079824155839840 - K2 * .001699155839840) * H;
    K4:= FXY * H; X:= "IF" LAST "THEN" B "ELSE" XL + H;
    Y:= YL + (ZL + K0 * .309016994374947 + K2 *
    .190983005625053) * H; K3:= FXY * H;
    Y:= YL + (ZL + K0 * .083333333333333 + K1 *
    .301502832395825 + K2 * .115163834270842) * H;
    K5:= FXY * H;
    DISCRY:= ABS(( - K0 * .5 + K1 * 1.809016994374947 +
    K2 * .690983005625053 - K4 * 2) * H);
    DISCRZ:= ABS((K0 - K3) * 2 - (K1 + K2) * 10 + K4 *
    16 + K5 * 4); TOLY:= ABSH * (ABS(ZL) * E1 + E2);
    TOLZ:= ABS(K0) * E3 + ABSH * E4;
    REJECT:= DISCRY > TOLY "OR" DISCRZ > TOLZ;
    FHY:= DISCRY / TOLY; FHZ:= DISCRZ / TOLZ;
    "IF" FHZ > FHY "THEN" FHY:= FHZ;
    MU:= 1 / (1 + FHY) + .45; "IF" REJECT "THEN"
    "BEGIN" "IF" ABSH <= HMIN "THEN"
       "BEGIN" D[1]:= D[1] + 1; Y:= YL; Z:= ZL;
          FIRST:= "TRUE"; "GOTO" NEXT
       "END";
       H:= MU * H; "GOTO" TEST
    "END";
    "IF" FIRST "THEN"
    "BEGIN" FIRST:= "FALSE"; HL:= H; H:= MU * H; "GOTO" ACC
    "END";
    FHY:= MU * H / HL + MU - MU1; HL:= H; H:= FHY * H;
ACC; MU1:= MU;
    Z:= ZL + (K0 + K3) * .083333333333333 + (K1 + K2) *
    .416666666666667;
NEXT; "IF" B ^= X "THEN"
    "BEGIN" XL:= X; YL:= Y; ZL:= Z; "GOTO" TEST "END";
    "IF" "NOT"LAST "THEN" D[2]:= H; D[3]:= X; D[4]:= Y; D[5]:= Z
"END" RK3;
       "EOP"
```

AUTHOR: J. A. ZONNEVELD.

CONTRIBUTORS: M. BAKKER AND I. BRINK.

INSTITUTE: MATHEMATICAL CENTRE.

RECEIVED: 730715.

BRIEF DESCRIPTION:

RK3N INTEGRATES THE VECTOR INITIAL PROBLEM
(D/DX) (D/DX) Y = F(X,Y), A <= X <= B OR B <= X <= A,
Y[J] (A) = YA[J], J = 1, .. ,N,
(D/DX) Y[J] (A) = ZA[J], J = 1, .. ,N.
A 5-TH ORDER RUNGE-KUTTA METHOD IS USED.

KEYWORDS:

RUNGE-KUTTA METHODS,
SECOND ORDER DIFFERENTIAL EQUATION,
INITIAL VALUE PROBLEM.

CALLING SEQUENCE:

THE HEADING OF THE PROCEDURE READS:

```
"PROCEDURE" RK3N(X,A,B,Y,YA,Z,ZA,FXYJ,J,E,D,FI,N);
"VALUE" B,FI,N;
"INTEGER" J,N;
"REAL" X,A,B,FXYJ;
"BOOLEAN" FI;
"ARRAY" Y,YA,Z,ZA,E,D;
```

```
      THE MEANING OF THE FORMAL PARAMETERS IS:
X:        <VARIABLE>;
          THE INDEPENDENT VARIABLE.
          UPON COMPLETION OF A CALL OF RK3N,
          IT IS EQUAL TO B;
A:        <ARITHMETIC EXPRESSION>;
          THE STARTING VALUE OF X;
B:        <ARITHMETIC EXPRESSION>;
          A VALUE PARAMETER,GIVING THE END VALUE OF X;
          B <= A IS ALLOWED.
Y:        <ARRAY IDENTIFIER>;
          "ARRAY" Y[1:N];
          THE VECTOR OF DEPENDENT VARIABLES;
          EXIT : THE VALUE OF Y[J](X) AT X = B;
YA:       <ARRAY IDENTIFIER>;
          "ARRAY" YA[1:N];
          ENTRY : THE STARTING VALUES OF Y[J],I.E. THE VALUES AT X=A;
Z:        <ARRAY IDENTIFIER>;
          "ARRAY" Z[1:N];
          THE DERIVATIVES OF THE DEPENDENT VARIABLES, Z[J] = DY[J]/DX;
          EXIT : THE VALUE OF Z[J](X) AT X = B;
ZA:       <ARRAY IDENTIFIER>;
          "ARRAY" ZA[1:N];
          ENTRY : THE STARTING VALUES OF Z[J],I.E. THE VALUES AT X=A;
FXYJ:     <ARITHMETIC EXPRESSION>;
          AN EXPRESSION DEPENDING ON X,Y[1],...,Y[N],J,
          GIVING THE VALUE OF (D/DX)(D/DX)Y[J];
J:        <VARIABLE>;
          A VARIABLE OF TYPE INTEGER,USED IN THE ACTUAL PARAMETER
          CORRESPONDING TO FXYJ,TO DENOTE THE NUMBER OF THE EQUATION
          REQUIRED (JENSEN'S DEVICE);
E:        <ARRAY IDENTIFIER>;
          "ARRAY" E[1:4*N];
          THE ELEMENT E[2*J-1] IS A RELATIVE AND E[2*J] IS AN ABSOLUTE
          TOLERANCE ASSOCIATED WITH Y[J];
          E[2*(N+J)-1] IS A RELATIVE AND E[2*(N+J)] IS AN ABSOLUTE
          TOLERANCE ASSOCIATED WITH Z[J];
D:        <ARRAY IDENTIFIER>;
          "ARRAY" D[1:2*N+3];
          EXIT:
          ENTIER(D[1]+.5) IS THE NUMBER OF STEPS SKIPPED;
          D[2] IS THE LAST STEP LENGTH USED;
          D[3] IS EQUAL TO B;
          D[4],...,D[N+3] ARE EQUAL TO Y[1],...,Y[N] FOR X=B;
          D[N+4],...,D[2*N+3] ARE EQUAL TO THE DERIVATIVES
          Z[1],...,Z[N] FOR X=B;
FI:       <BOOLEAN EXPRESSION>;
          IF FI="TRUE" THEN THE INTEGRATION STARTS AT A ,WITH A TRIAL
          STEP B-A;IF FI="FALSE" THEN THE INTEGRATION IS CONTINUED VIZ.
          WITH THE INITIAL CONDITIONS:X=D[3],Y[J]=D[J+3],Z[J]=D[N+J+3],
          AND STEP LENGTH H=D[2]*SIGN(B-D[3]); A,YA,ZA ARE IGNORED;
N:        <ARITHMETIC EXPRESSION>;
          THE NUMBER OF EQUATIONS.
```

DATA AND RESULTS:
RK3N INTEGRATES (D/DX)(D/DX)Y=F(X,Y) FROM X TO B,WITH,IF FI="TRUE" THEN X=A, Y[J]=YA[J], Z[J]=ZA[J].IF FI="FALSE" THEN X=D[3], Y[J]=D[J+3], Z[J]=D[N+3+J], USING A 5-TH ORDER RUNGE-KUTTA METHOD. UPON COMPLETION OF A CALL OF RK3N WE HAVE X=D[3]=B, Y[J]=D[J+3] THE VALUE OF THE DEPENDENT VARIABLES FOR X=B, Z[J]= D[N+3+J], THE VALUE OF THE DERIVATIVES OF Y[J] AT X=B.
RK3N USES AS ITS MINIMAL ABSOLUTE STEP LENGTH:
HMIN=MIN (E[2*J-1]*INT+E[2*J]) ,WITH 1<=J<=2*N AND INT= ABS(B-("IF" FI "THEN" A "ELSE" D[3])).
IF A STEP OF LENGTH ABS(H)<=HMIN IS REJECTED,A STEP SIGN(H)*HMIN IS SKIPPED.
A STEP IS REJECTED IF THE ABSOLUTE VALUE OF THE LAST TERM TAKEN INTO ACCOUNT IS GREATER THEN (ABS(Z[J])*E[2*J-1]+E[2*J])* ABS(H)/INT OR IF THAT TERM IS GREATER THEN (ABS(FXYJ)*E[2*(J+N)-1] +E[2*(J+N)])*ABS(H)/INT FOR ANY VALUE OF J, 1<=J<=N (INT=ABS(B-A)).
SEE REF[1].

PROCEDURES USED: NONE.

REQUIRED CENTRAL MEMORY:
EIGHT ARAYS OF ORDER N AND ONE OF ORDER 4 * N ARE USED.

RUNNING TIME:DEPENDS STRONGLY ON THE DIFFERENTIAL EQUATIONS TO BE SOLVED.

LANGUAGE:ALGOL 60.

METHOD AND PERFORMANCE:SEE REF[1].

REFERENCES:
[1]J.A.ZONNEVELD.
AUTOMATIC NUMERICAL INTEGRATION.
MATHEMATICAL CENTRE TRACT 8 (1970).

EXAMPLE OF USE:
THE SECOND ORDER (VECTOR) DIFFERENTIAL EQUATION

(D/DX)(D/DX)Y[1] = +Y[2],

(D/DX)(D/DX)Y[2] = -Y[1], X>=0,

Y[1] = Y[2] = 1,

(D/DX)Y[1] = (D/DX)Y[2] = 0, X = 0,

WHOSE EXACT SOLUTION IS GIVEN BY

```
Y[1]=COSH(X/SQRT(2))*COS(X/SQRT(2))+SINH(X/SQRT(2))*SIN(X/SQRT(2))
Y[2]=COSH(X/SQRT(2))*COS(X/SQRT(2))-SINH(X/SQRT(2))*SIN(X/SQRT(2))
```

CAN BE INTEGRATED BY RK3N BECAUSE THE SECOND DERIVATIVE IS NOT EXPRESSED IN THE FIRST. THE PROGRAM READS AS FOLLOWS:

```
"BEGIN" "INTEGER" K,B; "REAL" X; "BOOLEAN" FI;
    "ARRAY" Y,YA,Z[1:2],E[1:8],D[0:7];
    "INTEGER" "PROCEDURE" EVEN(N); "VALUE" N; "INTEGER" N;
    EVEN:= "IF" N//2 = N/2 "THEN" +1 "ELSE" -1;
"PROCEDURE" RK3N(X,A,B,Y,YA,Z,ZA,FXYJ,J,E,D,FI,N); "CODE"33015;
"PROCEDURE" EXACT(X,Y); "VALUE" X; "REAL" X; "ARRAY" Y;
"BEGIN" "INTEGER" I,N; "REAL" X2,TERM;
    Y[1]:=Y[2]:=0; TERM:=1; X2:= X*X*.5;
    "FOR" N:=1, N+1 "WHILE" ABS(TERM)>"-14 "DO"
    "BEGIN" "FOR" I:=1,2 "DO"
        Y[I]:=Y[I] + TERM*EVEN((I+N-2)//2);
        TERM:= TERM*X2  /N/(N*2-1)
    "END"
"END";
"FOR" K:=1,2,3,4,5,6,7,8 "DO" E[K]:="-7; FI:= "TRJE";
Y[1]:=Y[2]:=1; Z[1]:=Z[2]:=0; B:=0; AA: B:= B+1;
RK3N(X,0,B,Y,Y,Z,Z,"IF"K=1"THEN"Y[2]"ELSE"-Y[1],K,E,D,FI,2);
EXACT(X,YA); OUTPUT(61,"("//10B
"("ABS(YEXACT[1]-Y[1])+ABS(YEXACT[2]-Y[2])=")".10D"2D")",
ABS(Y[1]-YA[1])+ABS(YA[2]-Y[2]) );
FI:="FALSE" ; "IF" B<5 "THEN" "GO TO" AA
    "END"
    RESULTS:
    FOR X=1,2,3,4,5 THE FOLLOWING ERRORS ARE NOTICED (E[K]="-7,
    K=1,. ..,8):
            ABS(YEXACT[1]-Y[1])+ABS(YEXACT[2]-Y[2])=.0000000005"00
            ABS(YEXACT[1]-Y[1])+ABS(YEXACT[2]-Y[2])=.0000000018"00
            ABS(YEXACT[1]-Y[1])+ABS(YEXACT[2]-Y[2])=.0000000046"00
            ABS(YEXACT[1]-Y[1])+ABS(YEXACT[2]-Y[2])=.0000000126"00
            ABS(YEXACT[1]-Y[1])+ABS(YEXACT[2]-Y[2])=.0000000293"00
```

SOURCE TEXT(S):

```
"CODE""" 33015 ;
   "PROCEDURE" RK3N(X, A, B, Y, YA, Z, ZA, FXYJ, J, E, D,
   FI, N); "VALUE" B, FI, N; "INTEGER" J, N; "REAL" X, A, B, FXYJ;
   "BOOLEAN" FI; "ARRAY" Y, YA, Z, ZA, E, D;
"BEGIN" "INTEGER" JJ;
   "REAL" XL, H, HMIN, INT, HL, ABSH, FHM, DISCRY, DISCRZ,
   TOLY, TOLZ, MU, MU1, FHY, FHZ;
   "BOOLEAN" LAST, FIRST, REJECT;
   "ARRAY" YL, ZL, K0, K1, K2, K3, K4, K5[1:N], EE[1:4 *
   N];
   "IF" FI "THEN"
   "BEGIN" D[3]:= A;
      "FOR" JJ:= 1 "STEP" 1 "UNTIL" N "DO"
      "BEGIN" D[JJ + 3]:= YA[JJ]; D[N + JJ + 3]:= ZA[JJ]
      "END"
   "END";                                                    "COMMENT"
```

```
;

      D[1]:= 0; XL:= D[3];
      "FOR" JJ:= 1 "STEP" 1 "UNTIL" N "DO"
      "BEGIN" YL[JJ]:= D[JJ + 3]; ZL[JJ]:= D[N + JJ + 3] "END";
      "IF" FI "THEN" D[2]:= B - D[3]; ABSH:= H:= ABS(D[2]);
      "IF" B - XL < 0 "THEN" H:= - H; INT:= ABS(B - XL);
      HMIN:= INT * E[1] + E[2];
      "FOR" JJ:= 2 "STEP" 1 "UNTIL" 2 * N "DO"
      "BEGIN" HL:= INT * E[2 * JJ - 1] + E[2 * JJ];
         "IF" HL < HMIN "THEN" HMIN:= HL
      "END";
      "FOR" JJ:= 1 "STEP" 1 "UNTIL" 4 * N "DO" EE[JJ]:= E[JJ] / INT;
      FIRST:= REJECT:= "TRUE"; "IF" FI "THEN"
      "BEGIN" LAST:= "TRUE"; "GOTO" STEP "END";
TEST: ABSH:= ABS(H); "IF" ABSH < HMIN "THEN"
      "BEGIN" H:= "IF" H > 0 "THEN" HMIN "ELSE" - HMIN; ABSH:= HMIN
      "END";
      "IF" H >= B - XL "EQUIV" H >= 0 "THEN"
      "BEGIN" D[2]:= H; LAST:= "TRUE"; H:= B - XL;
         ABSH:= ABS(H)
      "END"
      "ELSE" LAST:= "FALSE";
STEP; "IF" REJECT "THEN"
      "BEGIN" X:= XL;
         "FOR" JJ:= 1 "STEP" 1 "UNTIL" N "DO" Y[JJ]:= YL[JJ];
         "FOR" J:= 1 "STEP" 1 "UNTIL" N "DO" K0[J]:= FXYJ * H
      "END"
      "ELSE"
      "BEGIN" FHY:= H / HL;
         "FOR" JJ:= 1 "STEP" 1 "UNTIL" N "DO" K0[JJ]:= K5[JJ] * FHY
      "END";
      X:= XL + .27639 3202250021 * H;
      "FOR" JJ:= 1 "STEP" 1 "UNTIL" N "DO" Y[JJ]:= YL[JJ] + (ZL[JJ]
      * .2763932022500 21 + K0[JJ] * .038196601125011) * H;
      "FOR" J:= 1 "STEP" 1 "UNTIL" N "DO" K1[J]:= FXYJ * H;
      X:= XL + .72360679774997 9 * H;
      "FOR" JJ:= 1 "STEP" 1 "UNTIL" N "DO" Y[JJ]:= YL[JJ] + (ZL[JJ]
      * .7236067977499 79 + K1[JJ] * .261803398874989) * H;
      "FOR" J:= 1 "STEP" 1 "UNTIL" N "DO" K2[J]:= FXYJ * H;
      X:= XL + H * .5;
      "FOR" JJ:= 1 "STEP" 1 "UNTIL" N "DO" Y[JJ]:= YL[JJ] + (ZL[JJ]
      * .5 + K0[JJ] * .046875 + K1[JJ] * .079824155839840
      - K2[JJ] * .00169 91558 39840) * H;
      "FOR" J:= 1 "STEP" 1 "UNTIL" N "DO" K4[J]:= FXYJ * H;
      X:= "IF" LAST "THEN" B "ELSE" XL + H;
      "FOR" JJ:= 1 "STEP" 1 "UNTIL" N "DO" Y[JJ]:= YL[JJ] + (ZL[JJ]
      + K0[JJ] * .309016994374947 + K2[JJ] *
      .190983005625053) * H;                                    "COMMENT"
```

```
;

    "FOR" J:= 1 "STEP" 1 "UNTIL" N "DO" K3[J]:= FXYJ * H;
    "FOR" JJ:= 1 "STEP" 1 "UNTIL" N "DO" Y[JJ]:= YL[JJ] + (ZL[JJ]
    + K0[JJ] * .083333333333333 + K1[JJ] * .30150
    2832395825 + K2[JJ] * .115163834270842) * H;
    "FOR" J:= 1 "STEP" 1 "UNTIL" N "DO" K5[J]:= FXYJ * H;
    REJECT:= "FALSE"; FHM:= 0;
    "FOR" JJ:= 1 "STEP" 1 "UNTIL" N "DO"
    "BEGIN" DISCRY:= ABS(( - K0[JJ] * .5 + K1[JJ] *
       1.809016994374947 + K2[JJ] * .690983005625053 -
       K4[JJ] * 2) * H);
       DISCRZ:= ABS((K0[JJ] - K3[JJ]) * 2 - (K1[JJ] +
       K2[JJ]) * 10 + K4[JJ] * 16 + K5[JJ] * 4);
       TOLY:= ABSH * (ABS(ZL[JJ]) * EE[2 * JJ - 1] +
       EE[2 * JJ]);
       TOLZ:= ABS(K0[JJ]) * EE[2 * (JJ + N) - 1] + ABSH
       * EE[2 * (JJ + N)];
       REJECT:= DISCRY > TOLY "OR" DISCRZ > TOLZ "OR" REJECT;
       FHY:= DISCRY / TOLY; FHZ:= DISCRZ / TOLZ;
       "IF" FHZ > FHY "THEN" FHY:= FHZ;
       "IF" FHY > FHM "THEN" FHM:= FHY
    "END";
    MU:= 1 / (1 + FHM) + .45; "IF" REJECT "THEN"
       "BEGIN" "IF" ABSH <= HMIN "THEN"
          "BEGIN" D[1]:= D[1] + 1;
             "FOR" JJ:= 1 "STEP" 1 "UNTIL" N "DO"
             "BEGIN" Y[JJ]:= YL[JJ]; Z[JJ]:= ZL[JJ] "END";
             FIRST:= "TRUE"; "GOTO" NEXT
          "END";
          H:= MU * H; "GOTO" TEST
       "END" REJ;
       "IF" FIRST "THEN"
       "BEGIN" FIRST:= "FALSE"; HL:= H; H:= MU * H; "GOTO" ACC
       "END";
       FHY:= MU * H / HL + MU - MU1; HL:= H; H:= FHY * H;
ACC:   MU1:= MU;
       "FOR" JJ:= 1 "STEP" 1 "UNTIL" N "DO" Z[JJ]:= ZL[JJ] + (K0[JJ]
       + K3[JJ]) * .083333333333333 + (K1[JJ] + K2[JJ]) *
       .416666666666667;
NEXT:  "IF" B ^= X "THEN"
       "BEGIN" XL:= X;
          "FOR" JJ:= 1 "STEP" 1 "UNTIL" N "DO"
          "BEGIN" YL[JJ]:= Y[JJ]; ZL[JJ]:= Z[JJ] "END";
          "GOTO" TEST
       "END";
       "IF" "NOT"LAST "THEN" D[2]:= H; D[3]:= X;
       "FOR" JJ:= 1 "STEP" 1 "UNTIL" N "DO"
       "BEGIN" D[JJ + 3]:= Y[JJ]; D[N + JJ + 3]:= Z[JJ] "END"
"END" RK3N;
      "EOP"
```

AUTHORS: P.A. BEENTJES, H.G.J. ROZENHART.

INSTITUTE: MATHEMATICAL CENTRE.

RECEIVED: 760201

BRIEF DESCRIPTION:

ARKMAT SOLVES AN INITIAL VALUE PROBLEM, GIVEN AS A SYSTEM OF FIRST ORDER (NON-LINEAR) DIFFERENTIAL EQUATIONS BY MEANS OF A STABILIZED RUNGE KUTTA METHOD;
IN PARTICULAR THIS PROCEDURE IS SUITABLE FOR THE INTEGRATION OF SYSTEMS WHERE THE DEPENDENT VARIABLE AND THE RIGHTHAND SIDE ARE STORED IN A RECTANGULAR ARRAY INSTEAD OF A VECTOR , I.E. DU / DT = F(T, U), WHERE U AND F ARE (N * M) MATRICES (SEE METHOD AND PERFORMANCE).

KEYWORDS:

MATRIX DIFFERENTIAL EQUATIONS,
INITIAL VALUE PROBLEMS,
EXPLICIT ONE-STEP METHODS,
STABILIZED RUNGE KUTTA METHODS.

CALLING SEQUENCE:

THE DECLARATION OF THE PROCEDURE IN THE CALLING PROGRAM READS;

```
"PROCEDURE" ARKMAT(T, TE, M, N, U, DER, TYPE, ORDER, SPR, OUT);
"VALUE" M, N, TYPE, ORDER; "INTEGER" M, N, TYPE, ORDER;
"REAL" T, TE, SPR; "ARRAY" U; "PROCEDURE" DER, OUT;
"CODE" 33066;
```

THE MEANING OF THE FORMAL PARAMETERS IS
T: «VARIABLE»;
THE INDEPENDENT VARIABLE T; IT MIGHT BE USED IN DER;
ENTRY: THE INITIAL VALUE TO;
EXIT : THE FINAL VALUE TE;

TE: «ARITHMETIC EXPRESSION»;
ENTRY: THE FINAL VALUE OF T;

M: «ARITHMETIC EXPRESSION»;
NUMBER OF COLUMNS OF U;

N: «ARITHMETIC EXPRESSION»;
NUMBER OF ROWS OF U;

U: «ARRAY IDENTIFIER»;
"ARRAY" U[1:N,1:M];
ENTRY: THE INITIAL VALUES OF THE SOLUTION OF THE SYSTEM OF DIFFERENTIAL EQUATIONS AT T=TO;
EXIT : THE VALUES OF THE SOLUTION AT T=TE;

DER: «PROCEDURE IDENTIFIER»;
THE HEADING OF THIS PROCEDURE READS:
"PROCEDURE" DER(T, V, FTV); "VALUE" T;
"REAL" T; "ARRAY" V, FTV;
THIS PROCEDURE MUST BE GIVEN BY THE USER AND PERFORMS AN EVALUATION OF THE RIGHTHAND SIDE F(T, V) OF THE SYSTEM; UPON COMPLETION OF DER,THE RIGHTHAND SIDE SHOULD BE STORED IN FTV[1:N,1:M];

TYPE: «VARIABLE»;
ENTRY: THE TYPE OF THE SYSTEM OF DIFFERENTIAL EQUATIONS TO BE SOLVED;
THE USER SHOULD SUPPLY ONE OF THE FOLLOWING VALUES;
1: IF NO SPECIFICATION OF THE TYPE CAN BE MADE;
2: IF THE EIGENVALUES OF THE JACOBIAN MATRIX OF THE RIGHTHAND SIDE ARE NEGATIVE REAL;
3: IF THE EIGENVALUES OF THE JACOBIAN MATRIX OF THE RIGHTHAND SIDE ARE PURELY IMAGINARY;

ORDER: «VARIABLE»;
THE ORDER OF THE RUNGE KUTTA METHOD USED;
ENTRY: FOR TYPE=2 THE USER MAY CHOOSE ORDER=1 OR ORDER=2;
ORDER SHOULD BE 2 FOR THE OTHER TYPES;
EXIT : IF ORDER IS SET TO ANOTHER VALUE,IT IS ASSUMED TO BE (IF TYPE=2 "THEN" 1 "ELSE" 2);

SPR: «ARITHMETIC EXPRESSION»;
ENTRY: THE SPECTRAL RADIUS OF THE JACOBIAN MATRIX OF THE RIGHTHAND SIDE, WHEN THE SYSTEM IS WRITTEN IN ONE DIMENSIONAL FORM (I.E. VECTORFORM);
THE INTEGRATION STEP WILL EQUAL CONSTANT/SPR (SEE DATA AND RESULTS);
IF NECESSARY SPR CAN BE UPDATED (AFTER EACH STEP) BY MEANS OF THE PROCEDURE OUT;

OUT: «PROCEDURE IDENTIFIER»
THE HEADING OF THIS PROCEDURE READS:
"PROCEDURE" OUT;
AFTER EACH INTEGRATION STEP PERFORMED, INFORMATION CAN BE OBTAINED OR UPDATED BY THIS PROCEDURE, E.G. THE VALUES OF T, U[1:N,1:M] AND SPR.

DATA AND RESULTS:

IF THE USER WANTS TO PERFORM THE INTEGRATION WITH A PRESCRIBED STEP H, HE HAS TO GIVE SPR THE VALUE CONSTANT/H, WHERE CONSTANT HAS THE FOLLOWING VALUES:
CONSTANT= 4.3 IF TYPE=1 AND ORDER=2;
CONSTANT= 156 IF TYPE=2 AND ORDER=1;
CONSTANT= 64 IF TYPE=2 AND ORDER=2;
CONSTANT= 8 IF TYPE=3 AND ORDER=2;

PROCEDURES USED:

ELMCOL = CP34023.
DUPMAT = CP31035.

REQUIRED CENTRAL MEMORY:

TWO AUXILIARY ARRAYS OF ORDER N*M ARE DECLARED.

RUNNING TIME:

THE PROCEDURE DER IS CALLED 9 TIMES PER STEP.

LANGUAGE: ALGOL60.

METHOD AND PERFORMANCE:

ARKMAT IS AN IMPLEMENTATION OF LOW ORDER STABILIZED RUNGE KUTTA METHODS (SEE REFERENCE[1]);
THE INTEGRATION STEPSIZE USED WILL DEPEND ON:
1. THE TYPE OF SYSTEM TO BE SOLVED (I.E. HYPERBOLIC OR PARABOLIC);
2. THE SPECTRAL RADIUS OF THE JACOBIAN MATRIX OF THE SYSTEM;
3. THE INDICATED ORDER OF THE PARTICULAR RUNGE KUTTA METHOD;
THE PROCEDURE ARKMAT IS ESPECIALLY INTENDED FOR SYSTEMS OF DIFFERENTIAL EQUATIONS ARISING FROM INITIAL BOUNDARY VALUE PROBLEMS IN TWO DIMENSIONS, E.G. WHEN THE METHOD OF LINES IS APPLIED TO THIS KIND OF PROBLEMS, THE RIGHTHAND SIDE OF THE RESULTING SYSTEM IS MUCH EASIER TO DESCRIBE IN MATRIX THAN IN VECTOR FORM; BECAUSE OF THIS FACT THE ARRAY OF DEPENDENT VARIABLES U IS A MATRIX, RATHER THAN A VECTOR.

REFERENCE:

[1]. P.J. VAN DER HOUWEN,
STABILIZED RUNGE KUTTA METHOD WITH LIMITED STORAGE REQUIREMENTS.
MATH. CENTR. REPORT TW 124/71.

EXAMPLE OF USE:

GIVEN THE FOLLOWING SYSTEM OF EQUATIONS:

DU / DT = V(T, X, Y),
(1)
DV / DT = D(DU / DX) / DX + D(DU / DY) / DY,

(ORIGINATING FROM THE INITIAL BOUNDARY VALUE PROBLEM
D(DU / DT) / DT = D(DU / DX) / DX + D(DU / DY) / DY,
ON THE DOMAIN 0 <= X <= PI , 0 <= Y <= 1),

WITH THE FOLLOWING BOUNDARY CONDITIONS:

U(T, 0, Y) = U(T, PI, Y) = U(T, X, 1) = 0,
U(T, X, 0) = SIN(X) * COS(SQRT(1 + PI * PI / 4) * T),

AND THE INITIAL VALUES:

U(0, X, Y) = SIN(X) * COS(PI * Y / 2),
V(0, X, Y) = 0;

BY APPLYING THE METHOD OF LINES TO PROBLEM (1), USING A TEN BY TEN GRID ON THE INDICATED DOMAIN, THE SYSTEM IS TRANSFORMED TO A MATRIX -DIFFERENTIAL EQUATION; THE SOLUTION OF THE LATTER PROBLEM AT T=1 IS COMPUTED BY THE FOLLOWING PROGRAM, USING A CONSTANT STEPSIZE .1;

```
"BEGIN" "REAL" HPI,H1,H2,H1K,H2K,T,TE;
   "INTEGER" I,J,N,M,TYP,ORDE,TEL;"ARRAY" U[1:20,1:10];
   "PROCEDURE" ARKMAT(T,TE,M,N,U,DER,TYPE,ORDER,SPR,OUT); "CODE" 33066;
   "PROCEDURE" INIMAT(LR,UR,LC,UC,A,X); "CODE" 31011;

   "PROCEDURE" DERIV(T,U,DU);"REAL" T;"ARRAY" U,DU;
   "BEGIN" "FOR" I:=2 "STEP" 1 "UNTIL" N-1 "DO"
      "FOR" J:=2 "STEP" 1 "UNTIL" M-1 "DO"
      "BEGIN" DU[I,J]:=U[I+N,J];
         DU[I+N,J]:=(U[I,J+1]-2*U[I,J]+U[I,J-1])/H1K+
                    (U[I+1,J]-2*U[I,J]+U[I-1,J])/H2K
      "END";
      "FOR" J:=1,M "DO"
      "BEGIN" INIMAT(N+1,N+N,J,J,DU,0);
      "FOR" I:=1 "STEP" 1 "UNTIL" N "DO" DU[I,J]:=U[N+1,J]
      "END";

      "FOR" I:=1,N "DO"
      "FOR" J:=2 "STEP" 1 "UNTIL" M-1 "DO"
      "BEGIN" DU[I,J]:=U[I+N,J];
         "IF" I=1 "THEN" DU[N+1,J]:=(U[1,J+1]-2*U[1,J]+U[1,J-1])/H1K+
                                    (2*U[2,J]-2*U[1,J])/H2K
                  "ELSE" DU[2*N,J]:=0
   "END"
"END" DERIV;
```

```
"PROCEDURE" OUT;
"BEGIN" TEL:=TEL+1;
   "IF" T=TE "THEN"
   "BEGIN" OUTPUT(61,"("//,3B,"("X")",7B,"("Y")",4B,
      "("U(1,X,Y)")",7B,"("U(1,X,Y)")",/,16B,"("COMPUTED")",7B,
      "("EXACT")",//")");
      OUTPUT(61,"("10(2(=D.3D2B),2(=D.6D6B),/)")",
      ((I=1)*H1,(I=1)*H2,U[I,I],SIN(H1*(I=1))*COS(HPI*H2*(I=1))*
      COS(T*SQRT(1+HPI*HPI)),I:=1:10));
      OUTPUT(61,"("/,"("NUMBER OF INTEGRATION STEPS: ")"
      ,ZZZD")",TEL);
      OUTPUT(61,"("//,"(" TYPE IS:")",ZD,"("  ORDER IS:")",
      ,ZD")",TYP,ORDE);
   "END";
"END"  OUT;

"PROCEDURE" START;
"BEGIN" "FOR" J:=1 "STEP" 1 "UNTIL" M "DO" U[N,J]:=SIN(H1*(J=1));
   "FOR" I:=1 "STEP" 1 "UNTIL" N "DO"
   "BEGIN" "REAL" COS1; COS1:=COS(H2*HPI*(I=1));
      "FOR" J:=1 "STEP" 1 "UNTIL" M "DO" U[I,J]:=U[N,J]*COS1
   "END"
      INIMAT(N+1,N+N,1,M,U,0)
   "END" START;

   HPI:=2*ARCTAN(1);H2:=1/9;H1:=(2*HPI)/9;N:=M:=10;
   H1K:=H1*H1;H2K:=H2*H2;TEL:=0;
   T:=0; TE:=1 ; START; TYP:=3; ORDE:=2;
   ARKMAT(T,TE,M,N+N,U,DERIV,TYP,ORDE,80,OUT)
"END"
```

THIS PROGRAM DELIVERS:

X	Y	U(1,X,Y) COMPUTED	U(1,X,Y) EXACT
0.000	0.000	0.000000	0.000000
0.349	0.111	-0.095201	-0.096735
0.698	0.222	-0.170723	-0.173474
1.047	0.333	-0.211983	-0.215398
1.396	0.444	-0.213228	-0.216663
1.745	0.556	-0.178920	-0.181802
2.094	0.667	-0.122388	-0.124360
2.443	0.778	-0.062138	-0.063139
2.793	0.889	-0.016787	-0.017057
3.142	1.000	0.000000	-0.000000

NUMBER OF INTEGRATION STEPS: 10

TYPE IS: 3 ORDER IS: 2

```
SOURCE TEXT(S);
"CODE" 33066;
"PROCEDURE" ARKMAT( T, TE, M, N, U, DER, TYPE, ORDER, SPR, OUT);
"VALUE" M,N,TYPE,ORDER;
"INTEGER" M,N,TYPE,ORDER;
"REAL" T,TE,SPR;
"ARRAY" U;
"PROCEDURE" DER,OUT;

"BEGIN" "INTEGER" SIG,L;
     "REAL" TAU;
     "ARRAY" LAMBDA[1:9],UH,DU[1:N,1:M];
     "BOOLEAN" LAST;

     "PROCEDURE" ELMCOL(L,U,I,J,A,B,X); "CODE" 34023;
     "PROCEDURE" DUPMAT(L,U,I,J,A,B);   "CODE" 31035;

     "PROCEDURE" ELMMAT(A,B,X); "VALUE" X; "ARRAY" A,B; "REAL" X;
     "FOR" L:=1 "STEP" 1 "UNTIL" M "DO" ELMCOL(1,N,L,L,A,B,X);

"PROCEDURE" INITIALIZE;
"BEGIN" "INTEGER" I;"REAL" LBD;
"SWITCH" TYPEODE:=NOTSPECIFIED2,PARABOLIC1,PARABOLIC2,HYPERBOLIC2;

"IF" TYPE"=2 "AND" TYPE"=3 "THEN" TYPE:=1;
"IF" TYPE"=2 "THEN" ORDER:=2 "ELSE" "IF" ORDER"=2 "THEN" ORDER:=1;
I:=1;
"GOTO" TYPEODE["IF" TYPE=1 "THEN" 1 "ELSE" TYPE+ORDER-1];

NOTSPECIFIED2: "FOR" LBD:=1/9,1/8,1/7,1/6,1/5,1/4,1/3,1/2,4.3 "DO"
               "BEGIN" LAMBDA[I]:=LBD; I:=I+1 "END";
               "GOTO" EXIT;

PARABOLIC1:    "FOR"LBD:=.1418519249"-2,.3404154076"-2,.0063118569
                        ,.0108279437 5,.0184273385 1,.0327850794 2,
                        .0653627415,.1691078577,156 "DO"
               "BEGIN" LAMBDA[I]:=LBD; I:=I+1 "END";
               "GOTO" EXIT;

PARABOLIC2:    "FOR" LBD:=.3534355908"-2,.8532600867"-2,.015956206
                        ,.0277222915 5,.0481258796 4,.0884868945 2,
                        .1863578961,.5,64 "DO"
               "BEGIN" LAMBDA[I]:=LBD; I:=I+1 "END";
               "GOTO" EXIT;

HYPERBOLIC2:   "FOR" LBD:=1/8,1/20,5/32,2/17,17/80,5/22,11/32,1/2,
                        8 "DO"
               "BEGIN" LAMBDA[I]:=LBD; I:=I+1 "END";
               "GOTO" EXIT;

                                                    "COMMENT"
```

```
      EXIT: SIG:=SIGN(TE-T)
   "END" INITIALIZE;

  "PROCEDURE" DIFFERENCE SCHEME;
  "BEGIN" "INTEGER" I;"REAL" MLT;

      DER(T,U,DU);
      "FOR" I:=1 "STEP" 1 "UNTIL" 8 "DO"
      "BEGIN" MLT:=LAMBDA[I]*TAU;
         DUPMAT(1,N,1,M,UH,U);
         ELMMAT(UH,DU,MLT);
         DER(T+MLT,UH,DU)
      "END";
      ELMMAT(U,DU,TAU);
      T:="IF" LAST "THEN" TE "ELSE" T+TAU;
   "END" DIFFERENCE SCHEME;

   INITIALIZE; LAST:="FALSE";

      STEP:
      TAU:=("IF" SPR=0 "THEN" ABS(TE-T) "ELSE" ABS(LAMBDA[9]/SPR))*SIG;
      "IF" T+TAU >= TE "EQUIV" TAU>=0 "THEN"
      "BEGIN" TAU:=TE-T;LAST:="TRUE" "END";
      DIFFERENCE SCHEME ; OUT;
      "IF" "NOT" LAST "THEN" "GOTO" STEP

"END" ARKMAT;
         "EOP"
```

AUTHOR: M. BAKKER.

INSTITUTE: MATHEMATICAL CENTRE, AMSTERDAM.

RECEIVED: 751231.

BRIEF DESCRIPTION:

THIS SECTION CONTAINS TWO PROCEDURES FOR THE SOLUTION OF SECOND ORDER SELF-ADJOINT LINEAR TWO POINT BOUNDARY VALUE PROBLEMS;

(1) FEM LAG SYM;

THIS PROCEDURE SOLVES THE DIFFERENTIAL EQUATION

$= (P(X)*Y')' + R(X)*Y = F(X),\ A \leq X \leq B,$

WITH BOUNDARY CONDITIONS

$E[1]*Y(A) + E[2]*Y'(A) = E[3],$

$E[4]*Y(B) + E[5]*Y'(B) = E[6].$

(2) FEM LAG;

THIS PROCEDURE SOLVES THE DIFFERENTIAL EQUATION

$= Y'' + R(X)*Y = F(X),\ A \leq X \leq B,$

WITH BOUNDARY CONDITIONS

$E[1]*Y(A) + E[2]*Y'(A) = E[3],$

$E[4]*Y(B) + E[5]*Y'(B) = E[6].$

KEY WORDS AND FRASES:

SECOND ORDER DIFFERENTIAL EQUATIONS,
TWO POINT BOUNDARY VALUE PROBLEMS,
SELF-ADJOINT BOUNDARY VALUE PROBLEMS,
RITZ-GALERKIN METHOD,
GLOBAL METHODS.

LANGUAGE: ALGOL 60.

REFERENCES:

[1] STRANG, G. AND G.J. FIX,
AN ANALYSIS OF THE FINITE ELEMENT METHOD,
PRENTICE-HALL, ENGLEWOOD CLIFFS, NEW JERSEY, 1973.

[2] BAKKER, M., EDITOR,
COLLOQUIUM ON DISCRETIZATION METHODS, CHAPTER 3 (DUTCH),
MATHEMATISCH CENTRUM, MC-SYLLABUS, TO APPEAR.

[3] HEMKER, P.W.,
GALERKIN'S METHOD AND LOBATTO POINTS,
MATHEMATISCH CENTRUM, REPORT 24/75 (1975).

[4] BABUSKA, I.,
NUMERICAL STABILITY IN PROBLEMS OF LINEAR ALGEBRA,
S.I.A.M. J. NUM. ANAL., VOL.9, P. 53-77 (1972).

SUBSECTION: FEM LAG SYM.

CALLING SEQUENCE:

THE HEADING OF THE PROCEDURE READS:

```
"PROCEDURE" FEM LAG SYM(X, Y, N, P, R, F, ORDER, E);
"VALUE" N, ORDER; "INTEGER" N, ORDER;
"ARRAY" X, Y, E;
"REAL" "PROCEDURE" P, R, F;
"CODE" 33300;
```

THE MEANING OF THE FORMAL PARAMETERS IS:

N: <ARITHMETIC EXPRESSION>;
THE UPPER BOUND OF THE ARRAYS X AND Y; N > 1;

X: <ARRAY IDENTIFIER>;
"ARRAY" X[0:N];
ENTRY: A = X[0] < X[1] < ... < X[N] = B IS A
PARTITION OF THE INTERVAL [A,B];

Y: <ARRAY IDENTIFIER>;
"ARRAY" Y[0:N];
EXIT: Y[I] (I = 0, 1, ... , N) IS THE APPROXIMATE
SOLUTION AT X[I] OF THE DIFFERENTIAL EQUATION

$$(1) \qquad = (P(X)*Y')' + R(X)*Y = F(X), \quad A < X < B,$$

WITH BOUNDARY CONDITIONS

$$(2) \qquad \begin{aligned} E[1]*Y(A) + E[2]*Y'(A) &= E[3], \\ E[4]*Y(B) + E[5]*Y'(B) &= E[6]; \end{aligned}$$

P: <PROCEDURE IDENTIFIER>;
THE HEADING OF P READS:
"REAL" "PROCEDURE" P(X); "VALUE" X; "REAL" X;
P(X) IS THE COEFFICIENT OF Y' IN (1);

R: <PROCEDURE IDENTIFIER>;
THE HEADING OF R READS:
"REAL" "PROCEDURE" R(X); "VALUE" X; "REAL" X;
R(X) IS THE COEFFICIENT OF Y IN (1);

F: <PROCEDURE IDENTIFIER>;
THE HEADING OF F READS:
"REAL" "PROCEDURE" F(X); "VALUE" X; "REAL" X;
F(X) IS THE RIGHT HAND SIDE OF (1);

ORDER: «ARITHMETIC EXPRESSION»;
ENTRY: ORDER DENOTES THE ORDER OF ACCURACY REQUIRED FOR THE APPROXIMATE SOLUTION OF (1)-(2); LET H = MAX(X[I] - X[I-1]); THEN ABS(Y[I] - Y(X[I])) «= C*H**ORDER, I = 0, ... , N; ORDER CAN BE CHOSEN EQUAL TO 2, 4 OR 6 ONLY;

E: «ARRAY IDENTIFIER»;
"ARRAY" E[1:6];
E[1], ... , E[6] DESCRIBE THE BOUNDARY CONDITIONS (2);
E[1] AND E[4] ARE NOT ALLOWWED TO VANISH BOTH.

PROCEDURES USED: NONE.

REQUIRED CENTRAL MEMORY:
FOUR AUXILIARY ARRAYS OF N REALS ARE USED.

RUNNING TIME:
LET K = ORDER/2; THEN

(A) K*N + 1 EVALUATIONS OF P(X), R(X) AND F(X) ARE NEEDED;

(B) ABOUT 17*2**(K-1)*N MULTIPLICATIONS/DIVISIONS ARE NEEDED.

DATA AND RESULTS:

THE PROCEDURE FEM LAG SYM HAS SOME RESTRICTIONS IN ITS USE:
(I) P(X) SHOULD BE POSITIVE ON THE CLOSED INTERVAL «X[J-1],X[J]»;
(II) P(X), R(X) AND F(X) ARE REQUIRED TO BE SUFFICIENTLY SMOOTH ON «X[0],X[N]» EXCEPT AT THE GRID POINTS WHERE P(X) SHOULD BE AT LEAST CONTINUOUS;
IN THAT CASE THE ORDER OF ACCURACY (2, 4, OR 6) IS PRESERVED;
(III) R(X) SHOULD BE NONNEGATIVE ON «X[0],X[N]»;
IF, HOWEVER, THE PROBLEM HAS PURE DIRICHLET BOUNDARY CONDITIONS (I.E. E[2] = E[5] = 0) THIS CONDITION CAN BE WEAKENED TO THE REQUIREMENT THAT

R(X) » - P0*(PI/(X[N] - X[0]))**2,

WHERE P0 IS THE MINIMUM OF P(X) ON «X[0],X[N]» AND PI HAS THE VALUE 3.14159...; HOWEVER, ONE SHOULD NOTE THAT THE PROBLEM MAY BE ILL-CONDITIONED WHEN R(X) IS QUITE NEAR THAT LOWER BOUND; FOR OTHER NEGATIVE VALUES OF R(X) THE EXISTENCE OF A SOLUTION REMAINS AN OPEN QUESTION;
(IV) THE USER SHOULD NOT EXPECT GREATER ACCURACY THAN 12 DECIMAL DUE TO THE LOSS OF DIGITS DURING THE EVALUATION OF THE MATRIX AND THE VECTOR OF THE LINEAR SYSTEM TO BE SOLVED AND DURING ITS REDUCTION TO A TRIDIAGONAL SYSTEM; WHEN THE SOLUTION OF THE PROBLEM IS NOT TOO WILD, THIS 12-DIGIT ACCURACY CAN ALREADY BE OBTAINED WITH A MODERATE MESH SIZE (E.G. « 0.1), PROVIDED THAT A SIXTH ORDER METHOD IS USED.

METHOD AND PERFORMANCE:

PROBLEM (1)-(2) IS SOLVED BY MEANS OF GALERKIN'S METHOD WITH CONTINUOUS PIECEWISE POLYNOMIALS (SEE [1], [2]); THE SOLUTION IS APPROXIMATED BY A FUNCTION WHICH IS CONTINUOUS ON THE CLOSED INTERVAL <X[0],X[N]> AND A POLYNOMIAL OF DEGREE LESS THAN OR EQUAL TO K (K = ORDER//2) ON EACH SEGMENT <X[J-1],X[J]> (J = 1, ..., N); THIS PIECEWISE POLYNOMIAL IS ENTIRELY DETERMINED BY THE VALUES IT HAS AT THE KNOTS X[J] AND ON (K-1) INTERIOR KNOTS ON EACH SEGMENT <X[J-1],X[J]>; THESE VALUES ARE OBTAINED BY THE SOLUTION OF AN (ORDER + 1)-DIAGONAL LINEAR SYSTEM WITH A SPECIALLY STRUCTURED MATRIX (SEE [2]); THE ENTRIES OF THE MATRIX AND THE VECTOR ARE INNER PRODUCTS WHICH ARE APPROXIMATED BY PIECEWISE (K+1)-POINT LOBATTO QUADRATURE (SEE [3]); THE EVALUATION OF THE MATRIX AND THE VECTOR IS DONE SEGMENT BY SEGMENT; ON EACH SEGMENT THE CONTRIBUTIONS TO THE ENTRIES OF THE MATRIX AND THE VECTOR ARE COMPUTED AND EMBEDDED IN THE GLOBAL MATRIX AND VECTOR; SINCE THE FUNCTION VALUES ON THE INTERIOR POINTS OF EACH SEGMENT ARE NOT COUPLED WITH THE FUNCTION VALUES OUTSIDE THAT SEGMENT, THE RESULTING LINEAR SYSTEM CAN BE REDUCED TO A TRIDIAGONAL SYSTEM BY MEANS OF STATIC CONDENSATION (SEE [2]); THE FINAL TRIDIAGONAL SYSTEM, SINCE IT IS OF FINITE DIFFERENCE TYPE, IS SOLVED BY MEANS OF BABUSKA'S METHOD (SEE [4]).

EXAMPLE OF USE:

WE SOLVE THE BOUNDARY VALUE PROBLEM

$$-(Y'*EXP(X))'+Y*COS(X)=EXP(X)*(SIN(X)-COS(X))+SIN(2*X)/2,$$
$$0 < X < PI = 3.14159265358979,\ Y(0) = Y(PI) = 0;$$

FOR THE BOUNDARY CONDITIONS THIS MEANS THAT

$$E[1] = E[4] = 1;\ E[2] = E[3] = E[5] = E[6] = 0;$$

THE ANALYTIC SOLUTION IS Y(X) = SIN(X); WE APPROXIMATE THE SOLUTION ON A UNIFORM GRID, I.E. X[I] = I*PI/N, I = 0, ..., N; WE CHOOSE N=10,20 AND COMPUTE FOR ORDER = 2,4,6 THE MAXIMUM ERROR; THE PROGRAM READS AS FOLLOWS:

```
"BEGIN" "INTEGER" N; "FOR" N:= 10, 20 "DO"
"BEGIN" "INTEGER" I, ORDER; "REAL" PI; "ARRAY" X, Y[0:N], E[1:6];
"REAL" "PROCEDURE" R(X); "VALUE" X; "REAL" X;
R:= COS(X);

"REAL" "PROCEDURE" P(X); "VALUE" X; "REAL" X;
P:= EXP(X);

"REAL" "PROCEDURE" F(X); "VALUE" X; "REAL" X;
F:= EXP(X)*(SIN(X)-COS(X)) + SIN(2*X)/2;

"PROCEDURE" FEM LAG SYM(X, Y, N, P, R, F, ORDER, E);
"CODE" 33300;
E[1]:= E[4]:= 1; E[2]:= E[3]:= E[5]:= E[6]:= 0;
PI:= 3.1415926535897 9;
"FOR" I:= 0 "STEP" 1 "UNTIL" N "DO" X[I]:= PI*I/N;
OUTPUT(61,"("//,6B"("N="")"D")",N);
"FOR" ORDER:= 2, 4, 6 "DO"
    "BEGIN" "REAL" RHO, D;
        FEM LAG SYM(X, Y, N, P, R, F, ORDER, E);
        RHO:= 0;
        "FOR" I:= 0 "STEP" 1 "UNTIL" N "DO"
        "BEGIN" D:= ABS(Y[I] - SIN(X[I]));
            "IF" RHO < D "THEN" RHO:= D
        "END";
        OUTPUT(61,"("/,16B"("ORDER="")"DD,4B"("MAX.ERROR= ")",
        D.DD"+ZD")",ORDER,RHO)
    "END"
"END"
"END"
```

RESULTS:

```
N=10
            ORDER=2   MAX. ERROR= 1.36" -2
            ORDER=4   MAX. ERROR= 7.55" -5
            ORDER=6   MAX. ERROR= 3.48" -8

N=20
            ORDER=2   MAX. ERROR= 3.41" -3
            ORDER=4   MAX. ERROR= 4.79" -6
            ORDER=6   MAX. ERROR= 5.51"-10
```

ONE OBSERVES THAT THE MAXIMUM ERROR DECREASES BY ABOUT 2**(-ORDER) WHEN THE MESH SIZE IS HALVED.

SUBSECTION: FEM LAG.

CALLING SEQUENCE:

THE HEADING OF THE PROCEDURE READS:

```
"PROCEDURE" FEM LAG(X, Y, N, R, F, ORDER, E);
"VALUE" N, ORDER; "INTEGER" N, ORDER;
"ARRAY" X, Y, E;
"REAL" "PROCEDURE" R, F;
"CODE" 33301;
```

THE MEANING OF THE FORMAL PARAMETERS IS:

N: <ARITHMETIC EXPRESSION>;
THE UPPER BOUND OF THE ARRAYS X AND Y; N > 1;

X: <ARRAY IDENTIFIER>;
"ARRAY" X[0:N];
ENTRY: A = X[0] < X[1] < ... < X[N] = B IS A
PARTITION OF THE SEGMENT [A,B];

Y: <ARRAY IDENTIFIER>;
"ARRAY" Y[0:N];
EXIT: Y[I] (I = 0, 1, ... , N) IS THE APPROXIMATE
SOLUTION AT X[I] OF THE DIFFERENTIAL EQUATION

(3) - Y''+ R(X)*Y = F(X), A < X < B,

WITH BOUNDARY CONDITIONS
(4) E[1]*Y(A) + E[2]*Y'(A) = E[3],
E[4]*Y(B) + E[5]*Y'(B) = E[6];

R: <PROCEDURE IDENTIFIER>;
THE HEADING OF R READS:
"REAL" "PROCEDURE" R(X); "VALUE" X; "REAL" X;
R(X) IS THE COEFFICIENT OF Y IN (3);

F: <PROCEDURE IDENTIFIER>;
THE HEADING OF F READS:
"REAL" "PROCEDURE" F(X); "VALUE" X; "REAL" X;
F(X) IS THE RIGHT HAND SIDE OF (3);

ORDER: <ARITHMETIC <EXPRESSION>>;
ENTRY: ORDER DENOTES THE ORDER OF ACCURACY REQUIRED FOR THE
APPROXIMATE SOLUTION OF (3)-(4); LET H = MAX(X[I] - X[I-1]);
THEN ABS(Y[I] - Y(X[I])) <= C*H**ORDER, I = 0, ... , N;
ORDER CAN CAN BE CHOSEN EQUAL TO 2, 4 OR 6 ONLY;

E: <ARRAY IDENTIFIER>;
"ARRAY" E[1:6];
E[1], ... , E[6] DESCRIBE THE BOUNDARY CONDITIONS (4);
E[1] AND E[4] ARE NOT ALLOWED TO VANISH BOTH.

PROCEDURES USED: NONE.

REQUIRED CENTRAL MEMORY:

FOUR AUXILIARY ARRAYS OF N REALS ARE USED.

RUNNING TIME:

LET K = ORDER/2; THEN

(A) K*N + 1 EVALUATIONS OF R(X) AND F(X) ARE NEEDED;

(B) ABOUT 12*2**(K-1)*N MULTIPLICATIONS/DIVISIONS ARE NEEDED.

DATA AND RESULTS: SEE PREVIOUS SUBSECTION.

METHOD AND PERFORMANCE: SEE PREVIOUS SUBSECTION.

EXAMPLE OF USE:

WE SOLVE THE BOUNDARY VALUE PROBLEM

= Y'' + Y*EXP(X) = SIN(X)*(1+EXP(X),
0 < X < PI = 3.1415926535897, Y(0) = Y(PI) = 0;

FOR THE BOUNDARY CONDITIONS THIS MEANS THAT

E[1] = E[4] = 1; E[2] = E[3] = E[5] = E[6] = 0;

THE ANALYTIC SOLUTION IS Y(X) = SIN(X); WE APPROXIMATE
THE SOLUTION ON A UNIFORM GRID, I.E. X[I] = I*PI/N,
I = 0, ..., N; WE CHOOSE N=10,20 AND COMPUTE FOR ORDER = 2,4,6
THE MAXIMUM ERROR; THE PROGRAM READS AS FOLLOWS:

```
"BEGIN" "INTEGER" N; "FOR" N:= 10, 20 "DO"
"BEGIN" "INTEGER" I, ORDER; "REAL" PI; "ARRAY" X, Y[0:N], E[1:6];
"REAL" "PROCEDURE" R(X); "VALUE" X; "REAL" X;
R:= EXP(X);

"REAL" "PROCEDURE" F(X); "VALUE" X; "REAL" X;
F:= SIN(X)*(1 + EXP(X));

"PROCEDURE" FEM LAG(X, Y, N, R, F, ORDER, E);
"CODE" 33301;
E[1]:= E[4]:= 1; E[2]:= E[3]:= E[5]:= E[6]:= 0;
PI:= 3.14159265358979;
"FOR" I:= 0 "STEP" 1 "UNTIL" N "DO" X[I]:= PI*I/N;
OUTPUT(61,"("//,6B"("N=")"D")",N);
"FOR" ORDER:= 2, 4, 6 "DO"
"BEGIN" "REAL" RHO, D;
    FEM LAG(X, Y, N, R, F, ORDER, E);
    RHO:= 0;
    "FOR" I:= 0 "STEP" 1 "UNTIL" N "DO"
        "BEGIN" D:= ABS(Y[I] - SIN(X[I]));
            "IF" RHO < D "THEN" RHO:= D
        "END";
        OUTPUT(61,"("/,16B"("ORDER=")"DD,4B"("MAX.ERROR= ")",
        D.DD"+ZD")",ORDER,RHO)
    "END"
"END"
"END"
```

RESULTS:

```
N=10
            ORDER=2     MAX. ERROR= 1.60" -3
            ORDER=4     MAX. ERROR= 1.55" -5
            ORDER=6     MAX. ERROR= 7.28"-10


N=20
            ORDER=2     MAX. ERROR= 4.01" -4
            ORDER=4     MAX. ERROR= 9.80" -7
            ORDER=6     MAX. ERROR= 9.38"-12
```

NOTICE THAT THE MAXIMUM ERROR DECREASES BY ABOUT
2**(-ORDER) WHEN THE MESH SIZE IS HALVED.

```
 SOURCE TEXT(S);

"CODE" 33300;
 "PROCEDURE" FEM LAG SYM(X, Y, N, P, R, F, ORDER, E);
 "INTEGER" N, ORDER;
"REAL" "PROCEDURE" P, R, F;
"ARRAY" X, Y, E;
"BEGIN" "INTEGER" L, L1;
  "REAL" XL1, XL, H, A12, B1, B2, TAU1, TAU2, CH, TL, G, YL, PP,
       P1, P2, P3, P4, R1, R2, R3, R4, F1, F2, F3, F4,
       E1, E2, E3, E4, E5, E6;
  "ARRAY" T, SUB, CHI, GI[0:N-1];

  "PROCEDURE" ELEMENT MAT VEC EVALUATION 1;
  "BEGIN" "REAL"  H2;
       "IF" L=1 "THEN"
       "BEGIN" P2:= P(XL1); R2:= R(XL1); F2:= F(XL1) "END";
       P1:= P2; P2:= P(XL); R1:= R2; R2:= R(XL); F1:= F2; F2:= F(XL);
       H2:= H/2; B1:= H2*F1; B2:= H2*F2; TAU1:= H2*R1; TAU2:= H2*R2;
       A12:= -0.5*(P1 + P2)/H
  "END" ELAN. M.V. EV.;

"PROCEDURE" ELEMENT MAT VEC EVALUATION 2;
"BEGIN" "REAL"  X2, H6, H15, B3, TAU3, C12, C32, A13, A22, A23;
     "IF" L=1 "THEN"
     "BEGIN" P3:= P(XL1); R3:= R(XL1); F3:= F(XL1) "END";
     X2:= (XL1 + XL)/2; H6:= H/6; H15:= H/1.5;
     P1:= P3; P2:= P(X2); P3:= P(XL);
     R1:= R3; R2:= R(X2); R3:= R(XL);
     F1:= F3; F2:= F(X2); F3:= F(XL);
     B1:= H6*F1; B2:= H15*F2; B3:= H6*F3;
     TAU1:= H6*R1; TAU2:= H15*R2; TAU3:= H6*R3;
     A12:= -(2*P1 + P3/1.5)/H; A13:= (0.5*(P1 + P3) - P2/1.5)/H;
     A22:= (P1 + P3)/H/0.375 + TAU2; A23:= -(P1/3 + P3)*2/H;
     "COMMENT" STATIC CONDENSATION;
     C12:= - A12/A22; C32:= - A23/A22; A12:= A13 + C32*A12;
     B1:= B1 + C12*B2; B2:= B3 + C32*B2;
     TAU1:= TAU1 + C12*TAU2; TAU2:= TAU3 + C32*TAU2
"END" ELEMENT MAT VEC EVALUATION 2;

"PROCEDURE" ELEMENT MAT VEC EVALUATION 3;
"BEGIN" "REAL"  X2, X3, H12, H24, DET, C12, C13, C42, C43,
     A13, A14, A22, A23, A24, A33, A34, B3, B4, TAU3, TAU4;
     "IF" L=1 "THEN"
     "BEGIN" P4:= P(XL1); R4:= R(XL1); F4:= F(XL1) "END";
     X2:= XL1 + 0.2763932025*H; X3:= XL - X2 + XL1;
     H12:= H/12; H24:= H/2.4;
     P1:= P4; P2:= P(X2); P3:= P(X3); P4:= P(XL);
     R1:= R4; R2:= R(X2); R3:= R(X3); R4:= R(XL);
     F1:= F4; F2:= F(X2); F3:= F(X3); F4:= F(XL);
     B1:= H12*F1; B2:= H24*F2; B3:= H24*F3; B4:= H12*F4;
     TAU1:= H12*R1; TAU2:= H24*R2; TAU3:= H24*R3; TAU4:= H12*R4;
                                                  "COMMENT"
```

```
A12:= -(+ 4.045084971874 50*P1
        + 0.575819171354 25*P3
        + 0.257514161979 11*P4)/H;
A13:= (+ 1.545084971874 7*P1
       - 1.507514161979 1*P2
       + 0.674180828645 8*P4)/H;
A14:= ((P2 + P3)/2.4 - (P1 + P4)/2)/H;
A22:= (5.454237476562*P1 + P3/.48 +.795762523437 62*P4)/H + TAU2;
A23:= - (P1 + P4)/(H*0.48);
A24:= (+ 0.674180828645 75*P1
       - 1.507514161979 10*P3
       + 1.545084971874 70*P4)/H;
A33:= (.795762523437 6*P1 + P2/.48 + 5.454237476562*P4)/H + TAU3;
A34:= -(+ 0.257514161979 11*P1
        + 0.575819171354 18*P2
        + 4.045084971874 7*P4)/H;
"COMMENT" STATIC CONDENSATION;
DET:= A22*A33 - A23*A23;
C12:= (A13*A23 - A12*A33)/DET;
C13:= (A12*A23 - A13*A22)/DET;
C42:= (A23*A34 - A24*A33)/DET;
C43:= (A24*A23 - A34*A22)/DET;
TAU1:= TAU1 + C12*TAU2 + C13*TAU3;
    TAU2:= TAU4 + C42*TAU2 + C43*TAU3;
    A12:= A14 + C42*A12 + C43*A13;
    B1:= B1 + C12*B2 + C13*B3;
    B2:= B4 + C42*B2 + C43*B3
"END" ELEMENT MAT VEC EVALUATION 3;

"PROCEDURE" BOUNDARY CONDITIONS;
"IF" L=1 "AND" E2 = 0 "THEN"
"BEGIN"  TAU1:= 1; B1:= E3/E1;B2:= B2 - A12*B1;
    TAU2:= TAU2 - A12; A12:= 0 "END"
"ELSE" "IF" L=1 "AND" E2 ≠ 0 "THEN"
"BEGIN" "REAL" AUX; AUX:= P1/E2; TAU1:= TAU1 - AUX*E1 ;
    B1:= B1 - E3*AUX
"END" "ELSE" "IF" L=N "AND" E5 = 0 "THEN"
"BEGIN" TAU2:= 1; B2:= E6/E4;
    B1:= B1 - A12*B2; TAU1:= TAU1 - A12; A12:= 0
"END" "ELSE" "IF" L=N "AND" E5 ≠ 0 "THEN"
"BEGIN" "REAL" AUX; AUX:= P2/E5;
    TAU2:= TAU2 + AUX*E4; B2:= B2 + AUX*E6
"END" B.C.1;

"PROCEDURE" FORWARD BABUSHKA;
"IF" L=1 "THEN"
"BEGIN" CHI[0]:= CH:= TL:= TAU1; T[0]:= TL;
     GI[0]:=  G:= YL:=   B1; Y[0]:= YL;
    SUB[0]:= A12; PP:= A12/(CH - A12);
    CH:= TAU2 - CH*PP; G:= B2 - G*PP; TL:= TAU2; YL:= B2
"END" "ELSE"
"BEGIN" CHI[L1]:= CH:= CH + TAU1;
     GI[L1]:=  G:=  G + B1;
```

"COMMENT"

```
      SUB[L1]:= A12; PP:= A12/(CH - A12);
      CH:= TAU2 - CH*PP; G:= B2 - G*PP;
      T[L1]:= TL + TAU1; TL:= TAU2;
      Y[L1]:= YL +  B1; YL:=  B2
      "END" FORWARD BABUSHKA 1;

      "PROCEDURE" BACKWARD BABUSHKA;
      "BEGIN" PP:= YL; Y[N]:= G/CH;
          G:= PP; CH:= TL; L:= N;
          "FOR" L:= L - 1 "WHILE" L >= 0 "DO"
          "BEGIN" PP:= SUB[L]; PP:= PP/(CH - PP);
              TL:= T[L]; CH:= TL - CH*PP;
              YL:= Y[L]; G:= YL - G*PP;
              Y[L]:=(GI[L] + G - YL)/(CHI[L] + CH - TL)
          "END"
      "END" BACKWARD BABUSHKA;

      L:= 0; XL:= X[0];
      E1:= E[1]; E2:= E[2]; E3:= E[3]; E4:= E[4]; E5:= E[5]; E6:= E[6];
      "FOR" L:= L + 1 "WHILE" L <= N "DO"
        "BEGIN" L1:= L - 1; XL1:= XL; XL:= X[L]; H:= XL - XL1;
            "IF" ORDER = 2 "THEN" ELEMENT MAT VEC EVALUATION 1 "ELSE"
            "IF" ORDER = 4 "THEN" ELEMENT MAT VEC EVALUATION 2 "ELSE"
                                  ELEMENT MAT VEC EVALUATION 3;
            "IF" L=1 "OR" L=N "THEN" BOUNDARY CONDITIONS;
            FORWARD BABUSHKA
        "END";
        BACKWARD BABUSHKA;
     "END" FEM LAG SYM;
           "EOP"


"CODE" 33301;
 "PROCEDURE" FEM LAG(X, Y, N, R, F, ORDER, E);
 "VALUE" N, ORDER; "INTEGER" N, ORDER;
 "REAL" "PROCEDURE" R, F;
 "ARRAY" X, Y, E;
 "BEGIN" "INTEGER" L, L1;
 "REAL" XL1, XL, H, A12, B1, B2, TAU1, TAU2, CH, TL, G, YL, PP,
   E1, E2, E3, E4, E5, E6;
 "ARRAY" T, SUB, CHI, GI[0: N-1];

 "PROCEDURE" ELEMENT MAT VEC EVALUATION 1;
 "BEGIN" "OWN" "REAL" F2, R2; "REAL" R1, F1, H2;
      "IF" L=1 "THEN"
      "BEGIN" F2:= F(XL1); R2:= R(XL1) "END";
      A12:= - 1/H; H2:= H/2;
      R1:= R2; R2:= R(XL); F1:= F2; F2:= F(XL);
      B1:= H2*F1; B2:= H2*F2; TAU1:= H2*R1; TAU2:= H2*R2
 "END" ELEMENT MAT VEC EVALUATION 1
```

```
"PROCEDURE" ELEMENT MAT VEC EVALUATION 2;
 "BEGIN" "OWN" "REAL" R3, F3;
    "REAL" R1, R2, F1, F2, X2, H6, H15,
    B3, TAU3, C12, A13, A22, A23;
    "IF" L=1 "THEN"
    "BEGIN" R3:= R(XL1); F3:= F(XL1) "END";
    X2:= (XL1 + XL)/2; H6:= H/6; H15:= H/1.5;
    R1:= R3; R2:= R(X2); R3:= R(XL);
    F1:= F3; F2:= F(X2); F3:= F(XL);
    B1:= H6*F1; B2:= H15*F2; B3:= H6*F3;
    TAU1:= H6*R1; TAU2:= H15*R2; TAU3:= R3*H6;
    A12:= A23:= -8/H/3; A13:= - A12/8; A22:= -2*A12 + TAU2;
    "COMMENT" STATIC CONDENSATION;
    C12:= - A12/A22; A12:= A13 + C12*A12;
    B2:= C12*B2; B1:= B1 + B2; B2:= B3 + B2;
    TAU2:= C12*TAU2; TAU1:= TAU1 + TAU2; TAU2:= TAU3 + TAU2
"END" ELEMENT MAT VEC EVALUATION2;

"PROCEDURE" ELEMENT MAT VEC EVALUATION 3;
"BEGIN" "OWN" "REAL" R4, F4;
"REAL" R1, R2, R3, F1, F2, F3, X2, X3, H12, H24,
DET, C12, C13, C42, C43, A13, A14, A22, A23, A24,
A33, A34, B3, B4, TAU3, TAU4;
"IF" L=1 "THEN"
"BEGIN" R4:= R(XL1); F4:= F(XL1) "END";
X2:= XL1 + 0.276393202250*H; X3:= XL - X2 + XL1;
R1:= R4; R2:= R(X2); R3:= R(X3); R4:= R(XL);
F1:= F4; F2:= F(X2); F3:= F(X3); F4:= F(XL);
H12:= H/12; H24:= H/2.4;
B1:= F1*H12; B2:= F2*H24; B3:= F3*H24; B4:= F4*H12;
TAU1:= R1*H12; TAU2:= R2*H24; TAU3:= R3*H24; TAU4:= R4*H12;
A12:= A34:= -4.878418305207B/H; A13:= A24:= 0.711751638541 2/H;
A14:= -0.166666666666 7/H; A23:= 25*A14;
A22:= -2*A23 + TAU2; A33:= -2*A23 + TAU3;
"COMMENT" STATIC CONDENSATION;
DET:= A22*A33 - A23*A23;
C12:= (A13*A23 - A12*A33)/DET;
C13:= (A12*A23 - A13*A22)/DET;
C42:= (A23*A34 - A24*A33)/DET;
C43:= (A24*A23 - A34*A22)/DET;
TAU1:= TAU1 + C12*TAU2 + C13*TAU3;
TAU2:= TAU4 + C42*TAU2 + C43*TAU3;
A12:= A14 + C42*A12 + C43*A13;
B1:= B1 + C12*B2 + C13*B3;
B2:= B4 + C42*B2 + C43*B3
"END" ELEMENT MAT VEC EVALUATION3
```

```
    "PROCEDURE" BOUNDARY CONDITIONS;
    "IF" L=1 "AND" E2 = 0 "THEN"
    "BEGIN" TAU1:= 1; B1:= E3/E1; B2:= B2 - A12*B1;
        TAU2:= TAU2 - A12; A12:= 0 "END"
    "ELSE" "IF" L=1 "AND" E2 =/= 0 "THEN"
    "BEGIN" TAU1:= TAU1 - E1/E2;
        B1:= B1 - E3/E2
    "END" "ELSE" "IF" L=N "AND" E5 = 0 "THEN"
    "BEGIN" TAU2:= 1; B2:= E6/E4; B1:= B1 - A12*B2;
        TAU1:= TAU1 - A12; A12:= 0
    "END" "ELSE" "IF" L=N "AND" E5 =/= 0 "THEN"
    "BEGIN" TAU2:= TAU2 + E4/E5;
        B2:= B2 + E6/E5
    "END" BOUNDARY CONDITIONS;

    "PROCEDURE" FORWARD BABUSHKA;
    "IF" L=1 "THEN"
    "BEGIN" CHI[0]:= CH:= TL:= TAU1; T[0]:= TL;
        GI[0]:= G:= YL:= B1; Y[0]:= YL;
        SUB[0]:= A12; PP:= A12/(CH - A12); CH:= TAU2 - CH*PP;
        G:= B2 - G*PP; TL:= TAU2; YL:= B2
    "END" "ELSE"
    "BEGIN" CHI[L1]:= CH:= CH + TAU1;
        GI[L1]:= G:= G + B1; SUB[L1]:= A12; PP:= A12/(CH - A12);
        CH:= TAU2 - CH*PP; G:= B2 - G*PP;
        T[L1]:= TL + TAU1; TL:= TAU2;
        Y[L1]:= YL +   B1; YL:=   B2
    "END" FORWARD BABUSHKA 1;

    "PROCEDURE" BACKWARD BABUSHKA;
    "BEGIN" PP:= YL; Y[N]:= G/CH;
        G:= PP; CH:= TL; L:= N;
        "FOR" L:= L - 1 "WHILE" L >= 0 "DO"
        "BEGIN" PP:= SUB[L]; PP:= PP/(CH - PP);
            TL:= T[L]; CH:= TL - CH*PP;
            YL:= Y[L]; G:= YL - G*PP;
            Y[L]:=((GI[L] + G) - YL)/((CHI[L] + CH) - TL)
        "END"
    "END" BACKWARD BABUSHKA;

    L:= 0; XL:= X[0];
    E1:= E[1]; E2:= E[2]; E3:= E[3]; E4:= E[4]; E5:= E[5]; E6:= E[6];
    "FOR" L:= L + 1 "WHILE" L <= N "DO"
    "BEGIN" L1:= L - 1; XL1:= XL; XL:= X[L]; H:= XL - XL1;
        "IF" ORDER = 2 "THEN" ELEMENT MAT VEC EVALUATION 1 "ELSE"
        "IF" ORDER = 4 "THEN" ELEMENT MAT VEC EVALUATION 2 "ELSE"
                              ELEMENT MAT VEC EVALUATION 3;
        "IF" L=1 "OR" L=N "THEN" BOUNDARY CONDITIONS;
        FORWARD BABUSHKA
    "END";
    BACKWARD BABUSHKA;
"END" FEM LAGR;
      "EOP"
```

AUTHOR: M. BAKKER.

INSTITUTE: MATHEMATICAL CENTRE, AMSTERDAM.

RECEIVED: 751231.

BRIEF DESCRIPTION:

THIS SECTION CONTAINS A PROCEDURE FOR THE SOLUTION OF SECOND ORDER SKEW-ADJOINT LINEAR TWO POINT BOUNDARY VALUE PROBLEMS;

FEM LAG SKEW;

THIS PROCEDURE SOLVES THE DIFFERENTIAL EQUATION

$$-Y'' + Q(X)*Y' + R(X)*Y = F(X), \quad A < X < B,$$

WITH BOUNDARY CONDITIONS

$$E[1]*Y(A) + E[2]*Y'(A) = E[3],$$

$$E[4]*Y(B) + E[5]*Y'(B) = E[6].$$

KEY WORDS AND FRASES:

SECOND ORDER DIFFERENTIAL EQUATIONS,
TWO POINT BOUNDARY VALUE PROBLEMS,
SKEW-ADJOINT BOUNDARY VALUE PROBLEMS,
GALERKIN'S METHOD,
GLOBAL METHODS.

LANGUAGE: ALGOL 60.

REFERENCES:

[1] STRANG, G. AND G.J. FIX,
AN ANALYSIS OF THE FINITE ELEMENT METHOD,
PRENTICE-HALL, ENGLEWOOD CLIFFS, NEW JERSEY, 1973.

[2] BAKKER, M., EDITOR,
COLLOQUIUM ON DISCRETIZATION METHODS, CHAPTER 3 (DUTCH),
MATHEMATISCH CENTRUM, MC-SYLLABUS, TO APPEAR.

[3] HEMKER, P.W.,
GALERKIN'S METHOD AND LOBATTO POINTS,
MATHEMATISCH CENTRUM, REPORT 24/75 (1975).

[4] BABUSKA, I.,
NUMERICAL STABILITY IN PROBLEMS OF LINEAR ALGEBRA,
S.I.A.M. J. NUM. ANAL., VOL.9, P. 53-77 (1972).

SUBSECTION: FEM LAG SKEW.

CALLING SEQUENCE:

THE HEADING OF THE PROCEDURE READS:

```
"PROCEDURE" FEM LAG SKEW(X, Y, N, Q, R, F, ORDER, E);
"VALUE" N, ORDER; "INTEGER" N, ORDER;
"ARRAY" X, Y, E;
"REAL" "PROCEDURE" Q, R, F;
"CODE" 33302;
```

THE MEANING OF THE FORMAL PARAMETERS IS:

N: ⟨ARITHMETIC EXPRESSION⟩;
THE UPPER BOUND OF THE ARRAYS X AND Y; N > 1;

X: ⟨ARRAY IDENTIFIER⟩;
"ARRAY" X[0:N];
ENTRY: A = X[0] < X[1] < ... < X[N] = B IS A
PARTITION OF THE INTERVAL [A,B];

Y: ⟨ARRAY IDENTIFIER⟩;
"ARRAY" Y[0:N];
EXIT: Y[I] (I = 0, 1, ..., N) IS THE APPROXIMATE
SOLUTION AT X[I] TO THE DIFFERENTIAL EQUATION

$$(1) \quad -Y'' + Q(X)*Y' + R(X)*Y = F(X), \quad A < X < B,$$

WITH BOUNDARY CONDITIONS

$$(2) \quad \begin{aligned} E[1]*Y(A) + E[2]*Y'(A) &= E[3], \\ E[4]*Y(B) + E[5]*Y'(B) &= E[6]; \end{aligned}$$

Q: ⟨PROCEDURE IDENTIFIER⟩;
THE HEADING OF Q READS:
"REAL" "PROCEDURE" Q(X); "VALUE" X; "REAL" X;
Q(X) IS THE COEFFICIENT OF Y' IN (1);

R: ⟨PROCEDURE IDENTIFIER⟩;
THE HEADING OF R READS:
"REAL" "PROCEDURE" R(X); "VALUE" X; "REAL" X;
R(X) IS THE COEFFICIENT OF Y IN (1);

F: ⟨PROCEDURE IDENTIFIER⟩;
THE HEADING OF F READS:
"REAL" "PROCEDURE" F(X); "VALUE" X; "REAL" X;
F(X) IS THE RIGHT HAND SIDE OF (1);

ORDER: «ARITHMETIC EXPRESSION»;
ENTRY: ORDER DENOTES THE ORDER OF ACCURACY REQUIRED FOR THE APPROXIMATE SOLUTION OF (1)=(2); LET H = MAX(X[I] - X[I-1]); THEN ABS(Y[I] - Y(X[I])) <= C*H**ORDER, I = 0, ... , N; ORDER CAN CAN BE CHOSEN EQUAL TO 2, 4 OR 6 ONLY;
E: «ARRAY IDENTIFIER»;
"ARRAY" E[1:6];
E[1], ... , E[6] DESCRIBE THE BOUNDARY CONDITIONS (2);
E[1] AND E[4] ARE NOT ALLOWWED TO VANISH BOTH.

PROCEDURES USED: NONE.

REQUIRED CENTRAL MEMORY:

FOUR AUXILIARY ARRAYS OF N REALS ARE USED.

RUNNING TIME:

LET K = ORDER/2; THEN

(A) K*N + 1 EVALUATIONS OF Q(X), R(X) AND F(X) ARE NEEDED;

(B) ABOUT 17*2**(K-1)*N MULTIPLICATIONS/DIVISIONS ARE NEEDED.

DATA AND RESULTS:
THE PROCEDURE FEM LAG SKEW HAS SOME RESTRICTIONS IN ITS USE:
(I) Q(X) IS NOT ALLOWED TO HAVE VERY LARGE VALUES IN SOME SENSE:
THE PRODUCT Q(X)*(X[J] - X[J-1]) SHOULD NOT BE TOO LARGE ON THE CLOSED INTERVAL «X[J-1],X[J]», OTHERWISE THE BOUNDARY VALUE PROBLEM MAY DEGENERATE TO A SINGULAR PERTURBATION OR BOUNDARY LAYER PROBLEM, FOR WHICH EITHER SPECIAL METHODS OR A SUITABLY CHOSEN GRID ARE NEEDED;
(II) Q(X), R(X) AND F(X) ARE REQUIRED TO BE SUFFICIENTLY DIFFERENTIABLE ON THE DOMAIN OF THE BOUNDARY VALUE PROBLEM; THEY ARE, HOWEVER, THE DERIVATIVES ARE ALLOWED TO HAVE DISCONTINUITIES AT THE GRID POINTS, IN WHICH CASE THE ORDER OF ACCURACY (2, 4 OR 6) IS PRESERVED;
(III) IF Q(X) AND R(X) SATISFY THE INEQUALITY R(X) >= Q'(X)/2, THE EXISTENCE OF A UNIQUE SOLUTION IS GUARANTEED, OTHERWISE THIS REMAINS AN OPEN QUESTION;
(IV) THE USER SHOULD NOT EXPECT GREATER ACCURACY THAN 12 DECIMALS DUE TO THE LOSS OF DIGITS DURING THE EVALUATION OF THE MATRIX AND THE VECTOR OF THE LINEAR SYSTEM TO BE SOLVED AND DURING ITS REDUCTION TO A TRIDIAGONAL SYSTEM; WHEN THE SOLUTION OF THE PROBLEM IS NOT TOO WILD, THIS 12-DIGITS ACCURACY CAN BE OBTAINED WITH A MODERATE MESH SIZE (E.G. < 0.1) ALREADY, PROVIDED A SIXTH ORDER METHOD IS USED.

METHOD AND PERFORMANCE:

PROBLEM (1)-(2) IS SOLVED BY MEANS OF GALERKIN'S METHOD WITH CONTINUOUS PIECEWISE POLYNOMIAL FUNCTIONS (SEE [1], [2]); THE SOLUTION IS APPROXIMATED BY A FUNCTION WHICH IS CONTINUOUS ON THE INTERVAL <X[0],X[N]> AND A POLYNOMIAL OF DEGREE LESS THAN OR EQUAL TO K (K = ORDER//2) ON EACH SEGMENT <X[J-1],X[J]> (J = 1, ..., N); THIS PIECEWISE POLYNOMIAL IS ENTIRELY DETERMINED BY THE VALUES IT HAS AT THE KNOTS X[J] AND ON (K-1) INTERIOR KNOTS ON EACH SEGMENT <X[J-1],X[J]>; THESE VALUES ARE OBTAINED BY THE SOLUTION OF AN (ORDER + 1)-DIAGONAL LINEAR SYSTEM WITH A SPECIALLY STRUCTURED MATRIX (SEE [2]); THE ENTRIES OF THE MATRIX AND THE VECTOR ARE INNER PRODUCTS WHICH ARE APPROXIMATED BY PICEWISE (K+1)-POINT LOBATTO QUADRATURE (SEE [3]); THE EVALUATION OF THE MATRIX AND THE VECTOR IS DONE SEGMENT BY SEGMENT; ON EACH SEGMENT THE CONTRIBUTIONS TO THE ENTRIES OF THE MATRIX AND THE VECTOR ARE COMPUTED AND EMBEDDED IN THE GLOBAL MATRIX AND VECTOR; SINCE THE FUNCTION VALUES ON THE INTERIOR POINTS OF EACH SEGMENT ARE NOT COUPLED WITH THE FUNCTION VALUES OUTSIDE THAT SEGMENT, THE RESULTING LINEAR SYSTEM CAN BE REDUCED TO A TRIDIAGONAL SYSTEM BY MEANS OF STATIC CONDENSATION (SEE [2]); SINCE THE FINAL TRIDIAGONAL SYSTEM IS OF FINITE DIFFERENCE TYPE, IT IS SOLVED BY MEANS OF BABUSKA'S METHOD (SEE [4]).

EXAMPLE OF USE:

WE SOLVE THE BOUNDARY VALUE PROBLEM

$$-Y'' + Y'*COS(X) + Y*EXP(X) = SIN(X)*(1 + EXP(X)) + COS(X)**2,$$
$$0 < X < PI = 3.1415926535897 9,\ Y(0) = Y(PI) = 0;$$

FOR THE BOUNDARY CONDITIONS THIS MEANS THAT

E[1] = E[4] = 1; E[2] = E[3] = E[5] = E[6] = 0;

THE ANALYTIC SOLUTION IS Y(X) = SIN(X); WE APPROXIMATE THE SOLUTION ON A UNIFORM GRID, I.E. X[I] = I*PI/N, I = 0, ..., N; WE CHOOSE N=10,20 AND COMPUTE FOR ORDER = 2,4,6 THE MAXIMUM ERROR; THE PROGRAM READS AS FOLLOWS:

```
"BEGIN" "INTEGER" N; "FOR" N:= 10, 20 "DO"
"BEGIN" "INTEGER" I, ORDER; "REAL" PI; "ARRAY" X, Y[0:N], E[1:6];
"REAL" "PROCEDURE" Q(X); "VALUE" X; "REAL" X;
Q:= COS(X);

"REAL" "PROCEDURE" R(X); "VALUE" X; "REAL" X;
R:= EXP(X);

"REAL" "PROCEDURE" F(X); "VALUE" X; "REAL" X;
F:= SIN(X)*(1 + EXP(X)) + COS(X)**2;

"PROCEDURE" FEM LAG SKEW(X, Y, N, Q, R, F, ORDER, E);
"CODE" 33302;
E[1]:= E[4]:= 1; E[2]:= E[3]:= E[5]:= E[6]:= 0;
PI:= 3.14159265358979;
"FOR" I:= 0 "STEP" 1 "UNTIL" N "DO" X[I]:= PI*I/N;
OUTPUT(61,"("//,6B"("N="")"D")",N);
"FOR" ORDER:= 2, 4, 6 "DO"
    "BEGIN" "REAL" RHO, D;
        FEM LAG SKEW(X, Y, N, Q, R, F, ORDER, E);
        RHO:= 0;
        "FOR" I:= 0 "STEP" 1 "UNTIL" N "DO"
        "BEGIN" D:= ABS(Y[I] - SIN(X[I]));
            "IF" RHO < D "THEN" RHO:= D
        "END";
        OUTPUT(61,"("/,16B"("ORDER="")"DD,4B"("MAX.ERROR= ")",
        D.DD"+ZD")",ORDER,RHO)
    "END"
"END"
"END"

RESULTS:

N=10
            ORDER=2   MAX. ERROR= 2.95" -3
            ORDER=4   MAX. ERROR= 2.56" -5
            ORDER=6   MAX. ERROR= 4.26" -8

N=20
            ORDER=2   MAX. ERROR= 7.55" -4
            ORDER=4   MAX. ERROR= 1.68" -6
            ORDER=6   MAX. ERROR= 6.76"-10

NOTICE THAT THE MAXIMUM ERROR DECREASES BY ABOUT
2**(-ORDER)  WHEN THE MESH SIZE IS HALVED.
```

```
SOURCE TEXT(9);

"CODE" 33302;
"PROCEDURE" FEM LAG SKEW(X, Y, N, Q, R, F, ORDER, E);
"INTEGER" N, ORDER;
"REAL" "PROCEDURE" Q, R, F;
"ARRAY" X, Y, E;
"BEGIN" "INTEGER" L, L1;
  "REAL" XL1, XL, H, A12, A21, B1, B2, TAU1, TAU2, CH, TL, G, YL, PP,
    E1, E2, E3, E4, E5, E6;
  "ARRAY" T, SUPER, SUB, CHI, GI[0:N-1];

  "PROCEDURE" ELEMENT MAT VEC EVALUATION 1;
  "BEGIN" "OWN" "REAL" Q2, R2, F2;
    "REAL" Q1, R1, F1, H2, S12;
    "IF" L=1 "THEN"
    "BEGIN" Q2:= Q(XL1); R2:= R(XL1); F2:= F(XL1) "END";
    H2:= H/2; S12:= - 1/H;
    Q1:= Q2; Q2:= Q(XL);
    R1:= R2; R2:= R(XL);
    F1:= F2; F2:= F(XL);
    B1:= H2*F1; B2:= H2*F2;
    TAU1:= H2*R1; TAU2:= H2*R2;
    A12:= S12 + Q1/2; A21:= S12 - Q2/2
"END" ELEMENT MAT VEC EV.;

"PROCEDURE" ELEMENT MAT VEC EVALUATION 2;
"BEGIN" "OWN" "REAL" Q3, R3, F3;
    "REAL" Q1, Q2, R1, R2, F1, F2, S12, S13, S22, X2, H6, H15,
      C12, C32, A13, A31, A22, A23, A32, B3, TAU3;
    "IF" L=1 "THEN"
    "BEGIN" Q3:= Q(XL1); R3:= R(XL1); F3:= F(XL1) "END";

    X2:= (XL1 + XL)/2; H6:= H/6; H15:= H/1.5;
    Q1:= Q3; Q2:= Q(X2); Q3:= Q(XL);
    R1:= R3; R2:= R(X2); R3:= R(XL);
    F1:= F3; F2:= F(X2); F3:= F(XL);
    B1:= H6*F1; B2:= H15*F2; B3:= H6*F3;
    TAU1:= H6*R1; TAU2:= H15*R2; TAU3:= H6*R3;
    S12:= - 1/H/0.375; S13:= - S12/8; S22:= - 2*S12;
    A12:= S12 + Q1/1.5; A13:= S13 - Q1/6;
    A21:= S12 - Q2/1.5; A23:= S12 + Q2/1.5; A22:= S22 + TAU2;
    A31:= S13 + Q3/6; A32:= S12 - Q3/1.5;
    "COMMENT" STATIC CONDENSATION;
    C12:= - A12/A22; C32:= - A32/A22;
    A12:= A13 + C12*A23; A21:= A31 + C32*A21;
    B1:= B1 + C12*B2; B2:= B3 + C32*B2;
    TAU1:= TAU1 + C12*TAU2; TAU2:= TAU3 + C32*TAU2
"END" ELEMENT MAT VEC EVALUATION 2
```

```
"PROCEDURE" ELEMENT MAT VEC EVALUATION 3;
"BEGIN" "OWN" "REAL" Q4, R4, F4;
    "REAL" Q1, Q2, Q3, R1, R2, R3, F1, F2, F3,
     S12, S13, S14, S22, S23, X2, X3, H12, H24,
     DET, C12, C13, C42, C43, A13, A14, A22, A23,
     A24, A31, A32, A33, A34, A41, A42, A43,
     B3, B4, TAU3, TAU4;

    "IF" L=1 "THEN"
    "BEGIN" Q4:= Q(XL1); R4:= R(XL1); F4:= F(XL1) "END";
    X2:= XL1 + 0.2763932O225*H; X3:= XL - X2 + XL1;
    H12:= H/12; H24:= H/2.4;
    Q1:= Q4; Q2:= Q(X2); Q3:= Q(X3); Q4:= Q(XL);
    R1:= R4; R2:= R(X2); R3:= R(X3); R4:= R(XL);
    F1:= F4; F2:= F(X2); F3:= F(X3); F4:= F(XL);
    S12:= -4.878418305208O/H; S13:= 0.711751638541 4/H;
    S14:= -.16666666666667/H; S23:= 25*S14; S22:= -2*S23;
    B1:= H12*F1; B2:= H24*F2; B3:= H24*F3; B4:= H12*F4;
    TAU1:= H12*R1; TAU2:= H24*R2; TAU3:= H24*R3; TAU4:= H12*R4;
    A12:= S12 + 0.6741808286457 8*Q1;
    A13:= S13 - 0.257514161979 12*Q1;
    A14:= S14 + Q1/12;
    A21:= S12 - 0.6741808286457 8*Q2;
    A22:= S22 + TAU2;
    A23:= S23 + 0.9316949906249 0*Q2;
    A24:= S13 - 0.257514161979 12*Q2;
    A31:= S13 + 0.257514161979 12*Q3;
    A32:= S23 - 0.9316949906249 0*Q3;
    A33:= S22 + TAU3;
    A34:= S12 + 0.6741808286457 8*Q3;
    A41:= S14 - Q4/12;
    A42:= S13 + 0.257514161979 12*Q4;
    A43:= S12 - 0.6741808286457 8*Q4;
    "COMMENT" STATIC CONDENSATION;
    DET:= A22*A33 - A23*A32;
    C12:= (A13*A32 - A12*A33)/DET;
    C13:= (A12*A23 - A13*A22)/DET;
    C42:= (A32*A43 - A42*A33)/DET;
    C43:= (A42*A23 - A43*A22)/DET;
    TAU1:= TAU1 + C12*TAU2 + C13*TAU3 ;
    TAU2:= TAU4 + C42*TAU2 + C43*TAU3;
    A12:= A14 + C12*A24 + C13*A34;
    A21:= A41 + C42*A21 + C43*A31;
    B1:= B1 + C12*B2 + C13*B3;
    B2:= B4 + C42*B2 + C43*B3
"END" ELEMENT MAT VEC EVALUATION 3
```

```
"PROCEDURE" BOUNDARY CONDITIONS;
"IF" L=1 "AND" E2 = 0 "THEN"
"BEGIN" TAU1:= 1; B1:= E3/E1; A12:= 0 "END"
"ELSE" "IF" L=1 "AND" E2 ¬= 0 "THEN"
"BEGIN" TAU1:= TAU1 - E1/E2; B1:= B1 - E3/E2
"END" "ELSE" "IF" L=N "AND" E5 = 0 "THEN"
"BEGIN" TAU2:= 1; A21:= 0; B2:= E6/E4;
"END" "ELSE" "IF" L=N "AND" E5 ¬= 0 "THEN"
"BEGIN" TAU2:= TAU2 + E4/E5; B2:= B2 + E6/E5
"END" B.C.1;

"PROCEDURE" FORWARD BABUSKA;
"IF" L=1 "THEN"
"BEGIN" CHI[0]:= CH:= TL:= TAU1; T[0]:= TL;
      GI[0]:= G:= YL:= B1; Y[0]:= YL;
      SUB[0]:= A21; SUPER[0]:= A12;
      PP:= A21/(CH - A12); CH:= TAU2 - CH*PP;
      G:= B2 - G*PP; TL:= TAU2; YL:= B2
"END" "ELSE"
"BEGIN" CHI[L1]:= CH:= CH + TAU1;
      GI[L1]:= G:= G + B1;
      SUB[L1]:= A21; SUPER[L1]:= A12;
      PP:= A21/(CH - A12); CH:= TAU2 - CH*PP;
      G:= B2 - G*PP; T[L1]:= TL + TAU1; TL:= TAU2;
      Y[L1]:= YL +  B1; YL:= B2
"END" FORWARD BABUSKA;

"PROCEDURE" BACKWARD BABUSKA;
"BEGIN"PP:= YL; Y[N]:= G/CH;
      G:= PP; CH:= TL; L:= N;
      "FOR" L:= L - 1 "WHILE" L >= 0 "DO"
      "BEGIN" PP:= SUPER[L]/(CH - SUB[L]);
        TL:= T[L]; CH:= TL - CH*PP;
        YL:= Y[L]; G:= YL - G*PP;
        Y[L]:=(GI[L] + G - YL)/(CHI[L] + CH - TL)  ;
      "END"
"END" BACKWARD BABUSKA;

L:= 0; XL:= X[0];
  E1:= E[1]; E2:= E[2]; E3:= E[3]; E4:= E[4]; E5:= E[5]; E6:= E[6];
  "COMMENT" ELEMENTWISE ASSEMBLAGE OF MATRIX AND VECTOR
    COMBINED WITH FORWARD BABUSKA SUBSTITUTION;
  "FOR" L:= L + 1 "WHILE" L <= N "DO"
  "BEGIN" XL1:= XL; L1:= L - 1; XL:= X[L]; H:= XL - XL1;
     "IF" ORDER = 2 "THEN" ELEMENT MAT VEC EVALUATION 1 "ELSE"
     "IF" ORDER = 4 "THEN" ELEMENT MAT VEC EVALUATION 2 "ELSE"
     ELEMENT MAT VEC EVALUATION 3;
     "IF" L=1 "OR" L=N "THEN" BOUNDARY CONDITIONS;
     FORWARD BABUSKA
  "END";
  BACKWARD BABUSKA;
"END" FEM LAGR;
      "EOP"
```

AUTHOR: M. BAKKER,

INSTITUTE: MATHEMATICAL CENTRE, AMSTERDAM.

RECEIVED: 751231.

BRIEF DESCRIPTION:

THIS SECTION CONTAINS A PROCEDURE FOR THE SOLUTION
OF FOURTH ORDER SELF-ADJOINT LINEAR TWO POINT
BOUNDARY VALUE PROBLEMS;

FEM HERM SYM;

THIS PROCEDURE SOLVES THE DIFFERENTIAL EQUATION

$$(P(X)*Y'')'' - (Q(X)*Y')' + R(X)*Y = F(X), \quad A < X < B,$$

WITH BOUNDARY CONDITIONS

$$Y(A) = E[1], \quad Y'(A) = E[2],$$

$$Y(B) = E[3], \quad Y'(B) = E[4].$$

KEY WORDS AND FRASES:

FOURTH ORDER DIFFERENTIAL EQUATIONS,
TWO POINT BOUNDARY VALUE PROBLEMS,
SELF-ADJOINT BOUNDARY VALUE PROBLEMS,
GALERKIN'S METHOD,
DIRICHLET BOUNDARY CONDITIONS,
GLOBAL METHODS.

LANGUAGE: ALGOL 60.

REFERENCES:

[1] STRANG, G. AND G.J. FIX,
AN ANALYSIS OF THE FINITE ELEMENT METHOD,
PRENTICE-HALL, ENGLE WOOD CLIFFS, NEW JERSEY, 1973.

[2] BAKKER, M., EDITOR,
COLLOQUIUM ON DISCRETIZATION METHODS, CHAPTER 3 (DUTCH),
MATHEMATISCH CENTRUM, MC-SYLLABUS, TO APPEAR.

[3] HEMKER, P.W.,
GALERKIN'S METHOD AND LOBATTO POINTS,
MATHEMATISCH CENTRUM, REPORT 24/75 (1975).

CALLING SEQUENCE:

THE HEADING OF THE PROCEDURE READS:

```
"PROCEDURE" FEM HER" SYM(X, Y, N, P, Q, R, F, ORDER, E);
"VALUE" N, ORDER; "INTEGER" N, ORDER;
"ARRAY" X, Y, E;
"REAL" "PROCEDURE" P, Q, R, F;
"CODE" 33303;
```

THE MEANING OF THE FORMAL PARAMETERS IS:

N: <ARITHMETIC EXPRESSION>;
THE UPPER BOUND OF THE ARRAY X; N > 1;

X: <ARRAY IDENTIFIER>;
"ARRAY" X[0:N];
ENTRY: A = X[0] < X[1] < ... < X[N] = B IS A
PARTITION OF THE INTERVAL [A,B];

Y: <ARRAY IDENTIFIER>;
"ARRAY" Y[1:2*N-2];
EXIT: Y[2*I-1] IS AN APPROXIMATION TO Y(X[I]),
Y[2*I] IS AN APPROXIMATION TO Y'(X[I]),
WHERE Y(X) IS THE SOLUTION OF THE DIFFERENTIAL EQUATION

$$(1)\quad (P(X)*Y'')'' - (Q(X)*Y')' + R(X)*Y = F(X) \quad , A < X < B,$$

WITH BOUNDARY CONDITIONS

$$(2)\quad \begin{aligned} Y(A) &= E[1],\ Y'(A) = E[2], \\ Y(B) &= E[3],\ Y'(B) = E[4]; \end{aligned}$$

P: <PROCEDURE IDENTIFIER>;
THE HEADING OF P READS:
"REAL" "PROCEDURE" P(X); "VALUE" X; "REAL" X;
P(X) IS THE COEFFICIENT OF Y'' IN (1);
P(X) SHOULD BE STRICTLY POSITIVE;

Q: <PROCEDURE IDENTIFIER>;
THE HEADING OF Q READS:
"REAL" "PROCEDURE" Q(X); "VALUE" X; "REAL" X;
Q(X) IS THE COEFFICIENT OF Y' IN (1);
Q(X) SHOULD BE NONNEGATIVE;

R: <PROCEDURE IDENTIFIER>;
THE HEADING OF R READS:
"REAL" "PROCEDURE" R(X); "VALUE" X; "REAL" X;
R(X) IS THE COEFFICIENT OF Y IN (1);
R(X) SHOULD BE NONNEGATIVE;

F: «PROCEDURE IDENTIFIER»;
THE HEADING OF F READS:
"REAL" "PROCEDURE" F(X); "VALUE" X; "REAL" X;
F(X) IS THE RIGHT HAND SIDE OF (1);

ORDER: «ARITHMETIC EXPRESSION»;
ENTRY: ORDER DENOTES THE ORDER OF ACCURACY REQUIRED FOR THE APPROXIMATE SOLUTION OF (1)-(2); LET H = MAX(X[I] - X[I-1]); THEN
ABS(Y[2*I-1]-Y(X[I])) <= C1 * H**ORDER,
ABS(Y[2*I]-Y'(X[I]) <= C2 * H**ORDER, I = 1,...,N-1;
ORDER CAN ONLY BE CHOSEN EQUAL TO 4, 6, 8;

E: «ARRAY IDENTIFIER»;
"ARRAY" E[1:4];
E[1], ... , E[4] DESCRIBE THE BOUNDARY CONDITIONS (2).

PROCEDURES USED: CHLDECSOLBND = CP 34333

REQUIRED CENTRAL MEMORY:

ONE AUXILIARY ARRAY OF 8*(N-1) REALS IS USED.

RUNNING TIME:

LET K = ORDER/2; THEN

(A) K*N + 1 EVALUATIONS OF P(X), Q(X), R(X) AND F(X) ARE NEEDED;

(B) ABOUT (ORDER-3)*50*N MULTIPLICATIONS/DIVISIONS ARE NEEDED;

(C) ONE CALL OF CHLDECSOLBND IS DONE.

DATA AND RESULTS:

THE PROCEDURE FEM HERM SYM HAS SOME RESTRICTIONS:
(I) P(X) SHOULD BE POSITIVE ON THE CLOSED INTERVAL «X[0],X[N]» AND Q(X) AND R(X) SHOULD BE NONNEGATIVE THERE;
(II) P(X), Q(X), R(X) AND F(X) ARE REQUIRED TO BE SUFFICIENTLY SMOOTH ON THE INTERVAL «X[0],X[N]» EXCEPT AT THE KNOTS, WHERE DISCONTINUITIES OF THE DERIVATIVES ARE ALLOWED; IN THAT CASE THE ORDER OF ACCURACY IS PRESERVED;
(III) THE USER SHOULD NOT EXPECT HIGHER ACCURACY THAN 12 DECIMALS DUE TO THE LOSS OF DIGITS DURING THE EVALUATION OF THE MATRIX AND VECTOR AND DURING THE REDUCTION TO A PENTADIAGONAL SYSTEM; THIS ACCURACY CAN BE REACHED VERY EASILY WHEN AN EIGTH ORDER METHOD IS USED

METHOD AND PERFORMANCE:

PROBLEM (1)-(2) IS SOLVED BY MEANS OF GALERKIN'S METHOD WITH CONTINUOUSLY DIFFERENTIABLE PIECEWISE POLYNOMIAL FUNCTIONS (SEE [1], [2]); THE SOLUTION IS APPROXIMATED BY A FUNCTION WHICH IS CONTINUOUSLY DIFFERENTIABLE ON THE CLOSED INTERVAL <X[0],X[N]> AND A POLYNOMIAL OF DEGREE LESS THAN OR EQUAL TO K (K = 1 + ORDER//2) ON EACH CLOSED SEGMENT <X[J-1],X[J]> (J = 1, ..., N);
THIS FUNCTION IS ENTIRELY DETERMINED BY THE VALUES OF THE ZEROETH AND FIRST DERIVATIVE AT THE KNOTS X[J] AND BY THE VALUES IT HAS AT (K-3) INTERIOR KNOTS ON EACH CLOSED SEGMENT <X[J-1],X[J]>; THE VALUES OF THE FUNCTION AND ITS DERIVATIVE AT THE KNOTS ARE OBTAINED BY THE SOLUTION OF AN (ORDER + 1)-DIAGONAL LINEAR SYSTEM OF (K-1)*N - 2 UNKNOWNS; THE ENTRIES OF THE MATRIX AND THE VECTOR ARE INNER PRODUCTS WHICH ARE APPROXIMATED BY PIECEWISE K-POINT LOBATTO QUADRATURE (SEE [3]); THE EVALUATION OF THE MATRIX AND VECTOR IS PERFORMED SEGMENT BY SEGMENT;
IF K > 3 THE RESULTING LINEAR SYSTEM CAN BE REDUCED TO A PENTADIAGONAL SYSTEM BY MEANS OF STATIC CONDENSATION; THIS IS POSSIBLE BECAUSE THE FUNCTION VALUES AT THE INTERIOR KNOTS ON EACH SEGMENT <X[J-1],X[J]> DO NOT DEPEND ON FUNCTION VALUES OUTSIDE THAT SEGMENT;
THE FINAL PENTADIAGONAL SYSTEM, SINCE THE MATRIX IS POSITIVE DEFINITE AND SYMMETRIC, IS SOLVED BY MEANS OF CHOLESKY'S DECOMPOSITION METHOD (SEE SECTION 3.1.2.1.1.2.1.3).

EXAMPLE OF USE:

WE SOLVE THE BOUNDARY VALUE PROBLEM

```
Y''''' = (Y''*COS(X))'' + Y*EXP(X) = SIN(X)*(1 + EXP(X) + COS(X)*2),
    0 < X < PI;
    Y(0) = Y(PI) = 0; Y'(0) = 1; Y'(PI) = -1;
                      PI = 3.1415926535897;
```

THE ANALYTIC SOLUTION IS Y(X) = SIN(X); WE APPROXIMATE THE SOLUTION ON A UNIFORM GRID, I.E. X[I] = I*PI/N, I = 0, ..., N; WE CHOOSE N = 5, 10 AND WE COMPUTE THE MAXIMUM DEVIATIONS FROM Y(X[I]) AND Y'(X[I]) FOR ORDER = 4, 6, 8;
THE PROGRAM READS AS FOLLOWS:

```
"BEGIN" "INTEGER" N; "FOR" N:= 5, 10 "DO"
"BEGIN" "INTEGER" I, ORDER; "REAL" PI; "ARRAY" X[0:N],
                                Y[1:2*N-2], E[1:4];

"REAL" "PROCEDURE" P(X); "VALUE" X; "REAL" X; P:= 1;

"REAL" "PROCEDURE" Q(X); "VALUE" X; "REAL" X;
Q:= COS(X);

"REAL" "PROCEDURE" R(X); "VALUE" X; "REAL" X;
R:= EXP(X);

"REAL" "PROCEDURE" F(X); "VALUE" X; "REAL" X;
F:= SIN(X)*(1 + EXP(X)+ 2*COS(X));

"PROCEDURE" FEM HERM SYM(X, Y, N, P, Q, R, F, ORDER, E);
"CODE" 33303;
E[1]:= E[3]:= 0; E[2]:= 1; E[4]:= - 1;
PI:= 3.14159265358979;
"FOR" I:= 0 "STEP" 1 "UNTIL" N "DO" X[I]:= PI*I/N;
OUTPUT(61,"("//,6B"("N="")"ZD")",N);
"FOR" ORDER:= 4, 6, 8 "DO"
"BEGIN" "REAL" RHO1, RHO2, D1, D2;
    FEM HERM SYM(X, Y, N, P, Q, R, F, ORDER, E);
    RHO1:= RHO2:= 0;
    "FOR" I:= 1 "STEP" 1 "UNTIL" N - 1 "DO"
    "BEGIN" D1:= ABS(Y[2*I-1] - SIN(X[I]));
        "IF" RHO1 < D1 "THEN" RHO1:= D1;
        D2:= ABS(Y[2*I] - COS(X[I]));
        "IF" RHO2 < D2 "THEN" RHO2:= D2
    "END";
    OUTPUT(61,"("/,16B"("ORDER="")"D,/,
    24B"("MAX ABS(Y[2*I-1]-Y(X[I]))= ")",D.3D"+ZD,
    /,24B"("MAX ABS(Y[2*I]-Y'(X[I]))= ")",D.3D"+ZD")",
        ORDER,RHO1,RHO2)
"END"
"END"
"END"
```

RESULTS:

```
N= 5
      ORDER=4
            MAX ABS(Y[2*I-1]-Y(X[I])) = 4.822" -4
            MAX ABS(Y[2*I]-Y'(X[I]))  = 4.548" -4
      ORDER=6
            MAX ABS(Y[2*I-1]-Y(X[I])) = 5.651" -6
            MAX ABS(Y[2*I]-Y'(X[I]))  = 2.035" -6
      ORDER=8
            MAX ABS(Y[2*I-1]-Y(X[I])) = 2.264" -8
            MAX ABS(Y[2*I]-Y'(X[I]))  = 1.600" -8

N=10
      ORDER=4
            MAX ABS(Y[2*I-1]-Y(X[I])) = 2.657" -5
            MAX ABS(Y[2*I]-Y'(X[I]))  = 2.870" -5
      ORDER=6
            MAX ABS(Y[2*I-1]-Y(X[I])) = 8.398" -8
            MAX ABS(Y[2*I]-Y'(X[I]))  = 3.572" -8
      ORDER=8
            MAX ABS(Y[2*I-1]-Y(X[I])) = 7.981"-11
            MAX ABS(Y[2*I]-Y'(X[I]))  = 6.796"-11

NOTICE THAT THE MAXIMUM ERROR IS DIVIDED BY
2**ORDER, WHEN THE MESH SIZE IS HALVED.
```

SOURCE TEXT(S):

```
"CODE" 33303;
 "PROCEDURE"  FEM HERM SYM(X, Y, N, P, Q, R, F, ORDER, E);
 "VALUE" N, ORDER; "INTEGER" N, ORDER;
"ARRAY" X, Y, E;
"REAL" "PROCEDURE"  P, Q, R, F;
"BEGIN" "INTEGER" L, N2, V, W;
  "ARRAY" A[1:8*(N - 1)], EM[2:3];
  "REAL" A11, A12, A13, A14, A22, A23, A24, A33, A34, A44,
    YA, YB, ZA, ZB,
    B1, B2, B3, B4, D1, D2, E1, R1, R2, XL1, XL;

  "PROCEDURE" CHLDECSOLBND(A, N, W, AUX, B); "CODE"34333;

  "PROCEDURE" ELEMENTMATVECEVALUATION;
  "IF"ORDER=4"THEN"
  "BEGIN" "REAL" X2, H, H2, H3, P1, P2,
      Q1, Q2, R1, R2, F1, F2,
      B11, B12, B13, B14, B22, B23, B24, B33, B34, B44,
      S11, S12, S13, S14, S22, S23, S24, S33, S34, S44,
      M11, M12, M13, M14, M22, M23, M24, M33, M34, M44;
      "OWN" "REAL"P3, Q3, R3, F3;

      H:= XL - XL1; H2:= H*H; H3:= H*H2;
      X2:= (XL1 + XL)/2;
      "IF"L=1"THEN"
      "BEGIN"P3:= P(XL1); Q3:= Q(XL1); R3:= R(XL1); F3:= F(XL1)
      "END";

      "COMMENT" ELEMENT BENDING MATRIX;
      P1:= P3; P2:= P(X2); P3:= P(XL);
      B11:= 6*(P1 + P3); B12:= 4*P1 + 2*P3;
      B13:= - B11; B14:= B11 - B12;
      B22:= (4*P1 + P2 + P3)/1.5; B23:= - B12; B24:= B12 - B22;
      B33:= B11; B34:= - B14; B44:= B14 - B24;

      "COMMENT" ELEMENT STIFFNESS MATRIX;
      Q1:= Q3; Q2:= Q(X2); Q3:= Q(XL);
      S11:= 1.5*Q2; S12:= Q2/4; S13:= - S11; S14:= S12;
      S24:= Q2/24; S22:= Q1/6 + S24; S23:= - S12;
      S33:= S11; S34:= - S12; S44:= S24 + Q3/6;

      "COMMENT" ELEMENT MASS MATRIX;
      R1:= R3; R2:= R(X2); R3:= R(XL);
      M11:= (R1 + R2)/6; M12:= R2/24; M13:= R2/6; M14:= - M12;
      M22:= R2/96; M23:= - M14; M24:= - M22;
      M33:= (R2 + R3)/6; M34:= M14; M44:= M22;

      "COMMENT" ELEMENT LOAD VECTOR;
      F1:= F3; F2:= F(X2); F3:= F(XL);
      B1:= H*(F1 + 2*F2)/6; B3:= H*(F3 + 2*F2)/6;
      B2:= H2*F2/12; B4:= - B2;
```

```
A11:= B11/H3 + S11/H + M11*H; A12:= B12/H2 + S12 + M12*H2;
A13:= B13/H3 + S13/H + M13*H; A14:= B14/H2 + S14 + M14*H2;
A22:= B22/H + S22*H + M22*H3; A23:= B23/H2 + S23 + M23*H2;
A24:= B24/H + S24*H + M24*H3; A34:= B34/H2 + S34 + M34*H2;
A33:= B33/H3 + S33/H + M33*H; A44:= B44/H + S44*H + M44*H3
"END" "ELSE" "IF"ORDER=6"THEN"
"BEGIN" "OWN" "REAL"P4, Q4, R4, F4;
    "REAL"H, H2, H3, X2, X3,
    P1, P2, P3, Q1, Q2, Q3,
    R1, R2, R3, F1, F2, F3,
    B11, B12, B13, B14, B15, B22, B23, B24, B25,
    B33, B34, B35, B44, B45, B55,
    S11, S12, S13, S14, S15, S22, S23, S24, S25,
    S33, S34, S35, S44, S45, S55,
    M11, M12, M13, M14, M15, M22, M23, M24, M25,
    M33, M34, M35, M44, M45, M55,
    A15, A25, A35, A45, A55, C1, C2, C3, C4, B5;
    "IF"L=1"THEN"
    "BEGIN"P4:= P(XL1); Q4:= Q(XL1); R4:= R(XL1); F4:= F(XL1)
    "END";

H:= XL = XL1; H2:= H*H; H3:= H*H2;
X2:= 0.2763932O225*H + XL1; X3:= XL1 + XL - X2;

"COMMENT" ELEMENT BENDING MATRIX;
P1:= P4; P2:= P(X2); P3:= P(X3); P4:= P(XL);
B11:= + 4.033333333333"+1*P1 + 1.112491386673B"-1*P2
      + 1.442208419466A"+1*P3 + 8.333333333333"+0*P4;
B12:= + 1.466666666667"+1*P1 - 3.319142509165"-1*P2
      + 2.798580917581B"+0*P3 + 1.666666666667"+0*P4;
B13:= + 1.833333333333"+1*(P1+P4)
      + 1.266666666667"+0*(P2+P3);
B15:= - (B11 + B13); B14:= - (B12 + B13 + B15/2);
B22:= + 5.333333333333"+0*P1 + 9.902734644167A"-1*P2
      + 5.430598689162A"-1*P3 + 3.333333333333"-1*P4;
B23:= + 6.666666666667"+0*P1 - 3.779127846416A"+0*P2
      + 2.457945130829A"-1*P3 + 3.666666666667"+0*P4;
B25:= - (B12 + B23); B24:= - (B22 + B23 + B25/2);
B33:= + 8.333333333333"+0*P1 + 1.442208419466A"+1*P2
      + 1.112491386672B"-1*P3 + 4.033333333333"+1*P4;
B35:= - (B13 + B33); B34:= - (B23 + B33 + B35/2);
B45:= - (B14 + B34); B44:= - (B24 + B34 + B45/2);
B55:= - (B15 + B35);
```

"COMMENT"

```
"COMMENT" ELEMENT STIFFNESS MATRIX;
Q1:= Q4; Q2:= Q(X2); Q3:= Q(X3); Q4:= Q(XL);
S11:= + 2.884416838933O"+0*Q2 + 2.2249827733448"-2*Q3;
S12:= + 2.567105187249B"-1*Q2 + 3.289481274999A"-3*Q3;
S13:= + 2.533333333333"-1*(Q2+Q3);
S14:= - 3.745355992500S"-2*Q2 - 2.254644007498B"-2*Q3;
S15:= - (S13 + S11);
S22:= + 8.333333333333"-2*Q1 + 2.284700655416A"-2*Q2
      + 4.863267791644S"-4*Q3;
S23:= + 2.254644007500B"-2*Q2 + 3.745355992487A"-2*Q3;
S24:= - 3.333333333333"-3*(Q2+Q3);
S25:= - (S12 + S23);
S331:= + 2.224982773347A"-2*Q2 + 2.884416838933O"+0*Q3;
S341:= - 3.289481275012B"-3*Q2 - 2.567105187249B"-1*Q3;
S351:= - (S13 + S33);
S44:= + 4.863267791678S"-4*Q2
      + 2.284700655416A"-2*Q3 + 8.333333333338"-2*Q4;
S45:= - (S14 + S34);
S55:= - (S15 + S35);
```

```
"COMMENT" ELEMENT MASS MATRIX;
R1:= R4; R2:= R(X2); R3:= R(X3); R4:= R(XL);
M11:= + 8.333333333333"-2*R1 + 1.012907608608A"-1*R2
      + 7.375905805838O"-3*R3;
M12:= + 1.329618127333A"-2*R2 + 1.370485393335A"-3*R3;
M13:= - 2.733333333333"-2*(R2+R3);
M14:= + 5.078689325833S"-3*R2 + 3.587977340833A"-3*R3;
M15:= + 1.314798711599A"-1*R2 - 3.547987115999A"-2*R3;
M22:= + 1.745355992500O"-3*R2 + 2.546440075005O"-4*R3;
M23:= - 3.587977340933S"-3*R2 - 5.078689325838S"-3*R3;
M24:= + 6.666666666667"-4*(R2+R3);
M25:= + 1.725902921333A"-2*R2 - 6.592362546672A"-3*R3;
M33:= + 7.375905805838O"-3*R2
      + 1.012907608608A"-1*R3 + 8.333333333333"-2*R4;
M34:= - 1.370485393333A"-3*R2 - 1.329618127333A"-2*R3;
M35:= - 3.547987115999S"-2*R2 + 1.314798711599A"-1*R3;
M44:= + 2.546440075000S"-4*R2 + 1.745355992499A"-3*R3;
M45:= + 6.592362546665S"-3*R2 - 1.725902921333O"-2*R3;
M55:= + .170666666667"+0*(R2+R3);
```

"COMMENT"

```
        "COMMENT" ELEMENT LOAD VECTOR;
        F1:= F4; F2:= F(X2); F3:= F(X3); F4:= F(XL);
        B1:= + 8.333333333333"-2*F1 + 2.054372986874 9"-1*F2
             - 5.543729868748 9"-2*F3;
        B2:= + 2.696723314583 2"-2*F2 - 1.030056647917 5"-2*F3;
        B3:= - 5.543729868748 9"-2*F2
             + 2.054372986874 9"-1*F3 + 8.333333333333"-2*F4;
        B4:= + 1.030056647916 5"-2*F2 - 2.696723314583 0"-2*F3;
        B5:= + 2.666666666667"-1*(F2+F3);

        A11:= H2*(H2*M11 + S11) + B11; A12:= H2*(H2*M12 + S12) + B12;
        A13:= H2*(H2*M13 + S13) + B13; A14:= H2*(H2*M14 + S14) + B14;
        A15:= H2*(H2*M15 + S15) + B15; A22:= H2*(H2*M22 + S22) + B22;
        A23:= H2*(H2*M23 + S23) + B23; A24:= H2*(H2*M24 + S24) + B24;
        A25:= H2*(H2*M25 + S25) + B25; A33:= H2*(H2*M33 + S33) + B33;
        A34:= H2*(H2*M34 + S34) + B34; A35:= H2*(H2*M35 + S35) + B35;
        A44:= H2*(H2*M44 + S44) + B44; A45:= H2*(H2*M45 + S45) + B45;
        A55:= H2*(H2*M55 + S55) + B55;

        "COMMENT" STATIC CONDENSATION;
        C1:= A15/A55; C2:= A25/A55; C3:= A35/A55; C4:= A45/A55;
        B1:= (B1 - C1*B5)*H; B2:= (B2 - C2*B5)*H2;
        B3:= (B3 - C3*B5)*H; B4:= (B4 - C4*B5)*H2;
        A11:= (A11 - C1*A15)/H3; A12:= (A12 - C1*A25)/H2;
        A13:= (A13 - C1*A35)/H3; A14:= (A14 - C1*A45)/H2;
        A22:= (A22 - C2*A25)/H; A23:= (A23 - C2*A35)/H2;
        A24:= (A24 - C2*A45)/H; A33:= (A33 - C3*A35)/H3;
        A34:= (A34 - C3*A45)/H2; A44:= (A44 - C4*A45)/H;
"END" "ELSE"
"BEGIN" "OWN" "REAL"P5, Q5, R5, F5;
    "REAL" X2, X3, X4, H, H2, H3,
    P1, P2, P3, P4, Q1, Q2, Q3, Q4,
    R1, R2, R3, R4, F1, F2, F3, F4,
    B11, B12, B13, B14, B15, B16, B22, B23, B24, B25, B26,
    B33, B34, B35, B36, B44, B45, B46, B55, B56, B66,
    S11, S12, S13, S14, S15, S16, S22, S23, S24, S25, S26,
    S33, S34, S35, S36, S44, S45, S46, S55, S56, S66,
    M11, M12, M13, M14, M15, M16, M22, M23, M24, M25, M26,
    M33, M34, M35, M36, M44, M45, M46, M55, M56, M66,
    C15, C16, C25, C26, C35, C36, C45, C46, B5, B6,
    A15, A16, A25, A26, A35, A36, A45, A46, A55, A56, A66, DET;
"IF"L=1"THEN"
"BEGIN"P5:= P(XL1); Q5:= Q(XL1); R5:= R(XL1); F5:= F(XL1)
"END";
H:= XL - XL1; H2:= H*H; H3:= H*H2;
X2:= XL1 + H*.172673164646; X3:= XL1 + H/2; X4:= XL1 + XL - X2;
                                                "COMMENT"
```

```
"COMMENT" ELEMENT BENDING MATRIX;
P1:= P5; P2:= P(X2); P3:= P(X3); P4:= P(X4); P5:= P(XL);
B11:= + 105.8*P1 + 9.8*P5      + 7.359312130351 3"-2*P2
      + 2.2755555555556"+1*P3 + 7.0565656088553"+0*P4;
B12:= + 27.6*P1  + 1.4*P5      - 3.4155482481 1"-1*P2
      + 2.8444444444444"+0*P3 + 1.0113960946522"+0*P4;
B13:= - 32.2*(P1 + P5)         - 7.2063492063505"-1*(P2 + P4)
      + 2.2755555555556"+1*P3;
B14:= + 4.6*P1 + 8.4*P5        + 1.0328641222944"-1*P2
      - 2.8444444444444"+0*P3 - 3.3445562534992"+0*P4;
B15:= - (B11 + B13); B16:= - (B12 + B13 + B14 + B15/2);
B22:= + 7.2*P1 + 0.2*P5        + 1.5851984028581"+0*P2
      + 3.5555555555556"-1*P3 + 1.4496032730059"-1*P4;
B23:= - 8.4*P1 - 4.6*P5        + 3.3445562534992"+0*P2
      + 2.8444444444444"+0*P3 - 1.0328641222944"-1*P4;
B24:= + 1.2*(P1 + P5)          - 4.7936507936508"-1*(P2 + P4)
      - 3.5555555555556"-1*P3;
B25:= - (B12 + B23); B26:= - (B22 + B23 + B24 + B25/2);
B33:= + 7.0565656088553"+0*P2 + 2.2755555555556"+1*P3
      + 7.359312130351 3"-2*P4 + 105.8*P5 + 9.8*P1;
B34:= - 1.4*P1 - 27.6*P5       - 1.0113960946522"+0*P2
      - 2.8444444444444"+0*P3 + 3.4155482481 1"-1*P4;
B35:= - (B13 + B33); B36:= - (B23 + B33 + B34 + B35/2);
B44:= +7.2*P5 + P1/5           + 1.4496032730059"-1*P2
      + 3.5555555555556"-1*P3 + 1.5851984028581"+0*P4;
B45:= - (B14 + B34); B46:= - (B24 + B34 + B44 + B45/2);
B55:= - (B15 + B35); B56:= - (B16 + B36);
B66:= - (B26 + B36 + B46 + B56/2);

"COMMENT" ELEMENT STIFFNESS MATRIX;
Q1:= Q5; Q2:= Q(X2); Q3:= Q(X3); Q4:= Q(X4); Q5:= Q(XL);
S11:= + 3.0242424037951"+0*Q2 + 3.153990913006 5"-2*Q4;
S12:= + 1.2575525581744"-1*Q2 + 4.1767169716742"-3*Q4;
S13:= - 3.0884353741496"-1*(Q2+Q4);
S14:= + 4.0899041243062"-2*Q2 + 1.2842455355577"-2*Q4;
S15:= - (S13 + S11);
S16:= + 5.9254861177068"-1*Q2 + 6.0512612719116"-2*Q4;
S22:= + 5.2292052865422"-3*Q2 + 5.5310763862796"-4*Q4 + Q1/20;
S23:= - 1.2842455355577"-2*Q2 - 4.0899041243062"-2*Q4;
S24:= + 1.7006802721088"-3*(Q2+Q4);
S25:= - (S12 + S23);
S26:= + 2.4639593097426"-2*Q2 + 8.0134681270641"-3*Q4;
S33:= + 3.153990913006 5"-2*Q2 + 3.0242424037951"+0*Q4;
S34:= - 4.1767169716742"-3*Q2 - 1.2575525581744"-1*Q4;
S35:= - (S13 + S33);
S36:= - 6.0512612719116"-2*Q2 - 5.9254861177068"-1*Q4;
S44:= + 5.5310763862796"-4*Q2 + 5.2292052865422"-3*Q4 + Q5/20;
S45:= - (S14 + S34);
S46:= + 8.0134681270641"-3*Q2 + 2.4639593097426"-2*Q4;
S55:= - (S15 + S35); S56:= -(S16 + S36);
S66:= + 1.1609977324263"-1*(Q2+Q4) + 3.5555555555556"-1*Q3;
                                                    "COMMENT"
```

```
"COMMENT" ELEMENT MASS MATRIX;
R1:= R5; R2:= R(X2); R3:= R(X3); R4:= R(X4); R5:= R(XL);
M11:= + 9.710702072731O"-2*R2 + 1.581025919918O"-3*R4 + R1/20;
M12:= + 8.235488946025 4"-3*R2 + 2.193215496007 1"-4*R4;
M13:= + 1.239067055393 6"-2*(R2+R4);
M14:= - 1.718846624996 8"-3*R2 - 1.050832675293 9"-3*R4;
M15:= + 5.308978971211 9"-2*R2 + 6.774155866106 0"-3*R4;
M16:= - 1.737771285607 6"-2*R2 + 2.217363001846 6"-3*R4;
M22:= + 6.984384617314 5"-4*R2 + 3.042451202934 9"-5*R4;
M23:= + 1.050832675294 7"-3*R2 + 1.718846624993 6"-3*R4;
M24:= - 1.457725947520 6"-4*(R2+R4);
M25:= + 4.502458967912 7"-3*R2 + 9.397179028337 4"-4*R4;
M26:= - 1.473775645278 0"-3*R2 + 3.075948872599 8"-4*R4;
M33:= + 1.581025919920 9"-3*R2 + 9.710702072729 0"-2*R4 + R5/20;
M34:= - 2.193215496013 1"-4*R2 - 8.235488946025 4"-3*R4;
M35:= + 6.774155866112 3"-3*R2 + 5.308978971211 2"-2*R4;
M36:= - 2.217363001849 2"-3*R2 + 1.737771285607 1"-2*R4;
M44:= + 3.042451202945 7"-5*R2 + 6.984384617315 8"-4*R4;
M45:= - 9.397179028354 2"-4*R2 - 4.502458967913 1"-3*R4;
M46:= + 3.075948872606 0"-4*R2 - 1.473775645277 8"-3*R4;
M55:= + 2.902494331069 7"-2*(R2+R4) + 3.555555555556"-1*R3;
M56:= + 9.500642840205 0"-3*(R4-R2);
M66:= + 3.109815354712 5"-3*(R2+R4);

"COMMENT" ELEMENT LOAD VECTOR;
F1:= F5; F2:= F(X2); F3:= F(X3); F4:= F(X4); F5:= F(XL);
B1:= + 1.625874809933 6"-1*F2 + 2.074585233996 9"-2*F4 + F1/20;
B2:= + 1.378878058923 3"-2*F2 + 2.877886077433 5"-3*F4;
B3:= + 2.074585233996 9"-2*F2 + 1.625874809933 6"-1*F4 + F5/20;
B4:= - 2.877886077433 5"-3*F2 - 1.378878058923 3"-2*F4;
B5:= + (F2 + F4)/11.25 + 3.555555555556"-1*F3;
B6:= + 2.909571869813 2"-2*(F4-F2);

A11:= H2*(H2*M11 + S11) + B11; A12:= H2*(H2*M12 + S12) + B12;
A13:= H2*(H2*M13 + S13) + B13; A14:= H2*(H2*M14 + S14) + B14;
A15:= H2*(H2*M15 + S15) + B15; A16:= H2*(H2*M16 + S16) + B16;
A22:= H2*(H2*M22 + S22) + B22; A23:= H2*(H2*M23 + S23) + B23;
A24:= H2*(H2*M24 + S24) + B24; A25:= H2*(H2*M25 + S25) + B25;
A26:= H2*(H2*M26 + S26) + B26; A33:= H2*(H2*M33 + S33) + B33;
A34:= H2*(H2*M34 + S34) + B34; A35:= H2*(H2*M35 + S35) + B35;
A36:= H2*(H2*M36 + S36) + B36; A44:= H2*(H2*M44 + S44) + B44;
A45:= H2*(H2*M45 + S45) + B45; A46:= H2*(H2*M46 + S46) + B46;
A55:= H2*(H2*M55 + S55) + B55; A56:= H2*(H2*M56 + S56) + B56;
A66:= H2*(H2*M66 + S66) + B66;
```

"COMMENT"

```
      "COMMENT" STATIC CONDENSATION;
      DET:= = A55*A66 + A56*A56;
      C15:= (A15*A66 - A16*A56)/DET; C16:= (A16*A55 - A15*A56)/DET;
      C25:= (A25*A66 - A26*A56)/DET; C26:= (A26*A55 - A25*A56)/DET;
      C35:= (A35*A66 - A36*A56)/DET; C36:= (A36*A55 - A35*A56)/DET;
      C45:= (A45*A66 - A46*A56)/DET; C46:= (A46*A55 - A45*A56)/DET;
      A11:= (A11 + C15*A15 + C16*A16)/H3;
      A12:= (A12 + C15*A25 + C16*A26)/H2;
      A13:= (A13 + C15*A35 + C16*A36)/H3;
      A14:= (A14 + C15*A45 + C16*A46)/H2;
      A22:= (A22 + C25*A25 + C26*A26)/H;
      A23:= (A23 + C25*A35 + C26*A36)/H2;
      A24:= (A24 + C25*A45 + C26*A46)/H;
      A33:= (A33 + C35*A35 + C36*A36)/H3;
      A34:= (A34 + C35*A45 + C36*A46)/H2;
      A44:= (A44 + C45*A45 + C46*A46)/H;
      B1:= (B1 + C15*B5 + C16*B6)*H; B2:= (B2 + C25*B5 + C26*B6)*H2;
      B3:= (B3 + C35*B5 + C36*B6)*H; B4:= (B4 + C45*B5 + C46*B6)*H2;
"END"EL.MATVECEVAL.;
L:= 1; W:= V:= 0; N2:= N + N - 2; XL1:= X[0]; XL:= X[1];
YA:= E[1]; ZA:= E[2]; YB:= E[3]; ZB:= E[4];
ELEMENTMATVECEVALUATION; EM[2]:= "-12;
R1:= B3 - A13*YA - A23*ZA; D1:= A33; D2:= A44;
R2:= B4 - A14*YA - A24*ZA; E1:= A34;
"FOR"L:= L + 1"WHILE"L<N"DO"
"BEGIN" XL1:= XL; XL:= X[L];
  ELEMENTMATVECEVALUATION;
  A[W + 1]:= D1 + A11; A[W + 4]:= E1 + A12;
  A[W + 7]:= A13; A[W + 10]:= A14;
  A[W + 5]:= D2 + A22; A[W + 8]:= A23;
  A[W + 11]:= A24; A[W + 14]:= 0;
  Y[V + 1]:= R1 + B1; Y[V + 2]:= R2 + B2;
  R1:= B3; R2:= B4; V:= V + 2; W:= W + 8;
  D1:= A33; D2:= A44; E1:= A34
"END";
L:= N; XL1:= XL; XL:= X[L]; ELEMENTMATVECEVALUATION;
Y[N2 - 1]:= R1 + B1 - A13*YB - A14*ZB;
  Y[N2]:= R2 + B2 - A23*YB - A24*ZB;
    A[W + 1]:= D1 + A11; A[W + 4]:= E1 + A12; A[W + 5]:= D2 + A22;
  CHLDECSOLBND(A, N2, 3, EM, Y)
"END" FEMHERM;
      "EOP"
```

AUTHORS: T.M.T.COOLEN AND R.PLOEGER.

INSTITUTE: MATHEMATICAL CENTRE.

RECEIVED: 740301.

BRIEF DESCRIPTION:

THIS SECTION CONTAINS TWO PROCEDURES :

RICHARDSON SOLVES A SYSTEM OF LINEAR EQUATIONS WITH A COEFFICIENT MATRIX HAVING POSITIVE REAL EIGENVALUES BY MEANS OF A NON-STATIONARY SECOND ORDER ITERATIVE METHOD: RICHARDSON'S METHOD. SINCE RICHARDSON'S METHOD IS PARTICULARLY SUITABLE FOR SOLVING A SYSTEM OF LINEAR EQUATIONS THAT IS OBTAINED BY DISCRETIZING A TWO-DIMENSIONAL ELLIPTIC BOUNDARY VALUE PROBLEM, THE PROCEDURE RICHARDSON IS PROGRAMMED IN SUCH A WAY THAT THE SOLUTION VECTOR IS GIVEN AS A TWO-DIMENSIONAL ARRAY U[J,L], LJ<=J<=UJ, LL<=L<=UL. THE COEFFICIENT MATRIX IS NOT STORED, BUT EACH ROW CORRESPONDING TO A PAIR (J,L) IS GENERATED WHEN NEEDED. RICHARSON CAN ALSO BE USED TO DETERMINE THE EIGENVALUE OF THE COEFFICIENT MATRIX CORRESPONDING TO THE DOMINANT EIGENFUNCTION.

ELIMINATION, USED IN CONNECTION WITH THE PROCEDURE RICHARDSON, (THIS SECTION) SOLVES A SYSTEM OF LINEAR EQUATIONS WITH A COEFFICIENT MATRIX HAVING POSITIVE REAL EIGENVALUES BY MEANS OF A NON-STATIONARY SECOND ORDER ITERATIVE METHOD, WHICH IS AN ACCELERATION OF RICHARDSON'S METHOD. IN GENERAL, ELIMINATION CANNOT BE USED BY ITSELF IN A SENSIBLE WAY. SINCE RICHARDSON'S METHOD AND ITS ACCELERATION ARE PARTICULARLY SUITABLE FOR SOLVING A SYSTEM OF LINEAR EQUATIONS THAT IS OBTAINED BY DISCRETIZING A TWO-DIMENSIONAL ELLIPTIC BOUNDARY VALUE PROBLEM, THE PROCEDURES RICHARDSON AND ELIMINATION ARE PROGRAMMED IN SUCH A WAY THAT THE SOLUTION VECTOR IS GIVEN AS A TWO-DIMENSIONAL ARRAY U[J,L], LJ<=J<=UJ, LL<=L<=UL. THE COEFFICIENT MATRIX IS NOT STORED, BUT EACH ROW CORSPONDING TO A PAIR (J,L) IS GENERATED WHEN NEEDED.

KEYWORDS:

DIFFERENTIAL EQUATION,
TWO-DIMENSIONAL BOUNDARY VALUE PROBLEM,
SYSTEM OF LINEAR EQUATIONS,
COEFFICIENT MATRIX HAVING POSITIVE REAL EIGENVALUES,
NON-STATIONARY SECOND ORDER ITERATIVE METHOD,
RICHARDSON'S METHOD,
ACCELERATION OF RICHARDSON'S METHOD,

SUBSECTION : RICHARDSON.

CALLING SEQUENCE:

THE HEADING OF THE PROCEDURE READS:
"PROCEDURE" RICHARDSON(U,LJ,UJ,LL,UL,INAP,RESIDUAL,A,B,N,DISCR,K,
RATECONV,DOMEIGVAL,OUT);
"VALUE" LJ,UJ,LL,UL,A,B;
"INTEGER" N,K,LJ,UJ,LL,UL;
"REAL" A,B,RATECONV,DOMIGVAL;
"BOOLEAN" INAP;
"ARRAY" U,DISCR;
"PROCEDURE" RESIDUAL, OUT;

THE MEANING OF THE FORMAL PARAMETERS IS:
U: <ARRAY IDENTIFIER>;
"ARRAY" U[LJ:UJ,LL:UL];
AFTER EACH ITERATION THE APPROXIMATE SOLUTION CALCULATED BY
THE PROCEDURE RICHARDSON IS STORED INTO U.
ENTRY: IF INAP IS CHOSEN TO BE "TRUE" THEN AN INITIAL
APPROXIMATION OF THE SOLUTION, OTHERWISE ARBITRARY;
EXIT: THE FINAL APPROXIMATION OF THE SOLUTION;
LJ,UJ: <ARITHMETIC EXPRESSION>;
LOWER AND UPPER BOUND FOR THE FIRST SUBSCRIPT OF U;
LL,UL: <ARITHMETIC EXPRESSION>;
LOWER AND UPPER BOUND FOR THE SECOND SUBSCRIPT OF U;
INAP: <BOOLEAN EXPRESSION>;
IF THE USER WISHES TO INTRODUCE AN INITIAL APPROXIMATION
INAP="TRUE" SHOULD BE CHOSEN; THE CHOICE INAP="FALSE" HAS
THE EFFECT THAT ALL COMPONENTS OF U ARE SET EQUAL TO 1
BEFORE THE FIRST ITERATION IS PERFORMED;
RESIDUAL: <PROCEDURE IDENTIFIER>;
THE HEADING OF THIS PROCEDURE READS :
"PROCEDURE" RESIDUAL(U); "ARRAY" U;
SUPPOSE THAT THE SYSTEM OF EQUATIONS AT HAND IS AU= F;
FOR ANY ENTRY U THE PROCEDURE RESIDUAL SHOULD CALCULATE
THE SO-CALLED RESIDUAL AU - F IN EACH POINT J,L, WHERE
LJ<=J<=UJ, LL<=L<=UL, AND SUBSTITUTE THESE VALUES IN THE
ARRAY U;
A,B: <ARITHMETIC EXPRESSION>;
IF ONE WISHES TO FIND THE SOLUTION OF THE BOUNDARY VALUE
PROBLEM, IN A AND B THE USER SHOULD GIVE A LOWER AND
UPPER BOUND FOR THE EIGENVALUES FOR WHICH THE CORRESPONDING
EIGENFUNCTIONS IN THE EIGENFUNCTION EXPANSION OF THE RESIDU
AL AU - F, WITH U = THE INITIAL APPROXIMATION, SHOULD BE
REDUCED; IF THE DOMINANT EIGENVALUE IS TO BE FOUND, ONE
SHOULD CHOOSE A GREATER THAN THIS EIGENVALUE (SEE HEADING
METHOD AND PERFORMANCE);

N: «ARITHMETIC EXPRESSION»;
N GIVES THE TOTAL NUMBER OF ITERATIONS TO BE PERFORMED; THE VALUE OF N SHOULD EITHER BE GIVEN, OR MADE DEPENDENT OF SOME JENSEN PARAMETER; E.G. K AND RATECONV CAN SERVE FOR THIS PURPOSE;

DISCR: «ARRAY IDENTIFIER»;
"ARRAY" DISCR[1:2];
AFTER EACH ITERATION THE PROCEDURE RICHARDSON DELIVERS IN DISCR[1] THE EUCLIDEAN NORM OF THE RESIDUAL, AND IN DISCR[2] THE MAXIMUM NORM OF THE RESIDUAL;

K: «VARIABLE»
K COUNTS THE NUMBER OF ITERATIONS RICHARDSON IS PERFORMING; IT CAN SERVE AS A JENSEN PARAMETER FOR N AND OUT;

RATECONV: «VARIABLE»;
AFTER EACH ITERATION THE AVERAGE RATE OF CONVERGENCE IS ASSIGNED TO RATECONV;

DOMEIGVAL: «VARIABLE»;
AFTER EACH ITERATION THE VALUE OF THE DOMINANT EIGENVALUE, IF PRESENT, IS ASSIGNED TO DOMEIGVAL; IF THERE IS NO DOMINANT EIGENVALUE, THE VALUE OF DOMEIGVAL IS MEANINGLESS, WHICH MANIFESTS ITSELF BY SHOWING NO CONVERGENCE TO A FIXED VALUE;

OUT: «PROCEDURE IDENTIFIER»;
THE HEADING OF THIS PROCEDURE, TO BE WRITTEN BY THE USER, READS :
"PROCEDURE" OUT(K); "VALUE" K; "INTEGER"K;
BY THIS PROCEDURE ONE HAS ACCESS TO THE FOLLOWING QUANTITIES:
FOR 0<=K<=N THE K-TH ITERAND IN U, THE EUCLIDEAN AND MAXIMUM NORM OF THE K-TH RESIDUAL IN DISCR[1] AND DISCR[2], RESPECTIVELY;
FOR 0<K<=N ALSO THE AVERAGE RATE OF CONVERGENCE AND THE APPROXIMATION TO THE DOMINANT EIGENVALUE, BOTH WITH RESPECT TO THE K-TH ITERAND U, IN RATECONV AND DOMEIGVAL, RESPECTIVELY;
MOREOVER, OUT CAN BE USED TO LET N BE DEPENDENT ON THE ACCURACY REACHED IN APPROXIMATING THE DOMINANT EIGENVALUE.

DATA AND RESULTS: SEE REF[1], [2].

PROCEDURES USED: NONE.

REQUIRED CENTRAL MEMORY:

EXECUTION FIELD LENGTH: CIRCA 75 + 3*(UJ - LJ + 1) * (UL - LL + 1).

RUNNING TIME:

DEPENDS STRONGLY ON THE BOUNDARY VALUE PROBLEM TO BE SOLVED.

LANGUAGE: ALGOL 60.

METHOD AND PERFORMANCE:

SUPPOSE THE SYSTEM OF EQUATIONS TO BE SOLVED READS AU = F, WHERE A IS A MATRIX HAVING POSITIVE REAL EIGENVALUES. DENOTING THE K-TH ITERATE BY U(K), U(K) BEING THE VECTOR U(K)[J,L], LJ<=J<=UJ, LL<=L<=UL, THE SO-CALLED RESIDUAL WITH RESPECT TO THE K-TH ITERATE IS DEFINED BY

R(K) = AU(K) - F.

A SECOND ORDER NON-STATIONARY ITERATIVE METHOD IS GIVEN BY

U(K+1) = BETA K * U(K) + (1 - BETA K) * U(K-1)
- OMEGA K * R(K),

OR, EQUIVALENTLY, UF U IS THE (UNKNOWN) EXACT SOLUTION OF AU = F,

U(K) - U = PK(A) (U(0) - U),

WHERE PK DENOTES A POLYNOMIAL OF DEGREE K. RICHARDSON'S METHOD CONSISTS OF CHOOSING THIS POLYNOMIAL IN SUCH A WAY THAT AMONGST ALL POLYNOMIALS PK(X) OF DEGREE K WITH PK(0)= 1 IT HAS MINIMAL MAXIMUM NORM OVER THE INTERVAL [C,D], WHERE C > 0 SHOULD BE CHOSEN TO BE A LOWER BOUND, AND D AN UPPER BOUND FOR THE EIGENVALUES OF A.
APPLICATION OF THIS POLYNOMIAL TO THE INITIAL ERROR U(0) - U HAS THE EFFECT THAT EACH COMPONENT OF THE INITIAL ERROR IN ITS EIGEN-FUNCTION EXPANSION IS REDUCED BY A FACTOR LESS OR EQUAL TO THE NORM OF THE POLYNOMIAL.
THE POLYNOMIALS

PK(X) = CK((A+B-2*X)/(A-B)) / CK((A+B)/(A-B))

WHERE CK(Y) DENOTES THE K-TH CHEBYSHEV POLYNOMIAL, HAVE THE DESIRED PROPERTIES. THUS, THE VALUES OF THE PARAMETERS BETA K AND OMEGA K MAY BE DETERMINED FROM THE RECURRENCE RELATIONS FOR CHEBESHEV POLYNOMIALS.
IN COMPUTATION U(K) - U IS NOT AVAILABLE, SO ONE USES R(K) AS A MEASURE FOR THE ERROR.

THE ELEMENTS OF THE MATRIX THE MATRIX A ARE NOT STORED, BUT GENERATED WHEN NEEDED. MORE PRECISELY, THIS MEANS THAT THE (UJ-LJ+1) * (UL-LL+1) COMPONENTS OF AU(K) - F ARE CALCULATED FOR EACH PAIR (J,L) LJ<J<UJ, LL<L<UL. THE USER SHOULD INTRODUCE THE EQUATION TO BE SOLVED IN THIS MANNER BY MEANS OF THE PROCEDURE RESIDUAL.
CLEARLY, THE METHOD IS PARTICULARLY SUITABLE FOR SPARSE MATRICES, FOR EXAMPLE MATRICES THAT ARE OBTAINED BY DISCRETIZING ELLIPTIC PARTIAL DIFFERENTIAL EQUATIONS.
THE SHARPER THE BOUNDS C AND D FOR THE EIGENVALUES OF A ARE, THE BETTER APPROXIMATE SOLUTION ONE GETS FOR A GIVEN VALUE OF K, SINCE THE ASYMPTOTIC RATE OF CONVERGENCE (K TO INFINITY) IS 2 * SQRT(C/D).

NOW LET ALPHA1 BE THE SMALLEST EIGENVALUE OF A. IF ONE CHOOSES C > ALPHA1, THEN, STARTING WITH ANY INITIAL APPROXIMATION, FOR A SUFFICIENTLY LARGE NUMBER OF ITERATIONS THE PROCEDURE RICHARDSON WILL DELIVER AN APPROXIMATE VALUE FOR THIS EIGENVALUE.

LET US EXPLAIN THIS FACT FOR THE CASE ALPHA1 < C < ALPHA2, WHERE ALPHA2 IS THE SECOND SMALLEST EIGENVALUE OF A, THE POLYNOMIAL PK(X) HAS SMALL MAXIMUM VALUE OVER THE INTERVAL [C,D] (WHICH, OF COURSE, DEPENDS ON K), BUT BECOMES LARGE FOR X < A. SO, IF ONE APPLIES PK(A) TO AN EIGENFUNCTION OF A, THIS EIGENFUNCTION WILL ONLY BE REDUCED CONSIDERABLY IF IT CORRESPONDS TO AN EIGENVALUE > C. CONSEQUENTLY, THE EIGENFUNCTION CORRESPONDING TO ALPHA1 WILL BECOME DOMINANT IN THE EIGENFUNCTION EXPANSION OF

PK(A) (U(0) - U)

FOR SUFFICIENTLY LARGE K.

SEE REF[1],[2] FOR DETAILS.

REFERENCES:

[1].T.M.T.COOLEN, P.W.HEMKER, P.J.VAN DER HOUWEN AND E.SLAGT.
ALGOL 60 PROCEDURES FOR INITIAL AND BOUNDARY VALUE PROBLEMS (DUTCH).
MC-SYLLABUS 20, MATHEMATICAL CENTRE, 1973, AMSTERDAM.

[2].P.J.VAN DER HOUWEN.
FINITE DIFFERENCE METHODS FOR SOLVING PARTIAL DIFFERENTIAL EQUATIONS.
MATHEMATICAL CENTRE TRACT NO. 20, 1968.

EXAMPLE OF USE:

THE APPROXIMATE SOLUTION OF THE BOUNDARY VALUE PROBLEM
- ((D/DX)**2 + (D/DY)**2) U(X,Y) = -2*(X*X+Y*Y), 0<X,Y<PI,
U(X,0) = 0, U(X,PI) = PI*PI*X*X, 0 < X < PI,
U(0,Y) = 0, U(PI,Y) = PI*PI*X*X, 0 < Y < PI,
WHICH HAS THE ANALYTICAL SOLUTION X*X*Y*Y, MAY BE OBTAINED BY THE FOLLOWING PROGRAM:

```
"BEGIN" "COMMENT" DIRICHLET PROBLEM FOR LAPLACE'S EQUATION;

  "PROCEDURE" RICHARDSON(U,LJ,UJ,LL,UL,INAP,RESIDUAL,A,B,N,DISCR,K,
  RATECONV,DOMEIGVAL,OUT); "CODE"33170;

  "PROCEDURE" RESIDUAL(U); "ARRAY" U;
  "BEGIN" "INTEGER" UJMIN1,ULMIN1,LJPLUS1;
  "REAL" U2; "REAL" "ARRAY" U1[LJ:UJ];
    UJMIN1:= UJ - 1; ULMIN1 := UL - 1; LJPLUS1:= LJ + 1;
    "FOR" J:= LJ "STEP" 1 "UNTIL" UJ "DO"
    "BEGIN" U1[J]:= U[J,LL]; U[J,LL]:= 0; "END";
```

```
    "FOR" L:= LL + 1 "STEP" 1 "UNTIL" ULMIN1 "DO"
    "BEGIN"  U1[LJ]:= U[LJ,L]; U[LJ,L]:= 0;
      "FOR"  J:= LJPLUS1"STEP" 1 "UNTIL" UJMIN1 "DO"
      "BEGIN" U2:= U[J,L];
        U[J,L]:=(4 * U2 - U1[J-1]  - U1[J] - U[J+1,L] - U[J,L+1])
        - F(J*H,L*H)*H2;
        U1[J]:= U2
      "END";
      U[UJ,L]:= 0;
    "END";
    "FOR" J:= LJ "STEP" 1 "UNTIL" UJ "DO" U[J,UL]:= 0
"END" RESIDUAL;

"REAL" "PROCEDURE" F(X,Y); "VALUE" X,Y; "REAL" X,Y;
F:= -2*(X*X + Y*Y);

"REAL" "PROCEDURE" ANALSOL(X,Y); "VALUE" X,Y; "REAL" X,Y;
ANALSOL:= X*X*Y*Y;

"PROCEDURE" INITAPPR(U,J,L,G); "INTEGER" J,L; "ARRAY" U; "REAL" G;
"FOR" J:= LJ "STEP" 1 "UNTIL" UJ "DO"
"FOR" L:= LL "STEP" 1 "UNTIL" UL "DO"
U[J,L]:= "IF" J=LJ "OR" J=UJ "OR" L=LL "OR" L=UL "THEN"G "ELSE" 1;

"PROCEDURE"OUT1(K); "VALUE" K; "INTEGER" K;
"IF" K = N "THEN" OUTPUT(61,"("//"(" K   DISCR[1]        DISCR[2]
RATECONV")",//,+ZDB,3(+.7D"+ZDB)")",K,DISCR[1],DISCR[2],RATECONV);
          2710=  "INTEGER" J,L,LJ,UJ,LL,UL,N,K;
"INTEGER" J,L,LJ,UJ,LL,UL,N,K;
"REAL" H,PI,D1,D2,H2,DOMEIGVAL,RATECONV,A,B;
"REAL" "ARRAY" DISCR[1:2];
OUTPUT(61,"("/"("GIVE LJ,UJ,LL,UL,N,A,B")"/")");
INPUT(60,"("")",LJ); INPUT(60,"("")",UJ);
INPUT(60,"("")",LL); INPUT(60,"("")",UL);
INPUT(60,"("")",N); INPUT(60,"("")",A); INPUT(60,"("")",B);

"BEGIN" "REAL" "ARRAY" U[LJ:UJ,LL:UL];
  PI:=3,1415 92653 58979; H:= PI/(UJ - LJ); H2:= H * H;
  INITAPPR(U,J,L,ANALSOL(J*H,L*H));
  RICHARDSON(U,LJ,UJ,LL,UL,"TRUE",RESIDUAL,A,B,N,DISCR,K,
  RATECONV ,DOMEIGVAL,OUT1);
 "END"
"END"
```

IT DELIVERS WITH
LJ = 0, UJ = 11, LL = 0, UL = 11, N = 50, A = ,163, B = 7,83
THE FOLLOWING RESULTS:

K	DISCR[1]	DISCR[2]	RATECONV
+50	+,1401828" -3	+,4666866" -4	+,2921718" +0

SUBSECTION : ELIMINATION.

CALLING SEQUENCE:

```
THE HEADING OF THE PROCEDURE READS:
"PROCEDURE" ELIMINATION(U,LJ,UJ,LL,UL,RESIDUAL,A,B,N,DISCR,K,
RATECONV,DOMEIGVAL,OUT);
"VALUE" LJ,UJ,LL,UL,A,B;
"INTEGER" N,K,LJ,UJ,LL,UL;
"REAL" A,B,RATECONV,DOMIGVAL;
"ARRAY" U,DISCR;
"PROCEDURE" RESIDUAL, OUT;

THE MEANING OF THE FORMAL PARAMETERS IS:
U:        <ARRAY IDENTIFIER>;
          "ARRAY" U[LJ:UJ,LL:UL];
          AFTER EACH ITERATION THE APPROXIMATE SOLUTION CALCULATED BY
          THE PROCEDURE ELIMINATION IS STORED INTO U;
          ENTRY: AN INITIAL APPROXIMATION OF THE SOLUTION, WHICH
          IS OBTAINED BY USE OF RICHARDSON;
          EXIT: THE FINAL APPROXIMATION OF THE SOLUTION;
LJ,UJ:    <ARITHMETIC EXPRESSION>;
          LOWER AND UPPER BOUND FOR THE FIRST SUBSCRIPT OF U;
LL,UL:    <ARITHMETIC EXPRESSION>;
          LOWER AND UPPER BOUND FOR THE SECOND SUBSCRIPT OF U;
RESIDUAL: <PROCEDURE IDENTIFIER>;
          THE HEADING OF THIS PROCEDURE READS :
          "PROCEDURE" RESIDUAL(U); "ARRAY" U;
          SUPPOSE THAT THE SYSTEM OF EQUATIONS AT HAND IS AU= F;
          FOR ANY ENTRY U THE PROCEDURE RESIDUAL SHOULD CALCULATE
          THE SO-CALLED RESIDUAL AU - F IN EACH POINT J,L, WHERE
          LJ<=J<=UJ, LL<=L<=UL, AND SUBSTITUTE THESE VALUES IN THE
          ARRAY U;
A,B:      <ARITHMETIC EXPRESSION>;
          A AND B SHOULD HAVE THE SAME VALUES AS IN THE PRECEDING
          CALL OF RICHARDSON (SEE DESCRIPTION OF RICHARDSON);
N:        <VARIABLE>;
          THE NUMBER OF ITERATIONS THE PROCEDURE ELIMINATION NEEDS
          TO ELIMINATE THE EIGENFUNCTION BELONGING TO THE DOMINANT
          EIGENVALUE, IS ASSIGNED TO N;
DISCR:    <ARRAY IDENTIFIER>;
          "ARRAY" DISCR[1:2];
          AFTER EACH ITERATION THE PROCEDURE  ELIMINATION DELIVERS
          IN DISCR[1] THE EUCLIDEAN NORM OF THE RESIDUAL, AND
          IN DISCR[2] THE MAXIMUM NORM OF THE RESIDUAL;
K:        <VARIABLE>
          K COUNTS THE NUMBER OF ITERATIONS ELIMINATION IS PERFORMING
          IT CAN SERVE AS A JENSEN PARAMETER FOR OUT;
RATECONV: <VARIABLE>;
          AFTER EACH ITERATION THE AVERAGE RATE OF CONVERGENCE IS
          ASSIGNED TO RATECONV;
```

DOMEIGVAL: <ARITHMETIC EXPRESSION>;
BEFORE A CALL OF ELIMINATION THE VALUE OF THE EIGENVALUE FOR WHICH THE CORRESPONDING EIGENFUNCTION HAS TO BE ELIMINATED, SHOULD BE ASSIGNED TO DOMEIGVAL; IF AFTER APPLICATION OF ELIMINATION THERE IS A NEW DOMINANT EIGENFUNCTION, THEN DOMEIGVAL WILL BE EQUAL TO THE CORRESPONDING EIGENVALUE; OTHERWISE, THE VALUE OF DOMEIGVAL BECOMES MEANINGLESS;

OUT: <PROCEDURE IDENTIFIER>;
THE HEADING OF THIS PROCEDURE, TO BE WRITTEN BY THE USER, READS :
"PROCEDURE" OUT(K); "VALUE" K; "INTEGER"K;
BY THIS PROCEDURE ONE HAS ACCESS TO THE FOLLOWING QUANTITIES:
FOR 0<=K<=N THE K-TH ITERAND IN U, THE EUCLIDEAN AND MAXIMUM NORM OF THE K-TH RESIDUAL IN DISCR[1] AND DISCR[2], RESPECTIVELY;
FOR 0<K<=N ALSO THE AVERAGE RATE OF CONVERGENCE WITH RESPECT TO THE K-TH ITERAND U, IN RATECONV;
FOR K = N, POSSIBLY THE DOMINANT EIGENVALUE OF THE COEFFICIENT MATRIX OF THE EQUATION AU= F, IN DOMEIGVAL.

DATA AND RESULTS: SEE REF[1],[2].

PROCEDURES USED:

RICHARDSON = CP33170,
TAN = CP35120,
TANH = CP35113,
ARCCOS = CP35122,
ZEROIN = CP34150.

REQUIRED CENTRAL MEMORY:

EXECUTION FIELD LENGTH: CIRCA 75 + 3*(UJ -LJ + 1) * (UL - LL + 1).

RUNNING TIME:

DEPENDS STRONGLY ON THE BOUNDARY VALUE PROBLEM TO BE SOLVED.

LANGUAGE: ALGOL 60.

METHOD AND PERFORMANCE:

SEE THIS HEADING IN THE DESCRIPTION OF THE PROCEDURE RICHARDSON, SOME ADDITIONAL REMARKS WILL BE MADE HERE.
IN ORDER TO USE ELIMINATION THE INITIAL APPROXIMATION OF THE SOLUTION OF

AU = F

IS FIRST TREATED BY MEANS OF RICHARDSON'S METHOD, WHERE C IS CHOSEN GREATER THAN THE SMALLEST EIGENVALUE. AFTER APPLICATION OF RICHARDSON, THE EIGENFUNCTION CORRESPONDING TO THIS EIGENVALUE HAS BECOME DOMINANT IN THE QUANTITY

PK(A) (U(0) - U),

WITH

PK(X) = CK((C+D-2*X)/(C-D)) / CK((C+D)/(C-D)),

WHEREAS THE CONTRIBUTION OF THE OTHER EIGENFUNCTIONS TO THE ERROR U(K) - U AND TO R(K) HAS BEEN REDUCED CONSIDERABLY. CONSEQUENTLY THE ERROR U(K) - U HAS VERY SMALL COMPONENTS IN THE SUBSPACE SPANNED BY ALL EIGENVECTORS BUT THE "FIRST", IN WHICH DIRECTION IT HAS A VERY LARGE COMPONENT.
THE CONTRIBUTION OF THE "FIRST" EIGENFUNCTION TO R(K) IS NOW "ELIMINATED" BY APPLICATION OF A POLYNOMIAL OPERATOR E(A) SUCH THAT E(X) HAS A ZERO IN THE FIRST EIGENVALUE.
THE POLYNOMIAL IS CHOSEN IN SUCH A WAY THAT A MAXIMAL RATE OF CONVERGENCE WITH RESPECT TO THE INITIAL APPROXIMATION USED IN RICHARDSON IS OBTAINED.

FOR DETAILS SEE REF[1],[2].

REFERENCES:

[1].T.M.T.COOLEN, P.W.HEMKER, P.J.VAN DER HOUWEN AND E.SLAGT.
ALGOL 60 PROCEDURES FOR INITIAL AND BOUNDARY VALUE PROBLEMS (DUTCH).
MC-SYLLABUS 20, MATHEMATICAL CENTRE, 1973, AMSTERDAM.

[2].P.J.VAN DER HOUWEN.
FINITE DIFFERENCE METHODS FOR SOLVING PARTIAL DIFFERENTIAL EQUATIONS.
MATHEMATICAL CENTRE TRACT NO. 20, 1968.

EXAMPLE OF USE:.

THE APPROXIMATE SOLUTION OF THE BOUNDARY VALUE PROBLEM
- ((D/DX)**2 + (D/DY)**2) U(X,Y) = -2*(X*X+Y*Y), 0<X,Y<PI,
U(X,0) = 0, U(X,PI) = PI*PI*X*X, 0 < X < PI,
U(0,Y) = 0, U(PI,Y) = PI*PI*X*X, 0 < Y < PI,
WHICH HAS THE ANALYTICAL SOLUTION X*X*Y*Y, MAY BE OBTAINED BY THE FOLLOWING PROGRAM:

```
"BEGIN" "COMMENT" DIRICHLET PROBLEM FOR LAPLACE'S EQUATION;

  "PROCEDURE" RICHARDSON(U,LJ,UJ,LL,UL,INAP,RESIDUAL,A,B,DISCR,K,
  RATECONV,DOMEIGVAL,OUT); "CODE"33170;
"PROCEDURE" ELIMINATION(U,LJ,UJ,LL,UL,RESIDUAL,A,B,DISCR,K,
RATECONV,DOMEIGVAL,OUT); "CODE"33171;

"PROCEDURE" RESIDUAL(U); "ARRAY" U;
"BEGIN" "INTEGER" UJMIN1,ULMIN1,LJPLUS1;
"REAL" U2; "REAL" "ARRAY" U1[LJ:UJ];
  UJMIN1:= UJ - 1; ULMIN1 := UL - 1; LJPLUS1:= LJ + 1;
  "FOR" J:= LJ "STEP" 1 "UNTIL" UJ "DO"
  "BEGIN" U1[J]:= U[J,LL]; U[J,LL]:= 0; "END";
  "FOR" L:= LL + 1 "STEP" 1 "UNTIL" ULMIN1 "DO"
  "BEGIN"  U1[LJ]:= U[LJ,L]; U[LJ,L]:= 0;
    "FOR"  J:= LJPLUS1"STEP" 1 "UNTIL" UJMIN1 "DO"
    "BEGIN" U2:= U[J,L];
      U[J,L]:=(4 * U2 - U1[J-1]  - U1[J] - U[J+1,L] - U[J,L+1])
      - F(J*H,L*H)*H2;
      U1[J]:= U2
    "END";
    U[UJ,L]:= 0;
  "END";
  "FOR" J:= LJ "STEP" 1 "UNTIL" UJ "DO" U[J,UL]:= 0
"END" RESIDUAL;

"REAL" "PROCEDURE" F(X,Y); "VALUE" X,Y; "REAL" X,Y;
F:= -2*(X*X + Y*Y);

"REAL" "PROCEDURE" ANALSOL(X,Y); "VALUE" X,Y; "REAL" X,Y;
ANALSOL:= X*X*Y*Y;

"PROCEDURE" INITAPPR(U,J,L,G); "INTEGER" J,L; "ARRAY" U; "REAL" G;
"FOR" J:= LJ "STEP" 1 "UNTIL" UJ "DO"
"FOR" L:= LL "STEP" 1 "UNTIL" UL "DO"
U[J,L]:= "IF" J=LJ "OR" J=UJ "OR" L=LL "OR" L=UL "THEN"G "ELSE" 1;

"PROCEDURE"OUT3(K); "VALUE" K; "INTEGER" K;
"IF" K=P "THEN" OUTPUT(61,"("//,+ZDB,3(+,7D"+ZDB)")",K,DISCR[1],
DISCR[2],RATECONV);

"PROCEDURE"OUT1(K); "VALUE" K; "INTEGER" K;
"IF" K=N "THEN" OUTPUT(61,"("//"("  K   DISCR[1]       DISCR[2]       R
TECONV")",//,+ZDB,3(+,7D"+ZDB)")",K,DISCR[1],DISCR[2],RATECONV);

"PROCEDURE" OUT2(K); "VALUE" K; "INTEGER" K;
"BEGIN"
  "IF" K = 0 "THEN" D1:= D2:= 1 "ELSE"
  "BEGIN" D2:= D1; D1:= DOMEIGVAL;
    N:= "IF" ABS((D1 - D2)/D2) < 10**(-Q) "THEN" K "ELSE" NN;
    OUT1(K)
  "END"
"END" OUT2;
```

```
"INTEGER" J,L,LJ,UJ,LL,UL,NN,N,P,K,Q;
"REAL" H,PI,D1,D2,H2,RATECONVR,RATECONVE,DOMEIGVAL,RATECONV,A,B,VAR;
"REAL" "ARRAY" DISCR[1:2];
OUTPUT(61,"("/"("GIVE LJ,UJ,LL,UL,N,Q,A,B")"/")");
INPUT(60,"("")",LJ); INPUT(60,"("")",UJ);
INPUT(60,"("")",LL); INPUT(60,"("")",UL);
INPUT(60,"("")", N); INPUT(60,"("")", Q);
INPUT(60,"("")", A); INPUT(60,"("")", B);

"BEGIN" "REAL" "ARRAY" U[LJ:UJ,LL:UL];
  PI:=3.1415 92653 58979; H:= PI/(UJ - LJ); H2:= H * H;
  INITAPPR(U,J,L,ANALSOL(J*H,L*H));
  NN:= N;
  RICHARDSON(U,LJ,UJ,LL,UL,"TRUE",RESIDUAL,A,B,N,DISCR,K,
  RATECONV ,DOMEIGVAL,OUT2); RATECONVR:= RATECONV;
  OUTPUT(61,"("//+.7D"+ZD4B"("DOMINANT EIGENVALUE")""")",DOMEIGVAL);
  ELIMINATION(U,LJ,UJ,LL,UL,RESIDUAL,A ,B,P,DISCR,K,
  RATECONV ,DOMEIGVAL,OUT3); RATECONVE:= RATECONV;
  NN:= N + P; OUTPUT(61,"("//+Z2D13B"("TOTAL NUMBER OF ITERATIONS")"
  ")",NN);
    OUTPUT(61,"("/+.7D"+ZD4B"("RATE OF CONVERGENCE WITH RESPECT TO")",
    /17B"("THE ZEROTH ITERAND OF RICHARDSON")""")",
    (N * RATECONVR + P * RATECONVE)/NN);
  "END"
"END"
```

```
    IT DELIVERS WITH
    LJ = 0, UJ = 11, LL = 0, UL = 11, N = 50, Q = 4, A = .326, B = 7.83
    THE FOLLOWING RESULTS:


   K    DISCR[1]        DISCR[2]        RATECONV

 +45 +.4998463" -1 +.8903863" -2 +.2009943" +0

 +.1620445" +0     DOMINANT EIGENVALUE

  +7 +.3563865" -5 +.6714375" -6 +.1360086" +1

  +52                TOTAL NUMBER OF ITERATIONS
+.3570259" +0        RATE OF CONVERGENCE WITH RESPECT TO
                     THE ZEROTH ITERAND OF RICHARDSON
```

SOURCE TEXT(S):

```
"CODE"33170;
  "PROCEDURE" RICHARDSON(U,LJ,UJ,LL,UL,INAP,RESIDUAL,A,B,N,DISCR,K,
  RATECONV,DOMEIGVAL,OUT); "VALUE" LJ,UJ,LL,UL,A,B;
  "INTEGER" N,K,LJ,UJ,LL,UL; "REAL" A,B,RATECONV,DOMEIGVAL; "BOOLEAN"
  INAP; "ARRAY" U,DISCR; "PROCEDURE" RESIDUAL,OUT;
  "BEGIN" "INTEGER" J,L; "REAL" X,Y,Z,Y0,C,D,ALFA,OMEGA,OMEGA0,
    EIGMAX,EIGEUCL,EUCLRES,MAXRES,RCMAX,RCEUCL,MAXRES0,EUCLRES0;
    "ARRAY" V,RES[LJ:UJ,LL:UL];
    "PROCEDURE" CALPAR;
    "COMMENT" CALPAR CALCULATES THE PARAMETERS ALFA AND OMEGA FOR
    EACH ITERATION;
    "BEGIN" ALFA:= Z/(Z - ALFA);
      OMEGA:= 1/(X - OMEGA * Y)
    "END" CALPAR;
    "PROCEDURE" ITERATION;
    "COMMENT" FIRST THE ITERATION FORMULA IS CONSTRUCTED;
    "BEGIN" "REAL" AUXV,AUXU,AUXRES,EUCLUV,MAXUV;
      EUCLUV:= EUCLRES:= MAXUV:= MAXRES:= 0;
    "FOR" J:= LJ "STEP" 1 "UNTIL" UJ "DO"
    "FOR" L:= LL "STEP" 1 "UNTIL" UL "DO" RES[J,L]:= V[J,L];
    RESIDUAL(RES);
    "FOR" J:= LJ "STEP" 1 "UNTIL" UJ "DO"
    "FOR" L:= LL "STEP" 1 "UNTIL" UL "DO"
    "BEGIN" AUXV:= U[J,L]; AUXU:= V[J,L]; AUXRES:= RES[J,L];
      AUXV:= ALFA * AUXU - OMEGA * AUXRES + (1 - ALFA) * AUXV;
      V[J,L]:= AUXV; U[J,L]:= AUXU;
      "COMMENT" THE NORMS OF THE K-TH RESIDUAL AND THE DIFFERENCE
      BETWEEN THE (K+1)-TH AND K-TH ITERAND ARE CALCULATED;
      AUXU:= ABS(AUXU - AUXV); AUXRES:= ABS(AUXRES);
      MAXUV:= "IF" MAXUV < AUXU "THEN" AUXU "ELSE" MAXUV;
      MAXRES:= "IF" MAXRES < AUXRES "THEN" AUXRES "ELSE" MAXRES;
      EUCLUV:= EUCLUV + AUXU * AUXU;
      EUCLRES:= EUCLRES + AUXRES * AUXRES;
    "END";
    EUCLUV:= SQRT(EUCLUV); EUCLRES:= SQRT(EUCLRES);
    DISCR[1]:= EUCLRES; DISCR[2]:= MAXRES;
    "COMMENT" DOMEIGVAL IS EVALUATED;
      MAXUV:= MAXRES/MAXUV; EUCLUV:= EUCLRES/EUCLUV;
      EIGMAX:= MAXUV * (C - MAXUV)/(.25 * D - MAXUV);
      EIGEUCL:= EUCLUV * (C - EUCLUV)/(.25 * D - EUCLUV);
      DOMEIGVAL:= .5 * (EIGMAX + EIGEUCL);
      "COMMENT" FINALLY THE RATE OF CONVERGENCE IS CALCULATED;
      RCEUCL:= -LN(EUCLRES/EUCLRES0)/K;
      RCMAX:= -LN(MAXRES/MAXRES0)/K;
      RATECONV:= .5 * (RCEUCL + RCMAX)
  "END" ITERATION;                                        "COMMENT"
```

```
"COMMENT" THE CONSTANTS FOR STARTING CALPAR ARE CALCULATED;
ALFA:= 2; OMEGA:= 4/(B + A); Y0:= (B + A)/(B - A);
X:= .5 * (B + A); Y:= (B - A) * (B - A)/16; Z:= 4 * Y0 * Y0;
"COMMENT" THE CONSTANTS NEEDED FOR DOMEIGVAL ARE CALCULATED;
C:= A * B; C:= SQRT(C); D:= SQRT(A) + SQRT(B); D:= D * D;
"COMMENT" THE INITIAL APPROXIMATION IS PUT INTO ARRAY U;
"IF" ^INAP "THEN"
"BEGIN" "FOR" J:= LJ "STEP" 1 "UNTIL" UJ "DO"
  "FOR" L:= LL "STEP" 1 "UNTIL" UL "DO" U[J,L]:= 1
"END";
"COMMENT" THE ZEROTH ITERATION IS NOW PERFORMED;
K:= 0;
"FOR" J:= LJ "STEP" 1 "UNTIL" UJ "DO"
"FOR" L:= LL "STEP" 1 "UNTIL" UL "DO" RES[J,L]:= U[J,L];
RESIDUAL(RES);
OMEGA0:= 2/(B+A);
"BEGIN" "REAL" AUXRES0;
  MAXRES0:= EUCLRES0:= 0;
  "FOR" J:= LJ "STEP" 1 "UNTIL" UJ "DO"
    "FOR" L:= LL "STEP" 1 "UNTIL" UL "DO"
    "BEGIN" AUXRES0:= RES[J,L];
      V[J,L]:= U[J,L] - OMEGA0 * AUXRES0;
      AUXRES0:= ABS(AUXRES0);
      MAXRES0:= "IF" MAXRES0 < AUXRES0 "THEN" AUXRES0 "ELSE" MAXRES0;
      EUCLRES0:= EUCLRES0 + AUXRES0 * AUXRES0
    "END";
  EUCLRES0:= SQRT(EUCLRES0)
  "END";
  DISCR[1]:= EUCLRES0; DISCR[2]:= MAXRES0;
  OUT(K);
  "IF" K >= N "THEN" "GOTO" FINALLY;
NEXT STEP;
  K:= K + 1; CALPAR; ITERATION; OUT(K);
  "IF" K < N "THEN" "GOTO" NEXT STEP;
FINALLY:
"END" RICHARDSON;
      "EOP"
```

```
"CODE"33171;
  "PROCEDURE" ELIMINATION(U,LJ,UJ,LL,UL,RESIDUAL,A,B,N,DISCR,K,RATECONV
  DOMEIGVAL,OUT); "VALUE" LJ,UJ,LL,UL,A,B; "INTEGER" LJ,UJ,LL,UL,N,K;
  "REAL" A,B,RATECONV,DOMEIGVAL; "ARRAY" U,DISCR;
"PROCEDURE" RESIDUAL,OUT;
"BEGIN" "REAL" PI,AUXCOS,C,D;
  "REAL" "PROCEDURE" ARCCOS(X); "CODE" 35122;
  "REAL" "PROCEDURE" TAN(X); "CODE" 35120;
  "REAL" "PROCEDURE" TANH(X); "CODE" 35113;
  "PROCEDURE" RICHARDSON(U,LJ,UJ,LL,UL,INAP,RESIDUAL,A,B,N,DISCR,
  K,RATECONV,DOMEIGVAL,OUT); "CODE"33170;
  "BOOLEAN" "PROCEDURE" ZEROIN(X,Y,FX,TOLX); "CODE"34150;
  "REAL" "PROCEDURE" OPTPOL(X); "VALUE" X; "REAL" X;
  "BEGIN" "REAL" W,Y;
    W:= (B * COS(.5*PI/X) + DOMEIGVAL) / (B - DOMEIGVAL);
    "IF" W < -1 "THEN" W:= -1;
    "IF" ABS(W) <= 1 "THEN"
    "BEGIN" Y:= ARCCOS(W);
      OPTPOL:= 2 * SQRT(A/B) + TAN(X*Y) *
      (Y - B*PI*SIN(.5*PI/X)*.5 / (X * (B-DOMEIGVAL) *
      SQRT(ABS(1-W*W))))
    "END" "ELSE"
    "BEGIN" Y:= LN(W + SQRT(ABS(W*W-1)));
      OPTPOL:= 2 * SQRT(A/B) - TANH(X*Y) * (Y + B*PI*SIN(.5*PI/X)*
      .5/(X*(B-DOMEIGVAL)*SQRT(ABS(W*W-1))))
    "END"
 "END" OPTPOL;
 PI:= 3,1415 92653 58979;
 C:= 1;
 "IF" OPTPOL(C) < 0 "THEN"
"BEGIN" D:= .5 * PI * SQRT(ABS(B/DOMEIGVAL));
 M: D:= D + D;
  "IF" ZEROIN(C,D,OPTPOL(C),C*"-3) "THEN" N:= ENTIER(C+.5)
  "ELSE" "GOTO" M;
 "END" "ELSE" N:= 1;
 AUXCOS:= COS(.5*PI/N);
 RICHARDSON(U,LJ,UJ,LL,UL,"TRUE",RESIDUAL,
 (2*DOMEIGVAL + B*(AUXCOS-1))/(AUXCOS+1),B,N,DISCR,K,RATECONV,
 DOMEIGVAL,OUT)
"END" ELIMINATION;
      "EOP"
```

AUTHOR : B. VAN DOMSELAAR.

INSTITUTE: MATHEMATICAL CENTRE.

RECEIVED: 750601.

BRIEF DESCRIPTION:

PEIDE ESTIMATES UNKNOWN VARIABLES IN A SYSTEM OF FIRST ORDER DIFFERENTIAL EQUATIONS; THE UNKNOWN VARIABLES MAY APPEAR NONLINEAR BOTH IN THE DIFFERENTIAL EQUATIONS AND ITS INITIAL VALUES; A SET OF OBSERVED VALUES OF SOME COMPONENTS OF THE SOLUTION OF THE DIFFERENTIAL EQUATIONS MUST BE GIVEN;

KEYWORDS:

PARAMETER ESTIMATION,
DIFFERENTIAL EQUATIONS,
INITIAL VALUE PROBLEM,
DATA FITTING.

CALLING SEQUENCE:

THE HEADING OF THIS PROCEDURE IS:

```
"PROCEDURE" PEIDE(N, M, NOBS, NBP, PAR, RV, BP, JTJINV, IN, OUT,
   DERIV, JAC DFDY, JACDFDP, CALL YSTART, DATA, MONITOR);
"VALUE" N,M,NOBS; "INTEGER" N,M,NOBS,NBP;
"ARRAY" PAR,RV,JTJINV,IN,OUT; "INTEGER" "ARRAY" BP;
"PROCEDURE" CALL YSTART,DATA,MONITOR;
"BOOLEAN" "PROCEDURE" DERIV,JAC DFDY,JAC DFDP;
```

THE MEANING OF THE FORMAL PARAMETERS IS:

N: <ARITHMETIC EXPRESSION>;
THE NUMBER OF DIFFERENTIAL EQUATIONS;

M: <ARITHMETIC EXPRESSION>;
THE NUMBER OF UNKNOWN VARIABLES;

NOBS: <ARITHMETIC EXPRESSION>;
THE NUMBER OF OBSERVATIONS; NOBS SHOULD SATISFY NOBS>=M;

NBP: <VARIABLE>;
ENTRY: THE NUMBER OF BREAK-POINTS; IF NO BREAK-POINTS ARE USED THEN SET NBP=0;
EXIT: WITH NORMAL TERMINATION OF THE PROCESS NBP=0; OTHERWISE, IF THE PROCESS HAS BEEN BROKEN OFF (SEE OUT[1]), THE VALUE OF NBP IS THE NUMBER OF BREAK-POINTS USED BEFORE THE PROCESS BROKE OFF;

```
PAR:        <ARRAY IDENTIFIER>;
            "ARRAY" PAR[1 : M+NBP];
            ENTRY: PAR[1:M] SHOULD CONTAIN AN INITIAL APPROXIMATION
                   TO THE REQUIRED PARAMETER VECTOR;
            EXIT:  PAR[1:M] CONTAINS THE CALCULATED PARAMETER VECTOR;
                   IF OUT[1]>0 AND NBP>0 THEN PAR[M+1:M+NBP] CONTAINS
                   THE VALUES OF THE NEWLY INTRODUCED PARAMETERS
                   BEFORE THE PROCESS BROKE OFF;
RV:         <ARRAY IDENTIFIER>;
            "ARRAY" RV[1 : NOBS+NBP];
            EXIT:  RV[1:NOBS] CONTAINS THE RESIDUAL VECTOR AT THE
                   CALCULATED MINIMUM; IF OUT[1]>0 AND NBP>0 THEN
                   RV[NOBS+1:NOBS+NBP] CONTAINS THE ADDITIONAL
                   CONTINUITY REQUIREMENTS AT THE BREAK-POINTS BEFORE
                   THE PROCESS BROKE OFF;
BP:         <ARRAY IDENTIFIER>;
            "INTEGER" "ARRAY" BP[0 : NBP];
            ENTRY: BP[I], I=1,...,NBP, SHOULD CORRESPOND TO THE INDEX
                   OF THAT TIME OF OBSERVATION WHICH WILL BE USED AS
                   A BREAK-POINT (1<=BP[I]<=NOBS); THE BREAK-POINTS
                   HAVE TO BE ORDERED SUCH THAT BP[I]<=BP[J] IF I<=J;
            EXIT:  WITH NORMAL TERMINATION OF THE PROCESS BP[1:NBP]
                   CONTAINS NO INFORMATION; OTHERWISE, IF OUT[1]>0
                   AND NBP>0 THEN BP[I], I=1,...,NBP, CONTAINS THE
                   INDEX OF THAT TIME OF OBSERVATION WHICH WAS USED
                   AS A BREAK-POINT BEFORE THE PROCESS BROKE OFF;
JTJINV:     <ARRAY IDENTIFIER>;
            "ARRAY" JTJINV[1 : M, 1 : M];
            EXIT:  THE INVERSE OF THE MATRIX  J' * J  WHERE J DENOTES
                   THE MATRIX OF PARTIAL DERIVATIVES DRV[I] / DPAR[K]
                   (I=1,...,NOBS ; K=1,...,M) AND J' DENOTES THE
                   TRANSPOSE OF J; THIS MATRIX CAN BE USED IF
                   ADDITIONAL INFORMATION ABOUT THE RESULT IS
                   REQUIRED; E.G. STATISTICAL DATA SUCH AS THE
                   COVARIANCE MATRIX, CORRELATION MATRIX AND
                   CONFIDENCE INTERVALS CAN EASILY BE CALCULATED FROM
                   JTJINV AND OUT[2];
IN:         <ARRAY IDENTIFIER>;
            "ARRAY" IN[0 : 6];
            ENTRY: IN THIS ARRAY THE USER SHOULD GIVE SOME DATA TO
                   CONTROL THE PROCESS;
            IN[0]: THE MACHINE PRECISION;
                   FOR THE CYBER 73 A SUITABLE VALUE IS "-14;
            IN[1]: THE RATIO: THE MINIMAL STEPLENGTH FOR THE
                   INTEGRATION OF THE DIFFERENTIAL EQUATIONS DIVIDED
                   BY THE DISTANCE BETWEEN TWO NEIGHBOURING
                   OBSERVATIONS; MOSTLY, A SUITABLE VALUE IS "-4;
            IN[2]: THE RELATIVE LOCAL ERROR BOUND FOR THE
                   INTEGRATION PROCESS; THIS VALUE SHOULD SATISFY
                   IN[2]<=IN[3]; THIS PARAMETER CONTROLS THE
                   ACCURACY OF THE NUMERICAL INTEGRATION; MOSTLY,
                   A SUITABLE VALUE IS IN[3]/100;
```

IN[3], IN[4]:
THE RELATIVE AND THE ABSOLUTE TOLERANCE FOR THE DIFFERENCE BETWEEN THE EUCLIDEAN NORM OF THE ULTIMATE AND PENULTIMATE RESIDUAL VECTOR RESPECTIVELY;
THE PROCESS IS TERMINATED IF THE IMPROVEMENT OF THE SUM OF SQUARES IS LESS THAN
IN[3] * (SUM OF SQUARES) + IN[4] * IN[4];
THESE TOLERANCES SHOULD BE CHOSEN IN ACCORDANCE WITH THE RELATIVE, RESP. ABSOLUTE ERRORS IN THE OBSERVATIONS;
NOTE THAT THE EJCLIDEAN NORM OF THE RESIDUAL VECTOR IS DEFINED AS THE SQUARE ROOT OF THE SUM OF SQUARES;

IN[5]: THE MAXIMUM NUMBER OF TIMES THAT THE INTEGRATION OF THE DIFFERENTIAL EQUATIONS IS PERFORMED;

IN[6]: A STARTING VALUE USED FOR THE RELATION BETWEEN THE GRADIENT AND THE GAUSS-NEWTON DIRECTION (SEE [1]); IF THE PROBLEM IS WELL CONDITIONED THEN A SUITABLE VALUE FOR IN[6] WILL BE 0.01; IF THE PROBLEM IS ILL CONDITIONED THEN IN[6] SHOULD BE GREATER, BUT THE VALUE OF IN[6] SHOULD SATISFY:
IN[0] < IN[6] <= 1/IN[0];

OUT: <ARRAY IDENTIFIER>;
"ARRAY" OUT[1 : 7];
EXIT : IN ARRAY OUT SOME BY-PRODUCTS ARE DELIVERED;
OUT[1]: THIS VALUE GIVES INFORMATION ABOUT THE TERMINATION OF THE PROCESS;
OUT[1]=0: NORMAL TERMINATION;
IF OUT[1]>0 THEN THE PROCESS HAS BEEN BROKEN OFF AND THIS MAY OCCUR BECAUSE OF THE FOLLOWING REASONS:
OUT[1]=1: THE NUMBER OF INTEGRATIONS PERFORMED EXCEEDED THE NUMBER GIVEN IN IN[5];
OUT[1]=2: THE DIFFERENTIAL EQUATIONS ARE VERY NONLINEAR; DURING AN INTEGRATION THE VALUE OF IN[1] WAS DECREASED BY A FACTOR 10000 AND IT IS ADVISED TO DECREASE IN[1], ALTHOUGH THIS WILL INCREASE COMPUTING TIME;
OUT[1]=3: A CALL OF DERIV DELIVERED THE VALUE FALSE;
OUT[1]=4: A CALL OF JAC DFDY DELIVERED THE VALUE FALSE;
OUT[1]=5: A CALL OF JAC DFDP DELIVERED THE VALUE FALSE;
OUT[1]=6: THE PRECISION ASKED FOR CAN NOT BE ATTAINED; THIS PRECISION IS POSSIBLY CHOSEN TOO HIGH, RELATIVE TO THE PRECISION IN WHICH THE RESIDUAL VECTOR IS CALCULATED (SEE IN[3]);
OUT[2]: THE EUCLIDEAN NORM OF THE RESIDUAL VECTOR CALCULATED WITH VALUES OF THE UNKNOWNS DELIVERED;

OUT[3]: THE EUCLIDEAN NORM OF THE RESIDUAL VECTOR
CALCULATED WITH THE INITIAL VALUES OF THE
UNKNOWN VARIABLES;
OUT[4]: THE NUMBER OF INTEGRATIONS PERFORMED, NEEDED TO
OBTAIN THE CALCULATED RESULT; IF OUT[4]=1 AND
OUT[1]>0 THEN THE MATRIX JTJINV CAN NOT BE USED;
OUT[5]: THE MAXIMUM NUMBER OF TIMES THAT THE REQUESTED
LOCAL ERROR BOUND WAS EXCEEDED IN ONE
INTEGRATION; IF IT IS A LARGE NUMBER, IT MAY BE
BETTER TO DECREASE THE VALUE OF IN[1];
OUT[6]: THE IMPROVEMENT OF THE EUCLIDEAN NORM OF THE
RESIDUAL VECTOR IN THE LAST ITERATION STEP OF THE
PROCESS OF MARQUARDT;
OUT[7]: THE CONDITION NUMBER OF J' * J , I.E. THE RATIO
OF ITS LARGEST TO SMALLEST EIGENVALUES;

DERIV: <PROCEDURE IDENTIFIER>;
THIS PROCEDURE DEFINES THE RIGHT HAND SIDE OF THE
DIFFERENTIAL EQUATIONS;
THE HEADING OF THIS PROCEDURE SHOULD BE:
"BOOLEAN" "PROCEDURE" DERIV(PAR, Y, T, DF); "VALUE" T;
"REAL" T; "ARRAY" PAR,Y,DF;
ENTRY: PAR,Y,T;
PAR[1:M] CONTAINS THE CURRENT VALUES OF THE
UNKNOWNS AND SHOULD NOT BE ALTERED;
Y[1:N] CONTAINS THE SOLUTIONS OF THE DIFFERENTIAL
EQUATIONS AT TIME T AND SHOULD NOT BE ALTERED;
EXIT: "ARRAY" DF[1 : N];
AN ARRAY ELEMENT DF[I] SHOULD CONTAIN THE RIGHT
HAND SIDE OF THE I-TH DIFFERENTIAL EQUATION;
AFTER A SUCCESSFUL CALL OF DERIV, THE BOOLEAN PROCEDURE
SHOULD DELIVER THE VALUE TRUE;
HOWEVER, IF DERIV DELIVERS THE VALUE FALSE, THEN THE
PROCESS IS TERMINATED (SEE OUT[1]);
HENCE, PROPER PROGRAMMING OF DERIV MAKES IT POSSIBLE TO
AVOID CALCULATION OF THE RIGHT HAND SIDE WITH VALJES OF
THE UNKNOWN VARIABLES WHICH CAUSE OVERFLOW IN THE
COMPUTATION;

JAC DFDY: <PROCEDURE IDENTIFIER>;
THE HEADING OF THIS PROCEDURE SHOULD BE:
"BOOLEAN" "PROCEDURE" JAC DFDY(PAR, Y, T, FY); "VALUE" T;
"REAL" T; "ARRAY" PAR,Y,FY;
ENTRY: PAR,Y,T;
SEE DERIV;
EXIT: "ARRAY" FY[1 : N,1 : N];
AN ARRAY ELEMENT FY[I,J] SHOULD CONTAIN THE
PARTIAL DERIVATIVE OF THE RIGHT HAND SIDE OF THE
I-TH DIFFERENTIAL EQUATION WITH RESPECT TO Y[J],
I.E. DF[I]/DY[J];
THE BOOLEAN VALUE SHOULD BE ASSIGNED TO THIS PROCEDURE
IN THE SAME WAY AS IT IS DONE FOR THE VALUE OF DERIV;

JAC DFDP: <PROCEDURE IDENTIFIER>;
THE HEADING OF THIS PROCEDURE SHOULD BE:
"BOOLEAN" "PROCEDURE" JAC DFDP(PAR, Y, T, FP); "VALUE" T;
"REAL" T; "ARRAY" PAR,Y,FP;

```
                ENTRY: PAR,Y,T;
                       SEE DERIV;
                EXIT:  "ARRAY" FP[1 : N,1 : M];
                       AN ARRAY ELEMENT FP[I,J] SHOULD CONTAIN THE
                         PARTIAL DERIVATIVE OF THE RIGHT HAND SIDE OF THE
                         I-TH DIFFERENTIAL EQUATION WITH RESPECT TO PAR[J],
                         I.E. DF[I]/DPAR[J];
                     THE BOOLEAN VALUE SHOULD BE ASSIGNED TO THIS PROCEDURE
                     IN THE SAME WAY AS IT IS DONE FOR THE VALUE OF DERIV;
          CALL YSTART: <PROCEDURE IDENTIFIER>;
                     THIS PROCEDURE DEFINES THE INITIAL VALUES OF THE INITIAL
                     VALUE PROBLEM;
                     THE HEADING OF THIS PROCEDURE SHOULD BE:
                     "BOOLEAN" "PROCEDURE" CALL YSTART(PAR, Y, YMAX);
                     "ARRAY" PAR,Y,YMAX;
                     ENTRY: PAR;
                            PAR[1:M] CONTAINS THE CURRENT VALUES OF THE
                            UNKNOWN VARIABLES AND SHOULD NOT BE ALTERED;
                     EXIT:  Y,YMAX;
                            Y[1:N] SHOULD CONTAIN THE INITIAL VALUES OF THE
                            CORRESPONDING DIFFERENTIAL EQUATIONS;
                            THE INITIAL VALUES MAY BE FUNCTIONS OF THE UNKNOWN
                            VARIABLES PAR; IN THAT CASE, THE INITIAL VALUES OF
                            DY/DPAR ALSO HAVE TO BE SUPPLIED; NOTE THAT
                            DY[I]/DPAR[J] CORRESPONDS WITH  Y[5*N+I*N+J]
                            (I=1,...,N , J=1,...,M);
                            YMAX[I], I=1,...,N, SHOULD CONTAIN A ROUGH
                            ESTIMATE TO THE MAXIMAL ABSOLUTE VALUE OF Y[I]
                            OVER THE INTEGRATION INTERVAL;
          DATA:        <PROCEDURE IDENTIFIER>;
                     THIS PROCEDURE TAKES THE DATA TO FIT INTO THE PROCEDURE
                     PEIDE;
                     THE HEADING OF THIS PROCEDURE SHOULD BE:
                     "PROCEDURE" DATA(NOBS, TOBS, OBS, COBS); "VALUE" NOBS;
                     "INTEGER" NOBS; "ARRAY" TOBS,OBS; "INTEGER" "ARRAY" COBS;
                     ENTRY: NOBS;
                            NOBS HAS THE SAME MEANING AS IN PEIDE;
                     EXIT:  "ARRAY" TOBS[0 : NOBS];
                            THE ARRAY ELEMENT TOBS[0] SHOULD CONTAIN THE TIME,
                     CORRESPONDING TO THE INITIAL VALUES OF Y GIVEN IN
                     THE PROCEDURE  CALL YSTART; AN ARRAY ELEMENT
                     TOBS[I], 1<=I<=NOBS, SHOULD CONTAIN THE I-TH TIME
                     OF OBSERVATION; THE OBSERVATIONS HAVE TO BE
                     ORDERED SUCH THAT TOBS[I]<=TOBS[J] IF I<=J;
                     "INTEGER" "ARRAY" COBS[1:NOBS];
                     AN ARRAY ELEMENT COBS[I] SHOULD CONTAIN THE
                     COMPONENT OF Y OBSERVED AT TIME TOBS[I]; NOTE THAT
                     1<=COBS[I]<=N;
                     "ARRAY" OBS[1:NOBS];
                     AN ARRAY ELEMENT OBS[I] SHOULD CONTAIN THE
                     OBSERVED VALUE OF THE COMPONENT COBS[I] OF Y AT
                     THE TIME TOBS[I];
```

MONITOR: <PROCEDURE IDENTIFIER>;
THIS PROCEDURE CAN BE USED TO OBTAIN INFORMATION ABOUT THE COURSE OF THE ITERATION PROCESS; IF NO INTERMEDIATE RESULTS ARE DESIRED, A DUMMY PROCEDURE SATISFIES;
THE HEADING OF THIS PROCEDURE SHOULD BE:
"PROCEDURE" MONITOR(POST,NCOL,NROW,PAR,RV,WEIGHT,NIS);
"VALUE" POST,NCOL,NROW,WEIGHT,NIS;
"INTEGER" POST,NCOL,NROW,WEIGHT,NIS; "ARRAY" PAR,RV;
INSIDE PEIDE, THE PROCEDURE MONITOR IS CALLED AT TWO DIFFERENT PLACES AND THIS IS DENOTED BY THE VALUE OF POST:

POST=1: MONITOR IS CALLED AFTER AN INTEGRATION OF THE DIFFERENTIAL EQUATIONS; AT THIS PLACE ARE AVAILABLE: THE CURRENT VALUES OF THE UNKNOWN VARIABLES PAR[1:NCOL], WHERE NCOL=M+NBP, THE CALCULATED RESIDUAL VECTOR RV[1:NROW], WHERE NROW=NOBS+NBP, AND THE VALUE OF NIS, WHICH IS THE NUMBER OF INTEGRATION STEPS PERFORMED DURING THE SOLUTION OF THE LAST INITIAL VALUE PROBLEM;

POST=2: MONITOR IS CALLED BEFORE A MINIMIZATION OF THE EUCLIDEAN NORM OF THE RESIDUAL VECTOR WITH THE PROCEDURE MARQUARDT (SEE SECTION 5.1.3.1.3) IS STARTED; AVAILABLE ARE THE CURRENT VALUES OF PAR[1:NCOL] AND THE VALUE OF THE WEIGHT, WITH WHICH THE CONTINUITY REQUIREMENTS AT THE BREAK-POINTS ARE ADDED TO THE ORIGINAL LEAST SQUARES PROBLEM.

DATA AND RESULTS: SEE REF[1].

PROCEDURES USED:

INIVEC = CP31010,
INIMAT = CP31011,
MULVEC = CP31020,
MULROW = CP31021,
DUPVEC = CP31030,
DUPMAT = CP31035,
VECVEC = CP34010,
MATVEC = CP34011,
ELMVEC = CP34020,
SOL = CP34051,
DEC = CP34300,
MARQUARDT = CP34440.

REQUIRED CENTRAL MEMORY :

IN THE BODY OF PEIDE (3 + NBP) * NOBS + N * (13 + N + 7 * M + 7 * NBP) ARRAY ELEMENTS ARE DECLARED.

RUNNING TIME : DEPENDS STRONGLY ON THE PROBLEM TO SOLVE.

LANGUAGE: ALGOL 60.

METHOD AND PERFORMANCE:
PEIDE ESTIMATES UNKNOWN VARIABLES IN THE SYSTEM OF DIFFERENTIAL EQUATIONS DY/DT (T, PAR) = F (T, Y, PAR), BY USING A SET OF OBSERVED VALUES OF Y; THE UNKNOWN VARIABLES PAR ARE OBTAINED IN THE LEAST SQUARES SENSE; AN ELEMENT OF THE RESIDUAL VECTOR IS DEFINED BY THE CALCULATED VALUE OF Y MINUS ITS OBSERVED VALUE; THE EUCLIDEAN NORM OF THE RESIDUAL VECTOR IS MINIMIZED BY THE ITERATION PROCESS OF MARQUARDT; THE DIFFERENTIAL EQUATIONS ARE SOLVED BY THE INTEGRATION PROCESS OF GEAR; A MULTIPLE SHOOTING TECHNIQUE HAS BEEN IMPLEMENTED TO IMPROVE BAD STARTING VALJES OF THE UNKNOWNS; IF THIS TECHNIQUE IS USED, ONE HAS TO GIVE SOME BREAK-POINTS, I.E. TIMES OF OBSERVATIONS WHERE A NEW INITIAL VALUE PROBLEM SHOULD BE STARTED; THE NEW INITIAL VALUES OF Y BECOME EXTRA UNKNOWN VARIABLES AND THE CONTINUITY REQUIREMENTS AT THE BREAK-POINTS ARE ADDED WITH SOME WEIGHTING FACTOR TO THE LEAST SQUARES PROBLEM; FOR DETAILS SEE REF[1].

REFERENCES:
[1]: B. VAN DOMSELAAR,
NONLINEAR PARAMETER ESTIMATION IN INITIAL VALUE PROBLEMS,
MATH. CENTRE, AMSTERDAM (TO APPEAR).

EXAMPLE OF USE:
THE PARAMETERS PAR[1:3] IN THE DIFFERENTIAL EQUATIONS
DY[1]/DT = - (1 - Y[2]) * Y[1] + EXP(PAR[2]) * Y[2],
DY[2]/DT = EXP(PAR[1]) * ((1 - Y[2]) * Y[1] - (EXP(PAR[2])+
+EXP(PAR[3])) * Y[2]),
WITH 23 OBSERVATIONS OF Y[2], MAY BE OBTAINED BY THE FOLLOWING PROGRAM, THAT CONSISTS OF
1: A CODE PROCEDURE WHICH TAKES CARE OF THE OUTPUT OF THE EXAMPLE PROGRAM. IT ALSO INTERPRETS THE NUMERICAL DATA THAT CAN BE USED TO OBTAIN STATISTICAL RESULTS;
2: THE USERS PROGRAM IN WHICH THE PROBLEM EXAMPLE IS DEFINED.

```
"CODE" 34445;
"PROCEDURE" COMMUNICATION(POST,FA,N,M,NOBS,NBP,PAR,RES,BP,JTJINV,
                          IN,OUT,WEIGHT,NIS);
"VALUE" POST,FA,N,M,NOBS,NBP,WEIGHT,NIS;
"INTEGER" POST,N,M,NOBS,NBP,WEIGHT,NIS; "REAL" FA;
"ARRAY" PAR,RES,BP,JTJINV,IN,OUT;
"BEGIN" "INTEGER" I,J; "REAL" C; "ARRAY" CONF[1:M];
 "REAL" "PROCEDURE" VECVEC(L,U,S,A,B); "CODE" 34010;
 "IF" POST=5 "THEN"
 "BEGIN" OUTPUT(61,"("*,/,10B,"("THE FIRST RESIDUAL VECTOR")",//,16B,
   "("I")",4B,"("RES[I]")",/")");
   "FOR" I:=1 "STEP" 1 "UNTIL" NOBS "DO"
   OUTPUT(61,"("15B,ZD,2B,+.4D"+ZD,/")",I,RES[I]);
 "END" "ELSE" "IF" POST=3 "THEN"
 "BEGIN" OUTPUT(61,"("*,/,
   "("THE EUCLIDEAN NORM OF THE RESIDUAL VECTOR:")",
   .7D"+ZD,2/,5B,"("CALCULATED PARAMETERS")",/")",
   SQRT(VECVEC(1,NOBS,0,RES,RES)));
   "FOR" I:=1 "STEP" 1 "UNTIL" M "DO"
   OUTPUT(61,"("9B,+.7D"+ZD,/")",PAR[I]);
   OUTPUT(61,"("/,
   "("NUMBER OF INTEGRATION STEPS PERFORMED: ")",ZZD,//")",NIS);
"END" "ELSE" "IF" POST=4 "THEN"
"BEGIN" "IF" NBP=0 "THEN" OUTPUT(61,"("*,//,5B,
   "("THE MINIMIZATION IS STARTED WITHOUT BREAK-POINTS")"")") "ELSE"
   "BEGIN" OUTPUT(61,"("*,5/,20B,
     "("THE MINIMIZATION IS STARTED WITH  W E I G H T ="")",ZD,
     3/")",WEIGHT);
     OUTPUT(61,"("/,5B,
     "("THE EXTRA PARAMETERS ARE THE OBSERVATIONS:")"")");
     "FOR" I:=1 "STEP" 1 "UNTIL" NBP "DO"
     OUTPUT(61,"("8B,ZD,2B")",BP[I]);
   "END";
   OUTPUT(61,"("6/,10B,
   "("STARTING VALUES OF THE PARAMETERS")",/")");
   "FOR" I:=1 "STEP" 1 "UNTIL" M "DO"
   OUTPUT(61,"("20B,+.7D"+ZD,/")",PAR[I]);
   OUTPUT(61,"("//,
   "("REL. TOLERANCE FOR THE EUCL. NORM OF THE RES. VECTOR:")"
   ,B,.7D"+ZD,/,
   "("ABS. TOLERANCE FOR THE EUCL. NORM OF THE RES. VECTOR:")"
   ,B,.7D"+ZD,/,"("RELATIVE STARTING VALUE OF LAMBDA")",19B,
   "(":")",B,.7D"+ZD")",IN[3],IN[4],IN[6])
"END" "ELSE" "IF" POST=1 "THEN"
"BEGIN"
OUTPUT(61,"("10B,"("STARTING VALUES OF THE PARAMETERS")",/")");
"FOR" I:=1 "STEP" 1 "UNTIL" M "DO"
OUTPUT(61,"("20B,+.7D"+ZD,/")",PAR[I]);
                                                    "COMMENT"
```

```
OUTPUT(61,"("2/,"("NUMBER OF EQUATIONS")",3B,"(":")",ZD,/,
"("NUMBER OF OBSERVATIONS:")",ZD,2/,
"("MACHINE PRECISION")",30B,"(":")",+.D"+ZD,/,
"("RELATIVE LOCAL ERROR BOUND FOR INTEGRATION")",5B,"(":")",+.D"+ZD,/,
"("RELATIVE TOLERANCE FOR RESIDUE")",17B,"(":")",+.2D"+ZD,/,
"("ABSOLUTE TOLERANCE FOR RESIDUE")",17B,"(":")",+.2D"+ZD,/,
"("MAXIMUM NUMBER OF INTEGRATIONS TO PERFORM")",6B,"(":")",ZZD,/,
"("RELATIVE STARTING VALUE OF LAMBDA")",14B,"(":")",+.2D"+ZD,/,
"("RELATIVE MINIMAL STEPLENGTH")",20B,"(":")",+.2D"+ZD,/")",
N,NOBS,IN[0],IN[2],IN[3],IN[4],IN[5],IN[6],IN[1]);
"IF" NBP=0 "THEN" OUTPUT(61,"("//,
"("THERE ARE NO BREAK-POINTS")"")") "ELSE"
"BEGIN" OUTPUT(61,"("//,
  "("BREAK-POINTS ARE THE OBSERVATIONS :")"")");
  "FOR" I:=1 "STEP" 1 "UNTIL" NBP "DO"
  OUTPUT(61,"("ZZD,B")",BP[I])
"END";
OUTPUT(61,"("//,
"("THE ALPHA-POINT OF THE F-DISTIBUTION :")",
ZD.DD")",FA);
"END" "ELSE" "IF" POST=2 "THEN"
"BEGIN" OUTPUT(61,"("*")"); "IF" OUT[1]=0 "THEN" OUTPUT(61,"("2/,
"("NORMAL TERMINATION OF THE PROCESS")"")")
"ELSE" "IF" OUT[1]=1 "THEN" OUTPUT(61,"("2/,
"("NUMBER OF INTEGRATIONS ALLOWED WAS EXCEEDED")"")")
"ELSE" "IF" OUT[1]=2 "THEN" OUTPUT(61,"("2/,
"("MINIMAL STEPLENGTH WAS DECREASED FOUR TIMES")"")")
"ELSE" "IF" OUT[1]=3 "THEN" OUTPUT(61,"("2/,
"("A CALL OF DERIV DELIVERED FALSE")"")")
"ELSE" "IF" OUT[1]=4 "THEN" OUTPUT(61,"("2/,
"("A CALL OF  JAC DFDY  DELIVERED FALSE ")"")")
"ELSE" "IF" OUT[1]=5 "THEN" OUTPUT(61,"("2/,
"("A CALL OF  JAC DFDP  DELIVERED FALSE ")"")")
"ELSE" "IF" OUT[1]=6 "THEN" OUTPUT(61,"("2/,
"("PRECISION ASKED FOR MAY NOT BE ATTAINED")"")");
"IF" NBP=0 "THEN" OUTPUT(61,"("2/,
"("LAST INTEGRATION WAS PERFORMED WITHOUT BREAK-POINTS")"")") "ELSE"
"BEGIN" OUTPUT(61,"("2/,
  "("THE PROCESS STOPPED WITH BREAK-POINTS: ")"")");
  "FOR" I:=1 "STEP" 1 "UNTIL" NBP "DO"
  OUTPUT(61,"("ZZD,B")",BP[I])
"END";
OUTPUT(61,"("4/,
"("EUCL. NORM OF THE LAST RESIDUAL VECTOR :")",.7D"+ZD,/,
"("EUCL. NORM OF THE FIRST RESIDUAL VECTOR:")",.7D"+ZD,/,
"("NUMBER OF INTEGRATIONS PERFORMED")",7B,"(":")",ZZD,/,
"("LAST IMPROVEMENT OF THE EUCLIDEAN NORM :")",.7D"+ZD,/,
"("CONDITON NUMBER OF  J'*J")",15B,"(":")",.7D"+ZD,/,
"("LOCAL ERROR BOUND WAS EXCEEDED (MAXIM.):")",ZZD,7/")",
OUT[2],OUT[3],OUT[4],OUT[6],OUT[7],OUT[5]);

                                                    "COMMENT"
```

```
"COMMENT" STATISTICS FOR THE PARAMETERS;
OUTPUT(61,"("//,B,"("PARAMETERS")",12B,"("CONFIDENCE INTERVAL")",
/")");
"FOR" I:=1 "STEP" 1 "UNTIL" M "DO"
"BEGIN" CONF[I]:=SQRT(M*FA*JTJINV[I,I]/(NOBS-M))*OUT[2];
   OUTPUT(61,"("+.7D"+ZD,12B,+.7D"+ZD,/")",PAR[I],CONF[I]);
"END";
C:="IF" NOBS=M "THEN" 0 "ELSE" OUT[2]*OUT[2]/(NOBS-M);
OUTPUT(61,"("5/,"("CORRELATION MATRIX")",11B,"("COVARIANCE MATRIX")",
/")");
"FOR" I:=1 "STEP" 1 "UNTIL" M "DO"
"BEGIN" "FOR" J:=1 "STEP" 1 "UNTIL" M "DO"
   "BEGIN" "IF" I=J "THEN" OUTPUT(61,"("29B")");
      "IF" I>J "THEN" OUTPUT(61,"("+.7D"+ZD,B")",
      JTJINV[I,J]/SQRT(JTJINV[I,I]*JTJINV[J,J]))
      "ELSE" OUTPUT(61,"("+.7D"+ZD,B")",JTJINV[I,J]*C)
   "END"; OUTPUT(61,"("/")");
"END"; OUTPUT(61,"("*")");

OUTPUT(61,"("3/,10B,"("THE LAST RESIDUAL VECTOR")",//,15B,
"("I")",4B,"("RES[I]")",/")");
"FOR" I:=1 "STEP" 1 "UNTIL" NOBS "DO"
 OUTPUT(61,"("14B,ZD,2B,+.4D"+ZD,/")",I,RES[I])
 "END"
"END" COMMUNICATION;
    "EOP"
```

THE USER PROGRAM READS:

```
"BEGIN" "INTEGER" I,M,N,NOBS,NBP; "REAL" TIME,FA;
 "ARRAY" PAR[1:6],RES[1:26],JTJINV[1:3,1:3],IN[0:6],OUT[1:7];
 "INTEGER" "ARRAY" BP[0:3];
 "PROCEDURE" PEIDE(N,M,NO,NB,P,R,BP,J,I,J,D,JDY,JDP,CY,DA,MO);
"CODE" 34444;
 "PROCEDURE" COMMUNICATION(P,F,M,N,NO,NP,PA,R,BP,J,I,O,W,VI);
"CODE" 34445;
```

```
"BOOLEAN" "PROCEDURE" JAC DFDP(PAR,Y,X,FP);
   "REAL" X; "ARRAY" PAR,Y,FP;
   "BEGIN" "REAL" Y2; Y2:=Y[2];
      FP[1,1]:=FP[1,3]:=0;
      FP[1,2]:=Y2*EXP(PAR[2]);
      FP[2,1]:=EXP(PAR[1])*(Y[1]*(1-Y2)-(EXP(PAR[2])+EXP(PAR[3]))*Y2);
      FP[2,2]:=-EXP(PAR[1]+PAR[2])*Y2;
      FP[2,3]:=-EXP(PAR[1]+PAR[3])*Y2;
      JAC DFDP:="TRUE"
      D" JAC DFDP
```

```
;

 "PROCEDURE" DATA(NOBS,TOBS,OBS,COBS);
    "VALUE" NOBS; "INTEGER" NOBS;
    "ARRAY" TOBS,OBS,COBS;
    "BEGIN" "INTEGER" I;
       TOBS[0]:=0;
       OUTPUT(61,"("*,4/,4B,"("THE OBSERVATIONS WERE:")",
       //,B,"("I")",3B,"("TOBS[I]")",3B,"("COBS[I]")",3B,
       "("OBS[I]")",/")");
       "FOR" I:=1 "STEP" 1 "UNTIL" NOBS "DO"
       "BEGIN"
          INPUT(60,"("3(N)")",TOBS[I],COBS[I],OBS[I]);
          OUTPUT(61,"("ZD,3B,ZD.4D,6B,D,6B,.4D,/")",I,TOBS[I],COBS[I],
          OBS[I])
       "END"
    "END" DATA;

"PROCEDURE" CALL YSTART(PAR,Y,YMAX);
    "ARRAY" PAR,Y,YMAX;
    "BEGIN" Y[1]:=YMAX[1]:=YMAX[2]:=1;
       Y[2]:=0
    "END" CALL YSTART;

"BOOLEAN" "PROCEDURE" DERIV(PAR,Y,X,DF);
    "REAL" X; "ARRAY" PAR,Y,DF;
    "BEGIN" "REAL" Y2; Y2:=Y[2];
       DF[1]:=-(1-Y2)*Y[1]+EXP(PAR[2])*Y2;
       DF[2]:=EXP(PAR[1])*((1-Y2)*Y[1]-(EXP(PAR[2])+EXP(PAR[3]))*Y2);
       DERIV:="TRUE"
    "END" DERIV;

"BOOLEAN" "PROCEDURE" JAC DFDY(PAR,Y,X,FY);
    "REAL" X; "ARRAY" PAR,Y,FY;
    "BEGIN" FY[1,1]:=-1+Y[2];
       FY[1,2]:=EXP(PAR[2])+Y[1];
       FY[2,1]:=EXP(PAR[1])*(1-Y[2]);
       FY[2,2]:=-EXP(PAR[1])*(EXP(PAR[2])+EXP(PAR[3])+Y[1]);
       JAC DFDY:="TRUE"
    "END" JAC DFDY;

"PROCEDURE" MONITOR(POST,NCOL,NROW,PAR,RES,WEIGHT,NIS);
 "VALUE" POST,NCOL,NROW,WEIGHT,NIS;
 "INTEGER" POST,NCOL,NROW,WEIGHT,NIS; "ARRAY" PAR,RES;;

OUTPUT(61,"("2/,30B,"("E S C E P - PROBLEM")",3/")");
M:= 3; N:=2; NOBS:=23; NBP:=3;
PAR[1]:=LN(1600); PAR[2]:=LN(.8); PAR[3]:=LN(1.2); IN[0]:="-14;
IN[3]:="-4; IN[4]:="-4; IN[5]:=50; IN[6]:="-2;
IN[1]:="-4; IN[2]:="-5;
BP[1]:=17; BP[2]:=19; BP[3]:=21;
                                                    "COMMENT"
```

```
FA:=4.94;
"COMMENT" FA DENOTES THE ALPHA-POINT OF THE FISHER-DISTRIBUTION;
COMMUNICATION(1,FA,N,M,NOBS,NBP,PAR,RES,BP,JTJINV,IN,OUT,0,0);
TIME:=CLOCK;

PEIDE(N,M,NOBS,NBP,PAR,RES,BP,JTJINV,IN,OUT,DERIV,JAC DFDY,JAC DFDP,
CALL YSTART,DATA,MONITOR);

TIME:=CLOCK-TIME;
COMMUNICATION(2,FA,N,M,NOBS,NBP,PAR,RES,BP,JTJINV,IN,OUT,0,0);
OUTPUT(61,"("3/,5B,
"("THE CALCULATION IN PEIDE CONSUMED")",B,ZZD.DD,2B,
"("SECONDS")",*")",TIME)
"END"
    "EOP"
```

THIS PROGRAM DELIVERS:

E S C E P - PROBLEM

STARTING VALUES OF THE PARAMETERS
+.7377759" +1
-.2231436" +0
+.1823216" +0

NUMBER OF EQUATIONS : 2
NUMBER OF OBSERVATIONS:23

MACHINE PRECISION	:+.1"-13
RELATIVE LOCAL ERROR BOUND FOR INTEGRATION	:+.1" -4
RELATIVE TOLERANCE FOR RESIDUE	:+.10" -3
ABSOLUTE TOLERANCE FOR RESIDUE	:+.10" -3
MAXIMUM NUMBER OF INTEGRATIONS TO PERFORM	: 50
RELATIVE STARTING VALUE OF LAMBDA	:+.10" -1
RELATIVE MINIMAL STEPLENGTH	:+.10" -3

BREAK-POINTS ARE THE OBSERVATIONS : 17 19 21

THE ALPHA-POINT OF THE F-DISTIBUTION : 4.94

THE OBSERVATIONS WERE:

I	TOBS[I]	COBS[I]	OBS[I]
1	0.0002	2	.1648
2	0.0004	2	.2753
3	0.0006	2	.3493
4	0.0008	2	.3990
5	0.0010	2	.4322
6	0.0012	2	.4545
7	0.0014	2	.4695
8	0.0016	2	.4795
9	0.0018	2	.4862
10	0.0020	2	.4907
11	0.0200	2	.4999
12	0.0400	2	.4998
13	0.0600	2	.4998
14	0.0800	2	.4998
15	0.1000	2	.4998
16	1.0000	2	.4986
17	2.0000	2	.4973
18	5.0000	2	.4936
19	10.0000	2	.4872
20	15.0000	2	.4808
21	20.0000	2	.4743
22	25.0000	2	.4677
23	30.0000	2	.4610

NORMAL TERMINATION OF THE PROCESS

LAST INTEGRATION WAS PERFORMED WITHOUT BREAK-POINTS

EUCL. NORM OF THE LAST RESIDUAL VECTOR :.1430776" -3
EUCL. NORM OF THE FIRST RESIDUAL VECTOR:.1331071" +1
NUMBER OF INTEGRATIONS PERFORMED : 12
LAST IMPROVEMENT OF THE EUCLIDEAN NORM :.2223694" -4
CONDITON NUMBER OF J'*J :.2582882" +3
LOCAL ERROR BOUND WAS EXCEEDED (MAXIM.): 37

PARAMETERS	CONFIDENCE INTERVAL
+.6907670" +1	+.3209313" -3
-.1003941" -1	+.1687774" -3
-.4605292" +1	+.1942501" -2

CORRELATION MATRIX

+.3851320" +0		
-.2170393" +0	-.6392889" +0	

COVARIANCE MATRIX

+.6949857" -8	+.1407628" -8	-.9129848" -8
	+.1922119" -8	-.1414245" -7
		+.2546094" -6

THE LAST RESIDUAL VECTOR

I	RES[I]
1	+.1748" -5
2	-.2905" -4
3	+.2814" -4
4	-.3879" -4
5	+.3069" -4
6	+.3101" -4
7	-.2019" -4
8	-.3887" -5
9	+.1052" -4
10	+.1391" -4
11	-.5109" -4
12	+.2384" -4
13	-.1156" -5
14	-.2616" -4
15	-.5116" -4
16	+.2244" -4
17	+.6794" -4
18	-.1418" -4
19	+.2087" -4
20	-.1980" -4
21	-.3476" -4
22	-.2245" -4
23	+.1886" -4

THE CALCULATION IN PEIDE CONSUMED 108.57 SECONDS

SOURCE TEXT(S):

```
"CODE" 34444;
"PROCEDURE" PEIDE(N,M,NOBS,NBP,PAR,RES,BP,JTJINV,IN,OUT,DERIV,JAC DFDY,
    JAC DFDP, CALL YSTART,DATA,MONITOR);
    "VALUE" N,M,NOBS; "INTEGER" N,M,NOBS,NBP;
    "ARRAY" PAR,RES,JTJINV,IN,OJT;
    "INTEGER" "ARRAY" BP;
    "PROCEDURE"  CALL YSTART,DATA,MONITOR;
    "BOOLEAN" "PROCEDURE" DERIV,JAC DFDY,JACDFDP;
    "BEGIN" "INTEGER" I,J,EXTRA,WEIGHT,NCOL,NROW,AWAY,NPAR,II,JJ,MAX,
       NFE,NIS;
       "REAL" EPS,EPS1,XEND,C,X,T,HMIN,HMAX,RES1,IN3,IN4,FAC3,FAC4;
       "ARRAY" AUX[1:3],OBS[1:NOBS],SAVE[-38:6*N],TOBS[0:NOBS],
       YP[1:NBP+NOBS,1:NBP+M],YMAX[1:N],Y[1:6*N*(NBP+M+1)],FY[1:N,1:N],
       FP[1:N,1:M+NBP];
       "INTEGER" "ARRAY" COBS[1:NOBS];
       "BOOLEAN" FIRST,SEC,CLEAN;

       "PROCEDURE" INIVEC(L,U,A,X); "CODE" 31010;
"PROCEDURE" INIMAT(L1,U1,L2,U2,A,X); "CODE" 31011;
"PROCEDURE" MULVEC(L,U,S,A,B,X); "CODE" 31020;
"PROCEDURE" MULROW(L,U,I,J,A,B,X); "CODE" 31021;
"PROCEDURE" DUPVEC(L,U,S,A,B); "CODE" 31030;
"PROCEDURE" DUPMAT(L1,U1,L2,U2,A,B); "CODE" 31035;
"REAL" "PROCEDURE" VECVEC(L,U,S,A,B); "CODE" 34010;
"REAL" "PROCEDURE" MATVEC(L,U,I,A,B); "CODE"34011;
"PROCEDURE" ELMVEC(L,U,S,A,B,X); "CODE"34020;
"PROCEDURE" SOL(A,N,P,B); "CODE" 34051;
"PROCEDURE" DEC(A,N,AUX,P); "CODE" 34300;
"PROCEDURE" MARQUARDT(M,N,P,R,C,F,J,I,O); "CODE" 34440;

"REAL" "PROCEDURE" INTERPOL(STARTINDEX,JUMP,K,TOBSDIF);
   "VALUE" STARTINDEX,JUMP,K,TOBSDIF;
   "INTEGER" STARTINDEX,JUMP,K; "REAL" TOBSDIF;
   "BEGIN" "INTEGER" I; "REAL" S,R; S:=Y[STARTINDEX]; R:=TOBSDIF;
      "FOR" I:=1 "STEP" 1 "UNTIL" K "DO"
      "BEGIN" STARTINDEX:=STARTINDEX+JUMP;
         S:=S+Y[STARTINDEX]*R; R:=R*TOBSDIF
   "END"; INTERPOL:=S
"END" INTERPOL;

"PROCEDURE" JAC DYDP(NROW,NCOL,PAR,RES,JAC,LOCFUNCT);
   "VALUE" NROW,NCOL; "INTEGER" NROW,NCOL;
   "ARRAY" PAR,RES,JAC; "PROCEDURE" LOCFUNCT;
   "BEGIN"
      DUPMAT(1,NROW,1,NCOL,JAC,YP)
   "END" JACOBIAN
```

```
;

      "BOOLEAN" "PROCEDURE" FUNCT(NROW,NCOL,PAR,RES);
         "VALUE" NROW,NCOL; "INTEGER" NROW,NCOL; "ARRAY" PAR,RES;
         "BEGIN" "INTEGER" L,K,KNEW,FAILS,SAME,KPOLD,N6,NNPAR,J5N,
         COBSII;
         "REAL" XOLD,HOLD,A0,TOLUP,TOL,TOLDWN,TOLCONV,H,CH,CHNEW,
         ERROR,DFI,TOBSDIF;
         "BOOLEAN" EVALUATE,EVALUATED,DECOMPOSE,CONV;
         "ARRAY" A[0:5],DELTA,LAST DELTA,DF,Y0[1:N],JACOB[1:N,1:N];
         "INTEGER" "ARRAY" P[1:N];

         "REAL" "PROCEDURE" NORM2(AI); "REAL" AI;
            "BEGIN" "REAL" S,A; S:= "-100;
               "FOR" I:= 1 "STEP" 1 "UNTIL" N "DO"
               "BEGIN" A:= AI/YMAX[I]; S:= S + A * A "END";
             NORM2:= S
            "END" NORM2;

         "PROCEDURE" RESET;
            "BEGIN" "IF" CH < HMIN/HOLD "THEN" CH:= HMIN/HOLD "ELSE"
               "IF" CH > HMAX/HOLD "THEN" CH:= HMAX/HOLD;
               X:= XOLD; H:= HOLD * CH; C:= 1;
               "FOR" J:= 0 "STEP" N "UNTIL" K*N "DO"
               "BEGIN" "FOR" I:= 1 "STEP" 1 "UNTIL" N "DO"
                  Y[J+I]:= SAVE[J+I] * C;
                  C:= C * CH
               "END";
               DECOMPOSE:="TRUE"
            "END" RESET;

         "PROCEDURE" ORDER;
            "BEGIN" C:= EPS * EPS; J:= (K-1) * (K + 8)/2 - 38;
               "FOR" I:= 0 "STEP" 1 "UNTIL" K "DO" A[I]:= SAVE[I+J];
               J:= J + K + 1;
               TOLUP  := C * SAVE[J];
               TOL    := C * SAVE[J + 1];
               TOLDWN := C * SAVE[J + 2];
               TOLCONV:= EPS/(2 * N * (K + 2));
               A0:= A[0];  DECOMPOSE:= "TRUE";
            "END" ORDER;

         "PROCEDURE" EVALUATE JACOBIAN;
            "BEGIN" EVALUATE:= "FALSE";
               DECOMPOSE:= EVALUATED:= "TRUE";
               "IF" "NOT" JAC DFDY(PAR,Y,X,FY) "THEN"
               "BEGIN" SAVE[-3]:=4; "GOTO" RETURN "END";
            "END" EVALUATE JACOBIAN
```

```
"PROCEDURE" DECOMPOSE JACOBIAN;
   "BEGIN" DECOMPOSE:= "FALSE";
      C:= -A0 * H;
      "FOR" J:= 1 "STEP" 1 "UNTIL" N "DO"
      "BEGIN" "FOR" I:= 1 "STEP" 1 "UNTIL" N "DO"
         JACOB[I,J]:= FY[I,J] * C;
         JACOB[J,J]:= JACOB[J,J] + 1
      "END";
      DEC(JACOB,N,AUX,P)
   "END" DECOMPOSE JACOBIAN;

"PROCEDURE" CALCULATE STEP AND ORDER;
   "BEGIN" "REAL" A1,A2,A3;
      A1:= "IF" K <= 1 "THEN" 0 "ELSE"
           0.75 * (TOLDWN/NORM2(Y[K*N+I])) ** (0.5/K);
      A2:= 0.80 * (TOL/ERROR) ** (0.5/(K + 1));
      A3:= "IF" K >= 5 "OR" FAILS ^= 0
           "THEN" 0 "ELSE"
           0.70 * (TOLUP/NORM2(DELTA[I] - LAST DELTA[I]))**
           (0.5/(K+2));

      "IF" A1 > A2 "AND" A1 > A3 "THEN"
      "BEGIN" KNEW:= K-1; CHNEW:= A1 "END" "ELSE"
      "IF" A2 > A3 "THEN"
      "BEGIN" KNEW:= K  ; CHNEW:= A2 "END" "ELSE"
      "BEGIN" KNEW:= K+1; CHNEW:= A3 "END"
   "END" CALCULATE STEP AND ORDER;

"IF" SEC "THEN" "BEGIN" SEC:="FALSE"; "GOTO" RETURN "END";
NPAR:=M; EXTRA:=NIS:=0; II:=1;
JJ:="IF" NBP=0 "THEN" 0 "ELSE" 1;
N6:=N*6;
INIVEC(-3,-1,SAVE,0);
INIVEC(N6+1,(6+M)*N,Y,0);
INIMAT(1,NOBS+NBP,1,M+NBP,YP,0);
T:=TOBS[1]; X:=TOBS[0];
CALL YSTART(PAR,Y,YMAX);
HMAX:=TOBS[1]-TOBS[0]; HMIN:=HMAX*IN[1];
EVALUATE JACOBIAN; NNPAR:=N*NPAR;

NEW START:
   K:= 1; KPOLD:=0; SAME:= 2; ORDER;
   "IF" "NOT" DERIV(PAR,Y,X,DF) "THEN"
   "BEGIN" SAVE[-3]:=3; "GOTO" RETURN "END";
   H:=SQRT(2 * EPS/SQRT(NORM2 (MATVEC(1,N,I,FY,DF))));
   "IF" H > HMAX "THEN" H:= HMAX "ELSE"
   "IF" H < HMIN "THEN" H:= HMIN;
   XOLD:= X; HOLD:= H; CH:= 1;
   "FOR" I:= 1 "STEP" 1 "UNTIL" N "DO"
   "BEGIN" SAVE[I]:=Y[I]; SAVE[N+I]:=Y[N+I]:=DF[I]*H "END";
   FAILS:= 0;
                                                    "COMMENT"
```

```
        "FOR" L:= 0 "WHILE" X < XEND "DO"
        "BEGIN" "IF" X + H <= XEND "THEN" X:= X + H "ELSE"
           "BEGIN" H:= XEND-X; X:= XEND; CH:= H/HOLD; C:= 1;
              "FOR" J:= N "STEP" N "UNTIL" K*N "DO"
              "BEGIN" C:= C* CH;
              "FOR" I:= J+1 "STEP" 1 "UNTIL" J+N "DO"
              Y[I]:= Y[I] * C
           "END";
           SAME:= "IF" SAME<3 "THEN" 3 "ELSE" SAME+1;
        "END";

        "COMMENT" PREDICTION;
        "FOR" L:= 1 "STEP" 1 "UNTIL" N "DO"
        "BEGIN" "FOR" I:= L "STEP" N "UNTIL" (K-1)*N+L "DO"
           "FOR" J:= (K-1)*N+L "STEP" -N "UNTIL" I "DO"
           Y[J]:= Y[J] + Y[J+N];
           DELTA[L]:= 0
        "END";  EVALUATED:= "FALSE";

        "COMMENT" CORRECTION AND ESTIMATION LOCAL ERROR;
        "FOR" L:= 1,2,3 "DO"
        "BEGIN" "IF" "NOT" DERIV(PAR,Y,X,DF) "THEN"
        "BEGIN" SAVE[-3]:=3; "GOTO" RETURN "END";
        "FOR" I:= 1 "STEP" 1 "UNTIL" N "DO"
        DF[I]:= DF[I] * H - Y[N+I];
        "IF" EVALUATE "THEN" EVALUATE JACOBIAN;
        "IF" DECOMPOSE "THEN" DECOMPOSE JACOBIAN;
        SOL(JACOB,N,P,DF);

        CONV:= "TRUE";
        "FOR" I:= 1 "STEP" 1 "UNTIL" N "DO"
        "BEGIN" DFI:= DF[I];
           Y[  I]:= Y[  I] + A0 * DFI;
           Y[N+I]:= Y[N+I] +      DFI;
           DELTA[I]:= DELTA[I] +  DFI;
           CONV:= CONV "AND" ABS(DFI) < TOLCONV * YMAX[I]
        "END";
        "IF" CONV "THEN"
        "BEGIN" ERROR:= NORM2(DELTA[I]);
           "GOTO" CONVERGENCE
        "END"
        "END";

        "COMMENT" ACCEPTANCE OR REJECTION;
        "IF" "NOT" CONV "THEN"
        "BEGIN" "IF" "NOT" EVALUATED "THEN" EVALUATE:= "TRUE"
           "ELSE"
           "BEGIN" CH:=CH/4; "IF" H<4*HMIN "THEN"
              "BEGIN" SAVE[-1]:= SAVE[-1]+10;
                 HMIN:=HMIN/10;
                                                    "COMMENT"
```

```
                    "IF" SAVE[-1]>40 "THEN" "GOTO" RETURN
                "END"
            "END";
            RESET
        "END" "ELSE" CONVERGENCE:

    "IF" ERROR > TOL "THEN"
    "BEGIN" FAILS:= FAILS + 1;
        "IF" H > 1.1 * HMIN "THEN"
        "BEGIN" "IF" FAILS > 2 "THEN"
            "BEGIN"  RESET; "GOTO" NEW START
            "END" "ELSE"
            "BEGIN" CALCULATE STEP AND ORDER;
                "IF" KNEW ^= K "THEN"
                "BEGIN" K:= KNEW; ORDER "END";
                CH:= CH * CHNEW; RESET
            "END"
        "END" "ELSE"
        "BEGIN" "IF" K = 1 "THEN"
            "BEGIN" "COMMENT" VIOLATE EPS CRITERION;
                SAVE[-2]:= SAVE[-2] + 1;
                SAME:= 4; "GOTO" ERROR TEST OK
            "END";
            K:=1; RESET; ORDER; SAME:= 2
        "END"
    "END" "ELSE" ERROR TEST OK:

    "BEGIN" FAILS:= 0;
        "FOR" I:= 1 "STEP" 1 "UNTIL" N "DO"
        "BEGIN" C:= DELTA[I];
            "FOR" L:= 2 "STEP" 1 "UNTIL" K "DO"
            Y[L*N+I]:= Y[L*N+I] + A[L] * C;
            "IF" ABS(Y[I]) > YMAX[I] "THEN"
            YMAX[I]:=  ABS(Y[I])
        "END";

        SAME:= SAME-1;
        "IF" SAME= 1 "THEN"
        DUPVEC(1,N,0,LAST DELTA,DELTA) "ELSE"
        "IF" SAME= 0 "THEN"
        "BEGIN" CALCULATE STEP AND ORDER;
    "IF" CHNEW > 1.1 "THEN"
    "BEGIN"
        "IF" K ^= KNEW "THEN"
        "BEGIN" "IF" KNEW > K "THEN"
            MULVEC(KNEW*N+1,KNEW*N+N,-KNEW*N,Y,DELTA,
            A[K]/KNEW);
            K:= KNEW; ORDER
        "END";
        SAME:= K+1;
        "IF" CHNEW * H > HMAX
        "THEN" CHNEW:= HMAX/H;
                                                    "COMMENT"
```

```
H:= H * CHNEW; C:= 1;
"FOR" J:= N "STEP" N "UNTIL" K*N "DO"
"BEGIN" C:= C * CHNEW;
   MULVEC(J+1,J+N,0,Y,Y,C)
   "END"; DECOMPOSE:="TRUE"
   "END"
   "ELSE" SAME:= 10
"END" OF A SINGLE INTEGRATION STEP OF Y;
NIS:=NIS+1;

"COMMENT" START OF A INTEGRATION STEP OF YP;
"IF" CLEAN "THEN"
"BEGIN" HOLD:=H; XOLD:=X; KPOLD:=K; CH:=1;
   DUPVEC(1,K*N+N,0,SAVE,Y)
"END" "ELSE"
"BEGIN" "IF" H^=HOLD "THEN"
   "BEGIN" CH:=H/HOLD; C:=1;
      "FOR" J:=N6+NNPAR "STEP" NNPAR "UNTIL"
      KPOLD*NNPAR+N6 "DO"
      "BEGIN" C:=C*CH;
         "FOR" I:=J+1 "STEP" 1 "UNTIL" J+NNPAR "DO"
         Y[I]:=Y[I]*C
   "END"; HOLD:=H
"END";
"IF" K>KPOLD "THEN"
INIVEC(N6+K*NNPAR+1,N6+K*NNPAR+NNPAR,Y,0);
XOLD:= X; KPOLD:= K; CH:= 1;
DUPVEC(1,K*N+N,0,SAVE,Y);
EVALUATE JACOBIAN;
DECOMPOSE JACOBIAN;
"IF" "NOT" JAC DFDP(PAR,Y,X,FP) "THEN"
"BEGIN" SAVE[-3]:=5; "GOTO" RETURN "END";
"IF" NPAR>M "THEN" INIMAT(1,N,M+1,NPAR,FP,0);

"COMMENT" PREDICTION;
"FOR" L:=0 "STEP" 1 "UNTIL" K-1 "DO"
"FOR" J:=K-1 "STEP" -1 "UNTIL" L "DO"
ELMVEC(J*NNPAR+N6+1,J*NNPAR+N6+NNPAR,NNPAR,Y,Y,1);

"COMMENT" CORRECTION;
   "FOR" J:=1 "STEP" 1 "UNTIL" NPAR "DO"
   "BEGIN" J5N:=(J+5)*N;
      DUPVEC(1,N,J5N,Y0,Y);
      "FOR" I:=1 "STEP" 1 "UNTIL" N "DO" DF[I]:=
      H*(FP[I,J]+MATVEC(1,N,I,FY,Y0))
      -Y[NNPAR+J5N+I];
      SOL(JACOB,N,P,DF);
      "FOR" L:=0 "STEP" 1 "UNTIL" K "DO"
      "BEGIN" I:=L*NNPAR+J5N;
         ELMVEC(I+1,I+N,-I,Y,DF,A[L])
      "END"
   "END"
"END";
```

"COMMENT"

```
                "FOR" L:=0 "WHILE" X>=T "DO"
                "BEGIN"
                   "COMMENT" CALCULATION OF A ROW OF THE JACOBIAN
                             MATRIX AND AN ELEMENT OF THE RESIDUAL
                             VECTOR;
                TOBSDIF:=(TOBS[II]-X)/H; COBSII:=COBS[II];
                RES[II]:=INTERPOL(COBSII,N,K,TOBSDIF)-OBS[II];
                "IF" "NOT" CLEAN "THEN"
                "BEGIN" "FOR" I:=1 "STEP" 1 "UNTIL" NPAR "DO"
                   YP[II,I]:=INTERPOL(COBSII+(I+5)*N,NNPAR,K,
                                      TOBSDIF);

                   "COMMENT" INTRODUCING OF BREAK-POINTS;
                   "IF" BP[JJ]^=II "THEN" "ELSE"
                   "IF" FIRST "AND" ABS(RES[II])<EPS1 "THEN"
                   "BEGIN" NBP:=NBP-1; DUPVEC(JJ,NBP,1,BP,BP);
                      BP[NBP+1]:=0
                   "END" "ELSE"
                   "BEGIN" EXTRA:=EXTRA+1;
                      "IF" FIRST "THEN" PAR[M+JJ]:=OBS[II];
                      "COMMENT" INTRODUCING A JACOBIAN ROW AND A
                                RESIDUAL VECTOR ELEMENT FOR
                                CONTINUITY REQUIREMENTS;
                         YP[NOBS+JJ,M+JJ]:=-WEIGHT;
                         MULROW(1,NPAR,NOBS+JJ,II,YP,YP,WEIGHT);
                         RES[NOBS+JJ]:=WEIGHT*(RES[II]+OBS[II]-
                                               PAR[M+JJ])
                      "END"
                   "END";

                   "IF" II=NOBS "THEN" "GOTO" RETURN "ELSE"
                   "BEGIN" T:=TOBS[II+1];
                      "IF" BP[JJ]=II "AND" JJ<NBP "THEN" JJ:=JJ+1;
                      HMAX:=T-TOBS[II]; HMIN:=HMAX*IN[1]; II:=II+1
                   "END";
                "END";

                "COMMENT" BREAK-POINTS INTRODUCE NEW INITIAL VALUES
                          FOR Y AND YP;
                "IF" EXTRA>0 "THEN"
                "BEGIN" "FOR" I:=1 "STEP" 1 "UNTIL" N "DO"
                   "BEGIN" Y[I]:=INTERPOL(I,N,K,TOBSDIF);
                      "FOR" J:=1 "STEP" 1 "UNTIL" NPAR "DO"
                      Y[I+(J+5)*N]:=INTERPOL(I+(J+5)*N,NNPAR,K,
                                             TOBSDIF)
                   "END";
                   "FOR" L:=1 "STEP" 1 "UNTIL" EXTRA "DO"
                   "BEGIN"  COBSII:=COBS[BP[NPAR-M+L]];
                      Y[COBSII]:=PAR[NPAR+L];
                      "FOR" I:=1 "STEP" 1 "UNTIL" NPAR+EXTRA "DO"
                      Y[COBSII+(5+I)*N]:=0;
                                                          "COMMENT"
```

```
      INIVEC(1+NNPAR+(L+5)*N,NNPAR+(L+6)*N,Y,0);
      Y[COBSII+(5+NPAR+L)*N]:=1
"END";
NPAR:=NPAR+EXTRA; EXTRA:=0;
      X:=TOBS[II-1]; EVALJATE JACOBIAN; NNPAR:=N*NPAR;
      "GOTO" NEW START
   "END"
"END"
"END" STEP;

RETURN:
   "IF" SAVE[-2]>MAX "THEN" MAX:=SAVE[-2];
   FUNCT:=SAVE[-1]<=40 "AND" SAVE[-3]=0;
   "IF" "NOT" FIRST "THEN"
   MONITOR(1,NCOL,NROW,PAR,RES,WEIGHT,NIS)
"END" FUNCT;


I:= -39;
"FOR" C:= 1,1,9,4,0,2/3,1,1/3,36,20.25,1,6/11,
          1,6/11,1/11,84.028,53.778,0.25,.48,1,.7,.2,.02,
          156.25, 108.51, .027778, 120/274, 1, 225/274,
          85/274, 15/274, 1/274, 0, 187.69, .0047361
"DO" "BEGIN" I:= I + 1; SAVE[I]:= C "END";

DATA(NOBS,TOBS,OBS,COBS); WEIGHT:=1;
FIRST:=SEC:="FALSE"; CLEAN:=NBP>0;
AUX[2]:="-12; EPS:=IN[2]; EPS1:="10;
XEND:=TOBS[NOBS]; OUT[1]:=0; BP[0]:=MAX:=0;

"COMMENT" SMOOTH INTEGRATION WITHOUT BREAK-POINTS;
"IF" "NOT" FUNCT(NOBS,M,PAR,RES) "THEN" "GOTO" ESCAPE;
RES1:=SQRT(VECVEC(1,NOBS,0,RES,RES)); NFE:=1;
"IF" IN[5]=1 "THEN"
"BEGIN" OUT[1]:=1; "GOTO" ESCAPE "END";

"IF" CLEAN "THEN"
"BEGIN" FIRST:="TRUE"; CLEAN:="FALSE";
   FAC3:=SQRT(SQRT(IN[3]/RES1)); FAC4:=SQRT(SQRT(IN[4]/RES1));
   EPS1:=RES1*FAC4;
   "IF" "NOT" FUNCT(NOBS,M,PAR,RES) "THEN" "GOTO" ESCAPE;
   FIRST:="FALSE"
"END" "ELSE" NFE:=0;

NCOL:=M+NBP; NROW:=NOBS+NBP;
SEC:="TRUE";
IN3:=IN[3]; IN4:=IN[4]; IN[3]:=RES1;

"BEGIN" "REAL" W; "ARRAY" AID[1:NCOL,1:NCOL];
   WEIGHT:=AWAY:=0;
   OUT[4]:=OUT[5]:=W:=0;
                                                    "COMMENT"
```

```
;

      "FOR" WEIGHT:=(SQRT(WEIGHT)+1)**2 "WHILE"
      WEIGHT^=16 "AND" NBP>0 "DO"
   "BEGIN" "IF" AWAY=0 "AND" W^=0 "THEN"
      "BEGIN" "COMMENT" IF NO BREAK-POINTS WERE OMITTED THEN ONE
                        FUNCTION EVALUATION IS SAVED;
         W:=WEIGHT/W;
         "FOR" I:=NOBS+1 "STEP" 1 "UNTIL" NROW "DO"
         "BEGIN" "FOR" J:=1 "STEP" 1 "UNTIL" NCOL "DO"
            YP[I,J]:=W*YP[I,J];
            RES[I]:=W*RES[I]
         "END"; SEC:="TRUE"; NFE:=NFE-1
      "END";

      IN[3]:=IN[3]*FAC3*WEIGHT; IN[4]:=EPS1;
      MONITOR(2,NCOL,NROW,PAR,RES,WEIGHT,NIS);
      MARQUARDT(NROW,NCOL,PAR,RES,AID,FUNCT,JAC DYDP,IN,OUT);
      "IF" OUT[1]>0 "THEN" "GOTO" ESCAPE;

      "COMMENT" THE RELATIVE STARTING VALUE OF LAMBDA IS
                ADJUSTED TO THE LAST VALUE OF LAMBDA USED;
      AWAY:=OUT[4]-OUT[5]-1;
      IN[6]:=IN[6] * 5**AWAY * 2**(AWAY-OUT[5]);

      NFE:=NFE+OUT[4];
      W:=WEIGHT; EPS1:=(SQRT(WEIGHT)+1)**2*IN[4]*FAC4;
      AWAY:=0;

      "COMMENT" USELESS BREAK-POINTS ARE OMITTED;
      "FOR" J:=1 "STEP" 1 "UNTIL" NBP "DO"
      "BEGIN" "IF" ABS(OBS[BP[J]]+RES[BP[J]]-PAR[J+M])<EPS1
         "THEN"
         "BEGIN" NBP:=NBP-1; DUPVEC(J,NBP,1,BP,BP);
            DUPVEC(J+M,NBP+M,1,PAR,PAR);
            J:=J-1; AWAY:=AWAY+1; BP[NBP+1]:=0
         "END"
      "END";
      NCOL:=NCOL-AWAY; NROW:=NROW-AWAY
   "END";

      IN[3]:=IN3; IN[4]:=IN4; NBP:=0; WEIGHT:=1;
      MONITOR(2,M,NOBS,PAR,RES,WEIGHT,NIS);
      MARQUARDT(NOBS,M,PAR,RES,JTJINV,FUNCT,JAC DYDP,IN,OUT);
      NFE:=OUT[4]+NFE
    "END";
  ESCAPE: "IF" OUT[1]=3 "THEN" OUT[1]:=2 "ELSE"
          "IF" OUT[1]=4 "THEN" OUT[1]:=6;
          "IF" SAVE[-3]^=0 "THEN" OUT[1]:=SAVE[-3];
          OUT[3]:=RES1;
          OUT[4]:=NFE;
          OUT[5]:=MAX
"END" PEIDE;
     "EOP"
```