Printed at the Mathematical Centre, 413 Kruislaan, Amsterdam.

The Mathematical Centre, founded the 11-th of February 1946, is a non-profit institution aiming at the promotion of pure mathematics and its applications. It is sponsored by the Netherlands Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O.).

MC SYLLABUS 47.6

NUMAL

NUMERICAL PROCEDURES IN ALGOL 60

VOLUME 5B, ANALYTICAL PROBLEMS, PART 2

P.W. HEMKER (ed.)

MATHEMATISCH CENTRUM AMSTERDAM 1981

1980 Mathematics subject classification: 65XX04, 68B99

ISBN 90 6196 217 X

INDEX	PROCEDURE	CODE	MNT/YR	RECORD NUMBER
5. 2. 1. 2. 4.OVER-DETERMINED PROBLEMS 3.PARAMETER ESTIMATION IN D.E. 1.P.E. IN INITIAL VALUE PROBLEMS	PEIDE	34444	OCT/75	259
2.INTEGRAL EQUATIONS 3.INTEGRO- DIFFERENTIAL EQS 4.DIFFERENCE EQUATIONS 5.CONVOLUTION EQUATIONS				

VERSION: 791231

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 XO;
        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 EIGENVALUE 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, C1)
        "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 XO;
              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=XO;
              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;
```

JACOBIAN: <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 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 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;
        PHIO:=PHI; COSPHI:=COS(PHIO); SINPHI:=SIN(PHIO);
        E:=EXP(B*COSPHI); ZI:=B*SINPHI-3*PHIO;
        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" PHIO:=PHI;
            COSPHI:=COS(PHIO); SINPHI:=SIN(PHIO);
            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*PHIO;
            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" PHIO^=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" BO:=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/BO)>.05 "OR" PHI^=PHIO;
        "IF" 1.1*H>=XE-X "THEN"
        "BEGIN" CHANGE:=LAST:="TRUE"; H:=XE-X "END";
        "IF" "NOT" CHANGE "THEN" H:=H*BO/B
    "END" STEPSIZE;

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

    BO:=PHIO:=-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 XO;
        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=XO;
        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 XO 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] = .765 876 320 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 XO;
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=XO;
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 XO 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(-47D8),2(48+.3D83D83D),-5Z0.3D,/")",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;                                          "COMMENT"
```

```
    "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"
```

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, NSJEV, 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 PROGRAM DELIVERS:

```
X = 50   N =  109   JEV =    3   LU =   12

Y1 = +.5976547958004"+00  REL. ERR. = .16"-06
Y2 = +.1402343310813"+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, R1, 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]:= 8;
           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:= ABTA + 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);
"VALUE" 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;
    YO:      <ARRAY IDENTIFIER>;
             "REAL" "ARRAY" YO[1:N];
             ENTRY: THE  INITIAL  VALUES  OF  THE SYSTEM OF DIFFERENTIAL
                    EQUATIONS: YO[I] AT T=TO;
    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];
    HO:      <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.
IMPLEX 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 JACOBIAN 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:=HO "ELSE"
    "BEGIN" "IF" HO>HMAX "THEN" H:=HMAX "ELSE" H:=HO;
        "IF" H>(TEND-TO)/4 "THEN" H:=(TEND-TO)/4;
    "END";
    HNEW:=H;
    ALF:=0; T:=TP:=TO;
    INIVEC(1,3,ERR,0); INIVEC(1,N,EHR,0);
    DUPROWVEC(1,N,1,R,YO);
    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,MO,M,U,SIGMA,TAUMIN,I,DERIVATIVE,
                                 K,DATA,ALFA,NORM,AETA,RETA,ETA,RHO,OUT);
    "INTEGER" MO,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 TO;
             EXIT : THE FINAL VALUE TE;
    TE:      <ARITHMETIC EXPRESSION>;
             THE FINAL VALUE OF T (TE >= T);
    MO,M:    <ARITHMETIC EXPRESSION>;
             INDICES OF THE FIRST AND 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 = TO;
             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 MO 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[MO : 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, MO, M AND THE ARRAY DATA SEE REFERENCES [2] AND [3].
AS FOR THE INDICES MO 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 MO 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 - MO.

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

    NUMBER OF STEPS: 424
    SOLUTION:  T= +7.38906    U(T) = +1.9999967
```

SOURCE TEXT(S):

```
"CODE"" 33040;
"PROCEDURE" MODIFIED TAYLOR(T,TE,MO,M,U,SIGMA,TAUMIN,I,DERIVATIVE,K,
                  DATA,ALFA,NORM,AETA,RETA,ETA,RHO,OUT);
"INTEGER" MO,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[MO: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 TO;
            EXIT : THE FINAL VALUE TE;
    TE:     <APITHMETIC 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 = TO;
            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,MO,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,"("4B0"-D,37.3D,/")",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,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;
"BEGIN" "INTEGER" KL;
    "REAL" Q,ECO,EC1,EC2,H,HI,HO,H1,H2,BETAN,T2,SIGMAL,PHIL;
    "REAL" "ARRAY" C,ROOMO: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:=MO "STEP" 1 "UNTIL" M "DO"
            "BEGIN" X:=ABS(W[J]); "IF" X>S "THEN" S:=X "END"
        "END" "ELSE"
        S:=SQRT(VECVEC(MO,M,O,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(MO,M,RO,O);
        "IF" I<4 "THEN" ELMVEC(MO,M,O,RO,C,BETHA[I]*HI);
        "IF" I=4 "THEN"
        "BEGIN" ELMVEC(MO,M,O,RO,C,-H);
            RHO:=NORMFUNCTION(NORM,RO);
            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 INITIAL VALUE PROBLEMS FOR SECOND ORDER ORDINARY DIFFERENTIAL EQUATIONS.

A. RK2 SOLVES AN IVP FOR A SINGLE SECOND ORDER ODE BY MEANS OF A 5-TH ORDER RUNGE-KUTTA METHOD.

B. RK2N SOLVES AN IVP FOR A SYSTEM OF SECOND ORDER ODE'S BY MEANS OF A 5-TH ORDER RUNGE-KUTTA METHOD

C. RK3 SOLVES AN IVP FOR A SINGLE SECOND ORDER ODE WITHOUT FIRST DERIVATIVE. RK3 IS BASED ON A 5-TH ORDER RUNGE-KUTTA METHOD.

D. RK3N SOLVES AN IOVP FOR A SYSTEM OF SECOND ORDER ODE'S WITHOUT FIRST DERIVATIVE. RK3N IS BASED ON A 5-TH ORDER RUNGE-KUTTA METHOD.

PROCEDURE : RK2.

AUTHOR: J.A.ZONNEVELD.

CONTRIBUTORS: M.BAKKER AND I.BRINK.

INSTITUTE: MATHEMATICAL CENTRE.

RECEIVED: 730715.

BRIEF DESCRIPTION:

RK2 INTEGRATES THE SCALAR INITIAL VALUE PROBLEM
(D/DX) (D/DX) Y = F(X, Y, (D/DX)Y), A<= X <=B OR B <= X <= A,
Y(A) AND (D/DX) Y(A) PRESCRIBED.

KEYWORDS:

INITIAL VALUE PROBLEM,
SECOND ORDER DIFFERENTIAL EQUATION.

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;
    "CODE" 33012;

    THE MEANING OF THE FORMAL PARAMETERS IS:
    X:  <VARIABLE>;
        THE INDEPENDENT VARIABLE;
    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;
        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;
        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], AND
        A, YA AND ZA ARE IGNORED.
```

PROCEDURES USED: NONE.

METHOD AND PERFORMANCE :
THE PROCEDURE, WHICH IS PROVIDED WITH STEPLENGTH AND ERROR CONTROL, IS BASED ON A 5-TH ORDER RUNGE-KUTTA METHOD.
A COMPLETE DESCRIPTION IS GIVEN IN [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(YL) * 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"
```

PROCEDURE : RK2N.

AUTHOR: J.A.ZONNEVELD.

CONTRIBUTORS: M.BAKKER AND I.BRINK.

INSTITUTE : MATHEMATICAL CENTRE.

RECEIVED: 730715.

BRIEF DESCRIPTION:

RK2N INTEGRATES THE VECTOR INITIAL VALUE PROBLEM
(D/DX) (D/DX) Y = F(X, Y, (D/DX) Y), A<= X <= B OR B <= X <= A,
Y[J] (A) AND (D/DX) Y[J] (A) PRESCRIBED FOR J=1,....N.

KEYWORDS :

INITIAL VALUE PROBLEM,
SECOND ORDER DIFFERENTIAL EQUATION.

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;
    "CODE" 33013;

    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[J],Z[J]  (J=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]), AND
          A, YA, ZA ARE IGNORED;
    N:    <ARITHMETIC EXPRESSION>;
          THE NUMBER OF EQUATIONS.
```

PROCEDURES USED: NONE.

REQUIRED CENTRAL MEMORY:
EIGHT ARRAYS OF ORDER N AND ONE OF ORDER 4 * N ARE USED.

METHOD AND PERFORMANCE :
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 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 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].

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,B,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]=+.00000000002955     Y[2]-YEXACT[2]=+.0000000000567
 Z[1]-ZEXACT[1]=-.00000000013770     Z[2]-ZEXACT[2]=-.0000000002422

      X=2.0000
 Y[1]-YEXACT[1]=-.00000000085294     Y[2]-YEXACT[2]=+.0000000001486
 Z[1]-ZEXACT[1]=+.00000000378800     Z[2]-ZEXACT[2]=-.0000000006509

      X=3.0000
 Y[1]-YEXACT[1]=-.00000000162707     Y[2]-YEXACT[2]=-.0000000004796
 Z[1]-ZEXACT[1]=+.00000000803265     Z[2]-ZEXACT[2]=+.0000000019380

      X=4.0000
 Y[1]-YEXACT[1]=-.00000000117993    Y[2]-YEXACT[2]=-.0000000008505
 Z[1]-ZEXACT[1]=+.00000000633393     Z[2]-ZEXACT[2]=+.0000000039114
```

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, KO, 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" KO[J]:= FXYZJ * H;
      X:= XL + H / 4.5;
      "FOR" JJ:= 1 "STEP" 1 "UNTIL" N "DO"
      "BEGIN" Y[JJ]:= YL[JJ] + (ZL[JJ] * 18 + KO[JJ] * 2) /
         81 * H; Z[JJ]:= ZL[JJ] + KO[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 + KO[JJ]) / 18 * H;
         Z[JJ]:= ZL[JJ] + (KO[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 + KO[JJ] + K2[JJ])
         / 16 * H;
         Z[JJ]:= ZL[JJ] + (KO[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 + KO[JJ] * 12 +
         K3[JJ] * 28) / 125 * H;
         Z[JJ]:= ZL[JJ] + (KO[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 + KO[JJ] * 21 +
         K2[JJ] * 92 + K4[JJ] * 55) / 336 * H;
         Z[JJ]:= ZL[JJ] + (KO[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(( - KO[JJ] * 21 + K2[JJ] * 108 -
         K3[JJ] * 112 + K4[JJ] * 25) / 56 * H);
         DISCRZ:= ABS(KO[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(KO[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 + KO[JJ] * 7 +
         K2[JJ] * 36 - K4[JJ] * 15) / 56 * HL;
         Z[JJ]:= ZL[JJ] + ( - KO[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 + KO[JJ] * 35 +
         K2[JJ] * 108 + K4[JJ] * 25) / 336 * HL;
         Z[JJ]:= ZL[JJ] + (KO[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"
```

PROCEDURE : RK3

AUTHOR: J.A.ZONNEVELD.

CONTRIBUTORS: M.BAKKER AND I.BRINK.

INSTITUTE: MATHEMATICAL CENTRE.

RECEIVED: 730715.

BRIEF DESCRIPTION:

RK3 INTEGRATES THE SCALAR 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) AND (D/DX) Y(A) PRESCRIBED.

KEYWORDS:

INITIAL VALUE PROBLEM,
SECOND ORDER DIFFERENTIAL EQUATION.

CALLING SEQUENCE:

```
    THE HEADING OF THE PROCEDURE READS:
    "PROCEDURE" RK3(X,A,B,Y,YA,Z,ZA,FXY,E,D,FI); "CODE" 33014;
    "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.
```

PROCEDURES USED : NONE.

METHOD AND PERFORMANCE :
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].

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; KO:= FXY * H "END"
      "ELSE" KO:= K5 * H / HL; X:= XL + .276393202250021 * H;
      Y:= YL + (ZL * .2763932022 50021 + KO *
      .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 + KO * .046875 + K1 *
      .079824155839840 - K2 * .001699155839840) * H;
      K4:= FXY * H; X:= "IF" LAST "THEN" B "ELSE" XL + H;
      Y:= YL + (ZL + KO * .309016994374947 + K2 *
      .190983005625053) * H; K3:= FXY * H;
      Y:= YL + (ZL + KO * .083333333333333 + K1 *
      .301502832395825 + K2 * .115163834270842) * H;
      K5:= FXY * H;
      DISCRY:= ABS(( - KO * .5 + K1 * 1.809016994374947 +
      K2 * .690983005625053 - K4 * 2) * H);
      DISCRZ:= ABS((KO - K3) * 2 - (K1 + K2) * 10 + K4 *
      16 + K5 * 4); TOLY:= ABSH * (ABS(ZL) * E1 + E2);
      TOLZ:= ABS(KO) * 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 + (KO + 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"
```

PROCEDURE : RK3N.

AUTHOR: J.A.ZONNEVELD.

CONTRIBUTORS: M.BAKKER AND I.BRINK.

INSTITUTE: MATHEMATICAL CENTRE.

RECEIVED: 730715.

BRIEF DESCRIPTION:

RK3N INTEGRATES THE VECTOR INITIAL VALUE PROBLEM
(D/DX) (D/DX) Y = F(X,Y), A <= X <= B OR B <= X <= A,
Y[J] (A) AND (D/DX) Y[J] (A) PRESCRIBED.

KEYWORDS:

INITIAL VALUE PROBLEM,
SECOND ORDER DIFFERENTIAL EQUATION.

CALLING SEQUENCE:

THE HEADING OF THE PROCEDURE READS:

```
"PROCEDURE" RK3N(X,A,B,Y,YA,Z,ZA,FXYJ,J,E,D,FI,N); "CODE" 33015;
"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.
```

PROCEDURES USED: NONE.

REQUIRED CENTRAL MEMORY:
EIGHT ARAYS OF ORDER N AND ONE OF ORDER 4 * N ARE USED.

METHOD AND PERFORMANCE :
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].

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:= "TRUE";
        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]
      * .276393202250021 + K0[JJ] * .038196601125011) * H;
      "FOR" J:= 1 "STEP" 1 "UNTIL" N "DO" K1[J]:= FXYJ * H;
      X:= XL + .723606797749979 * H;
      "FOR" JJ:= 1 "STEP" 1 "UNTIL" N "DO" Y[JJ]:= YL[JJ] + (ZL[JJ]
      * .723606797749979 + 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 9155839840) * 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]
    + KO[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(( - KO[JJ] * .5 + K1[JJ] *
       1.809016994374947 + K2[JJ] * .690983005625053 -
       K4[JJ] * 2) * H);
       DISCRZ:= ABS((KO[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(KO[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] + (KO[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;
        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.

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:

$$DU / DT = V(T, X, Y),$$
$$DV / DT = D(DU / DX) / DX + D(DU / DY) / DY, \quad (1)$$

(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
                             ,.01082794375,.01842733851,.03278507942,
                             .0653627415,.1691078577,156 "DO"
                   "BEGIN" LAMBDA[I]:=LBD; I:=I+1 "END";
                   "GOTO" EXIT;

    PARABOLIC2:    "FOR" LBD:=.3534355908"-2,.8532600867"-2,.015956206
                             ,.02772229155,.04812587964,.08848689452,
                             .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(1,1,1,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/ REVISED 791231.

BRIEF DESCRIPTION:

THIS SECTION CONTAINS THREE 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

$$-(P(X)*Y')' + R(X)*Y = F(X),\ A < X < B,$$

WITH BOUNDARY CONDITIONS

$$E[1]*Y(A) + E[2]*Y'(A) = E[3],$$

$$E[4]*Y(B) + E[5]*Y'(B) = E[6].$$

(2) FEM LAG;

THIS PROCEDURE SOLVES THE DIFFERENTIAL EQUATION

$$-Y'' + R(X)*Y = F(X),\ A < X < B,$$

WITH BOUNDARY CONDITIONS

$$E[1]*Y(A) + E[2]*Y'(A) = E[3],$$

$$E[4]*Y(B) + E[5]*Y'(B) = E[6].$$

(3) FEM LAG SPHER:

THIS PROCEDURE SOLVES THE DIFFERENTIAL EQUATION

WITH SPHERICAL COORDINATES

$$-(X^{NC}*Y')'/X^{NC} + R(X)*Y = F(X),\ A < X < B,$$

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,
SELF-ADJOINT BOUNDARY VALUE PROBLEMS,
RITZ-GALERKIN METHOD,
SPHERICAL COORDINATES,
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

$$(1) \qquad -(P(X)*Y')' + R(X)*Y = F(X),\ A < X < B,$$

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) > - PO*(PI/(X[N] - X[0]))**2,

WHERE PO 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:= 3.14159265358979;
    "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 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 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 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 = 3.14159265358979,
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:= 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,48"("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.

SUBSECTION: FEM LAG SPHER.

CALLING SEQUENCE:

THE HEADING OF THE PROCEDURE READS:

```
"PROCEDURE" FEM LAG SPHER(X, Y, N, NC, R, F, ORDER, E);
"VALUE" N, NC, ORDER; "INTEGER" N, NC, ORDER;
"ARRAY" X, Y, E;
"REAL" "PROCEDURE" R, F;
"CODE" 33308;
```

THE MEANING OF THE FORMAL PARAMETERS IS:

N: <ARITHMETIC EXPRESSION>;
THE UPPER BOUND OF THE ARRAYS X AND Y; N > 1;

NC: <EXPRESSION>;
IF NC = 0, CARTESIAN COORDINATES ARE USED;
IF NC = 1, POLAR COORDINATES ARE USED;
IF NC = 2, SPHERICAL COORDINATES ARE USED;

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) $\quad -(X^{NC}Y')'/X^{NC} + R(X)Y = F(X),\ A < X < B,$

WITH BOUNDARY CONDITIONS

(2)
$$E[1]Y(A) + E[2]Y'(A) = E[3],$$
$$E[4]Y(B) + E[5]Y'(B) = E[6];$$

R: <PROCEDURE IDENTIFIER>;
THE HEADING OF R READS:
"REAL" "PROCEDURE" R(X); "VALUE" X; "REAL" X;
R(X) IS THE COEFFICIENT OF Y IN (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 OR 4 ONLY;

E: <ARRAY IDENTIFIER>;
"ARRAY" E[1:6];
E[1], ... , E[6] DESCRIBE THE BOUNDARY CONDITIONS (2); 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 EVALUATIONS OF R(X) AND F(X) ARE NEEDED;
(B) IF NC > 0 AND ORDER=4, THEN N SQUARE ROOTS ARE EVALUATED;

DATA AND RESULTS:

THE PROCEDURE FEM LAG SPHER HAS SOME RESTRICTIONS IN ITS USE: R(X) AND F(X) ARE REQUIRED TO BE SUFFICIENTLY SMOOTH ON <X[0],X[N]> EXCEPT AT THE GRID POINTS; FURTHERMORE R(X) SHOULD BE NONNEGATIVE.

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 SOME PIECEWISE K-POINT GAUSSIAN QUADRATURE (SEE [4]); 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 [3]).

EXAMPLE OF USE:

WE SOLVE THE BOUNDARY VALUE PROBLEM

$$-(Y'*X**NC)'/X**NC + Y = 1 - X**4 + (12 + 4*NC)*X**2,$$
$$0 < X < 1; \quad Y'(0) = Y(1) = 0;$$

FOR THE BOUNDARY CONDITIONS THIS IMPLIES THAT

$$E[2] = E[4] = 1; \quad E[1] = E[3] = E[5] = E[6] = 0;$$

THE ANALYTIC SOLUTION IS Y(X) = 1 - X**4; WE APPROXIMATE
THE SOLUTION ON A UNIFORM GRID, I.E. X[I] = I/N, I = 0, ..., N;
I = 0, ..., N; WE CHOOSE N=10,20 AND COMPUTE FOR ORDER = 2,4
THE MAXIMUM ERROR; THE PROGRAM READS AS FOLLOWS:

```
"BEGIN" "INTEGER" N, NC;
    "FOR" N:= 10, 20 "DO" "FOR" NC:= 0, 1, 2 "DO"
"BEGIN" "INTEGER" I, ORDER; "ARRAY" X, Y[0:N], E[1:6];

    "REAL" "PROCEDURE" R(X); "VALUE" X; "REAL" X;
    R:= 1;

    "REAL" "PROCEDURE" F(X); "VALUE" X; "REAL" X;
    F:= (12 + 4*NC)*X**2 + 1 - X**4;

    "PROCEDURE" FEM LAG SPHER(X, Y, N, NC, R, F, ORDER, E);
    "CODE" 33308;

    E[2]:= E[4]:= 1; E[1]:= E[3]:= E[5]:= E[6]:= 0;
    "FOR" I:= 0 "STEP" 1 "UNTIL" N "DO" X[I]:= I/N;
    OUTPUT(61,"("//,6B"("N=")"ZZD,6B"("NC=")"ZD")",N,NC);
    "FOR" ORDER:= 2, 4 "DO"
    "BEGIN" "REAL" RHO, D;
        FEM LAG SPHER(X, Y, N, NC, R, F, ORDER, E);
        RHO:= 0;
        "FOR" I:= 0 "STEP" 1 "UNTIL" N "DO"
        "BEGIN" D:= ABS(Y[I] - 1 + X[I]**4);
            "IF" RHO < D "THEN" RHO:= D
        "END";
        OUTPUT(61,"("/,16B"(" ORDER=")"ZD,4B"("MAX.ERROR= ")",
        D.DD"+ZD")",ORDER,RHO)
    "END"
"END"
"END"
```

RESULTS:

N= 10	NC= 0		
	ORDER= 2	MAX.ERROR=	4.37" -3
	ORDER= 4	MAX.ERROR=	2.93" -6
N= 10	NC= 1		
	ORDER= 2	MAX.ERROR=	1.42" -2
	ORDER= 4	MAX.ERROR=	5.49" -5
N= 10	NC= 2		
	ORDER= 2	MAX.ERROR=	2.46" -2
	ORDER= 4	MAX.ERROR=	1.27" -4
N= 20	NC= 0		
	ORDER= 2	MAX.ERROR=	1.09" -3
	ORDER= 4	MAX.ERROR=	1.83" -7
N= 20	NC= 1		
	ORDER= 2	MAX.ERROR=	3.53" -3
	ORDER= 4	MAX.ERROR=	3.91" -6
N= 20	NC= 2		
	ORDER= 2	MAX.ERROR=	6.10" -3
	ORDER= 4	MAX.ERROR=	9.26" -6

ONE OBSERVES 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 + 0.27639320225*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:= -(+ 4.04508497187450*P1
               + 0.57581917135425*P3
               + 0.23751416197911*P4)/H;
      A13:= (+ 1.54508497187747*P1
              - 1.5075141619791*P2
              + 0.67418082864587*P4)/H;
      A14:= ((P2 + P3)/2.4 - (P1 + P4)/2)/H;
      A22:= (5.454237476562*P1 + P3/.48 +.79576252343762*P4)/H + TAU2;
      A23:= - (P1 + P4)/(H*0.48);
      A24:= (+ 0.67418082864575*P1
              - 1.50751416197910*P3
              + 1.54508497187470*P4)/H;
      A33:= (.7957625234376*P1 + P2/.48 + 5.454237476562*P4)/H + TAU3;
      A34:= -(+ 0.23751416197911*P1
               + 0.57581917135418*P2
               + 4.04508497187747*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 + 0.27639320225*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:= -4.878418305207 8/H; A13:= A24:= 0.7117516385412/H;
       A14:= -0.16666666666667/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"
```

```
"CODE" 33308;
 "PROCEDURE" FEM LAG SPHER(X, Y, N, NC, R, F, ORDER, E);
 "VALUE" N, NC, ORDER; "INTEGER" N, NC, 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,
     TAU3, B3, A13, A22, A23, C32, C12,
       E1, E2, E3, E4, E5, E6;
   "ARRAY" T, SUB, CHI, GI[0:N-1];

   "PROCEDURE" ELEMENT MAT VEC EVALUATION 1;
   "BEGIN" "REAL"  XM, VL, VR,WL, WR, PR, RM, FM, XL2, XLXR, XR2;
     "IF" NC = 0 "THEN" VL:= VR:= 0.5 "ELSE" "IF" NC = 1 "THEN"
     "BEGIN" VL:= (XL1*2 + XL)/6; VR:= (XL1 + XL*2)/6 "END" "ELSE"
     "BEGIN" XL2:= XL1*XL1/12; XLXR:=XL1*XL/6; XR2:=XL*XL/12;
       VL:= 3*XL2 + XLXR + XR2;
       VR:= 3*XR2 + XLXR + XL2
     "END";

     WL:= H*VL; WR:=H*VR; PR:= VR/(VL +VR);
     XM:= XL1 + H*PR; FM:= F(XM); RM:=R(XM);
     TAU1:= WL*RM; TAU2:=WR*RM;
     B1:= WL*FM; B2:= WR*FM; A12:= - (VL + VR)/H + H*(1 - PR)*PR*RM
   "END" ELEM. M.V. EV.;

   "PROCEDURE" ELEMENT MAT VEC EVALUATION 2;
   "BEGIN" "REAL" XLM, XRM, VLM, VRM, WLM, WRM, FLM, FRM,
     RLM, RRM, PL1, PL2, PL3, PR1, PR2, PR3, QL1, QL2, QL3,
     RLMPL1, RLMPL2, RLMPL3, RRMPR1, RRMPR2, RRMPR3,
     VLMQL1, VLMQL2, VLMQL3, VRMQR1, VRMQR2, VRMQR3,
     QR1, QR2,QR3;

     "IF" NC = 0 "THEN"
     "BEGIN" XLM:=XL1 + H*0.2113248654052; XRM:= XL1 + XL - XLM;
       VLM:= VRM:= 0.5;
       PL1:= PR3:= 0.455341801261 48; PL3:= PR1:= -0.122008467928 15;
       PL2:= PR2:= 1 - PL1 - PL3;
       QL1:= - 2.154700538379 25; QL3:= -0.154700538379 25;
       QL2:= - QL1 - QL3; QR1:= - QL3; QR3:= - QL1; QR2:= - QL2;
     "END" "ELSE" "IF" NC = 1 "THEN"
     "BEGIN" "REAL" A, A2, A3, A4, B, B2, B3, B4, P4H,
       P2, P3, P4, AUX1, AUX2;
       A:= XL1; A2:= A*A; A3:= A*A2; A4:= A*A3;
       B:= XL; B2:= B*B; B3:= B*B2; B4:= B*B3;
       P2:= 10*(A2 + 4*A*B + B2); P3:= 6*(A3 + 4*(A2*B + A*B2) + B3);
       P4:= SQRT(6*(A4 + 10*(A*B3 + A3*B) + 28*A2*B2 + B4));
       P4H:= P4*H; XLM:= (P3 - P4H)/P2; XRM:= (P3 + P4H)/P2;
       AUX1:= (A + B)/4; AUX2:= H*(A2 + 7*A*B + B2)/6/P4;
       VLM:= AUX1 - AUX2; VRM:= AUX1 + AUX2;
                                                          "COMMENT"
```

```
;

    "END" "ELSE"
    "BEGIN" "REAL" A, A2, A3, A4, A5, A6, A7, A8,
      B, B2, B3, B4, B5, B6, B7, B8, AB4, A2B3, A3B2, A4B,
      P4, P5, P8, P8H, AUX1, AUX2;
      A:= XL1; A2:= A*A; A3:= A*A2; A4:= A*A3; A5:= A*A4; A6:= A*A5;
        A7:= A*A6; A8:= A*A7;
      B:= XL; B2:= B*B; B3:= B*B2; B4:= B*B3; B5:= B*B4; B6:= B*B5;
        B7:= B*B6; B8:= B*B7;
      AB4:= A*B4; A2B3:= A2*B3; A3B2:= A3*B2; A4B:=A4*B;
      P4:= 15*(A4 + 4*(A3*B + A*B3) + 10*A2*B2 + B4);
      P5:= 10*(A5 + 4*(A4B + AB4) + 10*(A3B2 + A2B3) + B5);
      P8:= SQRT(10*(A8 + 10*(A7*B + A*B7) + 55*(A2*B6 + A6*B2)
          + 164*(A5*B3 +A3*B5) + 290*A4*B4 + B8));
      AUX1:= (A2 +A*B + B2)/6; P8H:= P8*H;
      AUX2:= (H*(A5 + 7*(A4B + AB4) + 28*(A3B2 + A2B3) + B5))/4.8/P8;
      XLM:= (P5 - P8H)/P4; XRM:= (P5 + P8H)/P4;
      VLM:= AUX1 - AUX2; VRM:= AUX1 + AUX2
    "END";

    "IF" NC > 0 "THEN"
    "BEGIN" "REAL" AUX, PLM, PRM;
      PLM:= (XLM - XL1)/H; PRM:= (XRM - XL1)/H;
      AUX:= 2*PLM - 1; PL1:= AUX*(PLM - 1); PL3:= AUX*PLM;
        PL2:= 1 - PL1 - PL3;
      AUX:= 2*PRM - 1; PR1:= AUX*(PRM - 1); PR3:= AUX*PRM;
        PR2:= 1 - PR1 - PR3;
      AUX:= 4*PLM; QL1:= AUX - 3; QL3:= AUX - 1; QL2:= - QL1 - QL3;
      AUX:= 4*PRM; QR1:= AUX - 3; QR3:= AUX - 1; QR2:= - QR1 - QR3;
    "END";

    WLM:= H*VLM; WRM:= H*VRM; VLM:= VLM/H; VRM:= VRM/H;
    FLM:= F(XLM)*WLM; FRM:= WRM*F(XRM);
    RLM:= R(XLM)*WLM; RRM:= WRM*R(XRM);
    TAU1:= PL1*RLM + PR1*RRM;
    TAU2:= PL2*RLM + PR2*RRM;
    TAU3:= PL3*RLM + PR3*RRM;
    B1:= PL1*FLM + PR1*FRM;
    B2:= PL2*FLM + PR2*FRM;
    B3:= PL3*FLM + PR3*FRM;
    VLMQL1:= QL1*VLM; VRMQR1:= QR1*VRM;
    VLMQL2:= QL2*VLM; VRMQR2:= QR2*VRM;
    VLMQL3:= QL3*VLM; VRMQR3:= QR3*VRM;
    RLMPL1:= RLM*PL1; RRMPR1:= RRM*PR1;
    RLMPL2:= RLM*PL2; RRMPR2:= RRM*PR2;
    RLMPL3:= RLM*PL3; RRMPR3:= RRM*PR3;
                                                    "COMMENT"
```

```
    A12:= VLMQL1*QL2 + VRMQR1*QR2 + RLMPL1*PL2 + RRMPR1*PR2;
    A13:= VLMQL1*QL3 + VRMQR1*QR3 + RLMPL1*PL3 + RRMPR1*PR3;
    A22:= VLMQL2*QL2 + VRMQR2*QR2 + RLMPL2*PL2 + RRMPR2*PR2;
    A23:= VLMQL2*QL3 + VRMQR2*QR3 + RLMPL2*PL3 + RRMPR2*PR3;
    "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" 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:= ("IF" NC = 0 "THEN" 1 "ELSE" X[0]**NC)/E2;
      B1:= B1 - E3*AUX; TAU1:= TAU1 - E1*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:= ("IF" NC = 0 "THEN" 1 "ELSE" X[N]**NC)/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;
      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
```

```
   "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"
                             ELEMENT MAT VEC EVALUATION 2;
       "IF" L=1 "OR" L=N "THEN" BOUNDARY CONDITIONS;
       FORWARD BABUSHKA
   "END";
   BACKWARD BABUSHKA;
"END" FEM LAG SPHER;
      "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

$$-Y'' + Q(X)*Y' + R(X)*Y = F(X),\ A < X < B,$$

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) - Y'' + Q(X)*Y' + R(X)*Y = F(X), A < X < B,

WITH BOUNDARY CONDITIONS

(2)
$$E[1]*Y(A) + E[2]*Y'(A) = E[3],$$
$$E[4]*Y(B) + E[5]*Y'(B) = E[6];$$

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

$$-Y'' + Y'*COS(X) + Y*EXP(X) = SIN(X)*(1 + EXP(X)) + COS(X)**2,$$
$$0 < X < PI = 3.14159265358979,\ Y(0) = Y(PI) = 0;$$

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" 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(S):

```
"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 + 0.27639320225*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:= -4.8784183052080/H; S13:= 0.7117516385414/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 + 0.67418082864578*Q1;
      A13:= S13 - 0.25751416197912*Q1;
      A14:= S14 + Q1/12;
      A21:= S12 - 0.67418082864578*Q2;
      A22:= S22 + TAU2;
      A23:= S23 + 0.93169499062490*Q2;
      A24:= S13 - 0.25751416197912*Q2;
      A31:= S13 + 0.25751416197912*Q3;
      A32:= S23 - 0.93169499062490*Q3;
      A33:= S22 + TAU3;
      A34:= S12 + 0.67418082864578*Q3;
      A41:= S14 - Q4/12;
      A42:= S13 + 0.25751416197912*Q4;
      A43:= S12 - 0.67418082864578*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

$$(P(X)*Y''(X))'' - (Q(X)*Y'(X))' + R(X)*Y = F(X), \quad A < X < B,$$

WITH BOUNDARY CONDITIONS

$$Y(A) = E[1], \; Y'(A) = E[2],$$

$$Y(B) = E[3], \; Y'(B) = E[4].$$

KEY WORDS AND PHRASES:

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 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;
"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 FCM 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 = 3.14159265358979;
```

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" P1; "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;
                                                        "COMMENT"
```

```
      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:= 0.27639320225*H + XL1; X3:= XL1 + XL - X2;

      "COMMENT" ELEMENT BENDING MATRIX;
      P1:= P4; P2:= P(X2); P3:= P(X3); P4:= P(XL);
      B11:= + 4.0333333333333"+1*P1 + 1.1124913866738"-1*P2
            + 1.4422084194664"+1*P3 + 8.3333333333333"+0*P4;
      B12:= + 1.4666666666667"+1*P1 - 3.3191425091659"-1*P2
            + 2.7985809175818"+0*P3 + 1.6666666666667"+0*P4;
      B13:= + 1.8333333333333"+1*(P1+P4)
            + 1.2666666666667"+0*(P2+P3);
      B15:= - (B11 + B13); B14:= - (B12 + B13 + B15/2);
      B22:= + 5.3333333333333"+0*P1 + 9.9027346441674"-1*P2
            + 5.4305986891624"-1*P3 + 3.3333333333333"-1*P4;
      B23:= + 6.6666666666667"+0*P1 - 3.7791278464167"+0*P2
            + 2.4579451308295"-1*P3 + 3.6666666666667"+0*P4;
      B25:= - (B12 + B23); B24:= - (B22 + B23 + B25/2);
      B33:= + 8.3333333333333"+0*P1 + 1.4422084194666"+1*P2
            + 1.1124913866726"-1*P3 + 4.0333333333333"+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:= + 2.6844168389330"+0*Q2 + 2.2249827733448"-2*Q3;
S12:= + 2.5671051872498"-1*Q2 + 3.2894812749994"-3*Q3;
S13:= + 2.5333333333333"-1*(Q2+Q3);
S14:= - 3.7453559925005"-2*Q2 - 2.2546440074988"-2*Q3;
S15:= - (S13 + S11);
S22:= + 8.3333333333333"-2*Q1 + 2.2847006554164"-2*Q2
      + 4.8632677916445"-4*Q3;
S23:= + 2.2546440075002"-2*Q2 + 3.7453559924873"-2*Q3;
S24:= - 3.3333333333333"-3*(Q2+Q3);
S25:= - (S12 + S23);
S33:= + 2.2249827733471"-2*Q2 + 2.6844168389330"+0*Q3;
S34:= - 3.2894812750127"-3*Q2 - 2.5671051872496"-1*Q3;
S35:= - (S13 + S33);
S44:= + 4.8632677916788"-4*Q2
      + 2.2847006554161"-2*Q3 + 8.3333333333338"-2*Q4;
S45:= - (S14 + S34);
S55:= - (S15 + S35);

"COMMENT" ELEMENT MASS MATRIX;
R1:= R4; R2:= R(X2); R3:= R(X3); R4:= R(XL);
M11:= + 8.3333333333333"-2*R1 + 1.0129076086083"-1*R2
      + 7.3759058058380"-3*R3;
M12:= + 1.3296181273333"-2*R2 + 1.3704853933353"-3*R3;
M13:= - 2.7333333333333"-2*(R2+R3);
M14:= + 5.0786893258335"-3*R2 + 3.5879773408333"-3*R3;
M15:= + 1.3147987115999"-1*R2 - 3.5479871159991"-2*R3;
M22:= + 1.7453559925000"-3*R2 + 2.5464400750059"-4*R3;
M23:= - 3.5079773408336"-3*R2 - 5.0786893258385"-3*R3;
M24:= + 6.6666666666667"-4*(R2+R3);
M25:= + 1.7259029213333"-2*R2 - 6.5923625466719"-3*R3;
M33:= + 7.3759058058380"-3*R2
      + 1.0129076086083"-1*R3 + 8.3333333333333"-2*R4;
M34:= - 1.3704853933333"-3*R2 - 1.3296181273333"-2*R3;
M35:= - 3.5479871159992"-2*R2 + 1.3147987115999"-1*R3;
M44:= + 2.5464400750008"-4*R2 + 1.7453559924997"-3*R3;
M45:= + 6.5923625466656"-3*R2 - 1.7259029213330"-2*R3;
M55:= + .17066666666667"+0*(R2+R3);
                                                    "COMMENT"
```

```
;

      "COMMENT" ELEMENT LOAD VECTOR;
      F1:= F4; F2:= F(X2); F3:= F(X3); F4:= F(XL);
      B1:= + 8.3333333333333"-2*F1 + 2.0543729868749"-1*F2
           - 5.5437298687489"-2*F3;
      B2:= + 2.6967233145832"-2*F2 - 1.0300566479175"-2*F3;
      B3:= - 5.5437298687489"-2*F2
           + 2.0543729868749"-1*F3 + 8.3333333333333"-2*F4;
      B4:= + 1.0300566479165"-2*F2 - 2.6967233145830"-2*F3;
      B5:= + 2.6666666666667"-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      + 7.3593121303513"-2*P2
      + 2.2755555555556"+1*P3 + 7.0565656088553"+0*P4;
B12:= + 27.6*P1  + 1.4*P5      - 3.41554824811"-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
      + 7.3593121303513"-2*P4 + 105.8*P5 + 9.8*P1;
B34:= - 1.4*P1 - 27.6*P5       - 1.0113960946522"+0*P2
      - 2.8444444444444"+0*P3 + 3.4155482481100"-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 + 3.1539909130065"-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:= + 3.1539909130065"-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:= + 9.7107020727310"-2*R2 + 1.5810259199180"-3*R4 + R1/20;
M12:= + 8.2354889460254"-3*R2 + 2.1932154960071"-4*R4;
M13:= + 1.2390670553936"-2*(R2+R4);
M14:= - 1.7188466249968"-3*R2 - 1.0508326752939"-3*R4;
M15:= + 5.3089789712119"-2*R2 + 6.7741558661060"-3*R4;
M16:= - 1.7377712856076"-2*R2 + 2.2173630018466"-3*R4;
M22:= + 6.9843846173145"-4*R2 + 3.0424512029349"-5*R4;
M23:= + 1.0509326752947"-3*R2 + 1.7188466249936"-3*R4;
M24:= - 1.4577259475206"-4*(R2+R4);
M25:= + 4.5024589679127"-3*R2 + 9.3971790283374"-4*R4;
M26:= - 1.4737756452780"-3*R2 + 3.0759488725998"-4*R4;
M33:= + 1.5810259199209"-3*R2 + 9.7107020727290"-2*R4 + R5/20;
M34:= - 2.1932154960131"-4*R2 - 8.2354889460254"-3*R4;
M35:= + 6.7741558661123"-3*R2 + 5.3089789712112"-2*R4;
M36:= - 2.2173630018492"-3*R2 + 1.7377712856071"-2*R4;
M44:= + 3.0424512029457"-5*R2 + 6.9843846173158"-4*R4;
M45:= - 9.3971790283542"-4*R2 - 4.5024589679131"-3*R4;
M46:= + 3.0759488726060"-4*R2 - 1.4737756452778"-3*R4;
M55:= + 2.9024943310657"-2*(R2+R4) + 3.5555555555556"-1*R3;
M56:= + 9.5006428402050"-3*(R4-R2);
M66:= + 3.1098153547125"-3*(R2+R4);

"COMMENT" ELEMENT LOAD VECTOR;
F1:= F5; F2:= F(X2); F3:= F(X3); F4:= F(X4); F5:= F(XL);
B1:= + 1.6258748099336"-1*F2 + 2.0745852339969"-2*F4 + F1/20;
B2:= + 1.3788780589233"-2*F2 + 2.8778860774335"-3*F4;
B3:= + 2.0745852339969"-2*F2 + 1.6258748099336"-1*F4 + F5/20;
B4:= - 2.8778860774335"-3*F2 - 1.3788780589233"-2*F4;
B5:= + (F2 + F4)/11.25       + 3.5555555555556"-1*F3;
B6:= + 2.9095718698132"-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"
```

AUTHOR: M. BAKKER.

INSTITUTE: MATHEMATICAL CENTRE, AMSTERDAM.

RECEIVED: 791231.

BRIEF DESCRIPTION:

THE PROCEDURE NONLIN FEMLAGSKEW SOLVES A NONLINEAR TWO POINT BOUNDARY VALUE PROBLEM WITH SPHERICAL COORDINATES.
IT SOLVES THE DIFFERENTIAL EQUATION

(X**NC*Y')'/X**NC = F(X, Y, Y'), 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,
BOUNDARY VALUE PROBLEMS,
RITZ-GALERKIN METHOD,
SPHERICAL COORDINATES,
GLOBAL METHODS.

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 27, 1976.

[3] BABUSKA, I.,
NUMERICAL STABILITY IN PROBLEMS OF LINEAR ALGEBRA,
S.I.A.M. J. NUM. ANAL., VOL.9, P. 53-77 (1972).

[4] BAKKER, M.,
GALERKIN METHODS IN SPHERICAL REGIONS, TO APPEAR.

SUBSECTION: NONLIN FEM LAG SKEW.

CALLING SEQUENCE:

THE HEADING OF THE PROCEDURE READS:

```
"PROCEDURE" NONLIN FEM LAG SKEW(X, Y, N, F, FY, FZ, NC, E);
"INTEGER" N, NC;
"REAL" "PROCEDURE" F, FY, FZ;
"ARRAY" X, Y, E;
"CODE" 33314;
```

THE MEANING OF THE FORMAL PARAMETERS IS:

N: <ARITHMETIC EXPRESSION>;
THE UPPER BOUND OF THE ARRAYS X AND Y; N > 1;

NC: <EXPRESSION>;
IF NC = 0, CARTESIAN COORDINATES ARE USED;
IF NC = 1, POLAR COORDINATES ARE USED;
IF NC = 2, SPHERICAL COORDINATES ARE USED;

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];
ENTRY: Y[I] (I = 0, 1, ... , N) IS AN INITIAL APPROXIMATE
SOLUTION AT X[I] OF THE DIFFERENTIAL EQUATION

(3) $(Y'*X**NC)'/X**NC = F(X,Y,Y')$, $A < X < B$,

WITH BOUNDARY CONDITIONS

(4) $E[1]*Y(A) + E[2]*Y'(A) = E[3]$,
$E[4]*Y(B) + E[5]*Y'(B) = E[6]$;

EXIT: Y[I] (I = 0, 1, ... , N) IS THE GALERKIN
SOLUTION AT X[I] OF THE (3)-(4);

F: <PROCEDURE IDENTIFIER>;
THE HEADING OF F READS:
"REAL" "PROCEDURE" F(X,Y,Z); "VALUE" X,Y,Z; "REAL" X,Y,Z;
F(X,Y,Z) IS THE RIGHT HAND SIDE OF (3);

```
FY:  <PROCEDURE IDENTIFIER>;
     THE HEADING OF FY READS:
     "REAL" "PROCEDURE" FY(X,Y,Z); "VALUE" X,Y,Z; "REAL" X,Y,Z;
     FY(X,Y,Z) IS THE DERIVATIVE OF F WITH RESPECT TO Y;

FZ:  <PROCEDURE IDENTIFIER>;
     THE HEADING OF FZ READS:
     "REAL" "PROCEDURE" FZ(X,Y,Z); "VALUE" X,Y,Z; "REAL" X,Y,Z;
     FZ(X,Y,Z) IS THE DERIVATIVE OF F WITH RESPECT TO Z;

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: DUPVEC CP 31030.

REQUIRED CENTRAL MEMORY:

FIVE AUXILIARY ARRAYS OF N REALS ARE USED.

RUNNING TIME:

LET IT BE THE NUMBER OF NEWTON ITERATIONS; THEN
IT*N EVALUATIONS OF F, FY, FZ ARE NEEDED;

DATA AND RESULTS:

THE FUNCTIONS F, FY AND FZ ARE REQUIRED TO BE SUFFICIENTLY SMOOTH IN THEIR VARIABLES ON THE INTERIOR OF EVERY SEGMENT <X[I],X[I+1]> (I = 0, ..., N - 1);

METHOD AND PERFORMANCE:

LET Y[0](X) BE SOME INITIAL APPROXIMATION OF Y(X); THEN
THE NONLINEAR PROBLEM IS SOLVED BY SUCCESIVELY SOLVING

$$-(D[K]'*X**NC)'/X**NC$$

$$+FY(X,Y[K](X),Y[K]'(X))*D[K](X)$$

$$+FZ(X,Y[K](X),Y[K]'(X))*D[K]'(X)$$

$$= (Y[K]'*X**NC)'/X**NC$$

$$-F(X,Y[K],Y[K]'(X)),\ X[0] < X < X[N],$$

$$E[1]*D[K](X[0]) + E[2]*D[K]'(X[0]) = 0;$$

$$E[4]*D[K](X[N]) + E[5]*D[K]'(X[N]) = 0;$$

WITH GALERKIN'S METHOD (SEE PREVIOUS SECTION) AND PUTTING

$$Y[K+1](X) = Y[K](X) + D[K](X),\ K = 0,1,...$$

THIS IS THE SO-CALLED NEWTON-KANTOROWITCH METHOD;

EXAMPLE OF USE:

WE SOLVE THE BOUNDARY VALUE PROBLEM

$$(Y'*X**2)'/X**2 = EXP(Y)+EXP(Y')-EXP(1-X**2)-EXP(2*X)-6;$$
$$0 < X < 1,\ Y'(0) = Y(1) = 0;$$

FOR THE BOUNDARY CONDITIONS THIS MEANS THAT

$$E[2] = E[4] = 1;\ E[1] = E[3] = E[5] = E[6] = 0;$$

THE ANALYTIC SOLUTION IS Y(X) = 1 - X**2; WE APPROXIMATE
THE SOLUTION ON A UNIFORM GRID, I.E. X[I] = I/N,
I = 0, ..., N; WE CHOOSE N=10,20 AND COMPUTE THE MAXIMUM ERROR;
THE PROGRAM READS AS FOLLOWS:

```
"BEGIN" "INTEGER" N, NC;
"FOR" NC:= 0,1,2 "DO"
"FOR" N:= 25, 50 "DO"
"BEGIN" "INTEGER" I;"ARRAY" X, Y[0:N], E[1:6]; "REAL" RHO, D;

    "REAL" "PROCEDURE" F(X,Y,Z); "VALUE" X,Y,Z; "REAL" X,Y,Z;
    F:= EXP(Y)+EXP(Z)-EXP(1-X**2)-EXP(-2*X)-2-2*NC;

    "REAL" "PROCEDURE" FY(X,Y,Z); "VALUE" X,Y,Z; "REAL" X,Y,Z;
    FY:= EXP(Y);

    "REAL" "PROCEDURE" FZ(X,Y,Z); "VALUE" X,Y,Z; "REAL" X,Y,Z;
    FZ:= EXP(Z);

    "PROCEDURE" NONLIN FEM LAG SKEW(X,Y,N,F,FY,FZ,NC,E);
    "CODE" 33314;
    E[2]:= E[4]:= 1; E[1]:= E[3]:= E[5]:= E[6]:= 0;
    "FOR" I:= 0 "STEP" 1 "UNTIL" N "DO"
    "BEGIN" X[I]:= I/N; Y[I]:= 0 "END";
    OUTPUT(61,"("//,4B"("N = ")"ZD,4B"("NC = ")"ZD")",N,NC);
    NONLIN FEM LAG SKEW(X, Y, N, F, FY, FZ, NC, E);
    RHO:= 0;
    "FOR" I:= 0 "STEP" 1 "UNTIL" N "DO"
    "BEGIN" D:= ABS(Y[I] - 1 + X[I]**2);
        "IF" RHO < D "THEN" RHO:= D
    "END";
    OUTPUT(61,"("24B"("MAX.ERROR= ")",D.DD"+ZD")",RHO)
"END"
"END"
```

RESULTS:

N = 25	NC = 0	MAX.ERROR= 2.47" -4
N = 50	NC = 0	MAX.ERROR= 6.19" -5
N = 25	NC = 1	MAX.ERROR= 1.41" -3
N = 50	NC = 1	MAX.ERROR= 3.99" -4
N = 25	NC = 2	MAX.ERROR= 2.44" -3
N = 50	NC = 2	MAX.ERROR= 7.02" -4

ONE OBSERVES THAT THE MAXIMUM ERROR DECREASES BY ABOUT 0.25 WHEN THE MESH SIZE IS HALVED.

SOURCE TEXT(S):

```
"CODE" 33314;
 "PROCEDURE" NONLIN FEM LAG SKEW(X, Y, N, F, FY, FZ, NC, E);
 "INTEGER" N, NC;
 "REAL" "PROCEDURE" F, FY, FZ;
 "ARRAY" X, Y, E;
 "BEGIN" "INTEGER" L, L1, IT;
   "REAL" XL1, XL, H, A12, A21, B1, B2, TAU1, TAU2, CH, TL, G, YL, PP,
     PLM, PRM, PL1, PL3, PL1PL2, PL1PL3, PL2PL2, PL2PL3,
     PR1PR2, PR1PR3, PR2PR3, PL1QL2, PL1QL3, PL2QL1, PL2QL2, PL2QL3,
     PL3QL1, PL3QL2, PR1QR2, PR1QR3, PR2QR1, PR2QR2, PR2QR3, PR3QR1,
     PR3QR2, H2RM, ZL1, ZL, E1, E2, E3, E4, E5, E6, EPS, RHO;
   "ARRAY" T, SUPER, SUB, CHI, GI[0:N-1], Z[0:N];

   "PROCEDURE" DUPVEC(L,U,S,A,B); "CODE" 31030;

   "PROCEDURE" ELEMENT MAT VEC EVALUATION 1;
   "BEGIN" "REAL"  XM,VL,VR,WL,WR,PR,QM,RM,FM,XL12,XL1XL,XL2,ZM,ZACCM;
     "IF" NC = 0 "THEN" VL:= VR:= 0.5 "ELSE" "IF" NC = 1 "THEN"
     "BEGIN" VL:= (XL1*2 + XL)/6; VR:= (XL1 + XL*2)/6 "END" "ELSE"
     "BEGIN" XL12:= XL1*XL1/12; XL1XL:=XL1*XL/6; XL2:=XL*XL/12;
       VL:= 3*XL12 + XL1XL + XL2;
       VR:= 3*XL2 + XL1XL + XL12
     "END";
     WL:= H*VL; WR:=H*VR; PR:= VR/(VL +VR);
     XM:= XL1 + H*PR; ZM:= PR*ZL + (1 - PR)*ZL1;
     ZACCM:= (ZL - ZL1)/H ; QM:= FZ(XM,ZM,ZACCM);
     RM:= FY(XM, ZM, ZACCM); FM:= F(XM,ZM,ZACCM);
     TAU1:= WL*RM; TAU2:=WR*RM;
     B1:= WL*FM - ZACCM*(VL +VR); B2:= WR*FM + ZACCM*(VL + VR);
     A12:= - (VL + VR)/H + VL*QM + (1 - PR)*PR*RM*(WL + WR);
     A21:= - (VL + VR)/H - VR*QM + (1 - PR)*PR*RM*(WL + WR);
   "END" ELEM. M.V. EV.;
```

"COMMENT"

```
"PROCEDURE" BOUNDARY CONDITIONS;
"IF" L=1 "AND" E2 = 0 "THEN"
"BEGIN" TAU1:= 1; B1:= A12:= 0 "END"
"ELSE" "IF" L=1 "AND" E2 ^= 0 "THEN"
"BEGIN" TAU1:= TAU1 - E1/E2
"END" "ELSE" "IF" L=N "AND" E5 = 0 "THEN"
"BEGIN" TAU2:= 1; B2:= A21:= 0
"END" "ELSE" "IF" L=N "AND" E5 ^= 0 "THEN"
"BEGIN" TAU2:= TAU2 + E4/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;
                                                    "COMMENT"
```

```
;
   DUPVEC(0,N,0,Z,Y);
     E1:= E[1]; E2:= E[2]; E3:= E[3]; E4:= E[4]; E5:= E[5]; E6:= E[6];
   "FOR" IT:= 1, IT + 1 "WHILE" EPS > RHO "DO"
   "BEGIN" L:= 0;XL:= X[0]; ZL:= Z[0];
     "FOR" L:= L + 1 "WHILE" L <= N "DO"
     "BEGIN" XL1:= XL; L1:= L - 1; XL:= X[L]; H:= XL - XL1;
        ZL1:= ZL; ZL:= Z[L];
        ELEMENT MAT VEC EVALUATION 1;
        "IF" L=1 "OR" L=N "THEN" BOUNDARY CONDITIONS;
        FORWARD BABUSKA
     "END";
     BACKWARD BABUSKA;
     EPS:= 0; RHO:= 1;
     "FOR" L:= 0 "STEP" 1 "UNTIL" N "DO"
     "BEGIN" RHO:= RHO + ABS(Z[L]);
       EPS:= EPS + ABS(Y[L]); Z[L]:= Z[L] - Y[L]
     "END";
     RHO:= "-14*RHO
   "END";
     DUPVEC(0,N,0,Y,Z)
"END" NONLIN FEM LAG SKEW;
       "EOP"
```

AUTHORS: T.M.T.COOLEN AND R.PLOEGER.

INSTITUTE: MATHEMATICAL CENTRE, AMSTERDAM.

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;
"CODE" 33170;

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 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 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:

APPROXIMATELY 3*(UJ - LJ + 1) * (UL - LL + 1) WORDS.

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, IF 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 EIGENFUNCTION 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 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);
  "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; "CODE" 33171;

    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:

APPROXIMATELY 3*(UJ - LJ + 1) * (UL - LL + 1) WORDS.

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, 1976, 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"+Z04B"("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"+Z04B"("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,YO,C,D,ALFA,OMEGA,OMEGAO,
    EIGMAX,EIGEUCL,EUCLRES,MAXRES,RCMAX,RCEUCL,MAXRESO,EUCLRESO;
    "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/EUCLRESO)/K;
      RCMAX:= -LN(MAXRES/MAXRESO)/K;
      RATECONV:= .5 * (RCEUCL + RCMAX)
    "END" ITERATION;                                    "COMMENT"
```

```
    "COMMENT" THE CONSTANTS FOR STARTING CALPAR ARE CALCULATED;
    ALFA:= 2; OMEGA:= 4/(B + A); YO:= (B + A)/(B - A);
    X:= .5 * (B + A); Y:= (B - A) * (B - A)/16; Z:= 4 * YO * YO;
    "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);
    OMEGAO:= 2/(B+A);
    "BEGIN" "REAL" AUXRESO;
      MAXRESO:= EUCLRESO:= 0;
      "FOR" J:= LJ "STEP" 1 "UNTIL" UJ "DO"
      "FOR" L:= LL "STEP" 1 "UNTIL" UL "DO"
      "BEGIN" AUXRESO:= RES[J,L];
        V[J,L]:= U[J,L] - OMEGAO * AUXRESO;
        AUXRESO:= ABS(AUXRESO);
        MAXRESO:= "IF" MAXRESO < AUXRESO "THEN" AUXRESO "ELSE" MAXRESO;
        EUCLRESO:= EUCLRESO + AUXRESO * AUXRESO
      "END";
    EUCLRESO:= SQRT(EUCLRESO)
    "END";
    DISCR[1]:= EUCLRESO; DISCR[2]:= MAXRESO;
    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, AMSTERDAM.

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;
"CODE" 34444;
```

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 EUCLIDEAN 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 VALUES 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+J*N+I]
                   (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.

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 VALUES 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,O,D,JDY,JDP,CY,DA,MO);
"CODE" 34444;
 "PROCEDURE" COMMUNICATION(P,F,M,N,NO,NP,PA,R,BP,J,I,O,W,NI);
"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"
    "END" 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,OUT;
   "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,AO,TOLUP,TOL,TOLDWN,TOLCONV,H,CH,CHNEW,
      ERROR,DFI,TOBSDIF;
      "BOOLEAN" EVALUATE,EVALUATED,DECOMPOSE,CONV;
      "ARRAY" A[0:5],DELTA,LAST DELTA,DF,YO[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));
            AO:= 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,1,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]; EVALUATE 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"
```

UITGAVEN IN DE SERIE MC SYLLABUS

Onderstaande uitgaven zijn verkrijgbaar bij het Mathematisch Centrum, 2e Boerhaavestraat 49 te Amsterdam-1005, tel. 020-947272.

MCS 1.1 F. GÖBEL & J. VAN DE LUNE, *Leergang Besliskunde, deel 1: Wiskundige basiskennis*, 1965. ISBN 90 6196 014 2.

MCS 1.2 J. HEMELRIJK & J. KRIENS, *Leergang Besliskunde, deel 2: Kansberekening*, 1965. ISBN 90 6196 015 0.

MCS 1.3 J. HEMELRIJK & J. KRIENS, *Leergang Besliskunde, deel 3: Statistiek*, 1966. ISBN 90 6196 016 9.

MCS 1.4 G. DE LEVE & W. MOLENAAR, *Leergang Besliskunde, deel 4: Markovketens en wachttijden*, 1966. ISBN 90 6196 017 7.

MCS 1.5 J. KRIENS & G. DE LEVE, *Leergang Besliskunde, deel 5: Inleiding tot de mathematische besliskunde*, 1966. ISBN 90 6196 018 5.

MCS 1.6a B. DORHOUT & J. KRIENS, *Leergang Besliskunde, deel 6a: Wiskundige programmering 1*, 1968. ISBN 90 6196 032 0.

MCS 1.6b B. DORHOUT, J. KRIENS & J.TH. VAN LIESHOUT, *Leergang Besliskunde, deel 6b: Wiskundige programmering 2*, 1977. ISBN 90 6196 150 5.

MCS 1.7a G. DE LEVE, *Leergang Besliskunde, deel 7a: Dynamische programmering 1*, 1968. ISBN 90 6196 033 9.

MCS 1.7b G. DE LEVE & H.C. TIJMS, *Leergang Besliskunde, deel 7b: Dynamische programmering 2*, 1970. ISBN 90 6196 055 X.

MCS 1.7c G. DE LEVE & H.C. TIJMS, *Leergang Besliskunde, deel 7c: Dynamische programmering 3*, 1971. ISBN 90 6196 066 5.

MCS 1.8 J. KRIENS, F. GÖBEL & W. MOLENAAR, *Leergang Besliskunde, deel 8: Minimaxmethode, netwerkplanning, simulatie*, 1968. ISBN 90 6196 034 7.

MCS 2.1 G.J.R. FÖRCH, P.J. VAN DER HOUWEN & R.P. VAN DE RIET, *Colloquium Stabiliteit van differentieschema's, deel 1*, 1967. ISBN 90 6196 023 1.

MCS 2.2 L. DEKKER, T.J. DEKKER, P.J. VAN DER HOUWEN & M.N. SPIJKER, *Colloquium Stabiliteit van differentieschema's, deel 2*, 1968. ISBN 90 6196 035 5.

MCS 3.1 H.A. LAUWERIER, *Randwaardeproblemen, deel 1*, 1967. ISBN 90 6196 024 X.

MCS 3.2 H.A. LAUWERIER, *Randwaardeproblemen, deel 2*, 1968. ISBN 90 6196 036 3.

MCS 3.3 H.A. LAUWERIER, *Randwaardeproblemen, deel 3*, 1968. ISBN 90 6196 043 6.

MCS 4 H.A. LAUWERIER, *Representaties van groepen*, 1968. ISBN 90 6196 037 1.

MCS 5 J.H. VAN LINT, J.J. SEIDEL & P.C. BAAYEN, *Colloquium Discrete wiskunde*, 1968.
ISBN 90 6196 044 4.

MCS 6 K.K. KOKSMA, *Cursus ALGOL 60*, 1969. ISBN 90 6196 045 2.

MCS 7.1 *Colloquium Moderne rekenmachines, deel 1*, 1969. ISBN 90 6196 046 0.

MCS 7.2 *Colloquium Moderne rekenmachines, deel 2*, 1969. ISBN 90 6196 047 9.

MCS 8 H. BAVINCK & J. GRASMAN, *Relaxatietrillingen*, 1969.
ISBN 90 6196 056 8.

MCS 9.1 T.M.T. COOLEN, G.J.R. FÖRCH, E.M. DE JAGER & H.G.J. PIJLS, *Elliptische differentiaalvergelijkingen, deel 1*, 1970.
ISBN 90 6196 048 7.

MCS 9.2 W.P. VAN DEN BRINK, T.M.T. COOLEN, B. DIJKHUIS, P.P.N. DE GROEN, P.J. VAN DER HOUWEN, E.M. DE JAGER, N.M. TEMME & R.J. DE VOGELAERE, *Colloquium Elliptische differentiaalvergelijkingen, deel 2*, 1970.
ISBN 90 6196 049 5.

MCS 10 J. FABIUS & W.R. VAN ZWET, *Grondbegrippen van de waarschijnlijkheidsrekening*, 1970. ISBN 90 6196 057 6.

MCS 11 H. BART, M.A. KAASHOEK, H.G.J. PIJLS, W.J. DE SCHIPPER & J. DE VRIES, *Colloquium Halfalgebra's en positieve operatoren*, 1971.
ISBN 90 6196 067 3.

MCS 12 T.J. DEKKER, *Numerieke algebra*, 1971. ISBN 90 6196 068 1.

MCS 13 F.E.J. KRUSEMAN ARETZ, *Programmeren voor rekenautomaten; De MC ALGOL 60 vertaler voor de EL X8*, 1971. ISBN 90 6196 069 X.

MCS 14 H. BAVINCK, W. GAUTSCHI & G.M. WILLEMS, *Colloquium Approximatietheorie*, 1971. ISBN 90 6196 070 3.

MCS 15.1 T.J. DEKKER, P.W. HEMKER & P.J. VAN DER HOUWEN, *Colloquium Stijve differentiaalvergelijkingen, deel 1*, 1972. ISBN 90 6196 078 9.

MCS 15.2 P.A. BEENTJES, K. DEKKER, H.C. HEMKER, S.P.N. VAN KAMPEN & G.M. WILLEMS, *Colloquium Stijve differentiaalvergelijkingen, deel 2*, 1973. ISBN 90 6196 079 7.

MCS 15.3 P.A. BEENTJES, K. DEKKER, P.W. HEMKER & M. VAN VELDHUIZEN, *Colloquium Stijve differentiaalvergelijkingen, deel 3*, 1975.
ISBN 90 6196 118 1.

MCS 16.1 L. GEURTS, *Cursus Programmeren, deel 1: De elementen van het programmeren*, 1973. ISBN 90 6196 080 0.

MCS 16.2 L. GEURTS, *Cursus Programmeren, deel 2: De programmeertaal ALGOL 60*, 1973. ISBN 90 6196 087 8.

MCS 17.1 P.S. STOBBE, *Lineaire algebra, deel 1*, 1974. ISBN 90 6196 090 8.

MCS 17.2 P.S. STOBBE, *Lineaire algebra, deel 2*, 1974. ISBN 90 6196 091 6.

MCS 17.3 N.M. TEMME, *Lineaire algebra , deel 3*, 1976. ISBN 90 6196 123 8.

MCS 18 F. VAN DER BLIJ, H. FREUDENTHAL, J.J. DE IONGH, J.J. SEIDEL & A. VAN WIJNGAARDEN, *Een kwart eeuw wiskunde 1946-1971, Syllabus van de Vakantiecursus 1971*, 1974. ISBN 90 6196 092 4.

MCS 19 A. HORDIJK, R. POTHARST & J.Th. RUNNENBURG, *Optimaal stoppen van Markovketens*, 1974. ISBN 90 6196 093 2.

MCS 20 T.M.T. COOLEN, P.W. HEMKER, P.J. VAN DER HOUWEN & E. SLAGT, *ALGOL 60 procedures voor begin- en randwaardeproblemen*, 1976. ISBN 90 6196 094 0.

MCS 21 J.W. DE BAKKER (red.), *Colloquium Programmacorrectheid*, 1975. ISBN 90 6196 103 3.

MCS 22 R. HELMERS, F.H. RUYMGAART, M.C.A. VAN ZUYLEN & J. OOSTERHOFF, *Asymptotische methoden in de toetsingstheorie; Toepassingen van naburigheid*, 1976. ISBN 90 6196 104 1.

MCS 23.1 J.W. DE ROEVER (red.), *Colloquium Onderwerpen uit de biomathematica, deel 1*, 1976. ISBN 90 6196 105 X.

MCS 23.2 J.W. DE ROEVER (red.), *Colloquium Onderwerpen uit de biomathematica, deel 2*, 1976. ISBN 90 6196 115 7.

MCS 24.1 P.J. VAN DER HOUWEN, *Numerieke integratie van differentiaalvergelijkingen, deel 1: Eenstapsmethoden*, 1974. ISBN 90 6196 106 8.

MCS 25 *Colloquium Structuur van programmeertalen*, 1976. ISBN 90 6196 116 5.

MCS 26.1 N.M. TEMME (ed.), *Nonlinear analysis, volume 1*, 1976. ISBN 90 6196 117 3.

MCS 26.2 N.M. TEMME (ed.), *Nonlinear analysis, volume 2*, 1976. ISBN 90 6196 121 1.

MCS 27 M. BAKKER, P.W. HEMKER, P.J. VAN DER HOUWEN, S.J. POLAK & M. VAN VELDHUIZEN, *Colloquium Discretiseringsmethoden*, 1976. ISBN 90 6196 124 6.

MCS 28 O. DIEKMANN, N.M. TEMME (EDS), *Nonlinear Diffusion Problems*, 1976. ISBN 90 6196 126 2.

MCS 29.1 J.C.P. BUS (red.), *Colloquium Numerieke programmatuur, deel 1A, deel 1B*, 1976. ISBN 90 6196 128 9.

MCS 29.2 H.J.J. TE RIELE (red.), *Colloquium Numerieke programmatuur, deel 2*, 1976. ISBN 90 6196 144 0

* MCS 30 P. GROENEBOOM, R. HELMERS, J. OOSTERHOFF & R. POTHARST, *Efficiency begrippen in de statistiek*, . ISBN 90 6196 149 1.

MCS 31 J.H. VAN LINT (red.), *Inleiding in de coderingstheorie*, 1976. ISBN 90 6196 136 X.

MCS 32 L. GEURTS (red.), *Colloquium Bedrijfssystemen*, 1976. ISBN 90 6196 137 8.

MCS 33 P.J. VAN DER HOUWEN, *Differentieschema's voor de berekening van waterstanden in zeeën en rivieren*, 1977. ISBN 90 6196 138 6.

MCS 34 J. HEMELRIJK, *Oriënterende cursus mathematische statistiek*, ISBN 90 6196 139 4.

MCS 35 P.J.W. TEN HAGEN (red.), *Colloquium Computer Graphics*, 1977. ISBN 90 6196 142 4.

MCS 36 J.M. AARTS, J. DE VRIES, *Colloquium Topologische Dynamische Systemen*, 1977. ISBN 90 6196 143 2.

MCS 37 J.C. van Vliet (red.), *Colloquium Capita Datastructuren*, 1978. ISBN 90 6196 159 9.

MCS 38.1 T.H. KOORNWINDER (ED.), *Representations of locally compact groups with applications*, 1979. ISBN 90 6196 161 0.

MCS 38.2 T.H. KOORNWINDER (ED.), *Representations of locally compact groups with applications*, 1979. ISBN 90 6196 181 5.

MCS 39 O.J. VRIEZE & G.L. Wanrooij, *Colloquium Stochastische Spelen*, 1978. ISBN 90 6196 167 X.

MCS 40 J. VAN TIEL, *Convexe Analyse*, 1979. ISBN 90 6196 187 4.

MCS 41 H.J.J. TE RIELE (ED.), *Colloquium Numerical Treatment of Integral Equations*, 1979. ISBN 90 6196 189 0.

MCS 42 J.C. VAN VLIET (RED.), *Colloquium Capita Implementatie van Programmeertalen*, 1980. ISBN 90 6196 191 2.

MCS 43 A.M. COHEN & H.A. WILBRINK, *Eindige groepen (Een inleidende cursus)*, 1980. ISBN 90 6196 203 X

MCS 44 J.G. VERWER (ED.), *Numerical solution of partial differential equations*, 1980. ISBN 90 6196 205 6.

MCS 45 P. KLINT (red.), *Colloquium hogere programmeertalen en computerarchitectuur*, 1980. ISBN 90 6196 206 4.

MCS 46.1 P.M.G. APERS (RED.), *Colloquium Databankorganisatie*, 1981. ISBN 90 6196 212 9.

MCS 47.1 P.W. HEMKER (ED.), *NUMAL numerical procedures in ALGOL 60*, Part I: *General information and indices*, 1981. ISBN 90 6196 217 X.

MCS 47.2 P.W. HEMKER (ED.), *NUMAL numerical procedures in ALGOL 60*, Part II: *Elementary procedures, algebraic evaluation*, 1981, ISBN 90 6196 217 X.

MCS 47.3 P.W. HEMKER (ED.), *NUMAL numerical procedures in ALGOL 60*, Part III: *Linear algebra, part I*, 1981. ISBN 90 6196 217 X.

MCS 47.4 P.W. HEMKER (ED.), *NUMAL numerical procedures in ALGOL 60*, Part IV: *Linear algebra, part II*, 1981. ISBN 90 6196 217 X.

MCS 47.5 P.W. HEMKER (ED.), *NUMAL numerical procedures in ALGOL 60*, Part V: *Analytical evaluations, analytical problems, part I*, 1981. ISBN 90 6196 217 X.

MCS 47.6 P.W. HEMKER (ED.), *NUMAL numerical procedures in ALGOL 60*, Part VI: *Analytical problems, part II*, 1981. ISBN 90 6196 217 X.

MCS 47.7 P.W. HEMKER (ED.), *NUMAL numerical procedures in ALGOL 60*, Part VII: *Special functions and constants, interpolation and approximation*, 1981. ISBN 90 6196 217 X.

De met een * gemerkte uitgaven moeten nog verschijnen.