

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 47.5

NUMAL
NUMERICAL PROCEDURES IN ALGOL 60

VOLUME 4, ANALYTICAL EVALUATIONS
VOLUME 5A, ANALYTICAL PROBLEMS, PART 1

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
VOLUME 4, VOLUME 5A.				
4. ANALYTIC EVALUATIONS				
1. EVAL. OF AN INFINITE SERIES	EULER	32010	JUL/74	131
	SUMPOSSERIES	32020	JUL/74	131
2. QUADRATURE				
1. ONE-DIMENSIONAL QUADRATURE	QADRAT	32070	JUL/74	133
	INTEGRAL	32051	JUL/74	135
2. MULTIDIMENSIONAL QUADRATURE				
3. GAUSSIAN QUADRATURE	TRICUB	32075	OCT/75	257
1. GENERAL WEIGHTS				
	RECCOF	31254	NOV/78	313
	GSSWTS	31253	NOV/78	313
	GSSWTSSYM	31252	NOV/78	313
2. SPECIAL WEIGHTS				
	GSSJACWGHTS	31425	NOV/76	291
	GSSLAGWGHTS	31427	NOV/76	291
3. NUMERICAL DIFFERENTIATION				
1. FUNCTIONS OF ONE VARIABLE				
2. FUNCTIONS OF MORE VARIABLES				
1. CALC. WITH DIFFERENCE FORMULAS	JACOBNNF	34437	OCT/74	213
	JACOBMF	34438	OCT/74	213
	JACOBBNDF	34439	OCT/74	213
5. ANALYTICAL PROBLEMS				
1. ANALYTICAL EQUATIONS				
1. NON-LINEAR EQUATIONS				
1.A SINGLE EQUATION				
1. NO DERIVATIVE AVAILABLE	ZEROIN	34150	OCT/75	215
	ZEROINRAT	34436	OCT/75	215
2. DERIVATIVE AVAILABLE	ZEROINDER	34453	OCT/75	233
2.A SYSTEM OF EQUATIONS				
1. AUXILIARY PROCEDURES				
2. JACOBIAN MATRIX NOT AVAILABLE	QUANEWBND	34430	OCT/74	217
	QUANEWBND1	34431	OCT/74	217
3. JACOBIAN MATRIX AVAILABLE				
3. POLYNOMIAL EQUATIONS				
SEE ALSO SECTION 3.6				
2. UNCONSTRAINED OPTIMIZATION				
1. FUNCTIONS OF ONE VARIABLE				
1. DERIVATIVE NOT AVAILABLE	MININ	34433	DEC/78	235
2. DERIVATIVE AVAILABLE	MININDER	34435	OCT/75	237
2. FUNCTIONS OF MORE VARIABLES				
1. AUXILIARY PROCEDURES	LINEMIN	34210	DEC/75	139
	RNK1UPD	34211	DEC/75	139
	DAVUPD	34212	DEC/75	139
	FLEUPD	34213	DEC/75	139
1. 2. 2. 2. NO DERIVATIVES AVAILABLE				

INDEX	PROCEDURE	CODE	MNT/YR	RECORD NUMBER
5. 1. 2. 2. 2.	PRAXIS	34432	OCT/75	239
3. GRADIENT AVAILABLE	RNK1MIN	34214	DEC/75	19
4. GRADIENT & JACOBIAN AVAILABLE	FLEMIN	34215	DEC/75	19
3. OVERDETERMINED NONLINEAR SYST.				
1. LEAST SQUARES SOLUTIONS				
SEE ALSO SECTION 7.				
1. AUXILIARY PROCEDURES				
2. JACOBIAN MATRIX NOT AVAILABLE				
SEE ALSO SECTION 5.1.2.2.2.				
3. JACOBIAN MATRIX AVAILABLE				
2. FUNCTIONAL EQUATIONS	MARQUARDT	34440	DEC/75	219
1. DIFFERENTIAL EQUATIONS	GSSNEWTON	34441	DEC/75	219
1. INITIAL VALUE PROBLEMS				
1. FIRST ORDER ORDINARY D.E.				
1. NO DERIVATIVES RHS AVAILABLE				
	RK1	33010	AUG/74	141
	RKE	33033	DEC/75	143
	RK4A	33016	AUG/74	145
	RK4NA	33017	AUG/74	147
	RK5NA	33018	AUG/74	149
	MULTISTEP	33080	AUG/74	151
	DIFFSYE	33180	AUG/74	153
	ARK	33061	DEC/75	155
	EFRK	33070	AUG/74	157

AUTHOR : J.W. DANIEL.
REVISOR : J. KOK.
INSTITUTE : MATHEMATICAL CENTRE.
RECEIVED : 730528 (EULER).
730917 (SUMPOSSERIES).

BRIEF DESCRIPTION :

THIS SECTION CONTAINS TWO PROCEDURES FOR THE SUMMATION OF CONVERGENT INFINITE SERIES:

EULER PERFORMS THE SUMMATION OF AN ALTERNATING SERIES.

SUMPOSSERIES PERFORMS THE SUMMATION OF A CONVERGENT SERIES WITH POSITIVE MONOTONICALLY DECREASING TERMS USING THE VAN WIJNGAARDEN TRANSFORMATION OF THE SERIES TO AN ALTERNATING SERIES.

KEYWORDS :

SUMMATION,
SERIES,
VAN WIJNGAARDEN TRANSFORMATION.

SUBSECTION : EULER.

CALLING SEQUENCE :

THE HEADING OF THE PROCEDURE IS :
"REAL" "PROCEDURE" EULER(AI, I, EPS, TIM);
"VALUE" EPS, TIM; "INTEGER" I, TIM; "REAL" AI, EPS;

EULER : DELIVERS THE COMPUTED SUM OF THE INFINITE SERIES
A[I], I = 0, 1, ...

THE MEANING OF THE FORMAL PARAMETERS IS:
AI :

<ARITHMETIC EXPRESSION>;

THE SUMMAND,

THIS EXPRESSION WILL BE DEPENDENT ON THE JENSEN
PARAMETER I;

AI IS THE I-TH TERM OF THE SERIES (I >= 0).

I :

<VARIABLE>;

JENSEN PARAMETER.

EPS, TIM :

<ARITHMETIC EXPRESSION>;

THE SUMMATION IS CONTINUED UNTIL TIM SUCCESSIVE TERMS

OF THE TRANSFORMED SERIES ARE IN ABSOLUTE VALUE LESS

THAN EPS.

PROCEDURES USED : NONE.

REQUIRED CENTRAL MEMORY :

EXECUTION FIELD LENGTH : 25.

LANGUAGE : ALGOL 60.

METHOD AND PERFORMANCE :

EULER PERFORMS THE SUMMATION OF AN ALTERNATING SEQUENCE BY APPLYING EULER'S TRANSFORMATION. BY THIS TRANSFORMATION THE SEQUENCE OF TERMS IS REPLACED BY THE SEQUENCE OF MEANS OF TWO SUCCESSIVE TERMS. IF NECESSARY THE NEW SEQUENCE IS AGAIN TRANSFORMED BY EULER'S TRANSFORMATION. THE SUMMATION STOPS WHEN TEN SUCCESSIVE TERMS OF THE (ONCE OR SEVERAL TIMES TRANSFORMED) SEQUENCE ARE IN ABSOLUTE VALUE LESS THAN EPS.

REFERENCES :

P. NAUR, ED. : REVISED REPORT ON THE ALGORITHMIC LANGUAGE ALGOL 60. COPENHAGEN (1964).

EXAMPLE OF USE :

THE PROGRAM :

```
"BEGIN" "INTEGER" K;  
  "REAL" "PROCEDURE" EULER(A, B, C, D); "CODE" 32010;  
  OUTPUT(61, "(" + .8D + "2D")",  
    EULER((- 1) ** K / (K + 1) ** 2, K, "- 6, 100))  
"END"
```

DELIVERS :

+ .82246703 + 00.

SUBSECTION : SUMPOSSERIES.

CALLING SEQUENCE :

THE HEADING OF THE PROCEDURE IS :

```
"REAL" "PROCEDURE" SUMPOSSERIES(AI, I, MAXADDUP, MAXZERO, MAXRECURS,
                                MACHEXP, TIM);
"VALUE" MAXADDUP, MAXZERO, MAXRECURS, MACHEXP, TIM;
"REAL" AI, I, MAXZERO; "INTEGER" MAXADDUP, MAXRECURS, MACHEXP, TIM;
```

SUMPOSSERIES : DELIVERS THE COMPUTED SUM OF THE INFINITE SERIES
 $A[i]$, $i = 1, 2, \dots$

THE MEANING OF THE FORMAL PARAMETERS IS:

AI : <ARITHMETIC EXPRESSION>;
 THE SUMMAND,
 THIS EXPRESSION SHOULD BE DEPENDENT ON THE JENSEN
 PARAMETER I;
 AI IS THE I-TH TERM OF THE SERIES ($i \geq 1$).

I : <VARIABLE>;
 JENSEN PARAMETER.

MAXADDUP : <ARITHMETIC EXPRESSION>;
 UPPER LIMIT FOR THE NUMBER OF STRAIGHTFORWARD ADDITIONS

MAXZERO, TIM :
 <ARITHMETIC EXPRESSION>;
 TOLERANCES EPS AND TIM NEEDED FOR A CALL OF THE
 PROCEDURE EULER (THIS SECTION). MAXZERO IS ALSO USED
 AS A TOLERANCE FOR MAXADDUP STRAIGHTFORWARD ADDITIONS.

MAXRECURS : <ARITHMETIC EXPRESSION>;
 UPPER LIMIT FOR THE RECURSION DEPTH OF THE
 VAN WIJNGAARDEN TRANSFORMATIONS.

MACHEXP : <ARITHMETIC EXPRESSION>;
 IN ORDER TO AVOID OVERFLOW AND EVALUATION OF THOSE
 TERMS WHICH CAN BE NEGLECTED, MACHEXP HAS TO BE THE
 LARGEST ADMISSIBLE VALUE FOR WHICH TERMS WITH INDEX
 $k * (2 ** MACHEXP)$ CAN BE COMPUTED (k IS SMALL).
 OTHERWISE OVERFLOW MIGHT OCCUR IN COMPUTING A VALUE FOR
 THE JENSEN PARAMETER I, WHICH CAN BE AN UNUSUALLY
 HIGH POWER OF 2.

PROCEDURES USED : EULER = CP32010.

REQUIRED CENTRAL MEMORY :

EXECUTION FIELD LENGTH : ABOUT 1000 * RECURSION DEPTH.

LANGUAGE : ALGOL 60.

METHOD AND PERFORMANCE :

WHEN THE TERMS A_i WITH INDICES $MAXADDUP + 1, \dots, MAXADDUP + TIM$ ARE ALL LESS THAN $MAXZERO$, CONVERGENCE IS ASSUMED AND `SUMPOSSERIES` DELIVERS THE SUM OF THE SERIES BY STRAIGHTFORWARD ADDITION UNTIL TIM SERIAL TERMS ARE LESS THAN $MAXZERO$. OTHERWISE THE VAN WIJNGAARDEN TRANSFORMATION IS APPLIED, YIELDING AN ALTERNATING SERIES WHICH IS SUMMED UP WITH EULER'S METHOD. SINCE THE TERMS OF THIS ALTERNATING SERIES ARE THEMSELVES INFINITE SERIES WITH POSITIVE TERMS, THE HERE DESCRIBED PROCESS IS RECURSIVELY CALLED FOR THE SUMMATION OF EACH TERM THAT IS WANTED BY EULER'S METHOD. HOWEVER, ONLY STRAIGHTFORWARD ADDITION IS APPLIED IF THE ALLOWED RECURSION LEVEL (SPECIFIED BY `MAXRECURS`) HAS BEEN REACHED. IN THE RECURSION THE PROCESS ASKS FOR TERMS A_i WITH INDICES OF THE TYPE $J * (2 ** K)$, IN WHICH K CAN BE VERY LARGE. IN ORDER TO AVOID OVERFLOW AN UPPER BOUND FOR K MUST BE GIVEN IN `MACHEXP`. IF K EXCEEDS THIS BOUND THE CORRESPONDING TERM IS TAKEN TO BE ZERO.

REFERENCES :

- [1] DANIEL, J.W. :
SUMMATION OF A SERIES OF POSITIVE TERMS BY CONDENSATION TRANSFORMATIONS. MATH. OF COMP. V.23, P.91-96 (1969).
- [2] WIJNGAARDEN, A. VAN :
COURSE SCIENTIFIC COMPUTING B, PROCESS ANALYSIS (DUTCH) MATHEMATISCH CENTRUM CR-18 (1965).

EXAMPLE OF USE :

THE PROGRAM :

```
"BEGIN" "COMMENT" 730808, EXAMPLE OF THE USE OF SUMPOSSERIES;
"REAL" "PROCEDURE" SUMPOSSERIES(A, B, C, D, E, F, G); "CODE"
32020;

"INTEGER" I;
OUTPUT(61, "(" / , +.12D"+DD)",
SUMPOSSERIES(1 / I ** 2, I, 100, "- 7, 8, 1068, 10))
"END"
```

DELIVERS :

```
+ .164493406604" +01
NUMBER OF TERMS USED : 462,
RECURSION DEPTH : 1.
```

SOURCE TEXT(S) :

```

"CODE" 32010;
"REAL""PROCEDURE" EULER(AI, I, EPS, TIM);
"VALUE" EPS, TIM; "INTEGER" I, TIM; "REAL" AI, EPS;
"BEGIN""INTEGER" K, N, T; "REAL" MN, MP, DS, SUM; "ARRAY" MCO:15];
  N:= T:= 0; I:= 0; MCO:= AI; SUM:= MCO / 2;
NEXT TERM: I:= I + 1; MN:= AI;
  "FOR" K:= 0 "STEP" 1 "UNTIL" N "DO"
  "BEGIN" MP:= (MN + M[K]) / 2; M[K]:= MN; MN:= MP "END";
  "IF" ABS(MN) < ABS(M[N]) & N < 15 "THEN"
  "BEGIN" DS:= MN / 2; N:= N + 1; M[N]:= MN "END" "ELSE" DS:= MN;
  SUM:= SUM + DS; T:= "IF" ABS(DS) < EPS "THEN" T + 1 "ELSE" 0;
  "IF" T < TIM "THEN" "GO TO" NEXT TERM;
  EULER:= SUM
"END" EULER;
  "EOP"

"CODE" 32020;
"REAL" "PROCEDURE" SUMPOSSERIES(AI, I, MAXADDUP, MAXZERO, MAXRECURS,
  MACHEXP, TIM);
"VALUE" MAXADDUP, MAXZERO, MAXRECURS, MACHEXP, TIM;
"REAL" AI, I, MAXZERO; "INTEGER" MAXADDUP, MAXRECURS, MACHEXP, TIM;
"BEGIN" "INTEGER" RECURS, VL, VL2, VL4;
  "REAL" "PROCEDURE" EULER(AI, I, EPS, TIM); "CODE" 32010;

  "REAL" "PROCEDURE" SUMUP(AI, I); "REAL" AI, I;
  "BEGIN" "INTEGER" J; "REAL" SUM, NEXTTERM;
  I:= MAXADDUP + 1; J:= 1;
  CHECK ADD: "IF" AI <= MAXZERO "THEN"
  "BEGIN""IF" J < TIM "THEN"
  "BEGIN" J:= J + 1; I:= I + 1; "GO TO" CHECK ADD "END"
  "END""ELSE"
  "IF" RECURS ^= MAXRECURS "THEN""GO TO" TRANSFORMSERIES;
  SUM:= 0; I:= 0; J:= 0;
  ADD LOOP: I:= I + 1; NEXTTERM:= AI;
  J:= "IF" NEXTTERM <= MAXZERO "THEN" J + 1 "ELSE" 0;
  SUM:= SUM + NEXTTERM;
  "IF" J < TIM "THEN""GO TO" ADD LOOP;
  SUMUP:= SUM; "GO TO" GOTSUM;
  TRANSFORMSERIES:
  "BEGIN""BOOLEAN" JODD; "INTEGER" J2; "ARRAY" V[1:VL];

  "REAL""PROCEDURE" BJK(J, K); "VALUE" J, K; "REAL" K;
  "INTEGER" J;
  "BEGIN""REAL" COEFF;
  "IF" K > MACHEXP "THEN" BJK:= 0 "ELSE"
  "BEGIN" COEFF:= 2 ** (K - 1); I:= J * COEFF;
  BJK:= COEFF * AI
  "END"
"END" BJK

```

```

"REAL""PROCEDURE" VJ(J); "VALUE" J; "INTEGER" J;
"BEGIN""REAL" TEMP, K;
  "IF" JODD "THEN"
    "BEGIN" JODD:= "FALSE"; RECURS:= RECURS + 1;
    TEMP:= VJ:= SUMUP(BJK(J, K), K);
    RECURS:= RECURS - 1;
    "IF" J <= VL "THEN" V[J]:= TEMP "ELSE"
    "IF" J <= VL2 "THEN" V[J - VL]:= TEMP
  "END""ELSE"
  "BEGIN" JODD:= "TRUE"; "IF" J > VL4 "THEN"
    "BEGIN" RECURS:= RECURS + 1;
    VJ:= - SUMUP(BJK(J, K), K); RECURS:= RECURS - 1
  "END""ELSE"
  "BEGIN" J2:= J2 + 1; I:= J2;
    "IF" J > VL2 "THEN" VJ:= - (V[J2 - VL] - AI) / 2
    "ELSE"
    "BEGIN" TEMP:= V["IF" J <= VL "THEN" J "ELSE"
      J - VL]:= (V[J2] - AI) / 2; VJ:= - TEMP
    "END"
  "END"
"END"
"END" VJ;

J2:= 0;
JODD:= "TRUE"; SUMUP:= EULER(VJ(J + 1), J, MAXZERO, TIM)
"END" TRANSFORMSERIES;
GOTSUM:
"END" SUMUP;

RECURS:= 0; VL:= 1000; VL2:= 2 * VL; VL4:= 2 * VL2;
SUMPOSSERIES:= SUMUP(AI, I)
"END" SUMPOSSERIES;
"EQP"

```

SECTION 4.2.1 CONTAINS TWO ALTERNATIVE PROCEDURES FOR THE COMPUTATION OF A DEFINITE INTEGRAL.

- A. THE PROCEDURE QADRAT USES HIGH ORDER INTEGRATION RULES (UP TO 16-TH ORDER) AND IS APPROPRIATE FOR THE EVALUATION OVER A FINITE INTERVAL.
- B. THE PROCEDURE INTEGRAL USES A 5-TH ORDER METHOD AND CAN ALSO BE USED TO CALCULATE THE INTEGRAL OVER A NUMBER OF CONSECUTIVE INTERVALS. MOREOVER THE PROCEDURE CAN BE USED FOR THE COMPUTATION OF THE DEFINITE INTEGRAL OVER AN INFINITE INTERVAL.

FOR A COMPARISON OF A NUMBER OF PROCEDURES THAT EVALUATE DEFINITE INTEGRALS : SEE REF[2].

REFERENCES :

- [1] T.J.DEKKER AND C.J.ROOTHART.
INTRODUCTION TO NUMERICAL ANALYSIS. (DUTCH).
MATH. CENTRE REPORT CR 244/74, AMSTERDAM.
- [2] C.J.ROOTHART AND H. FIOLET.
QUADRATURE PROCEDURES.
MATH. CENTRE REPORT MR 137/72, AMSTERDAM.

AUTHORS: C.J.ROOTHART.

CONTRIBUTORS: P.W.HEMKER.

INSTITUTE: MATHEMATICAL CENTRE.

RECEIVED: 730530.

BRIEF DESCRIPTION:

QADRAT COMPUTES THE DEFINITE INTEGRAL OF A FUNCTION OF ONE VARIABLE OVER A FINITE INTERVAL.

KEYWORDS:

INTEGRATION,
QUADRATURE,
DEFINITE INTEGRAL.

CALLING SEQUENCE:

THE HEADING OF THE PROCEDURE READS:
"REAL" "PROCEDURE" QADRAT (X, A, B, FX, E);
"VALUE" A, B; "REAL" X, A, B, FX; "ARRAY" E;

QADRAT: DELIVERS THE COMPUTED VALUE OF THE DEFINITE INTEGRAL FROM
A TO B OF THE FUNCTION F(X);

THE MEANING OF THE FORMAL PARAMETERS IS:

X: <VARIABLE>;
INTEGRATION VARIABLE; X CAN BE USED AS A JENSEN-PARAMETER
DURING THE EVALUATIONS OF FX;
A,B: <ARITHMETIC EXPRESSION>;
(A,B) DENOTES THE INTERVAL OF INTEGRATION;
B < A IS ALLOWED;
FX: <ARITHMETIC EXPRESSION>;
FX DENOTES THE INTEGRAND F(X). THIS EXPRESSION WILL BE
DEPENDENT ON THE JENSEN-PARAMETER X.
E: <ARRAY IDENTIFIER>;
"ARRAY" E[1:3];
ENTRY: E[1]: THE RELATIVE ACCURACY REQUIRED;
E[2]: THE ABSOLUTE ACCURACY REQUIRED;
EXIT: E[3]: THE NUMBER OF ELEMENTARY INTEGRATIONS WITH
 $H < ABS(B-A) * E[1]$.

PROCEDURES USED: NONE.

REQUIRED CENTRAL MEMORY:

EXECUTION FIELD LENGTH: CIRCA $16 + 9 * \text{RECURSION DEPTH}$.

RUNNING TIME: DEPENDS STRONGLY ON THE DEFINITE INTEGRAL TO COMPUTE.

LANGUAGE: ALGOL 60.

METHOD AND PERFORMANCE: SEE REF[1].

REFERENCES:

- [1]. C. J. ROOTHART AND H. FIOLET.
QUADRATURE PROCEDURES.
MATH. CENTRE, AMSTERDAM. REPORT MR 137/72.

EXAMPLE OF USE:

```
"BEGIN"  
  "REAL" "PROCEDURE" QADRAT(X,A,B,FX,E); "CODE" 32070;  
  "ARRAY" E[1:3]; "REAL" T,Q;  
  
  E[1]:= E[2]:= -. "9;  
  Q:= QADRAT(T,0,3.141592653589 ,SIN(T),E);  
  OUTPUT(61,"( / ,+.15D"+3D, 3B, 3ZD, /")",Q,E[3]);  
"END"
```

DELIVERS:

+ .200000000079740"+001 0

SOURCE TEXT(S):

```

"CODE" 32070;
"REAL" "PROCEDURE" QADRAT(X, A, B, FX, E);
"VALUE" A, B; "REAL" X, A, B, FX; "ARRAY" E;
"BEGIN" "REAL" F0, F2, F3, F5, F6, F7, F9,
      F14, V, W, HMIN, HMAX, RE, AE;

"REAL" "PROCEDURE" LINT(X0, XN, F0, F2, F3, F5, F6, F7, F9, F14);
"REAL" X0, XN, F0, F2, F3, F5, F6, F7, F9, F14;
"BEGIN" "REAL" H, XM, F1, F4, F8, F10, F11, F12, F13;
      XM:= (X0 + XN) / 2; H:= (XN - X0) / 32; X:= XM + 4 * H;
      F8:= FX; X:= XM - 4 * H; F11:= FX; X:= XM - 2 * H; F12:= FX;
      V:= 0.330580178199226 * F7 + 0.173485115707338 * (F6 + F8) +
      0.321105426559972*(F5 + F9) + 0.135007708341042 * (F3 + F11)
      + 0.165714514228223*(F2 + F12) + 0.393971460638127*- 1 * (F0
      + F14); X:= X0 + H; F1:= FX; X:= XM - H; F13:= FX;
      W:= 0.260652434656970 * F7 + 0.239053286684765 * (F6 + F8) +
      0.263062635477467*(F5 + F9) + 0.218681931383057 * (F3 + F11)
      + 0.275789764664284*- 1 * (F2 + F12) + 0.105575010053846* (F1
      + F13) + 0.157119426059518*- 1 * (F0 + F14);
      "IF" ABS(H) < HMIN "THEN" E[3]:= E[3] + 1;
      "IF" ABS(V - W) < ABS(W) * RE + AE "OR" ABS(H) < HMIN
      "THEN" LINT:= 4 * W "ELSE"
      "BEGIN" X:= X0 + 6 * H; F4:= FX; X:= XM - 6 * H; F10:= FX;
      V:= 0.245673430093324* F7 + 0.255786258286921* (F6 + F8) +
      0.228526063690406*(F5 + F9) + 0.500557131525460*- 1*(F4 +
      F10) + 0.177946487736780*(F3 + F11)+0.584014599347449*- 1
      * (F2 + F12) + 0.874830942871331*- 1 * (F1 + F13) +
      0.189642078548079*- 1 * (F0 + F14);
      LINT:= "IF" ABS(V - W) < ABS(V) * RE + AE "THEN" H * V
      "ELSE"
      LINT(X0, XM, F0, F1, F2, F3, F4, F5, F6, F7) - LINT(XN,
      XM, F14, F13, F12, F11, F10, F9, F8, F7)
      "END"
"END" LINT;

HMAX:= (B - A) / 16; "IF" HMAX=0 "THEN"
"BEGIN" QADRAT:= 0; "GOTO" RETURN "END";
RE:= E[1]; AE:= 2 * E[2] / ABS(B - A); E[3]:= 0;
HMIN:= ABS(B - A) * RE; X:= A; F0:= FX;
X:= A + HMAX; F2:= FX; X:= A + 2 * HMAX; F3:= FX;
X:= A + 4 * HMAX; F5:= FX; X:= A + 6 * HMAX; F6:= FX;
X:= A + 8 * HMAX; F7:= FX; X:= B - 4 * HMAX; F9:= FX; X:= B;
F14:= FX;
QADRAT:= LINT(A, B, F0, F2, F3, F5, F6, F7, F9, F14) * 16;
RETURN;
"END" QADRAT;
"EOB"

```


SECTION : 4.2.1.B

(JULY 1974)

PAGE 1

AUTHOR: H.N.GLORIE.

CONTRIBUTOR: H.FIDLET.

INSTITUTE: MATHEMATICAL CENTRE.

RECEIVED: 730606.

BRIEF DESCRIPTION:

INTEGRAL CALCULATES THE DEFINITE INTEGRAL OF A FUNCTION OF ONE VARIABLE, OVER A FINITE OR INFINITE INTERVAL OR OVER A NUMBER OF CONSECUTIVE INTERVALS.

KEYWORDS:

DEFINITE INTEGRAL,
INFINITE INTERVAL,
SIMPSON RULE,
RICHARDSON CORRECTION.

CALLING SEQUENCE:

THE HEADING OF THE PROCEDURE READS:
 "REAL" "PROCEDURE" INTEGRAL(X,A,B,FX,E,UA,UB);
 "VALUE" A,B;"REAL" X,A,B,FX;"ARRAY" E;"BOOLEAN" UA,UB;

INTEGRAL:

DELIVERS THE COMPUTED VALUE OF THE DEFINITE INTEGRAL OF THE FUNCTION FROM A TO B; AFTER SUCCESSIVE CALLS OF THE PROCEDURE, THE INTEGRAL OVER THE TOTAL INTERVAL IS DELIVERED, I.E. THE VALUE OF A IN THE LAST CALL WITH UA="TRUE" IS THE STARTING POINT OF THE INTERVAL.

THE MEANING OF THE FORMAL PARAMETERS IS:

X: <VARIABLE>;
 INTEGRATION VARIABLE; X CAN BE USED AS JENSEN-PARAMETER FOR FX.
 A,B: <ARITHMETIC EXPRESSION>;
 (A,B) DENOTES THE INTERVAL OF INTEGRATION; B<A IS ALLOWED;
 FX: <ARITHMETIC EXPRESSION>;
 THE INTEGRAND F(X);
 E: <ARRAY IDENTIFIER>;
 "ARRAY" E[1:6];
 ENTRY : E[1]:THE RELATIVE ACCURACY REQUIRED;
 E[2]:THE ABSOLUTE ACCURACY REQUIRED;
 EXIT: E[3]:THE NUMBER OF SKIPPED INTEGRATION STEPS;
 E[4]:THE VALUE OF THE INTEGRAL FROM A TO B;
 E[5]:"IF" UB "THEN" B "ELSE" 0;
 E[6]:"IF" UB "THEN" F(B) "ELSE" 0;
 UA: <BOOLEAN EXPRESSION>;
 DETERMINES THE STARTING POINT OF THE INTEGRATION;
 STARTING POINT:="IF" UA "THEN" A "ELSE" E[5];
 UB: <BOOLEAN EXPRESSION>;
 DETERMINES THE FINAL POINT OF THE INTEGRATION;
 FINAL POINT:="IF" UB "THEN" B "ELSE"
 "IF" B>A "THEN" +INFINITY "ELSE" -INFINITY;
 IN THE CASE UB="FALSE", THE VALUE OF B IS STILL RELEVANT
 (SEE METHOD AND PERFORMANCE).

PROCEDURES USED: NONE.

REQUIRED CENTRAL MEMORY:

EXECUTION FIELD LENGTH: CIRCA 16 + 5 * RECURSION LEVEL.

RUNNING TIME: DEPENDS STRONGLY ON THE INTEGRAL TO COMPUTE.

SOURCE TEXT(S):

```

"CODE" 32051;
"REAL" "PROCEDURE" INTEGRAL(X, A, B, FX, E, UA, UB);
"VALUE" A, B; "REAL" X, A, B, FX; "ARRAY" E; "BOOLEAN" UA, UB;
"BEGIN"
  "REAL" "PROCEDURE" TRANSF;
  "BEGIN" Z:= 1 / X; X:= Z + B1; TRANSF:= FX * Z * Z "END";
  "REAL" "PROCEDURE" QAD(FX); "REAL" FX;
  "BEGIN" "REAL" T, V, SUM, HMIN;
  "PROCEDURE" INT;
  "BEGIN" "REAL" X3, X4, F3, F4, H;
    X4:= X2; X2:= X1; F4:= F2; F2:= F1;
  ANEW: X:= X1:= (X0 + X2) * .5; F1:= FX;
    X:= X3:= (X2 + X4) * .5; F3:= FX; H:= X4 - X0;
    V:= (4 * (F1 + F3) + 2 * F2 + F0 + F4) * 15;
    T:= 6 * F2 - 4 * (F1 + F3) + F0 + F4;
    "IF" ABS(T) < ABS(V) * RE + AE "THEN"
      SUM:=SUM + (V - T) * H "ELSE"
        "IF" ABS(H) < HMIN "THEN" E[3]:= E[3] + 1
        "ELSE"
          "BEGIN" INT; X2:= X3; F2:= F3; "GOTO" ANEW "END";
          X0:= X4; F0:= F4
        "END" INT;
    HMIN:= ABS(X0 - X2) * RE; X:= X1:= (X0 + X2) * .5;
    F1:=FX;SUM:= 0; INT; QAD:= SUM / 180
  "END" QAD;
  "REAL" X0, X1, X2, F0, F1, F2, RE, AE, B1, Z;
  RE:= E[1]; "IF" UB "THEN" AE:= E[2] * 180 / ABS(B - A)
  "ELSE" AE:= E[2] * 90 / ABS(B - A); "IF" UA "THEN"
  "BEGIN" E[3]:= E[4]:= 0; X:= X0:= A; F0:= FX "END"
  "ELSE"
  "BEGIN" X:= X0:= A:= E[5]; F0:= E[6] "END";
  E[5]:= X:= X2:= B; E[6]:= F2:= FX; E[4]:= E[4] + QAD(FX);
  "IF" ^UB "THEN"
  "BEGIN" "IF" A < B "THEN"
    "BEGIN" B1:= B - 1; X0:= 1 "END"
    "ELSE"
      "BEGIN" B1:= B + 1; X0:= -1 "END";
      F0:= E[6]; E[5]:= X2:= 0; E[6]:= F2:= 0;
      AE:= E[2] * 90;
      E[4]:= E[4] - QAD(TRANSF)
    "END";
  INTEGRAL:= E[4]
"END" INTEGRAL;
"EOB"

```

AUTHOR : P.W. HEMKER.

CONTRIBUTOR : F.GROEN.

INSTITUTE : MATHEMATICAL CENTRE.

RECEIVED : 740620.

BRIEF DESCRIPTION :

TRICUB COMPUTES THE DEFINITE INTEGRAL OF A FUNCTION OF TWO VARIABLES OVER A TRIANGULAR DOMAIN.

KEYWORDS :

INTEGRATION,
QUADRATURE,
MOR EDIMENSIONAL QUADRATURE,
CUBATURE,
DEFINITE INTEGRAL.

CALLING SEQUENCE :

THE HEADING OF THE PROCEDURE READS :

```
"REAL" "PROCEDURE" TRICUB ( XI, YI, XJ, YJ, XK, YK, F, RE, AE );  
"VALUE" XI, YI, XJ, YJ, XK, YK, RE, AE;  
"REAL" XI, YI, XJ, YJ, XK, YK, RE, AE;  
"REAL" "PROCEDURE" F;  
"CODE" 32075;
```

TRICUB := THE COMPUTED VALUE OF THE DEFINITE INTEGRAL OF THE FUNCTION $F(x, y)$ OVER THE TRIANGULAR DOMAIN T WITH VERTICES (x_i, y_i) , (x_j, y_j) AND (x_k, y_k) .

THE MEANING OF THE FORMAL PARAMETERS IS :

XI, YI: <ARITHMETIC EXPRESSION>;
ENTRY : THE COORDINATES OF THE FIRST VERTEX OF THE
TRIANGULAR DOMAIN OF INTEGRATION;

XJ, YJ: <ARITHMETIC EXPRESSION>;
ENTRY : THE COORDINATES OF THE SECOND VERTEX OF THE
TRIANGULAR DOMAIN OF INTEGRATION;

XK, YK: <ARITHMETIC EXPRESSION>;
ENTRY : THE COORDINATES OF THE THIRD VERTEX OF THE
TRIANGULAR DOMAIN OF INTEGRATION;

REMARK: THE ALGORITHM IS SYMMETRICAL IN THE VERTICES; THIS
IMPLIES THAT THE RESULT OF THE PROCEDURE (ON ALL COUNTS) IS
INVARIANT FOR ANY PERMUTATION OF THE VERTICES.

F : <PROCEDURE IDENTIFIER>;
THE HEADING OF THIS PROCEDURE READS:
"REAL" "PROCEDURE" F (X, Y); "REAL" X, Y;
THIS PROCEDURE DEFINES THE INTEGRAND;

AE, RE: <ARITHMETIC EXPRESSION>;
ENTRY: THE REQUIRED ABSOLUTE AND RELATIVE ERROR
RESPECTIVELY. ONE SHOULD TAKE FOR "AE" AND "RE"
VALUES WHICH ARE GREATER THAN THE ABSOLUTE AND
RELATIVE ERROR IN THE COMPUTATION OF THE INTEGRAND F.

PROCEDURES USED : NONE.

REQUIRED CENTRAL MEMORY :

THE PROCESS IS PROGRAMMED RECURSIVELY. AT EACH RECURSION LEVEL 43
REAL NUMBERS ARE USED. HOWEVER, FOR ANY PROPERLY CHOSEN VALUES OF
RE AND AE THE RECURSION DEPTH WILL NOT EXCEED THE NUMBER OF BITS IN
A REAL'S MANTISSA.

RUNNING TIME : DEPENDS STRONGLY ON THE INTEGRAL TO COMPUTE.

METHOD AND PERFORMANCE :

A NESTED SEQUENCE OF CUBATURE RULES OF ORDER 2, 3, 4 AND 5
IS APPLIED. IF THE DIFFERENCE BETWEEN THE RESULT WITH THE
4-TH DEGREE RULE AND THE RESULT WITH THE 5-TH DEGREE RULE
IS TOO LARGE, THEN THE TRIANGLE IS DIVIDED INTO FOUR CONGRUENT
TRIANGLES. THIS PROCESS IS APPLIED RECURSIVELY IN ORDER TO
OBTAIN AN ADAPTIVE CUBATURE ALGORITHM.

REFERENCES :

- [1]. P. W. HEMKER,
A SEQUENCE OF NESTED CUBATURE RULES,
MATH. CENTRE, AMSTERDAM, REPORT NW 3/73.

EXAMPLE OF USE:

THE FOLLOWING PROGRAM EVALUATES THE INTEGRAL OF
 $F(X, Y) = \cos(X) * \cos(Y)$ OVER THE TRIANGLE T WITH
 VERTICES (0, 0), (0, PI/2) AND (PI/2, PI/2).
 ON EACH LINE ARE LISTED :

- A: THE REQUIRED RELATIVE AND ABSOLUTE PRECISION;
 B: THE COMPUTED VALUE OF THE INTEGRAL;
 C: THE NUMBER OF CALLS OF THE FUNCTION F,

```
"BEGIN"
"REAL" "PROCEDURE" TRICUB(A,B,C,D,E,F,G,H,I); "CODE" 32075;
"INTEGER" N,C,I,K; "REAL" PI,ACC,R,S;
"REAL" "PROCEDURE" E(X,Y); "REAL" X,Y;
"BEGIN" C:= C+1;
      "IF" C> 20000 "THEN" "GOTO" CC;
      E:= COS(X) * COS(Y);
"END" E;

PI:= 3.14159265359;
"FOR" ACC:= "-1","-2","-3","-4","-5","-6","-7","-8","-9","-10","-11" "DO"
"BEGIN" C:=0; OUTPUT(61,"(+.D"+ZD,2B,+.14D"+2ZD,2B,107D,/"")",
      ACC, TRICUB(0,0,0,PI/2,PI/2,PI/2,E,ACC,ACC),C);
"END";
CC: OUTPUT(61,"(*)");
"END"
```

RESULTS:

+0.1"	+0	+0.50063973801970"	+0	7
+0.1"	-1	+0.50063973801970"	+0	7
+0.1"	-2	+0.50063973801970"	+0	7
+0.1"	-3	+0.49999110261504"	+0	10
+0.1"	-4	+0.49999848959226"	+0	13
+0.1"	-5	+0.49999848959226"	+0	13
+0.1"	-6	+0.4999997378240"	+0	43
+0.1"	-7	+0.4999999209792"	+0	133
+0.1"	-8	+0.4999999893172"	+0	313
+0.1"	-9	+0.4999999985571"	+0	733
+0.1"	-10	+0.4999999998692"	+0	1723

SOURCE TEXT(S):

```

"CODE" 32075;
"REAL" "PROCEDURE" TRICUB(XI,YI,XJ,YJ,XK,YK,G,RE,AE);
"VALUE" XI,YI,XJ,YJ,XK,YK,RE,AE;
"REAL" XI,YI,XJ,YJ,XK,YK,RE,AE; "REAL" "PROCEDURE" G;
"BEGIN" "REAL" SURF,SURFMIN,XZ,YZ,GI,GJ,GK;

      "REAL" "PROCEDURE" INT(AX1,AY1,AF1,AX2,AY2,AF2,AX3,AY3,AF3,
        BX1,BY1,BF1,BX2,BY2,BF2,BX3,BY3,BF3,
        PX,PY,PF);
"VALUE" BX1,BY1,BF1,BX2,BY2,BF2,BX3,BY3,BF3,PX,PY,PF;
"REAL"  BX1,BY1,BF1,BX2,BY2,BF2,BX3,BY3,BF3,PX,PY,PF,
        AX1,AY1,AF1,AX2,AY2,AF2,AX3,AY3,AF3;
"BEGIN" "REAL" E,I3,I4,I5,A,B,C,SX1,SY1,SX2,SY2,SX3,SY3,
        CX1,CY1,CF1,CX2,CY2,CF2,CX3,CY3,CF3,
        DX1,DY1,DF1,DX2,DY2,DF2,DX3,DY3,DF3;

      A:= AF1 + AF2 + AF3; B:= BF1 + BF2 + BF3;
      I3:= 3 * A + 27 * PF + 8 * B;
      E:= ABS(I3) * RE + AE;

      "IF" SURF < SURFMIN "OR" ABS(5 * A + 45 * PF - I3) < E
      "THEN" INT:= I3 * SURF "ELSE"
      "BEGIN" CX1:= AX1 + PX; CY1:= AY1 + PY; CF1:= G(CX1,CY1);
              CX2:= AX2 + PX; CY2:= AY2 + PY; CF2:= G(CX2,CY2);
              CX3:= AX3 + PX; CY3:= AY3 + PY; CF3:= G(CX3,CY3);
              C:= CF1 + CF2 + CF3;
              I4:= A + 9 * PF + 4 * B + 12 * C;

      "IF" ABS(I3 - I4) < E "THEN" INT:= I4 * SURF "ELSE"
      "BEGIN" SX1:= .5 * BX1; SY1:= .5 * BY1;
              DX1:= AX1 + SX1; DY1:= AY1 + SY1; DF1:= G(DX1,DY1);
              SX2:= .5 * BX2; SY2:= .5 * BY2;
              DX2:= AX2 + SX2; DY2:= AY2 + SY2; DF2:= G(DX2,DY2);
              SX3:= .5 * BX3; SY3:= .5 * BY3;
              DX3:= AX3 + SX3; DY3:= AY3 + SY3; DF3:= G(DX3,DY3);

      I5:= (51 * A + 2187 * PF + 276 * B + 972 * C -
            768 * (DF1 + DF2 + DF3))/63;
"COMMENT"

```

```

"IF" ABS(I4 - I5) < E "THEN" INT:= I5 * SURF "ELSE".
"BEGIN" SURF:= .25 * SURF;

INT:=

INT(SX1,SY1,BF1,SX2,SY2,BF2,SX3,SY3,BF3,
DX1,DY1,DF1,DX2,DY2,DF2,DX3,DY3,DF3,
PX,PY,PF) +

INT(AX1,AY1,AF1,SX3,SY3,BF3,SX2,SY2,BF2,DX1,DY1,DF1,
AX1 + SX2,AY1 + SY2,G(AX1 + SX2,AY1 + SY2),
AX1 + SX3,AY1 + SY3,G(AX1 + SX3,AY1 + SY3),
.5 * CX1,.5 * CY1,CF1) +
INT(AX2,AY2,AF2,SX3,SY3,BF3,SX1,SY1,BF1,DX2,DY2,DF2,
AX2 + SX1,AY2 + SY1,G(AX2 + SX1,AY2 + SY1),
AX2 + SX3,AY2 + SY3,G(AX2 + SX3,AY2 + SY3),
.5 * CX2,.5 * CY2,CF2) +
INT(AX3,AY3,AF3,SX1,SY1,BF1,SX2,SY2,BF2,DX3,DY3,DF3,
AX3 + SX2,AY3 + SY2,G(AX3 + SX2,AY3 + SY2),
AX3 + SX1,AY3 + SY1,G(AX3 + SX1,AY3 + SY1),
.5 * CX3,.5 * CY3,CF3);

SURF:= 4 * SURF
"END"
"END"
"END"
"END" INT;

SURF:= 0.5 * ABS(XJ * YK - XK * YJ + XI * YJ -
XJ * YI + XK * YI - XI * YK);
SURFMIN:= SURF*RE; RE:= 30*RE; AE:= 30*AE/SURF;
XZ:= (XI + XJ + XK)/3; YZ:= (YI + YJ + YK)/3;
GI:= G(XI,YI); GJ:= G(XJ,YJ); GK:= G(XK,YK);
XI:= XI*.5; YI:= YI*.5; XJ:= XJ*.5;
YJ:= YJ*.5; XK:= XK*.5; YK:= YK*.5;

TRICUB:= INT(XI,YI,GI,XJ,YJ,GJ,XK,YK,GK,
XJ+XK,YJ+YK,G(XJ+XK,YJ+YK),
XK+XI,YK+YI,G(XK+XI,YK+YI),
XI+XJ,YI+YJ,G(XI+XJ,YI+YJ),
.5 * XZ,.5 * YZ,G(XZ,YZ))/60

"END" TRICUB;
"EQP"

```


AUTHOR: C.G. VAN DER LAAN.
CONTRIBUTORS: C.G. VAN DER LAAN AND M. VOORINTHOLT.
INSTITUTE: REKENCENTRUM RIJKSUNIVERSITEIT GRONINGEN.
RECEIVED: 780601.

BRIEF DESCRIPTION:

THIS SECTION CONTAINS THE PROCEDURES GSSWTS, GSSWTSSYM AND RECCOF. RECCOF CALCULATES FROM A GIVEN WEIGHT FUNCTION ON $[-1,1]$ THE RECURRENCE COEFFICIENTS OF THE CORRESPONDING ORTHOGONAL POLYNOMIALS; GSSWTS AND GSSWTSSYM CALCULATE FROM THE RECURRENCE COEFFICIENTS THE GAUSSIAN WEIGHTS OF THE CORRESPONDING WEIGHT FUNCTION.

KEYWORDS:

RECURRENCE COEFFICIENTS ORTHOGONAL POLYNOMIALS,
GAUSSIAN WEIGHTS,
GAUSSIAN QUADRATURE.

SUBSECTION: RECCOF.

CALLING SEQUENCE:

THE DECLARATION OF THE PROCEDURE IN THE CALLING PROGRAM READS:

```
"PROCEDURE" RECCOF(N,M,X,WX,B,C,L,SYM);  
"VALUE" N,M,SYM; "INTEGER" N,M; "BOOLEAN" SYM;  
"REAL" X,WX; "ARRAY" B,C,L;  
"CODE" 31254;
```

THE MEANING OF THE FORMAL PARAMETERS IS:

```
N: <ARITHMETIC EXPRESSION>;  
ENTRY: UPPER BOUND FOR THE INDICES OF THE ARRAYS B, C, L  
(N>=0);  
M: <ARITHMETIC EXPRESSION>;  
ENTRY: THE NUMBER OF POINTS USED IN THE GAUSS-CHEBYSHEV  
QUADRATURE RULE FOR CALCULATING THE APPROXIMATION  
OF THE INTEGRAL REPRESENTATIONS OF B[K],C[K]
```

```

SYM:      <BOOLEAN EXPRESSION>;
          ENTRY: "IF" SYM
           "THEN" WEIGHT FUNCTION ON [-1,1] IS EVEN
           "ELSE" WEIGHT FUNCTION ON [-1,1] IS NOT EVEN;
X,WX:    <ARITHMETIC EXPRESSION>;
          ENTRY: JENSEN VARIABLES WITH WX AN EXPRESSION OF X
           DENOTING THE WEIGHT FUNCTION ON [-1,1];
B,C,L:   <ARRAY IDENTIFIER>;
          "ARRAY" B,C,L[0:N];
EXIT:    THE APPROXIMATE RECURRENCE COEFFICIENTS FOR
           $P_{K+1}(X) = (X-B[K])*P_K(X) - C[K]*P_{K-1}(X),$ 
            $K=0,1,2,\dots,N,$ 
          AND THE APPROXIMATE SQUARE LENGTHS
            $X = +1$ 
           $L[K] = \int_{X=-1}^{X=+1} (W(X) * P_K(X) ** 2) DX$ 
            $X = -1$ 

```

PROCEDURES USED: ORTPOL = CP31044.

RUNNING TIME: PROPORTIONAL TO $M*N**2$.

METHOD AND PERFORMANCE:

THE RECURRENCE COEFFICIENTS ARE REPRESENTED BY

$$B[K] = \left(\int_{X=-1}^{X=+1} (W(X) * X * P_K(X) ** 2) DX \right) / L[K],$$

$$C[K] = L[K] / L[K-1],$$

WHERE $P_K(X)$ IS THE K-TH ORTHOGONAL POLYNOMIAL.

THE INTEGRALS ARE APPROXIMATED BY THE M-POINTS GAUSS-CHEBYSHEV RULE AS

$$\int_{X=-1}^{X=+1} (F(X))DX := \pi / M * \sum_{J=1}^M \sin(\theta_{A[J]}) * F(\cos(\theta_{A[J]}))$$

WITH $\theta_{A[J]} = (2*J-1) * \pi / (2*M)$ (SEE GAUTSCHI, 1968A).
THE VALUE OF M IS TO BE SUPPLIED BY THE USER.

REFERENCES:

GAUTSCHI, W. (1968A):
CONSTRUCTION OF GAUSS-CHRISTOFFEL FORMULAS.
MATH. COMP., 22, P.251-270.

GAUTSCHI, W. (1968B):
GAUSSIAN QUADRATURE FORMULAS.
COMM. ACM. CALGD 331.

EXAMPLE OF USE:

THE FOLLOWING PROGRAM DELIVERS AN APPROXIMATION FOR THE RECURSION COEFFICIENTS C[1] AND C[2], OF THE CHEBYSHEV POLYNOMIALS OF THE SECOND KIND;

```
"BEGIN"
  "PROCEDURE" RECCOF(N,M,X,WX,B,C,L,SYM);
  "VALUE" N,M,SYM; "INTEGER" N,M; "BOOLEAN" SYM;
  "REAL" X,WX; "ARRAY" B,C,L;
  "CODE" 31254;
  "REAL" X; "ARRAY" B,C,L[0:2];
  RECCOF(2,200,X,SQRT(1 - X**2),B,C,L,"TRUE");
  OUTPUT(61,"(M2/,2(3B,-ZD.3D)M)",C[1],C[2]);
"END";
```

RESULTS:

0.250 0.250

SUBSECTION: GSSWTS.

CALLING SEQUENCE:

THE DECLARATION OF THE PROCEDURE IN THE CALLING PROGRAM READS:

```
"PROCEDURE" GSSWTS(N,ZER,B,C,W);
"VALUE" N; "INTEGER" N; "ARRAY" ZER,B,C,W;
"CODE" 31253;
```

THE MEANING OF THE FORMAL PARAMETERS IS:

```
N:        <ARITHMETIC EXPRESSION>;
          ENTRY: THE NUMBER OF WEIGHTS TO BE COMPUTED; UPPER
                  BOUND FOR THE ARRAYS Z AND W (N>=1);
ZER:      <ARRAY IDENTIFIER>;
          "ARRAY" ZER[1:N];
          ENTRY: THE ZEROS OF THE N-TH DEGREE ORTHOGONAL POLYNOMIAL;
B,C:      <ARRAY IDENTIFIER>;
          "ARRAY" B[0:N-1], C[1:N-1];
          ENTRY: THE RECURRENCE COEFFICIENTS;
W:        <ARRAY IDENTIFIER>;
          "ARRAY" W[1:N];
          EXIT : THE GAUSSIAN WEIGHTS DIVIDED BY THE
                  INTEGRAL OVER THE WEIGHT FUNCTION.
```

PROCEDURES USED: ALLDRTPOL = CP 31045.

METHOD AND PERFORMANCE:

THE GAUSSIAN WEIGHTS OF AN N-POINTS RULE DIVIDED BY THE INTEGRAL OF THE WEIGHT FUNCTION MAY BE REPRESENTED AS

$$W[K] = 1 / (\dots ((P[N-1](Z)**2/C[N-1] + P[N-2](Z)**2)/C[N-2] + \dots \dots + P[1](Z)**2)/C[1] + 1) , K=1,2,\dots,N$$

WITH Z = K-TH ZERO OF P[N](X). (GAUTSCHI, 1970).

ALLZERORTPOL AND GSSWTS MAY BE USED TO GENERATE GAUSSIAN QUADRATURE RULES PROVIDED THE RECURRENCE COEFFICIENTS AND THE INTEGRAL OF THE WEIGHT FUNCTION ARE KNOWN. FOR EXAMPLE THE GAUSS-LAGUERRE QUADRATURE RULE APPLIED TO F MAY BE OBTAINED BY THE CALLS

```
"FOR" K:=1 "STEP" 1 "UNTIL" N-1 "DO"
"BEGIN"
  B[K]:=2*K+ALPHA+1;
  C[K]:=K*(K+ALPHA)
"END";
B[0]:=-ALPHA+1;
ALLZERORTPOL(N,ZER,B,C);
GSSWTS(N,ZER,B,C,W);
GAUSSRULE:=0;
"FOR" K:=1 "STEP" 1 "UNTIL" N "DO"
GAUSSRULE:=GAUSSRULE+W[K]*F(ZER[K]);
GAUSSRULE:=GAUSSRULE*GAMMA(ALPHA+1)
```

GAUSSRULE CONTAINS THE VALUE OF THE APPROXIMATING GAUSS QUADRATURE RULE AND ZER[1:N],W[1:N] CONTAIN THE GAUSSIAN ABSCISSA AND WEIGHTS.

IN THE FOLLOWING TABLE WE SUMMARIZE CLASSICAL QUADRATURE RULES

GAUSSIAN QUADRATURE	WEIGHT FUNCTION W(X)	RECURRENCE COEFFICIENTS B[K]	C[K]	INTEGRAL OF WEIGHT FUNCTION
LEGENDRE	1	0	$K^{**2}*(4*K^{**2}-1)$	2
CHEBYSHEV (1-ST KIND)	$1/\text{SQRT}(1-X^{**2})$	0	$1/2, K=1$ $1/4, K>1$	PI
CHEBYSHEV (2-ND KIND)	$\text{SQRT}(1-X^{**2})$	0	1/4	PI/2
JACOBI	$(1-X)^{\text{ALPHA}}*(1+X)^{\text{BETA}}$	$-(\text{ALPHA}^{**2}-K+\text{ALPHA}+\text{BETA})/(2*(K+\text{ALPHA}+\text{BETA}+2))$	$4*(1+\text{ALPHA})*(K+\text{ALPHA})*((2*K+\text{ALPHA}+\text{BETA})^{**2}-1)/(K+\text{BETA})*(K+\text{ALPHA}+\text{BETA})/((2*K+\text{ALPHA}+\text{BETA})^{**2}-1), K>1$	$2^{**}(\text{ALPHA}+\text{BETA}+1)*\text{GAMMA}(\text{ALPHA}+1)*\text{GAMMA}(\text{BETA}+1)/\text{GAMMA}(\text{ALPHA}+\text{BETA}+2)$
LAGUERRE	$\text{EXP}(-X)*X^{**\text{ALPHA}}$	$2*K+\text{ALPHA}+1$	$K*(K+\text{ALPHA})$	$\text{GAMMA}(\text{ALPHA}+1)$
HERMITE	$\text{EXP}(-X^{**2})$	0	K/2	SQRT(PI)

(THE INTEGRATION INTERVALS ARE: [-INFINITY, INFINITY] FOR HERMITE;
[0, INFINITY] FOR LAGUERRE;
[-1, 1] FOR THE OTHERS.)
FOR NON-CLASSICAL WEIGHT FUNCTIONS ON A FINITE INTERVAL THE RECURSION
COEFFICIENTS (AND THE SQUARE LENGTHS OF THE CORRESPONDING ORTHOGONAL-
POLYNOMIALS) CAN BE OBTAINED BY THE PROCEDURE RECCOF (THIS SECTION).

REFERENCES:

GAUTSCHI, W. (1970):
 GENERATION OF GAUSSIAN QUADRATURE RULES AND
 ORTHOGONAL POLYNOMIALS.
 IN: COLLOQUIUM APPROXIMATIETHEORIE,
 MC SYLLABUS 14.

EXAMPLE OF USE: SEE SUBSECTION GSSWTSSYM.

SUBSECTION: GSSWTSSYM.

CALLING SEQUENCE:

THE DECLARATION OF THE PROCEDURE IN THE CALLING PROGRAM READS:

```
"PROCEDURE" GSSWTSSYM(N,ZER,C,W);
"VALUE" N; "INTEGER" N; "ARRAY" ZER,C,W;
"CODE" 31252;
```

THE MEANING OF THE FORMAL PARAMETERS IS:

```
N: <ARITHMETIC EXPRESSION>;
   ENTRY: THE WEIGHTS OF AN N-POINTS GAUSS RULE ARE
         TO BE CALCULATED (BECAUSE OF SYMMETRY ONLY
         (N+1)/2 OF THE VALUES ARE DELIVERED);
ZER: <ARRAY IDENTIFIER>;
     "ARRAY" ZER[1:N//2]
     ENTRY: THE NEGATIVE ZEROS OF THE N-TH DEGREE ORTHOGONAL
           POLYNOMIAL (ZER[I] < ZER[I+1], I=1,2,...,N//2-1);
           (IF N IS ODD THEN 0 IS ALSO A ZERO.)
C: <ARRAY IDENTIFIER>;
   "ARRAY" C[1:N-1];
   ENTRY: THE RECURRENCE COEFFICIENTS;
W: <ARRAY IDENTIFIER>;
   "ARRAY" W[1:(N+1)//2];
   EXIT : PART OF THE GAUSSIAN WEIGHTS DIVIDED BY THE
         INTEGRAL OF THE WEIGHT FUNCTION.
         (NOTE THAT W[N+1-K]=W[K] , K=1,2,...,(N+1)//2.)
```

PROCEDURES USED: ALLORTPOLSYM = CP 31049.

METHOD AND PERFORMANCE: SEE SUBSECTION GSSWTS; THIS PROCEDURE IS
 SUPPLIED FOR STORAGE ECONOMICAL REASONS.

REFERENCES: SEE SUBSECTION GSSWTS.

EXAMPLE OF USE :

THE FOLLOWING PROGRAM DELIVERS THE GAUSSIAN WEIGHTS FOR THE 5-POINTS GAUSS-CHEBYSHEV QUADRATURE RULE BY MEANS OF THE PROCEDURE GSSWTSSYM (C[1]=0.5; C[K]=0.25, K=2,3,...; ZER[I] = $\cos((2*(N-I) - 1) / (2*N) * \pi)$, I=1,2,...,N/2.

```
"BEGIN"
  "PROCEDURE" GSSWTSSYM(N,ZER,C,W);
  "VALUE" N; "INTEGER" N; "ARRAY" ZER,C,W;
  "CODE" 31252;
  "REAL" PI; "INTEGER" I;
  "ARRAY" ZER[1:2], W[1:3], C[1:4];
  PI:=4*ARCTAN(1);
  C[1]:=.5;
  "FOR" I:=2 "STEP" 1 "UNTIL" 4 "DO"
  C[I]:=.25;
  ZER[1]:=COS(.9*PI);
  ZER[2]:=COS(.7*PI);
  GSSWTSSYM(5,ZER,C,W);
  OUTPUT(61,("2/,5(3B,-ZD.3D)"),W[1]*PI,W[2]*PI,W[3]*PI,
  W[2]*PI,W[1]*PI);
"END";

RESULTS:
  0.628      0.628      0.628      0.628      0.628
```

SOURCE TEXT(S):

```

"CODE" 31254;
"PROCEDURE" RECCDF(N,M,X,WX,B,C,L,SYM);
"VALUE" N,M,SYM;"INTEGER" N,M;"BOOLEAN" SYM;
"REAL" X,WX;"ARRAY" B,C,L;
"BEGIN" "INTEGER" I,J,UP;"REAL" R,S,PIM,H,HH,ARG,SA;
"REAL" "PROCEDURE" ORTPOL(N,X,B,C);
"VALUE" N,X;"INTEGER" N;"REAL" X;"ARRAY" B,C;
"CODE" 31044;
PIM:=4*ARCTAN(1)/M;
"IF" SYM "THEN" "BEGIN"
"FOR" J:=0 "STEP" 1 "UNTIL" N"DO"
"BEGIN" R:=B[J]:=0;UP:=M "DIV" 2;
"FOR" I:=1 "STEP" 1 "UNTIL" UP"DO"
"BEGIN" ARG:=(I-.5)*PIM;X:=COS(ARG);
R:=R+SIN(ARG)*WX*ORTPOL(J,X,B,C)**2;
"END";"IF" UP*2=M "THEN" L[J]:=2*R*PIM "ELSE"
"BEGIN" X:=0;L[J]:=(2*R+WX*ORTPOL(J,0,B,C)**2)*PIM "END";
C[J]:="IF" J=0 "THEN" 0 "ELSE" L[J]/L[J-1];
"END" "END" "ELSE"
"FOR" J:=0 "STEP" 1 "UNTIL" N "DO"
"BEGIN" R:=S:=0;UP:=M"DIV" 2;
"FOR" I:=1 "STEP" 1 "UNTIL" UP "DO"
"BEGIN" ARG:=(I-.5)*PIM;SA:=SIN(ARG);X:=COS(ARG);
H:=WX*ORTPOL(J,X,B,C)**2;X:=-X;HH:=WX*ORTPOL(J,X,B,C)**2;
R:=R+(H+HH)*SA;S:=S+(HH-H)*X*SA;
"END";B[J]:=S*PIM;
"IF" UP*2=M "THEN" L[J]:=R*PIM"ELSE"
"BEGIN" X:=0;L[J]:=(R+WX*ORTPOL(J,0,B,C)**2)*PIM "END";
C[J]:="IF" J=0 "THEN" 0 "ELSE" L[J]/L[J-1];
"END";
"END" RECCDF;
"EDP"

```

```

"CODE" 31253;
"PROCEDURE" GSSWTS(N,ZER,B,C)RESULTS:(W);
"VALUE" N; "INTEGER" N;
"ARRAY" ZER,B,C,W;
"BEGIN"
  "INTEGER" J,K; "REAL" S; "ARRAY" P(0:N-1);
  "PROCEDURE" ALLORTPOL(N,X,B,C,P);
  "VALUE" N,X; "INTEGER" N; "REAL" X; "ARRAY" B,C,P;
  "CODE" 31045;
  "FOR" J:=1 "STEP" 1 "UNTIL" N "DO"
  "BEGIN"
    ALLORTPOL(N=1,ZER[J],B,C,P);
    S:=0.0;
    "FOR" K:=N-1 "STEP" -1 "UNTIL" 1 "DO"
    S:=(S+P[K]**2)/C[K];
    W[J]:=1/(1+S);
  "END"
"END" GSSWTS;
"EQP"

```

```

"CODE" 31252;
"PROCEDURE" GSSWTSSYM(N,ZER,C)RESULTS:(W);
"VALUE" N; "INTEGER" N;
"ARRAY" ZER,C,W;
"BEGIN"
  "PROCEDURE" ALLORTPOLSYM(N,X,C,P);
  "VALUE" N,X; "INTEGER" N; "REAL" X; "ARRAY" C,P;
  "CODE" 31049;
  "INTEGER" LOW,UP,DUMMY;
  "ARRAY" P(0:N-1);
  LOW:=1; UP:=N;
  "FOR" DUMMY:=1
  "WHILE" LOW < UP "DO"
  "BEGIN" "INTEGER" I; "REAL" S;
  ALLORTPOLSYM(N=1,ZER[LOW],C,P);
  S:=P[N-1]**2;
  "FOR" I:=N-1 "STEP" -1 "UNTIL" 1 "DO"
  S:=S/C[I] + (P[I-1])**2;
  W[LOW]:=1/S; LOW:=LOW+1; UP:=UP-1;
  "END";
  "IF" LOW = UP
  "THEN" "BEGIN" "INTEGER" TWOI; "REAL" S; S:=1.0;
  "FOR" TWOI:=N-1 "STEP" -2 "UNTIL" 2 "DO"
  S:=S*C[TWOI-1]/C[TWOI]+1;
  W[LOW]:=1/S;
  "END";
"END" GSSWTSSYM;
"EQP"

```


AUTHORS : M. BAKKER.

INSTITUTE : MATHEMATICAL CENTRE, AMSTERDAM.

RECEIVED : 760131.

BRIEF DESCRIPTION :

THIS SECTION CONTAINS THE FOLLOWING PROCEDURES :

(1) GSS JAC WGHTS :

GIVEN THE TWO PARAMETERS ALFA AND BETA, THIS PROCEDURE CALCULATES THE N ZEROS OF THE N-TH JACOBI POLYNOMIAL AND THE CORRESPONDING GAUSS-CHRISTOFFEL NUMBERS NEEDED FOR THE N-POINT GAUSS-JACOBI QUADRATURE OVER $[-1, +1]$ WITH WEIGHT FUNCTION

$$W(X) = (1-X)**ALFA*(1+X)**BETA;$$

(2) GSS LAG WGHTS :

GIVEN THE PARAMETER ALFA, THIS PROCEDURE CALCULATES THE N ZEROS OF THE N-TH LAGUERRE POLYNOMIAL AND THE GAUSS-CHRISTOFFEL NUMBERS NEEDED FOR THE N-POINT GAUSS-LAGUERRE QUADRATURE OF $F(X)$ OVER $(0, INFINITY)$ WITH RESPECT TO THE WEIGHT FUNCTION

$$W(X) = X**ALFA*EXP(-X).$$

THESE PROCEDURES CAN BE USED FOR GAUSSIAN QUADRATURE-RULES OF THE JACOBI AND LAGUERRE TYPE. LET THE WEIGHT FUNCTION $W(X)$ AND THE INTERVAL (A, B) DETERMINE THE SYSTEM OF POLYNOMIALS ORTHOGONAL ON (A, B) WITH RESPECT TO $W(X)$. THEN THE N-POINT GAUSSIAN QUADRATURE RULE APPROXIMATES THE INTEGRAL

$$\int_{X=A}^{X=B} F(X) W(X) DX$$

BY THE EXPRESSION

$$\sum_{J=1}^{J=N} W[J] F(X[J])$$

WHERE THE ABSCISSAS $X[J]$ ARE THE ZEROS OF THE N-TH POLYNOMIAL AND $W[J]$ ARE THE CORRESPONDING GAUSS-CHRISTOFFEL NUMBERS.

KEYWORDS:

GAUSSIAN QUADRATURE,
ZEROS OF ORTHOGONAL POLYNOMIALS,
GAUSS-CHRISTOFFEL NUMBERS,
GAUSSIAN WEIGHTS.

LANGUAGE: ALGOL 60.

REFERENCES:

- [1] M. ABRAMOWITZ AND I. A. STEGUN,
HANDBOOK OF MATHEMATICAL FUNCTIONS, CH. 22,
- [2] J. STÖRER,
EINFÜHRUNG IN DIE NUMERISCHE MATHEMATIK 1,
SPRINGER VERLAG, BERLIN, HEIDELBERG, GÖTTINGEN.

SUBSECTION: GSS JAC WGHTS.

CALLING SEQUENCE:

THE DECLARATION OF THE PROCEDURE IN THE CALLING PROGRAM READS:

```
"PROCEDURE" GSS JAC WGHTS(N, ALFA, BETA, X, W);  
"VALUE" N, ALFA, BETA;  
"INTEGER" N; "REAL" ALFA, BETA;  
"CODE" 31425;
```

THE MEANING OF THE FORMAL PARAMETERS IS:

```
N: <ARITHMETIC EXPRESSION>;  
    THE UPPER BOUND OF THE ARRAYS X AND W; N>=1;  
ALFA, BETA: <ARITHMETIC EXPRESSION>;  
    THE PARAMETERS OF THE WEIGHT FUNCTION FOR  
    THE JACOBI POLYNOMIALS; ALFA, BETA > -1;  
X: <ARRAY IDENTIFIER>;  
    "ARRAY" X[1:N];  
    EXIT: X[I] IS THE I-TH ZERO OF THE N-TH JACOBI POLYNOMIAL;  
W: <ARRAY IDENTIFIER>;  
    "ARRAY" W[1:N];  
    EXIT: W[I] IS THE GAUSS-CHRISTOFFEL NUMBER  
    ASSOCIATED WITH THE I-TH ZERO OF THE N-TH JACOBI POLYNOMIAL.
```


PROCEDURES USED:

GAMMA = CP 35061;
ALL JAC ZER = CP 31370.

REQUIRED CENTRAL MEMORY:

TWO AUXILIARY ARRAYS OF N REALS ARE USED.

RUNNING TIME: ROUGHLY PROPORTIONAL TO N CUBED.

METHOD AND PERFORMANCE:

AS IS WELL-KNOWN, THE GAUSSIAN QUADRATURE RULES ARE BASED ON THE ZEROS OF ORTHOGONAL POLYNOMIALS. PROCEDURES FOR THE COMPUTATION OF THESE ZEROS CAN BE FOUND IN SECTION 3.6.2. AFTER THE COMPUTATION OF THE ZEROS OF THE JACOBI POLYNOMIAL THE GAUSSIAN WEIGHTS ARE COMPUTED OF THE FORMULA

$$W_{II} = 1 / \left(\sum_{J=0}^{J=N-1} P(J, \text{ALFA}, \text{BETA}, X[II])^2 \right)$$

WHERE P(J, ALFA, BETA, X[II]) IS THE J-TH ORTHONORMAL JACOBI POLYNOMIAL; SEE FURTHER [2], CH. III.

EXAMPLE OF USE:

THE PROGRAM

```
"BEGIN" "COMMENT" EVALUATION OF THE INTEGRAL
  TO X=1
  INTEGRAL (1+X)**2 * (1-X) * EXP(X) DX
  FROM X=-1
  BY MEANS OF FIVE POINT GAUSS-JACOBI QUADRATURE.
  THE EXACT VALUE IS 2*EXP(1)-10/EXP(1);
  "REAL" ALFA, BETA, INT; "INTEGER" N; "ARRAY" X, W(1:5);
  "REAL" "PROCEDURE" F(X); "VALUE" X; "REAL" X; F:=EXP(X);
  "PROCEDURE" GSS JAC WGHTS (N, ALFA, BETA, X, W); "CODE" 31425;
  ALFA:= 1; BETA:= 2; N:= 5; INT:= 0;
  GSS JAC WGHTS( N, ALFA, BETA, X, W);
  "FOR" N:= 1 "STEP" 1 "UNTIL" 5 "DO" INT:= INT + W[N] * F(X[N]);
  OUTPUT(61, "(" /, 4B+D. 4D"+ZD)", INT - 2 * EXP(1) + 10 / EXP(1))
  "END"
```

PRINTS THE FOLOWING RESULT:

-1.5932"-10

SUBSECTION: GSS LAG WGHTS.

CALLING SEQUENCE:

THE DECLARATION OF THE PROCEDURE IN THE CALLING PROGRAM READS:

```
"PROCEDURE" GSS LAG WGHTS (N, ALFA, X, W);  
"VALUE" N, ALFA;  
"INTEGER" N; "REAL" ALFA; "ARRAY" X, W;  
"CODE" 31427;
```

THE MEANING OF THE FORMAL PARAMETERS IS:

```
N: <ARITHMETIC EXPRESSION>;  
    THE UPPER BOUND OF THE ARRAYS X AND W; N>=1;  
ALFA: <ARITHMETIC EXPRESSION>;  
    THE PARAMETER OF THE WEIGHT FUNCTION FOR THE  
    LAGUERRE POLYNOMIALS;  
    ALFA>-1;  
X: <ARRAY IDENTIFIER>;  
    "ARRAY" X[I: N];  
    EXIT: X[I] IS THE I-TH ZERO OF THE N-TH  
    LAGUERRE POLYNOMIAL;  
W: <ARRAY IDENTIFIER>;  
    "ARRAY" W[I: N];  
    EXIT: W[I] IS THE GAUSSIAN WEIGHT CORRESPONDING  
    WITH THE I-TH ZERO OF THE N-TH LAGUERRE POLYNOMIAL.
```

PROCEDURES USED:

```
GAMMA          = CP 35061,  
ALL LAG ZER    = CP 31371.
```

REQUIRED CENTRAL MEMORY:

TWO AUXILIARY ARRAYS OF N REALS ARE USED.

RUNNING TIME:

ROUGHLY PROPORTIONAL TO N CUBED.

METHOD AND PERFORMANCE:

THE ZEROS AND WEIGHTS ARE COMPUTED IN THE SAME
WAY AS IN THE PROCEDURE GSS JAC WGHTS.

EXAMPLE OF USE:

THE PROGRAM

```
"BEGIN" "COMMENT" COMPUTATION OF THE INTEGRAL FROM 0 TO INFINITY OF
SIN(X)*EXP(-X) BY MEANS OF A TEN POINT GAUSS-LAGUERRE
QUADRATURE.THE EXACT VALUE IS 0.5;
"REAL" INT;"INTEGER" N;"ARRAY" X, W[1:10];
"REAL""PROCEDURE" F(X); "VALUE"X; "REAL"X; F:=-SIN(X);
"PROCEDURE" GSS LAG WGHTS(N, ALFA, X, W); "CODE" 31427;
GSS LAG WGHTS(10, 0, X, W); INT:=0;
"FOR" N:=10 "STEP" -1 "UNTIL" 1 "DO" INT:= INT + W[N] * F(X[N]);
OUTPUT(61, "(-D,4D"-ZD)", INT-0.5)
"END"
```

PRINTS THE RESULT:

2.0497" -7

SOURCE TEXTS:

```

"CODE" 31425;
"PROCEDURE" GSS JAC WGHTS(N, ALFA, BETA, X, W);
"VALUE" N, ALFA, BETA; "INTEGER" N; "REAL" ALFA, BETA;
"ARRAY" X, W;
"IF" ALFA = BETA "THEN"
"BEGIN" "INTEGER" I, J, M;
"ARRAY" B[1:N - 1]; "REAL" RO, R1, R2, S, HO, ALFA2, XI;
"REAL" "PROCEDURE" GAMMA(X); "CODE" 35061;
"PROCEDURE" ALL JAC ZER(N, ALFA, BETA, ZER); "CODE" 31370;

ALL JAC ZER(N, ALFA, ALFA, X); ALFA2:= 2*ALFA;
HO:= 2**((ALFA2 + 1)*GAMMA(1 + ALFA)**2/GAMMA(ALFA2 + 2));
B[1]:= 1/SQRT(3 + ALFA2); M:= N - (N//2);
"FOR" I:= 2 "STEP" 1 "UNTIL" N - 1 "DO"
B[I]:= SQRT(I*(I + ALFA2)/(4*(I + ALFA)**2 - 1));
"FOR" I:= 1 "STEP" 1 "UNTIL" M "DO"
"BEGIN" XI:= ABS(X[I]); RO:= 1; R1:= XI/B[1];
S:= 1 + R1*R1;
"FOR" J:= 2 "STEP" 1 "UNTIL" N - 1 "DO"
"BEGIN" R2:= (XI*R1 - B[J - 1]*RO)/B[J];
RO:= R1; R1:= R2; S:= S + R2*R2
"END";
W[I]:= W[N + 1 - I]:= HO/S
"END"
"END" "ELSE"
"BEGIN" "INTEGER" I, J; "ARRAY" A, B[0:N];
"REAL" MIN, SUM, HO, RO, R1, R2, XI, ALFABETA;
"PROCEDURE" ALL JAC ZER(N, ALFA, BETA, ZER); "CODE" 31370;
"REAL" "PROCEDURE" GAMMA(X); "CODE" 35061;
ALFABETA:= ALFA + BETA; MIN:= (BETA - ALFA)*ALFABETA;
B[0]:= 0; SUM:= ALFABETA + 2; A[0]:= (BETA - ALFA)/SUM;
A[1]:= MIN /SUM/(SUM + 2);
B[1]:= 2*SQRT((1 + ALFA)*(1 + BETA)/(SUM + 1))/SUM;
"FOR" I:= 2 "STEP" 1 "UNTIL" N - 1 "DO"
"BEGIN" SUM:= I + 1 + ALFABETA;
A[I]:= MIN/SUM/(SUM + 2);
B[I]:= (2/SUM)*
SQRT(I*(SUM - I)*(I + ALFA)*(I + BETA)/(SUM*SUM - 1))
"END";
HO:= 2**((ALFABETA + 1)*GAMMA(1 + ALFA)*GAMMA(1 + BETA)/
GAMMA(2 + ALFABETA));
ALL JAC ZER(N, ALFA, BETA, X);
"FOR" I:= 1 "STEP" 1 "UNTIL" N "DO"
"BEGIN" XI:= X[I]; RO:= 1; R1:= (XI - A[0])/B[1];
SUM:= 1 + R1*R1;
"FOR" J:= 2 "STEP" 1 "UNTIL" N - 1 "DO"
"BEGIN" R2:= ((XI - A[J - 1])*R1 - B[J - 1]*RO)/B[J];
SUM:= SUM + R2*R2; RO:= R1; R1:= R2
"END";
W[I]:= HO/SUM
"END"
"END" GSS JAC WGHTS;
"EQP"

```

```
"CODE" 31427;
"PROCEDURE" GSS LAG WGHTS(N, ALFA, X, W);
"VALUE" N, ALFA; "INTEGER" N; "REAL" ALFA; "ARRAY" X, W;
"BEGIN" "INTEGER" I, J; "REAL" HO, S, RO, R1, R2, XI;
"ARRAY" A, B[0:N];
"PROCEDURE" ALL LAG ZER(N, ALFA, X); "CODE" 31371;
"REAL" "PROCEDURE" GAMMA(X); "CODE" 35061;
A[0]:= 1 + ALFA; A[1]:= 3 + ALFA; B[1]:= SQRT(A[0]);
"FOR" I:= 2 "STEP" 1 "UNTIL" N - 1 "DO"
"BEGIN" A[I]:= I + I + ALFA + 1;
      B[I]:= SQRT(I*(I + ALFA))
"END";
ALL LAG ZER(N, ALFA, X); HO:= GAMMA(1 + ALFA);
"FOR" I:= 1 "STEP" 1 "UNTIL" N "DO"
"BEGIN" XI:= X[I]; RO:= 1;
      R1:= (XI - A[0])/B[1]; S:= 1 + R1*R1;
"FOR" J:= 2 "STEP" 1 "UNTIL" N - 1 "DO"
"BEGIN" R2:= ((XI - A[J - 1])*R1 - B[J - 1]*RO)/B[J];
      RO:= R1; R1:= R2; S:= S + R2*R2
"END";
      WEI:= HO/S
"END"
"END" GSS LAG WGHTS;
"EOF"
```


AUTHOR: J.C.P.BUS.

INSTITUTE: MATHEMATICAL CENTRE.

RECEIVED: 740218.

BRIEF DESCRIPTION:

THIS SECTION CONTAINS PROCEDURES FOR CALCULATING THE DERIVATIVES OF FUNCTIONS OF MORE VARIABLES, USING DIFFERENCE FORMULAS;
JACOBNNF CALCULATES THE JACOBIAN MATRIX OF AN N-DIMENSIONAL FUNCTION OF N VARIABLES USING FORWARD DIFFERENCES;
JACOBMMF CALCULATES THE JACOBIAN MATRIX OF AN N-DIMENSIONAL FUNCTION OF M VARIABLES USING FORWARD DIFFERENCES;
JACOBNNDF CALCULATES THE JACOBIAN MATRIX OF AN N-DIMENSIONAL FUNCTION OF N VARIABLES, IF THIS JACOBIAN IS KNOWN TO BE A BAND MATRIX AND HAVE TO BE STORED ROW-WISE IN A ONE-DIMENSIONAL ARRAY.

KEYWORDS:

NUMERICAL DIFFERENTIATION,
FUNCTIONS OF MORE VARIABLES,
DIFFERENCE FORMULAS.

SUBSECTION: JACOBNNF.

CALLING SEQUENCE:

THE HEADING OF THIS PROCEDURE READS:
 "PROCEDURE" JACOBNNF(N, X, F, JAC, I, DI, FUNCT);
 "VALUE" N; "INTEGER" I, N; "REAL" DI; "ARRAY" X, F, JAC;
 "PROCEDURE" FUNCT;

THE MEANING OF THE FORMAL PARAMETERS IS:

N: <ARITHMETIC EXPRESSION>;
 THE NUMBER OF INDEPENDENT VARIABLES AND THE DIMENSION OF
 THE FUNCTION;
 X: <ARRAY IDENTIFIER>;
 "ARRAY" X[1:N];
 ENTRY: THE POINT AT WHICH THE JACOBIAN HAS TO BE CALCULATED
 F: <ARRAY IDENTIFIER>;
 "ARRAY" F[1:N];
 ENTRY: THE VALUES OF THE FUNCTION-COMPONENTS AT THE POINT
 GIVEN IN ARRAY X;
 JAC: <ARRAY IDENTIFIER>;
 "ARRAY" JAC[1:N, 1:N];
 EXIT: THE JACOBIAN MATRIX IN SUCH A WAY THAT THE PARTIAL
 DERIVATIVE OF F[I] TO X[J] IS GIVEN IN
 JAC[I, J], I, J = 1, ..., N;
 I: <INTEGER VARIABLE>;
 A JENSEN PARAMETER; DI MAY BE DEPENDENT OF I;
 DI: <ARITHMETIC EXPRESSION>;
 THE PARTIAL DERIVATIVES TO X[I] ARE APPROXIMATED WITH
 FORWARD DIFFERENCES, USING AN INCREMENT TO THE I-TH
 VARIABLE THAT EQUALS THE VALUE OF DI, I = 1, ..., N;
 FUNCT: <PROCEDURE IDENTIFIER>;
 THE HEADING OF THIS PROCEDURE SHOULD READ:
 "PROCEDURE" FUNCT(N, X, F);
 "VALUE" N; "INTEGER" N; "ARRAY" X, F;
 THE MEANING OF THE FORMAL PARAMETERS IS:
 N: <ARITHMETIC EXPRESSION>;
 THE NUMBER OF INDEPENDENT VARIABLES OF THE FUNCTION F;
 X: <ARRAY IDENTIFIER>;
 THE INDEPENDENT VARIABLES ARE GIVEN IN X[1:N];
 F: <ARRAY IDENTIFIER>;
 AFTER A CALL OF FUNCT THE FUNCTION COMPONENTS SHOULD BE
 GIVEN IN F[1:N].

PROCEDURES USED: NONE.

REQUIRED CENTRAL MEMORY :

EXECUTION FIELD LENGTH: JACOBNNF DECLARES ONE AUXILIARY ARRAY OF ORDER N.

RUNNING TIME: PROPORTIONAL TO $N ** 2$.

LANGUAGE: ALGOL 60.

METHOD AND PERFORMANCE :

JACOBNNF CALCULATES THE JACOBIAN MATRIX OF AN N-DIMENSIONAL FUNCTION OF N VARIABLES; THE ELEMENTS OF THIS MATRIX, WHICH ARE THE PARTIAL DERIVATIVES OF THE FUNCTION, ARE CALCULATED USING FORWARD DIFFERENCES WITH AN INCREMENT TO THE I-TH VARIABLE OF DI , ($I = 1, \dots, N$).

EXAMPLE OF USE:

LET F BE DEFINED BY:

$F[1] = X[1] ** 3 + X[2]$,

$F[2] = 10 * X[2]$;

THE JACOBIAN MATRIX AT THE POINT (2, 1) MAY BE CALCULATED AND PRINTED BY THE FOLLOWING PROGRAM:

```
"BEGIN"
  "INTEGER" I; "ARRAY" JAC[1:2, 1:2], X, F[1:2];
  "PROCEDURE" JACOBNNF(N, X, F, JAC, I, DI, FU); "CODE" 34437;
  "PROCEDURE" F1(N, X, F); "VALUE" N; "INTEGER" N; "ARRAY" X, F;
  "BEGIN" F[1] := X[1] ** 3 + X[2]; F[2] := X[2] * 10 "END" F1;
  X[1] := 2; X[2] := 1; F1(2, X, F);
  JACOBNNF(2, X, F, JAC, I, "IF" I = 1 "THEN" "-6" "ELSE" 1, F1);
  OUTPUT(71, "(**," 4B, "(THE CALCULATED JACOBIAN IS:)", //,
  2(4B, 2(N), /)"", JAC[1, 1], JAC[1, 2], JAC[2, 1], JAC[2, 2])
"END"
```

RESULTS:

THE CALCULATED JACOBIAN IS:

```
+1.2000005938262"+001 +1.0000000000000"+000
+0.0000000000000"+000 +1.0000000000000"+001
```

SUBSECTION: JACOBDMF.

CALLING SEQUENCE:

THE HEADING OF THIS PROCEDURE READS:
 "PROCEDURE" JACOBDMF(N, M, X, F, JAC, I, DI, FUNCT);
 "VALUE" N, M; "INTEGER" I, N, M; "REAL" DI; "ARRAY" X, F, JAC;
 "PROCEDURE" FUNCT;

THE MEANING OF THE FORMAL PARAMETERS IS:

N: <ARITHMETIC EXPRESSION>;
 THE NUMBER OF FUNCTION COMPONENTS;
 M: <ARITHMETIC EXPRESSION>;
 THE NUMBER OF INDEPENDENT VARIABLES;
 X: <ARRAY IDENTIFIER>;
 "ARRAY" X[I:M];
 ENTRY: THE POINT AT WHICH THE JACOBIAN HAS TO BE CALCULATED
 F: <ARRAY IDENTIFIER>;
 "ARRAY" F[I:N];
 ENTRY: THE VALUES OF THE FUNCTION-COMPONENTS AT THE POINT
 GIVEN IN ARRAY X;
 JAC: <ARRAY IDENTIFIER>;
 "ARRAY" JAC[I:N, 1:M];
 EXIT: THE JACOBIAN MATRIX IN SUCH A WAY THAT THE PARTIAL
 DERIVATIVE OF F[I] TO X[J] IS GIVEN IN
 JAC[I, J], I = 1, ..., N, J = 1, ..., M;
 I: <INTEGER VARIABLE>;
 A JENSEN PARAMETER; DI MAY BE DEPENDENT OF I;
 DI: <ARITHMETIC EXPRESSION>;
 THE PARTIAL DERIVATIVES TO X[I] ARE APPROXIMATED WITH
 FORWARD DIFFERENCES, USING AN INCREMENT TO THE I-TH
 VARIABLE THAT EQUALS THE VALUE OF DI, I = 1, ..., M;
 FUNCT: <PROCEDURE IDENTIFIER>;
 THE HEADING OF THIS PROCEDURE READS :
 "PROCEDURE" FUNCT(N, M, X, F);
 "VALUE" N, M; "INTEGER" N, M; "ARRAY" X, F;
 THE MEANING OF THE FORMAL PARAMETERS IS :
 N: <ARITHMETIC EXPRESSION>;
 THE NUMBER OF FUNCTION COMPONENTS;
 M: <ARITHMETIC EXPRESSION>;
 THE NUMBER OF INDEPENDENT VARIABLES OF THE FUNCTION F;
 X: <ARRAY IDENTIFIER>;
 THE INDEPENDENT VARIABLES ARE GIVEN IN X[I:M];
 F: <ARRAY IDENTIFIER>;
 AFTER A CALL OF FUNCT THE FUNCTION COMPONENTS SHOULD BE
 GIVEN IN F[I:N].

PROCEDURES USED: NONE.

EXECUTION FIELD LENGTH: JACOBNMF DECLARES ONE AUXILIARY ARRAY OF ORDER N.

RUNNING TIME: PROPORTIONAL TO $N * M$.

LANGUAGE: ALGOL 60.

METHOD AND PERFORMANCE:

JACOBNMF CALCULATES THE JACOBIAN MATRIX OF AN N-DIMENSIONAL FUNCTION OF M VARIABLES; THE ELEMENTS OF THIS MATRIX, WHICH ARE THE PARTIAL DERIVATIVES OF THE FUNCTION, ARE CALCULATED USING FORWARD DIFFERENCES WITH AN INCREMENT TO THE I-TH VARIABLE OF DI , ($I = 1, \dots, M$).

EXAMPLE OF USE:

```

LET F BE DEFINED BY:
    F[1]= X[1] ** 3 + X[2],
    F[2]= 10 * X[2] + X[2] * X[1],
    F[3]= X[1] * X[2];
THE JACOBIAN MATRIX AT THE POINT (2, 1) MAY BE CALCULATED AND
PRINTED BY THE FOLLOWING PROGRAM:

"BEGIN"
  "INTEGER" I; "ARRAY" JAC[1:3, 1:2], X[1:2], F[1:3];
  "PROCEDURE" JACOBNMF(N, X, F, JAC, I, DI, FU); "CODE" 34438;
  "PROCEDURE" F1(N, M, X, F); "VALUE" N, M; "INTEGER" N, M;
  "ARRAY" X, F;
  "BEGIN" F[1]:= X[1] ** 3 + X[2];
    F[2]:= X[2] * 10 + X[2] * X[1] ** 2; F[3]:= X[1] * X[2]
  "END" F1;
  X[1]:= 2; X[2]:= 1; F1(3, 2, X, F);
  JACOBNMF(3, 2, X, F, JAC, I, "IF" I=2 "THEN" 1 "ELSE" "-5, F1);
  OUTPUT(71, ("*,4B,("THE CALCULATED JACOBIAN IS:)",//,
    3(4B,2(N),/))", JAC[1, 1], JAC[1, 2], JAC[2, 1], JAC[2, 2],
    JAC[3, 1], JAC[3, 2])
"END"

```

RESULTS:

THE CALCULATED JACOBIAN IS:

```

+1.2000060002038"+001   +1.0000000000000"+000
+4.0000100000270"+000   +1.4000000000000"+001
+1.00000000003174"+000  +2.0000000000000"+000

```

SUBSECTION: JACOBNDNF.

CALLING SEQUENCE:

THE HEADING OF THIS PROCEDURE READS :
 "PROCEDURE" JACOBNDNF(N, LW, RW, X, F, JAC, I, DI, FUNCT);
 "VALUE" N, LW, RW; "INTEGER" N, I, LW, RW; "REAL" DI;
 "ARRAY" X, F, JAC; "PROCEDURE" FUNCT;

THE MEANING OF THE FORMAL PARAMETERS IS:

N: <ARITHMETIC EXPRESSION>;
 THE NUMBER OF INDEPENDENT VARIABLES AND THE DIMENSION OF
 THE FUNCTION;
 LW: <ARITHMETIC EXPRESSION>;
 THE NUMBER OF CODIAGONALS TO THE LEFT OF THE MAIN DIAGONAL
 OF THE JACOBIAN MATRIX, WHICH IS KNOWN TO BE A BAND MATRIX;
 RW: <ARITHMETIC EXPRESSION>;
 THE NUMBER OF CODIAGONALS TO THE RIGHT OF THE MAIN DIAGONAL
 OF THE JACOBIAN MATRIX;
 X: <ARRAY IDENTIFIER>;
 "ARRAY" X[1:N];
 ENTRY: THE POINT AT WHICH THE JACOBIAN HAS TO BE CALCULATED
 F: <ARRAY IDENTIFIER>;
 "ARRAY" F[1:N];
 ENTRY: THE VALUES OF THE FUNCTION-COMPONENTS AT THE POINT
 GIVEN IN ARRAY X;
 JAC: <ARRAY IDENTIFIER>;
 "ARRAY" JAC [1 : (LW + RW) * (N - 1) + N];
 EXIT: THE JACOBIAN MATRIX IN SUCH A WAY THAT THE (I, J)-TH
 ELEMENT OF THE JACOBIAN, I.E. THE PARTIAL DERIVATIVE OF
 F[I] TO X[J], IS GIVEN IN
 JAC((LW + RW) * (I - 1) + J), FOR I = 1, ..., N
 J = MAX(1, I - LW), ..., MIN(N, I + RW);
 I: <INTEGER VARIABLE>;
 A JENSEN PARAMETER; DI MAY BE DEPENDENT OF I;
 DI: <ARITHMETIC EXPRESSION>;
 THE PARTIAL DERIVATIVES TO X[I] ARE APPROXIMATED WITH
 FORWARD DIFFERENCES, USING AN INCREMENT TO THE I-TH
 VARIABLE THAT EQUALS THE VALUE OF DI, I = 1, ..., N;

FUNCT: <PROCEDURE IDENTIFIER>;
THE HEADING OF THIS PROCEDURE READS :
"PROCEDURE" FUNCT(N, L, U, X, F);
"VALUE" N, L, U; "INTEGER" N, L, U; "ARRAY" X, F;
THE MEANING OF THE FORMAL PARAMETERS IS :
N: <ARITHMETIC EXPRESSION>;
THE NUMBER OF FUNCTION COMPONENTS;
L,U: <ARITHMETIC EXPRESSION>;
LOWER AND UPPER BOUND OF THE FUNCTION COMPONENT
SUBSCRIPT;
X: <ARRAY IDENTIFIER>;
THE INDEPENDENT VARIABLES ARE GIVEN IN X[1:N];
F: <ARRAY IDENTIFIER>;
AFTER A CALL OF FUNCT THE FUNCTION COMPONENTS F[I],
I = L, ..., U, SHOULD BE GIVEN IN F[L:U].

PROCEDURES USED: NONE.

EXECUTION FIELD LENGTH: JACOBNMF DECLARES ONE AUXILIARY ARRAY OF
MAXIMUM ORDER $LW + RW + 1$;

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

LANGUAGE: ALGOL 60.

METHOD AND PERFORMANCE:

JACOBNBDF CALCULATES THE JACOBIAN MATRIX OF AN N-DIMENSIONAL
FUNCTION OF N VARIABLES, IF THIS JACOBIAN IS KNOWN TO BE A BAND
MATRIX AND HAVE TO BE STORED ROW-WISE IN A ONE-DIMENSIONAL ARRAY;
THE ELEMENTS OF THIS JACOBIAN MATRIX ARE CALCULATED USING FORWARD
DIFFERENCES, WITH AN INCREMENT TO THE I-TH VARIABLE OF DI , ($I = 1,$
..., N).

EXAMPLE OF USE:

LET F BE DEFINED BY:
 $F[1] = (3 - 2 * X[1]) * X[1] + 1 - 2 * X[2]$,
 $F[I] = (3 - 2 * X[I]) * X[I] + 1 - X[I-1] - 2 * X[I+1]$, $I = 2, 3, 4$,
 $F[5] = 4 - 2 * X[5] - X[4]$;
 THE TRIDIAGONAL JACOBIAN MATRIX AT THE POINT X, GIVEN BY $X[I] = -1$,
 $I = 1, \dots, 5$, MAY BE CALCULATED AND PRINTED BY THE FOLLOWING
 PROGRAM:

```
"BEGIN"
  "INTEGER" I; "ARRAY" JAC[1:13], X, F[1:5];
  "PROCEDURE" JACBNBDF(N, L, R, X, F, J, I, D, G); "CODE" 34439;
  "PROCEDURE" F1(N, L, U, X, F); "VALUE" N, L, U;
  "INTEGER" N, L, U; "ARRAY" X, F;
  "BEGIN" "INTEGER" I;
    "FOR" I:= L "STEP" 1 "UNTIL" ("IF" U = 5 "THEN" 4 "ELSE" U)
      "DO"
        "BEGIN" F[I]:= (3 - 2 * X[I]) * X[I] + 1 - 2 * X[I + 1];
          "IF" I ^= 1 "THEN" F[I]:= F[I] - X[I - 1]
        "END";
        "IF" U = 5 "THEN" F[5]:= 4 - X[4] - X[5] * 2
      "END" F1;

  "PROCEDURE" LIST(ITEM); "PROCEDURE" ITEM;
  "BEGIN" "INTEGER" I;
    ITEM("THE CALCULATED TRIDIAGONAL JACOBIAN IS:");
    "FOR" I:= 1 "STEP" 1 "UNTIL" 13 "DO" ITEM(JAC[I])
  "END" LIST;

  "PROCEDURE" LAYOUT;
  FORMAT("(", "/", 4B, 40S, "/", 4B, 2(+.5D"+D2B), /, 4B, 3(+.5D"+D2B), /,
  16B, 3(+.5D"+D2B), /, 28B, 3(+.5D"+D2B), /, 40B, 2(+.5D"+D2B), /")");
  "FOR" I := 1 "STEP" 1 "UNTIL" 5 "DO" X[I]:= -1;
  F1(5, 1, 5, X, F);
  JACBNBDF(5, 1, 1, X, F, JAC, I, "IF" I = 5 "THEN" 1 "ELSE"
  "-6, F1);
  OUTLIST(71, LAYOUT, LIST)
"END"
```

RESULTS:

THE CALCULATED TRIDIAGONAL JACOBIAN IS:

+ .70000"+1	- .20000"+1				
- .10000"+1	+ .70000"+1	- .20000"+1			
	- .10000"+1	+ .70000"+1	- .20000"+1		
		- .10000"+1	+ .70000"+1	- .20000"+1	
			- .10000"+1	- .20000"+1	

SOURCE TEXT(S):

```

"CODE" 34437;
"PROCEDURE" JACOBNNF(N, X, F, JAC, I, DI, FUNCT); "VALUE" N;
"INTEGER" N, I; "REAL" DI; "ARRAY" X, F, JAC; "PROCEDURE" FUNCT;
"BEGIN" "INTEGER" J; "REAL" STEP, AID; "ARRAY" F1[1:N];
  "FOR" I:= 1 "STEP" 1 "UNTIL" N "DO"
    "BEGIN" STEP:= DI; AID:= X[I]; X[I]:= AID + STEP;
      STEP:= 1 / STEP; FUNCT(N, X, F1);
      "FOR" J:= 1 "STEP" 1 "UNTIL" N "DO"
        JAC[J,I]:= (F1[J] - F[J]) * STEP; X[I]:= AID
      "END"
    "END" JACOBNNF;
  "EJP"

"CODE" 34438;
"PROCEDURE" JACOBMMF(N, M, X, F, JAC, I, DI, FUNCT); "VALUE" N, M;
"INTEGER" N, M, I; "REAL" DI; "ARRAY" X, F, JAC; "PROCEDURE" FUNCT;
"BEGIN" "INTEGER" J; "REAL" STEP, AID; "ARRAY" F1[1:N];
  "FOR" I:= 1 "STEP" 1 "UNTIL" M "DO"
    "BEGIN" STEP:= DI; AID:= X[I]; X[I]:= AID + STEP;
      STEP:= 1 / STEP; FUNCT(N, M, X, F1);
      "FOR" J:= 1 "STEP" 1 "UNTIL" N "DO"
        JAC[J,I]:= (F1[J] - F[J]) * STEP; X[I]:= AID
      "END"
    "END" JACOBMMF;
  "EJP"

"CODE" 34439;
"PROCEDURE" JACOBNNDF(N, LW, RW, X, F, JAC, I, DI, FUNCT);
"VALUE" N, LW, RW; "INTEGER" I, N, LW, RW; "REAL" DI;
"ARRAY" X, F, JAC; "PROCEDURE" FUNCT;
"BEGIN" "INTEGER" J, K, L, U, T, B; "REAL" AID, STEPI;
  L:= 1; U:= LW + 1; T:= RW + 1; B:= LW + RW;
  "FOR" I:= 1 "STEP" 1 "UNTIL" N "DO"
    "BEGIN" "ARRAY" F1[L:U];
      STEPI:= DI; AID:= X[I]; X[I]:= AID + STEPI;
      FUNCT(N, L, U, X, F1); X[I]:= AID;
      K:= I + ("IF" I <= T "THEN" 0 "ELSE" I - T) * B;
      "FOR" J:= L "STEP" 1 "UNTIL" U "DO"
        "BEGIN" JAC[K]:= (F1[J] - F[J]) / STEPI; K:= K + B "END";
        "IF" I >= T "THEN" L:= L + 1;
        "IF" U < N "THEN" U:= U + 1
      "END"
    "END" JACOBNNDF;
  "EJP"

```


M1062C4 //// END OF LIST ////
M1062C4 //// END OF LIST ////

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

CONTRIBUTOR: J. C. P. BUS.

INSTITUTE: MATHEMATICAL CENTRE.

RECEIVED: 750615.

BRIEF DESCRIPTION:

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

KEYWORDS:

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

SUBSECTION: ZEROIN.

CALLING SEQUENCE:

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

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

THE MEANING OF THE FORMAL PARAMETERS IS:

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

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

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

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

PROCEDURES USED: DWARF = CP30003;

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

LANGUAGE: ALGOL 60.

METHOD AND PERFORMANCE:

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

EXAMPLE OF USE:

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

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

RESULT:

CALCULATED ZERO: +.489702748548240"+000

SUBSECTION: ZEROINRAT.

CALLING SEQUENCE:

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

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

$$T(X) = ABS(X) * RE + AE,$$
 WHERE RE AND AE ARE THE REQUIRED RELATIVE AND ABSOLUTE PRECISION RESPECTIVELY);

THE MEANING OF THE FORMAL PARAMETERS IS:

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

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

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

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

PROCEDURES USED: DWARF = CP30003;

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

LANGUAGE: ALGOL 60.

METHOD AND PERFORMANCE:

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

EXAMPLE OF USE:

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

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

RESULT:

CALCULATED ZERO: +.489702748548240"+000

REFERENCES:

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

SOURCE TEXT(S):

```

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

```

```

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

```


AUTHOR: T.J. DEKKER.

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

INSTITUTE: UNIVERSITY OF AMSTERDAM.

RECEIVED: 750615.

BRIEF DESCRIPTION:

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

KEYWORDS:

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

CALLING SEQUENCE:

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

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

THE MEANING OF THE FORMAL PARAMETERS IS:

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

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

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

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

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

PROCEDURES USED: DWARF = CP30003;

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

RUNNING TIME: SEE METHOD AND PERFORMANCE.

LANGUAGE: ALGOL 60.

METHOD AND PERFORMANCE:

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

THE PERFORMANCE IS AS FOLLOWS:

THE NUMBER OF EVALUATIONS OF F_X , DF_X AND TOL_X IS AT MOST $4 \log(\text{ABS}(X - Y)) / \text{TAU}$,

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

FOR A SIMPLE ZERO OF F , THE ASYMPTOTIC ORDER OF CONVERGENCE APPROXIMATELY EQUALS 2.414.

EXAMPLE OF USE:

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

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

RESULT:

CALCULATED ZERO AND FUNCTION VALUE:

+0.489702748548240"+000 -0.444089209850060"-015

OTHER STRADDLING APPROXIMATION AND FUNCTION VALUE:

+0.489702748548250"+000 +0.177635683940030"-013

REFERENCES:

- [1]: BUS, J.C.P. AND DEKKER, T.J.,
TWO EFFICIENT ALGORITHMS WITH GUARANTEED CONVERGENCE FOR
FINDING A ZERO OF A FUNCTION.
MATHEMATICAL CENTRE, REPORT NW 13/74, AMSTERDAM (1974),
(ALSO TO APPEAR IN TOMS 1975).
- [2]: OSTROWSKI, A.M.,
SOLUTION OF EQUATIONS AND SYSTEMS OF EQUATIONS.
ACADEMIC PRESS 1966. CHAPTERS 3 AND 11.

SOURCE TEXT(S):

```

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

```

SECTION : 5.1.1.2.2

(OCTOBER 1974)

PAGE 1

AUTHOR: J.C.P.BUS.

INSTITUTE: MATHEMATICAL CENTRE.

RECEIVED: 740218.

BRIEF DESCRIPTION:

THIS SECTION CONTAINS TWO PROCEDURES FOR SOLVING SYSTEMS OF NON-LINEAR EQUATIONS, OF WHICH THE JACOBIAN IS KNOWN TO BE A BAND MATRIX.

QUANEWBND ASKS FOR AN APPROXIMATION OF THE JACOBIAN AT THE INITIAL GUESS.

QUANEWBND1 CALCULATES AN APPROXIMATION OF THE JACOBIAN AT THE INITIAL GUESS, USING FORWARD DIFFERENCES.

KEYWORDS:

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

SUBSECTION: QUANEWBND.

CALLING SEQUENCE:

THE HEADING OF THIS PROCEDURE READS :

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

THE MEANING OF THE FORMAL PARAMETERS IS :

```
N: <ARITHMETIC EXPRESSION>;
    THE NUMBER OF INDEPENDENT VARIABLES; THE NUMBER OF
    EQUATIONS SHOULD ALSO BE EQUAL TO N;
LW: <ARITHMETIC EXPRESSION>;
    THE NUMBER OF CODIAGONALS TO THE LEFT OF THE MAIN DIAGONAL
    OF THE JACOBIAN;
RW: <ARITHMETIC EXPRESSION>;
    THE NUMBER OF CODIAGONALS TO THE RIGHT OF THE MAIN DIAGONAL
    OF THE JACOBIAN;
X: <ARRAY IDENTIFIER>;
    "ARRAY" X[1:N];
    ENTRY: AN INITIAL ESTIMATE OF THE SOLUTION OF THE SYSTEM
           THAT HAS TO BE SOLVED;
           EXIT: THE CALCULATED SOLUTION OF THE SYSTEM;
F: <ARRAY IDENTIFIER>;
    "ARRAY" F[1:N];
    ENTRY: THE VALUES OF THE FUNCTION COMPONENTS AT THE
           INITIAL GUESS;
           EXIT: THE VALUES OF THE FUNCTION COMPONENTS AT THE
                 CALCULATED SOLUTION;
JAC: <ARRAY IDENTIFIER>;
    "ARRAY" JAC[1:(LW + RW) * (N - 1) + N];
    ENTRY: AN APPROXIMATION OF THE JACOBIAN AT THE INITIAL
           ESTIMATE OF THE SOLUTION;
           EXIT: AN APPROXIMATION OF THE JACOBIAN AT THE CALCULATED
                 SOLUTION;
    AN APPROXIMATION OF THE (I, J)-TH ELEMENT OF THE JACOBIAN
    IS GIVEN IN JAC[(LW + RW) * (I - 1) + J], FOR I = 1, ..., N
    AND J = MAX(1, I - LW), ..., MIN(N, I + RW);
```

```

FUNCT: <PROCEDURE IDENTIFIER>;
THE HEADING OF THIS PROCEDURE READS :
"BOOLEAN" "PROCEDURE" FUNCT(N, L, U, X, F);
"VALUE" N, L, U; "INTEGER" N, L, U; "ARRAY" X, F;
THE MEANING OF THE FORMAL PARAMETERS IS :
N: <ARITHMETIC EXPRESSION>;
    THE NUMBER OF INDEPENDENT VARIABLES OF THE FUNCTION F;
L, U: <ARITHMETIC EXPRESSION>;
    LOWER AND UPPER BOUND OF THE FUNCTION COMPONENT
    SUBSCRIPT;
X: <ARRAY IDENTIFIER>;
    THE INDEPENDENT VARIABLES ARE GIVEN IN X[1:N];
F: <ARRAY IDENTIFIER>;
    AFTER A CALL OF FUNCT THE FUNCTION COMPONENTS F[I],
    I = L, ..., U, SHOULD BE GIVEN IN F[L:U];
IF THE VALUE OF THE PROCEDURE IDENTIFIER EQUALS FALSE,
THEN THE EXECUTION OF QUANEWBN0 WILL BE TERMINATED, WHILE
THE VALUE OF OUT[5] IS SET EQUAL TO 2;
IN: <ARRAY IDENTIFIER>;
"ARRAY" INC[4];
ENTRY :
IN THIS AUXILIARY ARRAY ONE SHOULD GIVE THE FOLLOWING
VALUES FOR CONTROLLING THE PROCESS:
INC[0]: THE MACHINE PRECISION;
INC[1]: THE RELATIVE PRECISION ASKED FOR;
INC[2]: THE ABSOLUTE PRECISION ASKED FOR; IF THE VALUE,
    DELIVERED IN OUT[5] EQUALS ZERO, THEN THE LAST
    CORRECTION VECTOR D, SAY, WHICH IS A MEASURE FOR
    THE ERROR IN THE SOLUTION, SATISFIES THE INEQUALITY
     $NORM(D) \leq NORM(X) * INC[1] + INC[2]$ ,
    WHEREBY X DENOTES THE CALCULATED SOLUTION, GIVEN IN
    ARRAY X AND  $NORM(.)$  DENOTES THE EUCLIDIAN NORM;
    HOWEVER, WE CAN NOT GUARANTEE THAT THE TRUE ERROR IN
    THE SOLUTION SATISFIES THIS INEQUALITY, ESPECIALLY
    IF THE JACOBIAN IS (NEARLY) SINGULAR AT THE
    SOLUTION;
INC[3]: THE MAXIMUM VALUE OF THE NORM OF THE RESIDUAL
    VECTOR ALLOWED; IF  $OUT[5] = 0$ , THEN THIS RESIDUAL
    VECTOR F, SAY, SATISFIES:  $NORM(F) \leq INC[3]$ ;
INC[4]: THE MAXIMUM NUMBER OF FUNCTION COMPONENT
    EVALUATIONS ALLOWED;  $L - U + 1$  FUNCTION COMPONENT
    EVALUATIONS ARE COUNTED EACH CALL OF
    FUNCT(N, L, U, X, F); IF  $OUT[5]=1$ , THEN THE PROCESS
    IS TERMINATED, BECAUSE THE NUMBER OF EVALUATIONS
    EXCEEDED THE VALUE GIVEN IN INC[4];

```

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

PROCEDURES USED:

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

EXECUTION FIELD LENGTH:

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

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

LANGUAGE: ALGOL 60.

METHOD AND PERFORMANCE :

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

EXAMPLE OF USE: SEE QUANEWBND1 (THIS SECTION).

REFERENCES :

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

SUBSECTION: QUANEWBND1.

CALLING SEQUENCE :

THE HEADING OF THIS PROCEDURE READS :

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

THE MEANING OF THE FORMAL PARAMETERS IS:

```
N: <ARITHMETIC EXPRESSION>;  
THE NUMBER OF INDEPENDENT VARIABLES; THE NUMBER OF  
EQUATIONS SHOULD ALSO BE EQUAL TO N;  
LW: <ARITHMETIC EXPRESSION>;  
THE NUMBER OF CODIAGONALS TO THE LEFT OF THE MAIN DIAGONAL  
OF THE JACOBIAN;  
RW: <ARITHMETIC EXPRESSION>;  
THE NUMBER OF CODIAGONALS TO THE RIGHT OF THE MAIN DIAGONAL  
OF THE JACOBIAN;  
X: <ARRAY IDENTIFIER>;  
"ARRAY" X[1:N];  
ENTRY: AN INITIAL ESTIMATE OF THE SOLUTION OF THE SYSTEM  
THAT HAS TO BE SOLVED;  
EXIT: THE CALCULATED SOLUTION OF THE SYSTEM;  
F: <ARRAY IDENTIFIER>;  
"ARRAY" F[1:N];  
EXIT: THE VALUES OF THE FUNCTION COMPONENTS AT THE  
CALCULATED SOLUTION;
```

```

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

```

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

PROCEDURES USED:

JACOBNBDF = CP34439,
 QUANEWBND = CP34430.

EXECUTION FIELD LENGTH:

QUANEWBND1 DECLARES AN AUXILIARY ARRAY OF DIMENSION ONE AND ORDER
 $N + (N - 1) * (LW + RW)$.

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

LANGUAGE: ALGOL 60.

METHOD AND PERFORMANCE:

QUANEWBND1 USES JACOBNBDF (SECTION 4.3.2.1) TO CALCULATE AN
 INITIAL APPROXIMATION OF THE JACOBIAN MATRIX AT THE INITIAL GUESS
 GIVEN IN X[1:N] AND SOLVES THE NONLINEAR SYSTEM BY CALLING
 QUANEWBND (THIS SECTION).

EXAMPLE OF USE:

```

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

```

```

"BEGIN"
  "PROCEDURE" QUANWBND1(N, L, R, X, F, FU, I, D); "CODE" 34431;
  "BOOLEAN" "PROCEDURE" FUN(N, L, U, X, F); "VALUE" N, L, U;
  "INTEGER" N, L, U; "ARRAY" X, F;
  "BEGIN" "INTEGER" I; "REAL" X1, X2, X3;
    X1:= "IF" L = 1 "THEN" 0 "ELSE" X[L - 1]; X2:= X[L];
    X3:= "IF" L = N "THEN" 0 "ELSE" X[L + 1];
    "FOR" I:= L "STEP" 1 "UNTIL" U "DO"
      "BEGIN" F[I]:= (3 - 2 * X2) * X2 + 1 - X1 - X3 * 2;
        X1:= X2; X2:= X3;
        X3:= "IF" 1 <= N - 2 "THEN" X[I + 2] "ELSE" 0
      "END"; FUN:= "TRUE"
  "END" FUN;

  "INTEGER" I; "ARRAY" X, F[1:600], IN[0:5], OUT[1:5];
  "FOR" I := 1 "STEP" 1 "UNTIL" 600 "DO" X[I]:= -1;
  IN[0]:= "-14; IN[1]:= IN[2]:= IN[3]:= "-6; IN[4]:= 2000;
  IN[5]:= 0.001;
  QUANWBND1(600, 1, 1, X, F, FUN, IN, OUT);
  OUTPUT(71, "(//, "(NORM RESIDUAL VECTOR: ")"+.15D"+3D,/,
  "( LENGTH OF LAST STEP: ")"+.15D"+3D,/,
  "( NUMBER OF FUNCTION COMPONENT EVALUATIONS: )"5ZD,/,
  "( NUMBER OF ITERATIONS: )"4ZD/, "(REPORT: )"WD/)",
  OUT[2], OUT[1], OUT[3], OUT[4], OUT[5])
"END"

```

RESULTS:

```

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

```

REFERENCES:

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

SOURCE TEXT(S):

```

"CODE" 34430;
  "PROCEDURE" QUANEWBND(N, LW, RW, X, F, JAC, FUNCT, IN, OUT);
  "VALUE" N, LW, RW; "INTEGER" N, LW, RW;
  "ARRAY" X, F, JAC, IN, OUT; "BOOLEAN" "PROCEDURE" FUNCT;
  "BEGIN" "INTEGER" L, IT, FCNT, FMAX, ERR, B;
    "REAL" MACHEPS, RELTOL, ABSTOL, TOLRES, ND, MZ, RES;
    "ARRAY" DELTAC1:NJ;

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

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

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

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

    "PROCEDURE" UPDATE JAC;
    "BEGIN" "INTEGER" I, J, K, R, M; "REAL" MUL, CRIT;
      "ARRAY" PP, S[1:N];
      CRIT:= ND * MZ;
      "FOR" I:= 1 "STEP" 1 "UNTIL" N "DO" PP[I]:= DELTAC1I ** 2;
      R:= 1; K:= 1; M:= RW + 1;
      "FOR" I:= 1 "STEP" 1 "UNTIL" N "DO"
        "BEGIN" MUL:= 0; "FOR" J:= R "STEP" 1 "UNTIL" M "DO"
          MUL:= MUL + PP[IJ]; J:= R - K;
          "IF" ABS(MUL) > CRIT "THEN"
            ELMVEC(K, M - J, J, JAC, DELTA, F[I] / MUL); K:= K + B;
          "IF" I > LW "THEN" R:= R + 1 "ELSE" K:= K - 1;
          "IF" M < N "THEN" M:= M + 1
        "END"
    "END" UPDATEJAC;

```

```

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

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

```

AUTHOR : J. C. P. BUS

INSTITUTE : MATHEMATICAL CENTRE.

RECEIVED : 741101.

BRIEF DESCRIPTION :

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

KEYWORDS :

MINIMIZATION,
FUNCTIONS OF ONE VARIABLE.

CALLING SEQUENCE :

THE HEADING OF THIS PROCEDURE IS :
"REAL" "PROCEDURE" MININ(X, A, B, FX, TOLX);
"REAL" X, A, B, FX, TOLX;
"CODE" 34433;

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

THE MEANING OF THE FORMAL PARAMETERS IS :

X : <REAL VARIABLE>;
A JENSEN VARIABLE; THE ACTUAL PARAMETERS FOR FX AND TOLX
DEPEND ON X;
EXIT : THE CALCULATED APPROXIMATION OF THE POSITION OF THE
MINIMUM;
A, B : <REAL VARIABLE>;
ENTRY : THE ENDPOINTS OF THE INTERVAL ON WHICH A MINIMUM
IS SEARCHED FOR;
EXIT : THE ENDPOINTS OF THE INTERVAL WITH LENGTH LESS THAN
 $4 * TOL(X)$ SUCH THAT $A < X < B$;
FX : <ARITHMETIC EXPRESSION>;
THE FUNCTION IS GIVEN BY THE ACTUAL PARAMETER FX, WHICH
DEPENDS ON X;
TOLX : <ARITHMETIC EXPRESSION>;
THE TOLERANCE IS GIVEN BY THE ACTUAL PARAMETER TOLX, WHICH MAY
DEPEND ON X; A SUITABLE TOLERANCE FUNCTION IS : $ABS(X)*RE + AE$,
WHERE RE IS THE RELATIVE PRECISION DESIRED AND AE IS AN ABSOLUTE
PRECISION WHICH SHOULD NOT BE CHOSEN EQUAL TO ZERO.

DATA AND RESULTS :

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

PROCEDURES USED : NONE.

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

METHOD AND PERFORMANCE :

MININ IS A SLIGHTLY MODIFIED VERSION OF THE ALGORITHM GIVEN IN [1].

EXAMPLE OF USE:

THE FOLLOWING PROGRAM MAY BE USED TO CALCULATE THE MINIMUM OF THE FUNCTION $F(X) = \sum_{I=1}^{20} ((I^2 - 5)/(X - I^2))^2$; $I = 1(1)20$ ON THE INTERVAL $[1 + TOL, 4 - TOL]$ (SEE [1]).

```
"BEGIN"
  "REAL" "PROCEDURE" MININ(X, A, B, FX, TOLX); "CODE" 34433;
  "REAL" M, X, A, B; "INTEGER" CNT;
  "REAL" "PROCEDURE" F(X); "VALUE" X; "REAL" X;
  "BEGIN" "INTEGER" I; "REAL" S;
    S:= 0; "FOR" I:= 1 "STEP" 1 "UNTIL" 20 "DO"
      S:= S + ((I * 2 - 5) / (X - I ** 2)) ** 2;
    CNT:= CNT + 1; F:= S
  "END" F;
  "REAL" "PROCEDURE" TOL(X); "VALUE" X; "REAL" X;
  TOL:= ABS(X) * "-7 + "-7;
  A:= 1 + TOL(1); B:= 4 - TOL(4); CNT:= 0;
  M:= MININ(X, A, B, F(X), TOL(X));
  OUTPUT(61, "("4B, "("MINIMUM IS ")", N, /4B,
    "("FOR X IS ")", N, /4B,
    "("IN THE INTERVAL WITH ENDPOINTS ")", /8B, 2(N), /4B,
    "("THE NUMBER OF FUNCTION EVALUATIONS NEEDED IS ")", 2ZD, /)"",
  M, X, A, B, CNT)
"END"
```


RESULTS :

MINIMUM IS +3.6766990169019"+000
 FOR X IS +3.0229153397991"+000
 IN THE INTERVAL WITH ENDPNTS
 +3.0229149365075"+000 +3.0229157410906"+000
 THE NUMBER OF FUNCTION EVALUATIONS NEEDED IS 13

SOURCE TEXT:

```

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

```


AUTHOR: J. C. P. BUS.

INSTITUTE: MATHEMATICAL CENTRE.

RECEIVED: 741101.

BRIEF DESCRIPTION:

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

KEYWORDS :

MINIMIZATION,
FUNCTIONS OF ONE VARIABLE.

CALLING SEQUENCE:

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

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

THE MEANING OF THE FORMAL PARAMETERS IS:

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

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

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

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

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

DATA AND RESULTS:

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

PROCEDURES USED: NONE.

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

LANGUAGE: ALGOL 60.

METHOD AND PERFORMANCE:

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

REFERENCES:

- [1]: BRENT, R.P.
ALGORITHMS FOR MINIMIZATION WITHOUT DERIVATIVES. CH.5.
PRENTICE HALL, 1973.
- [2]: DAVIDON, W.C.
VARIABLE METRIC METHODS FOR MINIMIZATION.
REP. A.N.L. 5990, 1959.

EXAMPLE OF USE:

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

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

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

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

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

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

RESULTS:

```
MINIMUM IS +3.6766990169021"+000
FOR X IS +3.0229155250302"+000
AND Y IS +3.0229151227386"+000
THE NUMBER OF FUNCTION EVALUATIONS NEEDED IS 9
```

SOURCE TEXT(S):

```

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

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

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

```

AUTHOR: J.C.P.BUS.

INSTITUTE: MATHEMATICAL CENTRE.

RECEIVED: 730620.

BRIEF DESCRIPTION:

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

KEYWORDS:

AUXILIARY PROCEDURE.

SUBSECTION: LINEMIN.

CALLING SEQUENCE:

THE HEADING OF THIS AUXILIARY PROCEDURE IS:
"PROCEDURE" LINEMIN(N, X, D, ND, ALFA, G, FUNCT, FO, F1, DFO, DF1,
EVLMAX, STRONGSEARCH, IN);
"VALUE" N, ND, FO, DFO, STRONGSEARCH;
"INTEGER" N, EVLMAX; "BOOLEAN" STRONGSEARCH;
"REAL" ND, ALFA, FO, F1, DFO, DF1;
"ARRAY" X, D, G, IN; "REAL" "PROCEDURE" FUNCT;"CODE" 34210;

THE MEANING OF THE FORMAL PARAMETERS IS:

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

```

G:      <ARRAY IDENTIFIER>;
        "ARRAY" G[1 : N];
        EXIT: THE GRADIENT OF F AT THE CALCULATED APPROXIMATION
              OF THE MINIMUM;
FUNCT:  <PROCEDURE IDENTIFIER>;
        THE HEADING OF THIS PROCEDURE SHOULD BE:
        "REAL" "PROCEDURE" FUNCT(N, X, G); "VALUE" N;
        "INTEGER" N; "ARRAY" X, G;
        A CALL OF FUNCT SHOULD EFFECTUATE :
        1: FUNCT:= F(X);
        2: THE VALUE OF G[I], (I = 1, ..., N), BECOMES THE VALUE
           OF THE I - TH COMPONENT OF THE GRADIENT OF F AT X;
FO:     <ARITHMETIC EXPRESSION>;
        ENTRY: THE VALUE OF H(0), (SEE ALFA);
F1:     <VARIABLE>;
        ENTRY: THE VALUE OF H(ALFA0);
        EXIT: THE VALUE OF H(ALFAM), (SEE ALFA);
DFO:    <ARITHMETIC EXPRESSION>;
        ENTRY: THE VALUE OF THE DERIVATIVE OF H AT ALFA = 0;
DF1:    <VARIABLE>;
        ENTRY: THE VALUE OF THE DERIVATIVE OF H AT ALFA = ALFA0;
        EXIT: THE VALUE OF THE DERIVATIVE OF H AT ALFA = ALFAM;
EVLMAX: <VARIABLE>;
        ENTRY: THE MAXIMUM ALLOWED NUMBER OF CALLS OF FUNCT;
        EXIT: THE NUMBER OF TIMES FUNCT HAS BEEN CALLED;
STRONGSEARCH:
        <BOOLEAN EXPRESSION>;
        IF THE VALUE OF STRONGSEARCH IS TRUE, THEN THE PROCESS
        MAKES USE OF TWO STOPPING CRITERIA:
        A: THE NUMBER OF TIMES FUNCT HAS BEEN CALLED EXCEEDS THE
           GIVEN VALUE OF EVLMAX;
        B: AN INTERVAL IS FOUND WITH LENGTH LESS THAN TWO TIMES
           THE PRESCRIBED PRECISION, ON WHICH A MINIMUM IS EXPECTED;
        IF THE VALUE OF STRONGSEARCH IS FALSE, THE PROCESS MAKES
        ALSO USE OF A THIRD STOPPING CRITERION :
        C:  $\mu \leq (H(\text{ALFAK}) - H(\text{ALFA0})) / (\text{ALFAK} * \text{DFO}) \leq 1 - \mu$ ,
           WHEREBY ALFAK IS THE CURRENT ITERATE AND MU A
           PRESCRIBED CONSTANT;
IN:     <ARRAY IDENTIFIER>;
        ENTRY:
        "ARRAY" IN[1:3];
        IN[1]: THE RELATIVE PRECISION, EPSR, NECESSARY FOR THE
              STOPPING CRITERION B, (SEE STRONGSEARCH);
        IN[2]: THE ABSOLUTE PRECISION, EPSA, NECESSARY FOR THE
              STOPPING CRITERION B, (SEE STRONGSEARCH);
        THE PRESCRIBED PRECISION, EPS, AT ALFA = ALFAK IS GIVEN BY:
         $\text{EPS} = \text{NORM} ( X_0 + \text{ALPHA} * D ) * \text{EPSR} + \text{EPSA}$ , WHERE
         $\text{NORM} ( . )$  DENOTES THE EUCLIDEAN NORM.
        IN[3]: THE PARAMETER MU NECESSARY FOR STOPPING CRITERION C;
              THIS PARAMETER MUST SATISFY:  $0 < \mu < 0.5$ ; IN
              PRACTICE, A CHOICE OF  $\mu = 0.0001$  IS ADVISED.

```


DATA AND RESULTS:

LINEMIN CALCULATES AN APPROXIMATION OF A MINIMUM OF A HIGHER - DIMENSIONAL FUNCTION ON A GIVEN LINE; THE QUANTITY DFO MUST SATIFY: $DFO < 0$; IF MOREOVER $Df1 > 0$, THEN THE PROCEDURE WILL YIELD A RESULT THAT SATISFIES ONE OF THE CHOSEN STOPPING CRITERIA, (SEE STRONGSEARCH), OTHERWISE WE CAN NOT GUARANTEE SUCH A RESULT.

PROCEDURES USED:

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

REQUIRED CENTRAL MEMORY:
N WORDS.

METHOD AND PERFORMANCE:

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

REFERENCES:

- [1] BUS, J. C. P.
MINIMIZATION OF FUNCTIONS OF SEVERAL VARIABLES (DUTCH).
MATHEMATICAL CENTRE, AMSTERDAM, NR 29/72 (1972).
- [2] DAVIDON, W. C.
VARIABLE METRIC METHOD FOR MINIMIZATION.
ARGONNE NAT. LAB. REPORT, ANL 5990 (1959).
- [3] FLETCHER, R.
A NEW APPROACH TO VARIABLE METRIC ALGORITHMS.
COMP. J. 6, (1963), P.163 - 168.
- [4] GOLDSTEIN, A. A. AND PRICE, J. F.
AN EFFECTIVE ALGORITHM FOR MINIMIZATION.
NUMER. MATH. 10, (1967), P.184 - 189.

SUBSECTION: RNK1UPD.

CALLING SEQUENCE:

THE HEADING OF THIS AUXILIARY PROCEDURE IS:
"PROCEDURE" RNK1UPD(H, N, V, C); "VALUE" N, C;
"INTEGER" N; "REAL" C; "ARRAY" H, V;
"CODE" 34211;

THE MEANING OF THE FORMAL PARAMETERS IS:

N: <ARITHMETIC EXPRESSION>;
THE ORDER OF THE SYMMETRIC MATRIX, WHOSE UPPERTRIANGLE IS
STORED COLUMNWISE IN THE ONE - DIMENSIONAL ARRAY H;
C: <ARITHMETIC EXPRESSION>;
SEE V;
V: <ARRAY IDENTIFIER>;
"ARRAY" V[1 : N];
THE GIVEN MATRIX IS UPDATED (ANOTHER MATRIX IS ADDED TO IT)
WITH A SYMMETRIC MATRIX, U, OF RANK ONE, DEFINED BY:
$$U_{I,J} = C * V_{I1} * V_{J1};$$

H: <ARRAY IDENTIFIER>;
"ARRAY" H[1 : N * (N + 1) // 2];
ENTRY: THE UPPERTRIANGLE (STORED COLUMNWISE, I.E. :
$$A_{I,J} = H[(J-1)*J//2+I], 1 \leq I \leq J \leq N$$

OF THE SYMMETRIC MATRIX THAT HAS TO BE UPDATED;
EXIT: THE UPPERTRIANGLE (STORED COLUMNWISE) OF THE
UPDATED MATRIX.

PROCEDURES USED:

ELMVEC = CP34020.

REQUIRED CENTRAL MEMORY:

NO AUXILIARY ARRAYS ARE DECLARED IN RNK1UPD.

SUBSECTION: DAVUPD.

CALLING SEQUENCE:

THE HEADING OF THIS AUXILIARY PROCEDURE IS:
"PROCEDURE" DAVUPD(H, N, V, W, C1, C2); "VALUE" N, C1, C2;
"INTEGER" N; "REAL" C1, C2; "ARRAY" H, V, W;
"CODE" 34212;

THE MEANING OF THE FORMAL PARAMETERS IS:

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

PROCEDURES USED: NONE.

REQUIRED CENTRAL MEMORY:

NO AUXILIARY ARRAYS ARE DECLARED IN DAVUPD.

SUBSECTION: FLEUPD.

CALLING SEQUENCE:

THE HEADING OF THIS AUXILIARY PROCEDURE IS:
 "PROCEDURE" FLEUPD(H, N, V, W, C1, C2); "VALUE" N, C1, C2;
 "INTEGER" N; "REAL" C1, C2; "ARRAY" H, V, W;
 "CODE" 34213;

THE MEANING OF THE FORMAL PARAMETERS IS:

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

PROCEDURE USED: NONE.

REQUIRED CENTRAL MEMORY:

NO AUXILIARY ARRAYS ARE DECLARED IN FLEUPD.

SOURCE TEXT(S):

```

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

  RELTOL:= INC1; ABSTOL:= INC2; MU:= INC3; EVL:= 0;
  ALFAO:= 0; OLDF:= FO; OLDDF:= DFO; Y:= ALFA; NOTININT:= "TRUE";
  DUPVEC(1, N, 0, XO, X);
  EPS:= (SQRT(VECVEC(1, N, 0, X, X)) * RELTOL + ABSTOL) / ND;
  Q:= (F1 - FO) / (ALFA * DFO);
INT: "IF" NOTININT "THEN" NOTININT:= DF1 < 0 "AND" Q > MU;
  AID:= ALFA; "IF" DF1 >= 0 "THEN"
  "BEGIN" Z:= 3 * (OLDF - F1) / ALFA + OLDDF + DF1;
    W:= SQRT(Z ** 2 - OLDDF * DF1);
    ALFA:= ALFA * (1 - (DF1 + W - Z) / (DF1 - OLDDF + W * 2));
    "IF" ALFA < EPS "THEN" ALFA:= EPS "ELSE"
    "IF" AID - ALFA < EPS "THEN" ALFA:= AID - EPS
  "END" CUBIC INTERPOLATION
  "ELSE" "IF" NOTININT "THEN"
  "BEGIN" ALFAO:= ALFA:= Y; OLDDF:= DF1; OLDF:= F1 "END"
  "ELSE" ALFA:= 0.5 * ALFA; Y:= ALFA + ALFAO;
  DUPVEC(1, N, 0, X, XO); ELMVEC(1, N, 0, X, D, Y);
  EPS:= (SQRT(VECVEC(1, N, 0, X, X)) * RELTOL + ABSTOL) / ND;
  F:= FUNCT(N, X, G); EVL:= EVL + 1; DF:= VECVEC(1, N, 0, D, G);
  Q:= (F - FO) / (Y * DFO);
  "IF" ("IF" NOTININT "OR" STRONGSEARCH "THEN" "TRUE" "ELSE"
  Q < MU "OR" Q > 1 - MU) "AND" EVL < EVLMAX "THEN"
  "BEGIN" "IF" NOTININT "OR" DF > 0 "OR" Q < MU "THEN"
    "BEGIN" DF1:= DF; F1:= F "END"
    "ELSE"
    "BEGIN" ALFAO:= Y; ALFA:= AID - ALFA; OLDDF:= DF; OLDF:= F
    "END";
    "IF" ALFA > EPS * 2 "THEN" "GOTO" INT
  "END";
  ALFA:= Y; EVLMAX:= EVL; DF1:= DF; F1:= F
"END" LINEMIN;
"END"

```

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

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

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

AUTHOR: J.C.P. BUS.

INSTITUTE: MATHEMATICAL CENTRE.

RECEIVED: 741101.

BRIEF DESCRIPTION:

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

KEYWORDS:

MINIMIZATION,
FUNCTION OF SEVERAL VARIABLES.

CALLING SEQUENCE:

THE HEADING OF THIS PROCEDURE IS:
"PROCEDURE" PRAXIS(N, X, FUNCT, IN, OUT); "VALUE" N;
"INTEGER" N; "ARRAY" X, IN, OUT; "REAL" "PROCEDURE" FUNCT; "CODE" 34432;

THE MEANING OF THE FORMAL PARAMETERS IS:

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

FUNCT SHOULD DELIVER THE VALUE OF THE FUNCTION TO BE
MINIMIZED, AT THE POINT GIVEN BY X[1:N];

THE MEANING OF THE FORMAL PARAMETERS IS:

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

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

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

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

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

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

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

INC[9]: IF THE PROBLEM IS KNOWN TO BE ILL-CONDITIONED (SEE [1]), THEN THE VALUE OF INC[9] SHOULD BE NEGATIVE, OTHERWISE $INC[9] \geq 0$;

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

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

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

PROCEDURES USED:

INIVEC	=	CP31010,
INIMAT	=	CP31011,
DUPVEC	=	CP31030,
DUPMAT	=	CP31035,
DUPCOLVEC	=	CP31034,
MULROW	=	CP31021,
MULCOL	=	CP31022,
VECVEC	=	CP34010,
TAMMAT	=	CP34014,
MATTAM	=	CP34015,
ICHROWCOL	=	CP34033,
ELMVECCOL	=	CP34021,
QRISNGVALDEC	=	CP34273,
SETRANDOM	=	CP11014,
RANDOM	=	CP11015,
DWARF	=	CP30003.

REQUIRED CENTRAL MEMORY:

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

RUNNING TIME:

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

METHOD AND PERFORMANCE:

THIS PROCEDURE IS ADOPTED FROM [1].

REFERENCES:

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

SOURCE TEXT(S):

```

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

  "PROCEDURE" INIVEC(L, U, A, X); "CODE" 31010;
  "PROCEDURE" INIMAT(L, U, K, V, A, X); "CODE" 31011;
  "PROCEDURE" DUPVEC(L, U, K, A, X); "CODE" 31030;
  "PROCEDURE" DUPMAT(L, U, K, V, A, B); "CODE" 31035;
  "PROCEDURE" DUPCOLVEC(L, U, K, A, B); "CODE" 31034;
  "PROCEDURE" MULROW(L, U, I, J, A, B, X); "CODE" 31021;
  "PROCEDURE" MULCOL(L, U, I, J, A, B, X); "CODE" 31022;
  "REAL" "PROCEDURE" VECVEC(L, U, S, A, B); "CODE" 34010;
  "REAL" "PROCEDURE" TAMMAT(L, U, I, J, A, B); "CODE" 34014;
  "REAL" "PROCEDURE" MATTAM(L, U, I, J, A, B); "CODE" 34015;
  "PROCEDURE" ICHROWCOL(L, U, I, J, A); "CODE" 34033;
  "PROCEDURE" ELHVECCOL(L, U, I, A, B, X); "CODE" 34021;
  "INTEGER" "PROCEDURE" QRISNGVALDEC(A, M, N, VAL, V, EM); "CODE" 34273;
  "PROCEDURE" SETRANDOM(X); "CODE" 11014;
  "REAL" "PROCEDURE" RANDOM; "CODE" 11015;
  "REAL" "PROCEDURE" DWARF; "CODE" 30003;

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

```

```
"PROCEDURE" MIN(J, NITS, D2, X1, F1, FK); "VALUE" J, NITS, FK;
"INTEGER" J, NITS; "REAL" D2, X1, F1; "BOOLEAN" FK;
"BEGIN"
```

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

```
  "INTEGER" K; "BOOLEAN" DZ;
  "REAL" X2, XM, FO, F2, FM, D1, T2, S, SF1, SX1;
  SF1:= F1; SX1:= X1;
  K:= 0; XM:= 0; FO:= FM; DZ:= D2 < RELTOL;
  S:= SQRT(VECVEC(1,N,0,X,X));
  T2:= M4 * SQRT(ABS(FX) / ("IF" DZ "THEN" DMIN "ELSE" D2)
  + S * LDT) + M2 * LDT; S:= S * M4 + ABSTOL;
  "IF" DZ "AND" T2 > S "THEN" T2:= S;
  "IF" T2 < SMALL "THEN" T2:= SMALL;
  "IF" T2 > 0.01 * H "THEN" T2:= 0.01 * H;
  "IF" FK "AND" F1 < FM "THEN"
    "BEGIN" XM:= X1; FM:= F1 "END";
  "IF" ^ FK "OR" ABS(X1) < T2 "THEN"
    "BEGIN" X1:= "IF" X1 > 0 "THEN" T2 "ELSE" -T2;
      F1:= FLIN(X1)
    "END";
```

```
  "IF" F1 < FM "THEN"
    "BEGIN" XM:= X1; FM:= F1 "END";
  LO: "IF" DZ "THEN"
    "BEGIN" "COMMENT" EVALUATE FLIN AT ANOTHER POINT
      AND ESTIMATE THE SECOND DERIVATIVE;
      X2:= "IF" FO < F1 "THEN" -X1 "ELSE" X1 * 2;
      F2:= FLIN(X2); "IF" F2 < FM "THEN"
        "BEGIN" XM:= X2; FM:= F2 "END";
      D2:= (X2*(F1-FO)-X1*(F2-FO))/(X1*X2*(X1-X2))
    "END";
    "COMMENT" ESTIMATE FIRST DERIVATIVE AT 0;
    D1:= (F1-FO)/X1-X1*D2; DZ:= "TRUE";
    X2:= "IF" D2 < SMALL "THEN"
      ("IF" D1 < 0 "THEN" H "ELSE" -H)
    "ELSE" -0.5*D1/D2;
    "IF" ABS(X2) > H "THEN" X2:= "IF" X2 > 0 "THEN" H "ELSE" -H;
```

```
"COMMENT"
```

```

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

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

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

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

```

"COMMENT"

```

ABSTOL:=T2:= SMALL+ABS(ABSTOL); DMIN:= SMALL;
"IF" H<ABSTOL*100"THEN"H:=ABSTOL*100; LDT:=H;
INIMAT(1,N,1,N,V,0);
"FOR" I:=1"STEP"1"UNTIL"N"DO"V[I,I]:= 1;
D[I]:= QDO:= 0; DUPVEC(1,N,0,Q1,X);
INIVEC(1,N,Q0,0);

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

```

```

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

```


AUTHOR: J.C.P.BUS.

INSTITUTE: MATHEMATICAL CENTRE.

RECEIVED: 730620.

BRIEF DESCRIPTION:

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

KEYWORDS:

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

SUBSECTION: RNK1MIN.

CALLING SEQUENCE:

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

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

THE MEANING OF THE FORMAL PARAMETERS IS:

N: <ARITHMETIC EXPRESSION>;
 THE NUMBER OF VARIABLES OF THE FUNCTION TO BE MINIMIZED;
 X: <ARRAY IDENTIFIER>;
 "ARRAY" X[1 : N];
 THE INDEPENDENT VARIABLES;
 ENTRY: AN APPROXIMATION OF A MINIMUM OF THE FUNCTION;
 EXIT: THE CALCULATED MINIMUM OF THE FUNCTION;
 G: <ARRAY IDENTIFIER>;
 "ARRAY" G[1 : N];
 EXIT: THE GRADIENT OF THE FUNCTION AT THE CALCULATED
 MINIMUM;
 H: <ARRAY IDENTIFIER>;
 "ARRAY" H[1 : N * (N + 1) // 2];
 THE UPPERTRIANGLE OF AN APPROXIMATION OF THE INVERSE
 HESSIAN IS STORED COLUMNWISE IN H (I.E. THE I,J-TH ELEMENT=
 H((J - 1) * J // 2 + I), 1 <= I <= J <= N);
 IF INC6] > 0 INITIALIZING OF H WILL BE DONE AUTOMATICALLY
 AND THE INITIAL APPROXIMATION OF THE INVERSE HESSIAN WILL
 EQUAL THE UNITMATRIX MULTIPLIED WITH THE VALUE OF INC6];
 IF INC6] < 0 NO INITIALIZING OF H WILL BE DONE AND THE USER
 SHOULD GIVE IN H AN APPROXIMATION OF THE INVERSE HESSIAN,
 AT THE STARTING POINT; THE UPPERTRIANGLE OF AN APPROXIMATION
 OF THE INVERSE HESSIAN AT THE CALCULATED MINIMUM IS
 DELIVERED IN H;
 FUNCT: <PROCEDURE IDENTIFIER>;
 THE HEADING OF THIS PROCEDURE SHOULD BE:
 "REAL" "PROCEDURE" FUNCT(N, X, G); "VALUE" N;
 "INTEGER" N; "ARRAY" X, G;
 A CALL OF FUNCT MUST EFFECTUATE IN:
 1: FUNCT BECOMES THE VALUE OF THE FUNCTION TO BE MINIMIZED
 AT THE POINT X;
 2: THE VALUE OF G[I], (I = 1, ..., N), BECOMES THE VALUE
 OF THE I - TH COMPONENT OF THE GRADIENT OF THE FUNCTION
 AT X;

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

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

DATA AND RESULTS:

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

PROCEDURES USED:

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

REQUIRED CENTRAL MEMORY:

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

RUNNING TIME:

DEPENDS STRONGLY ON THE PROBLEM TO BE SOLVED;

LANGUAGE: ALGOL 60.

METHOD AND PERFORMANCE:

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

SUBSECTION: FLEMIN.

CALLING SEQUENCE:

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

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

THE MEANING OF THE FORMAL PARAMETERS IS:

N: <ARITHMETIC EXPRESSION>;
 THE NUMBER OF VARIABLES OF THE FUNCTION TO BE MINIMIZED;

X: <ARRAY IDENTIFIER>;
 "ARRAY" X[1 : N];
 THE INDEPENDENT VARIABLES;
 ENTRY: AN APPROXIMATION OF A MINIMUM OF THE FUNCTION;
 EXIT: THE CALCULATED MINIMUM OF THE FUNCTION;

G: <ARRAY IDENTIFIER>;
 "ARRAY" G[1 : N];
 EXIT: THE GRADIENT OF THE FUNCTION AT THE CALCULATED
 MINIMUM;

H: <ARRAY IDENTIFIER>;
 "ARRAY" H[1 : N * (N + 1) // 2];
 THE UPPERTRIANGLE OF AN APPROXIMATION OF THE INVERSE
 HESSIAN IS STORED COLUMNWISE IN H (I.E. THE I, J-TH ELEMENT =
 H[(J - 1) * J // 2 + I], 1 <= I <= J <= N);
 IF IN[6] > 0 INITIALIZING OF H WILL BE DONE AUTOMATICALLY
 AND THE INITIAL APPROXIMATION OF THE INVERSE HESSIAN WILL
 EQUAL THE UNITMATRIX MULTIPLIED WITH THE VALUE OF IN[6]; IF
 IN[6] < 0 NO INITIALIZING OF H WILL BE DONE AND THE USER
 SHOULD GIVE IN H AN APPROXIMATION OF THE INVERSE HESSIAN
 AT THE STARTING POINT;
 THE UPPERTRIANGLE OF AN APPROXIMATION OF THE INVERSE
 HESSIAN AT THE CALCULATED MINIMUM IS DELIVERED IN H;

FUNCT: <PROCEDURE IDENTIFIER>;
 THE HEADING OF THIS PROCEDURE SHOULD BE :
 "REAL" "PROCEDURE" FUNCT(N, X, G); "VALUE" N;
 "INTEGER" N; "ARRAY" X, G;
 A CALL OF FUNCT SHOULD EFFECTUATE IN:
 1: FUNCT BECOMES THE VALUE OF THE FUNCTION TO BE MINIMIZED
 AT THE POINT X;
 2: THE VALUE OF G[I], (I = 1, ..., N), BECOMES THE VALUE
 OF THE I - TH COMPONENT OF THE GRADIENT OF THE FUNCTION
 AT X;

```

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

```

DATA AND RESULTS:

USUALLY THE CALCULATED SOLUTION WILL SATISFY:

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

PROCEDURES USED:

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

REQUIRED CENTRAL MEMORY:

EXECUTION FIELD LENGTH: $3 * N + 19$ WORDS.

RUNNING TIME:

DEPENDS STRONGLY ON THE PROBLEM TO BE SOLVED;

LANGUAGE: ALGOL 60.

METHOD AND PERFORMANCE:

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

REFERENCES:

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

EXAMPLE OF USE:

THE MINIMUM OF THE FUNCTION:

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

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

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

DELIVERS:

LEAST VALUE: +.200699798801180"-018

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

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

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

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

LEAST VALUE: +.811973499921290"-016

X: +.99999999758770"+000 +.999999998616780"+000

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

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

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

SOURCE TEXT(S):

```

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

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

```

```

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

```

```

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

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

```

```

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

```


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

CONTRIBUTORS: B. VAN DOMSELAAR, J. KOK.

INSTITUTE: MATHEMATICAL CENTRE.

RECEIVED: 741101.

BRIEF DESCRIPTION:

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

KEYWORDS:

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

SUBSECTION: MARQUARDT.

CALLING SEQUENCE:

THE HEADING OF THIS PROCEDURE IS:

```
"PROCEDURE" MARQUARDT(M, N, PAR, RV, JJINV, FUNCT, JACOBIAN, IN,
  OUT); "VALUE" M, N; "INTEGER" M, N;
"ARRAY" PAR, RV, JJINV, IN, OUT; "BOOLEAN" "PROCEDURE" FUNCT;
"PROCEDURE" JACOBIAN;
"CODE" 34440;
```

THE MEANING OF THE FORMAL PARAMETERS IS:

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


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

JACOBIAN: <PROCEDURE IDENTIFIER>;
 THE HEADING OF THIS PROCEDURE SHOULD BE:
 "PROCEDURE" JACOBIAN(M, N, PAR, RV, JAC, LOCFUNCT);
 "VALUE" M, N; "INTEGER" M, N; "ARRAY" PAR, RV, JAC;
 "PROCEDURE" LOCFUNCT;
 ENTRY: M, N, PAR, RV, LOCFUNCT;
 FOR M, N, PAR SEE: FUNCT;
 RV CONTAINS THE RESIDUAL VECTOR OBTAINED WITH THE CURRENT VALUES OF THE UNKNOWN AND SHOULD NOT BE ALTERED;
 A CALL OF LOCFUNCT(M, N, PAR, RV) IS EQUIVALENT WITH A CALL OF THE USER-DEFINED PROCEDURE FUNCT(M, N, PAR, RV), BUT, IN ADDITION, THIS CALL IS COUNTED TO THE TOTAL NUMBER OF CALLS OF FUNCT (SEE OUT[4]) AND, MOREOVER, IF FUNCT DELIVERS THE VALUE FALSE THEN THE PROCESS IS TERMINATED;

EXIT: "ARRAY" JAC[1 : M, 1 : N];
 UPON COMPLETION OF A CALL OF JACOBIAN, JAC SHOULD CONTAIN THE PARTIAL DERIVATIVES $DRV_{IJ} / DP_{AR}[J]$, OBTAINED WITH THE CURRENT VALUES OF THE UNKNOWN VARIABLES GIVEN IN PAR[1:N];

IT IS A PREREQUISITE FOR THE PROPER OPERATION OF THE PROCEDURE MARQUARDT THAT THE PRECISION OF THE ELEMENTS OF THE MATRIX JAC IS AT LEAST THE PRECISION DEFINED BY IN[3] AND IN[4];

IN: <ARRAY IDENTIFIER>;
 "ARRAY" INCO : 6];
 ENTRY: IN THIS ARRAY THE USER SHOULD GIVE SOME DATA TO CONTROL THE PROCESS;
 INCO]: THE MACHINE PRECISION;
 FOR THE CYBER 73 A SUITABLE VALUE IS 10^{-14} ;
 IN[1], IN[2] ARE NOT USED BY THE PROCEDURE MARQUARDT;
 IN[3], IN[4]:
 THE RELATIVE AND ABSOLUTE TOLERANCE FOR THE DIFFERENCE BETWEEN THE EUCLIDEAN NORM OF THE ULTIMATE AND PENULTIMATE RESIDUAL VECTOR;
 THE PROCESS IS TERMINATED IF THE IMPROVEMENT OF THE SUM OF SQUARES IS LESS THAN
 $IN[3] * (\text{SUM OF SQUARES}) + IN[4] * IN[4]$;
 THESE TOLERANCES SHOULD BE CHOSEN GREATER THAN THE CORRESPONDING ERRORS OF THE CALCULATED RESIDUAL VECTOR;
 NOTE THAT THE EUCLIDEAN NORM OF THE RESIDUAL VECTOR IS DEFINED AS THE SQUARE ROOT OF THE SUM OF SQUARES;
 IN[5]: THE MAXIMUM NUMBER OF CALLS OF FUNCT ALLOWED;

INC6]: A STARTING VALUE USED FOR THE RELATION BETWEEN THE GRADIENT AND THE GAUSS-NEWTON DIRECTION (SEE [2]); IF THE PROBLEM IS WELL CONDITIONED THEN A SUITABLE VALUE FOR INC6] WILL BE 0.01; IF THE PROBLEM IS ILL CONDITIONED THEN INC6] SHOULD BE GREATER, BUT THE VALUE OF INC6] SHOULD SATISFY: $INC0] < INC6] \leq 1/INC0]$;
 OUT: <ARRAY IDENTIFIER>; *ARRAY* OUT[1 : 7];
 EXIT : IN ARRAY OUT SOME BY-PRODUCTS ARE DELIVERED;
 OUT[1]: THIS VALUE GIVES INFORMATION ABOUT THE TERMINATION OF THE PROCESS;
 OUT[1]=0: NORMAL TERMINATION;
 OUT[1]=1: THE PROCESS HAS BEEN BROKEN OFF, BECAUSE THE NUMBER OF CALLS OF FUNCT EXCEEDED THE NUMBER GIVEN IN INC5];
 OUT[1]=2: THE PROCESS HAS BEEN BROKEN OFF, BECAUSE A CALL OF FUNCT DELIVERED THE VALUE FALSE;
 OUT[1]=3: FUNCT BECAME FALSE WHEN CALLED WITH THE INITIAL ESTIMATES OF PAR[1:N]; THE ITERATION PROCESS WAS NOT STARTED AND SO JJINV[1:N,1:N] CAN NOT BE USED;
 OUT[1]=4: THE PROCESS HAS BEEN BROKEN OFF, BECAUSE THE PRECISION ASKED FOR CAN NOT BE ATTAINED; THIS PRECISION IS POSSIBLY CHOSEN TOO HIGH, RELATIVE TO THE PRECISION IN WHICH THE RESIDUAL VECTOR IS CALCULATED (SEE INC3]);
 OUT[2]: THE EUCLIDEAN NORM OF THE RESIDUAL VECTOR CALCULATED WITH VALUES OF THE UNKNOWN DELIVERED;
 OUT[3]: THE EUCLIDEAN NORM OF THE RESIDUAL VECTOR CALCULATED WITH THE INITIAL VALUES OF THE UNKNOWN VARIABLES;
 OUT[4]: THE NUMBER OF CALLS OF FUNCT NECESSARY TO OBTAIN THE CALCULATED RESULT;
 OUT[5]: THE TOTAL NUMBER OF ITERATIONS PERFORMED; NOTE THAT IN EACH ITERATION ONE EVALUATION OF THE JACOBIAN MATRIX HAD TO BE MADE;
 OUT[6]: THE IMPROVEMENT OF THE EUCLIDEAN NORM OF THE RESIDUAL VECTOR IN THE LAST ITERATION STEP;
 OUT[7]: THE CONDITION NUMBER OF $J' * J$, I.E. THE RATIO OF ITS LARGEST TO SMALLEST EIGENVALUES;

DATA AND RESULTS:

IF THIS PROCEDURE IS USED FOR CURVE FITTING THEN THE RELATIVE ACCURACY IN THE CALCULATION OF THE RESIDUAL VECTOR DEPENDS STRONGLY ON THE ERRORS IN THE EXPERIMENTAL DATA AND THIS SHOULD BE REFLECTED IN THE PARAMETERS INC3] AND INC4];
 THE MATRIX JJINV CAN BE USED IF SOME STATISTICAL INFORMATION ABOUT THE FITTED PARAMETERS IS REQUIRED; THE STANDARD DEVIATION, COVARIANCE MATRIX AND CORRELATION MATRIX MAY BE CALCULATED EASILY FROM JJINV ;

PROCEDURES USED:

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

REQUIRED CENTRAL MEMORY:

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

RUNNING TIME:

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

LANGUAGE: ALGOL 60.

METHOD AND PERFORMANCE:

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

REFERENCES:

- [1] D. W. MARQUARDT,
AN ALGORITHM FOR LEAST-SQUARES ESTIMATION OF NONLINEAR
PARAMETERS,
J. SIAM 11 (1963), PP.431-441.
- [2] J. C. P. BUIS, B. VAN DOMSELAAR, J. KOK,
NONLINEAR LEAST SQUARES ESTIMATION,
MATHEMATICAL CENTRE, AMSTERDAM. (TO APPEAR)

EXAMPLE OF USE:

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

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

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

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

  IN(0):=" -14; IN(3):=" -4; IN(4):=" -1; IN(5):=75; IN(6):=" -2;
  "FOR" I:=1 "STEP" 1 "UNTIL" 6 "DO"
    INPUT(60,"(2(N),/)",X[I],Y[I]);
  PAR[1]:=580; PAR[2]:=" -180; PAR[3]:=" -0.160;
  MARQUARDT(6,3,PAR,RV,JJINV,EXPFUNCT,JACOBIAN,IN,OUT);
  OUTPUT(61,"(3/,"(X[I] Y[I])",/6(8+D,5B,3D,0,/)2/,
    ("PARAMETERS")",/3(+0.3D+ZD,/)2/,("OUT:"),/7(5B+D.6D"+ZD,
    /)2/,("LAST RESIDUAL VECTOR")",/6(6B+3Z.0,/)")",X[1],Y[1],
    X[2],Y[2],X[3],Y[3],X[4],Y[4],X[5],Y[5],X[6],Y[6],PAR[1],PAR[2],
    PAR[3],OUT[7],OUT[2],OUT[6],OUT[3],OUT[4],OUT[5],OUT[1],RV[1],
    RV[2],RV[3],RV[4],RV[5],RV[6])
  "END"
  "EOP"
```

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

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

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

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

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

SUBSECTION : GSSNEWTON.

CALLING SEQUENCE :

THE HEADING OF THE PROCEDURE READS :

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

THE MEANING OF THE FORMAL PARAMETERS IS :

```
M      : <ARITHMETIC EXPRESSION>;
        : THE NUMBER OF EQUATIONS;
N      : <ARITHMETIC EXPRESSION>;
        : THE NUMBER OF UNKNOWN IN THE M EQUATIONS (N <= M);
PAR    : <ARRAY IDENTIFIER>; "ARRAY" PAR[1 : N];
        : THE UNKNOWN OF THE EQUATIONS.
        : ENTRY : AN APPROXIMATION TO A LEAST SQUARES SOLUTION
        : OF THE SYSTEM.
RV     : EXIT : THE CALCULATED LEAST SQUARES SOLUTION;
        : <ARRAY IDENTIFIER>; "ARRAY" RV[1 : M];
        : EXIT : THE RESIDUAL VECTOR OF THE SYSTEM AT THE
        : CALCULATED SOLUTION;
JJINV  : <ARRAY IDENTIFIER>; "ARRAY" JJINV[1 : N,1 : N];
        : EXIT : THE INVERSE OF THE MATRIX  $J^T * J$ , WHERE J
        : IS THE JACOBIAN MATRIX AT THE SOLUTION AND  $J^T$  IS
        : J TRANSPOSED;
```

FUNCT :

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

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

ENTRY: M, N, PAR;

M, N HAVE THE SAME MEANING AS IN THE PROCEDURE
GSSNEWTON;

"ARRAY" PAR[1:N] CONTAINS THE CURRENT VALUES OF
THE UNKNOWN AND SHOULD NOT BE ALTERED.

EXIT: "ARRAY" RV[1 : M];

UPON COMPLETION OF A CALL OF FUNCT, THIS ARRAY RV
SHOULD CONTAIN THE RESIDUAL VECTOR, OBTAINED WITH
THE CURRENT VALUES OF THE UNKNOWN.

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

OTHERWISE FUNCT SHOULD DELIVER THE VALUE TRUE;

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

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

THE MEANING OF THE PARAMETERS OF JACOBIAN IS :

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

A CALL OF THE PROCEDURE JACOBIAN SHOULD DELIVER THE JACOBIAN MATRIX EVALUATED WITH THE CURRENT ESTIMATES OF THE UNKNOWN VARIABLES GIVEN IN PAR IN SUCH A WAY, THAT THE PARTIAL DERIVATIVE $D(RV)[I] / DPAR[J]$ IS DELIVERED IN $JAC[I, J], I = 1(1)M, J = 1(1)N.$

FOR THE CALCULATION OF THE DERIVATIVES ONE CAN USE THE VALUES OF THE CURRENT ESTIMATES OF THE UNKNOWNNS AS GIVEN IN PAR AND THE RESIDUAL VECTOR AS GIVEN IN RV.

ONE CAN ALSO USE THE PROCEDURE FUNCT (PARAMETER OF GSSNEWTON) THROUGH CALLS OF THE PROCEDURE LOCFUNCT (PARAMETER OF JACOBIAN). THIS PARAMETER OF JACOBIAN MAY BE USED WHEN THE JACOBIAN MATRIX IS APPROXIMATED USING (FORWARD) DIFFERENCES. AN APPROPRIATE PROCEDURE TO THIS PURPOSE IS JACOBNMF (SECTION 4.3.2.1). SUCH A PROCEDURE MAY BE USED ONLY IF THE MATRIX ELEMENTS ARE COMPUTED SUFFICIENTLY ACCURATE;

```
IN : <ARRAY IDENTIFIER>; "ARRAY" INCO : 7];
      IN THIS ARRAY TOLERANCES AND CONTROL PARAMETERS SHOULD
      BE GIVEN.
      ENTRY :
      INCO : THE MACHINE PRECISION. FOR CALCULATION ON THE
      CYBER 73 A SUITABLE VALUE IS  $10^{-14}$ .
```

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

OUT : <ARRAY IDENTIFIER>; "ARRAY" OUT[1 : 9];
 IN ARRAY OUT INFORMATION ABOUT THE TERMINATION OF THE
 PROCESS IS DELIVERED.
 EXIT :
 OUT[1] :
 THE PROCESS WAS TERMINATED BECAUSE (OUT[1] =)
 1. THE NORM OF THE RESIDUAL VECTOR IS SMALL WITH
 RESPECT TO INC[4],
 2. THE CALCULATED NEWTON STEP IS SUFFICIENTLY SMALL
 (SEE INC[1], INC[2]),
 3. THE CALCULATED STEP WAS COMPLETELY DAMPED (HALVED)
 IN INC[7] SUCCESSIVE ITERATIONS,
 4. OUT[4] EXCEEDS INC[5], THE MAXIMUM ALLOWED NUMBER OF
 CALLS OF FUNCT,
 5. THE JACOBIAN WAS NOT FULL-RANK (SEE OUT[8]),
 6. FUNCT DELIVERED FALSE AT A NEW VECTOR OF
 ESTIMATES OF THE UNKNOWNNS,
 7. FUNCT DELIVERED FALSE IN A CALL FROM JACOBIAN.

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

DATA AND RESULTS :

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

FOR A CURVE FITTING PROBLEM, SAY :

"ESTIMATE PARAMETERS $PAR[1], \dots, PAR[N]$ OF A FUNCTION

" $Y = F(X; PAR[1], \dots, PAR[N])$, WHEN A SET OF DATA $(X[I], Y[I])$,

" $I = 1(1)M$, HAS TO BE FITTED."

THE FOLLOWING SYSTEM OF M EQUATIONS IN THE N UNKNOWN PARAMETERS $PAR[1], \dots, PAR[N]$ CAN BE DERIVED :

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

PROCEDURES USED :

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

REQUIRED CENTRAL MEMORY :

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

RUNNING TIME :

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

LANGUAGE : ALGOL 60.

METHOD AND PERFORMANCE :

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

REFERENCES :

- [1] H.O. HARTLEY :
THE MODIFIED GAUSS-NEWTON METHOD.
TECHNOMETRICS, V.3 (1961), PP. 269 - 280.
- [2] H. SPAETH :
THE DAMPED TAYLOR'S SERIES METHOD FOR MINIMIZING A SUM OF
SQUARES AND FOR SOLVING SYSTEMS OF NONLINEAR EQUATIONS.
ALGORITHM 315, COLLECTED ALGORITHMS FROM CACM,
COMMUNICATIONS OF THE ACM, VOL. 10 (NOV. 1967), PP. 726 - 728.
- [3] J.C.P. BUS, B. VAN DOMSELAAR, J. KOK :
NONLINEAR LEAST SQUARES ESTIMATION.
MATHEMATICAL CENTRE (TO APPEAR).

EXAMPLE OF USE :

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

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

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

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

```

IN[0]:= "-14; IN[1]:= IN[2]:= "-6; IN[5]:= 75; IN[4]:="-10;
IN[6]:= 14; IN[7]:= 1;
"FOR" I:= 1 "STEP" 1 "UNTIL" 6 "DO"
INPUT(1, "(("2(N),/)"",X[I],Y[I]);
PAR[1]:= 580; PAR[2]:= - 180; PAR[3]:= - 0.160;
GSSNEWTON(6, 3, PAR, G, V, EXPFUNCT, JACOBIAN, IN, OUT);
OUTPUT(61, "("3/48,"("X[I] Y[I])",/, 6(5B+D, 583D.D, /), 2/,
4B("PARAMETERS")",/, 3(4B+D.3D"+ZD,/)2/, 4B("OUT:"),/,
3(9B+D.6D"+?D,/) 5(14B3ZD,/) 9B+D.6D"+ZD, 2/48,
("LAST RESIDUAL VECTOR")",/, 6(10B+3Z.D,/)")", X[1], Y[1],
X[2], Y[2], X[3], Y[3], X[4], Y[4], X[5], Y[5], X[6], Y[6], PAR[1], PAR[2],
PAR[3], OUT[6], OUT[2], OUT[3], OUT[4], OUT[5], OUT[1], OUT[7], OUT[8],
OUT[9], G[1], G[2], G[3], G[4], G[5], G[6])
"END"
"["]P"

```

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

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

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

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

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

SOURCE TEXTS :

```

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

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

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

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

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

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

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

```

"COMMENT"

;

```

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

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

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

OUT[7]:=(LAMBDA/(LAMBDA MIN+IN[0]))**2;
OUT[2]:=SQRT(FPAR);
OUT[6]:=SQRT(RES+FPAR)-OUT[2];

ESCAPE:
OUT[4]:=FE;
OUT[5]:=IT-1;
OUT[1]:=ERR
"END" MARQUARDT;
"END"

```

```

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

"BEGIN" "INTEGER" I, J, INR, MIT, TEXT,
IT, ITMAX, INRMAX, TIM, FEVAL, FEVALMAX;
"REAL" RHO, RES1, RES2, RN, RELTOLPAR, ABSTOLPAR, ABSTOLRES,
STAP, NORMX;
"BOOLEAN" CONV, TESTTHF, DAMPING ON;
"ARRAY" JAC[1:M + 1, 1:N], PR, AID, SOL[1 : N], FU2[1 : M],
AUX2 : 5];
"INTEGER""ARRAY" CI[1:N];

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

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

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

```

```

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

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

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

```


SECTION 5.2.1.1.1.1 CONTAINS NINE PROCEDURES FOR INITIAL-VALUE PROBLEMS FOR FIRST ORDER ORDINARY DIFFERENTIAL EQUATIONS.

- A. RK1 SOLVES AN IVP FOR A SINGLE ODE BY MEANS OF A 5-TH ORDER RUNGE-KUTTA METHOD.
- B. RKE SOLVES AN IVP FOR A SYSTEM OF ODE'S BY MEANS OF A 5-TH ORDER RUNGE-KUTTA METHOD.
- C. RK4A SOLVES AN IVP FOR A SINGLE ODE BY MEANS OF A 5-TH ORDER RUNGE-KUTTA METHOD. RK4A INTERCHANGES THE DEPENDENT AND INDEPENDENT VARIABLE.
- D. RK4NA SOLVES AN IVP FOR A SYSTEM OF ODE'S BY MEANS OF A 5-TH ORDER RUNGE-KUTTA METHOD. RK4NA INTERCHANGES THE DEPENDENT AND ONE INDEPENDENT VARIABLE.
- E. RK5NA SOLVES AN IVP FOR A SYSTEM OF ODE'S BY MEANS OF A 5-TH ORDER RUNGE-KUTTA METHOD. RK5NA USES THE ARC LENGTH AS INTEGRATION VARIABLE.
- F. MULTISTEP SOLVES AN IVP FOR A SYSTEM OF ODE'S BY MEANS OF A LINEAR MULTISTEP METHOD. IT USES EITHER THE BACKWARD DIFFERENTIATION METHODS, OR THE ADAMS-BASHFORTH-MOULTON-METHOD.
- G. DIFFSYS SOLVES AN IVP FOR A SYSTEM OF ODE'S BY MEANS OF A HIGH ORDER EXTRAPOLATION METHOD BASED ON THE MODIFIED MIDPOINT RULE.
- H. ARK SOLVES AN IVP FOR A LARGE SYSTEM OF ODE'S WHICH IS OBTAINED FROM SEMI-DISCRETIZATION OF AN INITIAL BOUNDARY VALUE PROBLEM FOR A PARABOLIC OR HYPERBOLIC EQUATION. ARK IS BASED ON STABILIZED, EXPLICIT RUNGE-KUTTA METHODS OF LOW ORDER.
- I. EFRK SOLVES AN IVP FOR A SYSTEM OF ODE'S BY MEANS OF AN EXPONENTIALLY FITTED EXPLICIT RUNGE-KUTTA METHOD OF FIRST, SECOND OR THIRD ORDER.

RK1 AND RKE ARE INTENDED FOR NON-STIFF EQUATIONS, WHILE RK4A, RK4NA AND RK5NA ARE INTENDED FOR NON-STIFF EQUATIONS WHERE DERIVATIVES BECOME VERY LARGE, SUCH AS IN THE NEIGHBOURHOOD OF SINGULARITIES. MULTISTEP CAN BE USED FOR NON-STIF, AS WELL AS STIFF EQUATIONS. DIFFSYS SHOULD BE USED FOR PROBLEMS FOR WHICH A VERY HIGH ACCURACY IS DESIRED. ARK IS RECOMMENDED FOR THE INTEGRATION OF SEMI-DISCRETE PARABOLIC OR HYPERBOLIC PROBLEMS. EFRK IS A SPECIAL PURPOSE PROCEDURE FOR STIFF EQUATIONS WITH A KNOWN, CLUSTERED EIGENVALUE SPECTRUM. EXCEPT EFRK, ALL PROCEDURES PERFORM STEPSIZE CONTROL.

PROCEDURE : RK1

AUTHOR: J. A. ZONNEVELD.

CONTRIBUTORS: M. BAKKER AND I. BRINK.

INSTITUTE: MATHEMATICAL CENTRE.

RECEIVED: 730715.

BRIEF DESCRIPTION:

RK1 SOLVES AN INITIAL VALUE PROBLEM FOR A SINGLE FIRST ORDER
ORDINARY DIFFERENTIAL EQUATION $dy/dx = f(x,y)$.
THE EQUATION IS SUPPOSED TO BE NON-STIFF.

KEYWORDS:

INITIAL VALUE PROBLEM.
SINGLE FIRST ORDER ORDINARY DIFFERENTIAL EQUATIONS.

CALLING SEQUENCE:

THE HEADING OF THE PROCEDURE READS:
"PROCEDURE" RK1(X,A,B,Y,YA,FX,E,D,FI);
"VALUE" B,FI;
"REAL" X,A,B,Y,YA,FX;
"BOOLEAN" FI;
"ARRAY" E,D;
"CODE" 33010;

THE MEANING OF THE FORMAL PARAMETERS IS:

X: <VARIABLE>;
THE INDEPENDENT VARIABLE.
UPON COMPLETION OF A CALL
X IS EQUAL TO B;
A: <ARITHMETIC EXPRESSION>;
ENTRY: THE INITIAL VALUE OF X;
B: <ARITHMETIC EXPRESSION>;
ENTRY: A VALUE PARAMETER, GIVING THE END VALUE OF X;
Y: <VARIABLE>;
THE DEPENDENT VARIABLE;

YA: <ARITHMETIC EXPRESSION>;
 ENTRY : THE VALUE OF Y AT X=A;
 FXY: <ARITHMETIC EXPRESSION>;
 AN EXPRESSION, DEPENDING ON X AND Y, GIVING THE VALUE OF DY/DX;
 E: <ARRAY IDENTIFIER>;
 "ARRAY" E[1:2];
 ENTRY:
 E[1] : A RELATIVE TOLERANCE,
 E[2] : AN ABSOLUTE TOLERANCE ;
 D: <ARRAY IDENTIFIER>;
 "ARRAY" D[1:4];
 EXIT:
 ENTIER(D[1]+.5) GIVES THE NUMBER OF STEPS SKIPPED;
 D[2] : EQUALS THE STEP LENGTH;
 D[3] : IS EQUAL TO B;
 D[4] : IS EQUAL TO Y(B);
 FI: <BOOLEAN EXPRESSION>;
 IF FI="TRUE" RK1 INTEGRATES FROM X=A TO X=B WITH INITIAL VALUE
 VALUE Y(A)=YA AND TRIAL STEP B-A. IF FI="FALSE" RK1 INTEGRATES
 FROM X=D[3] TO X=B WITH INITIAL VALUE Y(D[3])=D[4] AND FIRST
 STEP H=D[2]*SIGN(B-D[3]), WHILE A AND YA ARE IGNORED.

PROCEDURES USED: NONE.

REQUIRED CENTRAL MEMORY : NEGLIGIBLE SMALL.

METHOD AND PERFORMANCE:
 RK1 IS BASED ON AN EXPLICIT, 5-TH ORDER RUNGE-KUTTA METHOD AND
 IS PROVIDED WITH STEPLENGTH AND ERRORCONTROL. THE ERRORCONTROL
 IS BASED ON THE LAST TERM OF THE TAYLOR SERIES WHICH IS TAKEN INTO
 ACCOUNT. A STEP IS REJECTED IF THE ABSOLUTE VALUE OF THIS LAST TERM
 GREATER THEN $(ABS(FXY)*E[1]+E[2])*ABS(H)/INT$, WHERE
 $INT = ABS(B - ("IF" FI "THEN" A "ELSE" D[3]))$ DENOTES THE LENGTH
 OF THE INTEGRATION INTERVAL, OTHERWISE, A STEP IS ACCEPTED.
 RK1 USES AS ITS MINIMAL ABSOLUTE STEPLENGTH $HMIN = E[1]*INT+E[2]$.
 IF A STEP OF LENGTH $ABS(H) = HMIN$ IS REJECTED, THE STEP IS SKIPPED.
 FOR FURTHER DETAILS SEE [1].

REFERENCES:

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

EXAMPLE OF USE:

THE SOLUTION OF THE DIFFERENTIAL EQUATION $dy/dx = -y$
 WITH INITIAL CONDITION $y(0)=1$ AT $x=1$ IS COMPUTED
 BY MEANS OF THE FOLLOWING PROGRAM:

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

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

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

IT DELIVERS WITH E[1]=E[2]="-4:
X=.100000000000"01

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

SOURCE TEXT(S):

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

```

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

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

```

SECTION : 5.2.1.1.1.1.B

(FEBRUARY 1979)

PAGE 1

PROCEDURE : RKE.

AUTHOR: P.A. BEENTJES.

INSTITUTE: MATHEMATICAL CENTRE.

RECEIVED: 740520.

BRIEF DESCRIPTION:

RKE SOLVES AN INITIAL VALUE PROBLEM FOR A SYSTEM OF FIRST
ORDER ORDINARY DIFFERENTIAL EQUATIONS $dy / dx = f(x,y)$.
THE SYSTEM IS SUPPOSED TO BE NON-STIFF.

KEYWORDS:

INITIAL VALUE PROBLEM.
SYSTEM OF FIRST ORDER ORDINARY DIFFERENTIAL EQUATIONS.

CALLING SEQUENCE:

THE HEADING OF THE PROCEDURE READS:

```
"PROCEDURE" RKE (X, XE, N, Y, DER, DATA, FI, OUT);
"VALUE" N, FI; "INTEGER" N; "REAL" X, XE; "BOOLEAN" FI;
"ARRAY" Y, DATA;
"PROCEDURE" DER, OUT;
"CODE" 33033;
```

RKE INTEGRATES THE SYSTEM OF ORDINARY DIFFERENTIAL EQUATIONS
 $dy/dx = F(x, y)$, FROM $x = x_0$ TO $x = x_e$ WHERE $y(x_0) = y_0$.

THE MEANING OF THE FORMAL PARAMETERS IS:

```
X: <VARIABLE>;
    THE INDEPENDENT VARIABLE;
    ENTRY: THE INITIAL VALUE X0;
XE: <ARITHMETIC EXPRESSION>;
    THE FINAL VALUE OF X;
N: <ARITHMETIC EXPRESSION>;
    THE NUMBER OF EQUATIONS OF THE SYSTEM;
Y: <ARRAY IDENTIFIER>;
    "ARRAY" Y[1 : N];
    THE DEPENDENT VARIABLES;
    ENTRY: THE INITIAL VALUES AT X = X0;
    EXIT : THE VALUES OF THE SOLUTION AT X = XE;
DER: <PROCEDURE IDENTIFIER>;
    THE HEADING OF THIS PROCEDURE READS:
    "PROCEDURE" DER (T, V); "VALUE" T; "REAL" T; "ARRAY" V;
    THIS PROCEDURE PERFORMS AN EVALUATION OF THE RIGHT HAND
    SIDE OF THE SYSTEM WITH DEPENDENT VARIABLES V[1 : N] AND
    INDEPENDENT VARIABLE T; UPON COMPLETION OF DER
    THE RIGHT HAND SIDE SHOULD BE OVERWRITTEN ON V[1 : N];
DATA: <ARRAY IDENTIFIER>;
    "ARRAY" DATA[1 : 6];
    IN ARRAY DATA ONE SHOULD GIVE:
    DATA[1]: THE RELATIVE TOLERANCE;
    DATA[2]: THE ABSOLUTE TOLERANCE;
    AFTER EACH STEP DATA[3:6] CONTAINS :
    DATA[3]: THE STEPLENGTH USED FOR THE LAST STEP;
    DATA[4]: THE NUMBER OF INTEGRATION STEPS PERFORMED;
    DATA[5]: THE NUMBER OF INTEGRATION STEPS REJECTED;
    DATA[6]: THE NUMBER OF INTEGRATION STEPS SKIPPED;
    IF UPON COMPLETION OF RKE DATA[6] > 0,
    RESULTS SHOULD BE CONSIDERED MOST CRITICALLY;
FI: <BOOLEAN EXPRESSION>;
    IF FI = "TRUE" THE INTEGRATION STARTS AT X0 WITH A
    TRIAL STEP XE - X0; IF FI = "FALSE" THE INTEGRATION IS
    CONTINUED WITH A STEPLENGTH DATA[3] * SIGN(XE - X0);
OUT: <PROCEDURE IDENTIFIER>;
    THE HEADING OF THIS PROCEDURE READS:
    "PROCEDURE" OUT;
    AFTER EACH INTEGRATION STEP PERFORMED OUT CAN BE USED TO
    OBTAIN INFORMATION FROM THE SOLUTION PROCES, E.G. THE VALUES
    OF X, Y[1 : N] AND DATA[3 : 6]. POUT CAN ALSO BE USED TO
    UPDATE DATA.
```


PROCEDURES USED : NONE.

REQUIRED CENTRAL MEMORY:

CIRCA 5 * N MEMORY PLACES.

METHOD AND PERFORMANCE:

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

REFERENCES:

- [1]. R. ENGLAND.
ERROR ESTIMATES FOR RUNGE KUTTA TYPE SOLUTIONS TO SYSTEMS OF ORDINARY DIFFERENTIAL EQUATIONS.
THE COMPUTER JOURNAL , VOLUME 12, P 166 - 169, 1969.
- [2]. J.A. ZONNEVELD.
AUTOMATIC NUMERICAL INTEGRATION.
MATH. CENTRE TRACT 8(1970).
- [3]. P.A. BEENTJES.
SOME SPECIAL FORMULAS OF THE ENGLAND CLASS OF FIFTH ORDER RUNGE - KUTTA SCHEMES.
MATH. CENTRE REPORT NW 24/74.

EXAMPLE OF USE:

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

$$\begin{aligned}DX / DT &= Y - Z, \\DY / DT &= X * X + 2 * Y + 4 * T, \\DZ / DT &= X * X + 5 * X + Z * Z + 4 * T,\end{aligned}$$

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

CAN BE OBTAINED BY THE FOLLOWING PROGRAM:

```
"BEGIN" "REAL" T, TE; "ARRAY" Y(1 : 3), DATA(1 : 6);

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

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

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

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

THIS PROGRAM DELIVERS:

T = +1

RELATIVE AND ABSOLUTE ERRORS IN X, Y AND Z :

RE(X)	RE(Y)	RE(Z)	AE(X)	AE(Y)	AE(Z)
.37 ⁻⁶	.15 ⁻⁵	.13 ⁻⁵	.91 ⁻⁶	.13 ⁻⁴	.11 ⁻⁴
NUMBER OF INTEGRATION STEPS PERFORMED : 9					
NUMBER OF INTEGRATION STEPS SKIPPED : 0					
NUMBER OF INTEGRATION STEPS REJECTED : 5					

T = -1

RELATIVE AND ABSOLUTE ERRORS IN X, Y AND Z :

RE(X)	RE(Y)	RE(Z)	AE(X)	AE(Y)	AE(Z)
.22 ⁻⁶	.52 ⁻⁷	.19 ⁻⁶	.75 ⁻⁷	.55 ⁻⁷	.77 ⁻⁷
NUMBER OF INTEGRATION STEPS PERFORMED : 10					
NUMBER OF INTEGRATION STEPS SKIPPED : 0					
NUMBER OF INTEGRATION STEPS REJECTED : 7					

SOURCE TEXT(S):

```
"CODE" 33033;
"PROCEDURE" RKE (X, XE, N, Y, DER, DATA, FI, OUT);
"VALUE" FI, N; "INTEGER" N; "REAL" X, XE;
"BOOLEAN" FI; "ARRAY" Y, DATA;
"PROCEDURE" DER, OUT;
"BEGIN" "INTEGER" J;
"REAL" XT, H, HMIN, INT, HL, HT, ABSH, FHM, DISCR, TOL, MU,
MU1, FH, E1, E2;
"BOOLEAN" LAST, FIRST, REJECT;
"ARRAY" KO, K1, K2, K3, K4[1:N];
"IF" FI "THEN"
"BEGIN" DATA[3] := XE - X; DATA[4] := DATA[5] := DATA[6] := 0 "END";
ABSH := H := ABS(DATA[3]);
"IF" XE < X "THEN" H := - H; INT := ABS(XE - X);
HMIN := INT * DATA[1] + DATA[2];
E1 := 12 * DATA[1] / INT; E2 := 12 * DATA[2] / INT;
FIRST := "TRUE"; REJECT := "FALSE"; "IF" FI "THEN"
"BEGIN" LAST := "TRUE"; "GOTO" STEP "END";
TEST: ABSH := ABS(H); "IF" ABSH < HMIN "THEN"
"BEGIN" H := SIGN (XE - X) * HMIN; ABSH := HMIN "END";
"IF" H >= XE - X "EQUIV" H >= 0 "THEN"
"BEGIN" LAST := "TRUE"; H := XE - X; ABSH := ABS(H) "END"
"ELSE" LAST := "FALSE";
```

"COMMENT"

```

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

```

SECTION : 5.2.1.1.1.1.C

(FEBRUARY 1979)

PAGE 1

PROCEDURE : RK4A.

AUTHOR: J.A.ZONNEVELD.

CONTRIBUTORS: M.BAKKER AND I.BRINK.

INSTITUTE: MATHEMATICAL CENTRE,

RECEIVED: 730715.

BRIEF DESCRIPTION:

RK4A SOLVES AN INITIAL VALUE PROBLEM FOR A SINGLE FIRST ORDER ORDINARY DIFFERENTIAL EQUATION $dy/dx = f(x,y)$, WHERE $f(x,y)$ MAY BECOME LARGE, E.G. IN THE NEIGHBOURHOOD OF A SINGULARITY. RK4A INTERCHANGES THE DEPENDENT AND INDEPENDENT VARIABLE. THE EQUATION IS SUPPOSED TO BE NON-STIFF.

KEYWORDS:

INITIAL VALUE PROBLEM,
SINGLE FIRST ORDER ORDINARY DIFFERENTIAL EQUATION
LARGE DERIVATIVE VALUES.

CALLING SEQUENCE:

THE HEADING OF THE PROCEDURE READS:
"PROCEDURE" RK4A(X, XA, B, Y, YA, FXY, E, D, FI, XDIR, POS);
"VALUE" FI, XDIR, POS;
"BOOLEAN" FI, XDIR, POS;
"REAL" X, XA, B, Y, YA, FXY;
"ARRAY" E, D;
"CODE" 33016;

THE MEANING OF THE FORMAL PARAMETERS IS:

X: <VARIABLE>;
THE INDEPENDENT VARIABLE ;
UPON COMPLETION OF A CALL X IS EQUAL TO THE MOST RECENT
VALUE OF THE INDEPENDENT VARIABLE;

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

B: <ARITHMETIC EXPRESSION>;
B DEPENDS ON X AND Y.
THE EQUATION B=0 FULFILLED WITHIN THE TOLERANCES E[4] AND
E[5] SPECIFIES THE END OF THE INTEGRATION INTERVAL.
AT THE END OF EACH INTEGRATION STEP B IS EVALUATED AND IS
TESTED FOR CHANGE OF SIGN;

Y: <VARIABLE>;
THE DEPENDENT VARIABLE;

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

FXY: <ARITHMETIC EXPRESSION>;
FX Y GIVES THE VALUE OF DY/DX;
FX Y USES X AND Y AS JENSEN PARAMETERS.

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

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

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

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

PROCEDURES USED : ZEROIN = CP34150.

METHOD AND PERFORMANCE:

RK4A IS BASED ON AN EXPLICIT, 5-TH ORDER RUNGE-KUTTA METHOD AND INTERCHANGES THE DEPENDENT AND INDEPENDENT INTEGRATION VARIABLE. IF $ABS(F(X,Y)) < 1$, X IS USED AS INTEGRATION VARIABLE, OTHERWISE Y. THE PROCEDURE IS PROVIDED WITH STEP SIZE AND ERROR CONTROL. FOR DETAILS, E.G. HOW TO USE ARRAY E AND HOW TO SPECIFY THE ENDPOINT SEE [1] (RK4A IS A SLIGHTLY ADAPTED VERSION OF RK4).

REFERENCES:

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

EXAMPLE OF USE:

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

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

THE PROGRAM PRINTS THE FOLLOWING RESULTS:

X=1.9999998554 EXACTLY: 2.0000000000

Y=-1.9999995347427

Y-X*(1-X)=0.0000000313

SOURCE TEXT(S):

```

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

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

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

```

"COMMENT"


```

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

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

```

```

AGAIN: ABSH:= ABS(H); "IF" ARSH < HMIN "THEN"
  "BEGIN" H:= SIGN(H) * HMIN; ABSH:= HMIN "END";
  "IF" IV "THEN"
    "BEGIN" RKSTEP(X, XL, H, Y, YL, ZL, FXY, I);
    TOL:= E[2] * ABS(KO) + E[3] * ABSH
  "END"
  "ELSE"
    "BEGIN" RKSTEP(Y, YL, H, X, XL, 1 / ZL, 1 / FXY, I);
    TOL:= E[0] * ABS(KO) + E[1] * ABSH
  "END";
REJ:= DISCR > TOL; MU:= TOL / (TOL + DISCR) + .45;
"IF" REJ "THEN"
  "BEGIN" "IF" ABSH <= HMIN "THEN"
    "BEGIN" "IF" IV "THEN"
      "BEGIN" X:= XL + H; Y:= YL + KO "END"
    "ELSE"
      "BEGIN" X:= XL + KO; Y:= YL + H "END";
    D[1]:= D[1] + 1; FIRST:= "TRUE"; "GOTO" NEXT
  "END";
  H:= H * MU; I:= 0; "GOTO" AGAIN
"END" REJ;
"IF" FIRST "THEN"
  "BEGIN" FIRST:= FIR; HL:= H; H:= MU * H; "GOTO" ACCEPT
"END";
FHM:= MU * H / HL + MU - MU1; HL:= H; H:= FHM * H;
ACCEPT: "IF" IV "THEN" RKSTEP(X, XL, HL, Y, YL, ZL, FXY,
2) "ELSE" RKSTEP(Y, YL, HL, X, XL, ZL, 1 / FXY, 2);
MU1:= MU;
NEXT: "IF" FIR "THEN"
  "BEGIN" FIR:= "FALSE"; CONDO:= B;
  "IF" "NOT"(FI "OR" REJ) "THEN" H:= D[2]
"END"
"ELSE"
  "BEGIN" D[2]:= H; COND1:= B;
  "IF" CONDO * CJND1 <= 0 "THEN" "GOTO" ZERO;
  CONDO:= COND1
"END";
D[3]:= XL:= X; D[4]:= YL:= Y; "GOTO" CHANGE;
ZERO: E1[1]:= E[4]; E1[2]:= E[5];
S1:= "IF" IV "THEN" X "ELSE" Y;
S:= "IF" IV "THEN" XL "ELSE" YL ;
ZERODIN(S, S1, FZERO, ABS(E1[1]*S)+ABS(E1[2]));
S1:= "IF" IV "THEN" X "ELSE" Y ;
"IF" IV "THEN" RKSTEP(X, XL, S - XL, Y, YL, ZL, FXY, 3)
"ELSE" RKSTEP(Y, YL, S - YL, X, XL, ZL, 1 / FXY,
3); D[3]:= X; D[4]:= Y
"END" RK4A;
"EQP"

```

SECTION : 5.2.1.1.1.1.D

(FEBRUARY 1979)

PAGE 1

PROCEDURE : RK4NA.

AUTHOR: J.A.ZONNEVELD.

CONTRIBUTORS: M.BAKKER AND I.BRINK.

INSTITUTE: MATHEMATICAL CENTRE.

RECEIVED: 730715.

BRIEF DESCRIPTION:

RK4NA SOLVES AN INITIAL VALUE PROBLEM FOR A SYSTEM OF FIRST ORDER ORDINARY DIFFERENTIAL EQUATIONS $dy/dx = f(x,y)$, OF WHICH THE DERIVATIVE COMPONENTS ARE SUPPOSED TO BECOME LARGE, E.G. IN THE NEIGHBOURHOOD OF SINGULARITIES. RK4NA INTERCHANGES THE INDEPENDENT VARIABLE AND ONE DEPENDENT VARIABLE. THE SYSTEM IS SUPPOSED TO BE NON-STIFF.

KEYWORDS:

INITIAL VALUE PROBLEM,
SYSTEM OF FIRST ORDER ORDINARY DIFFERENTIAL EQUATIONS,
LARGE DERIVATIVE COMPONENTS.

CALLING SEQUENCE:

THE HEADING OF THE PROCEDURE READS:
 "PROCEDURE" RK4NA(X, XA, B, FXJ, J, E, D, FI, N, L, POS);
 "VALUE" FI, N, L, POS;
 "INTEGER" J, N, L;
 "BOOLEAN" FI, POS;
 "REAL" B, FXJ;
 "ARRAY" X, XA, E, D;
 "CODE" 33017;

THE MEANING OF THE FORMAL PARAMETERS IS:

X: <ARRAY IDENTIFIER>;
 "ARRAY" X(0:N);
 X(0) IS THE INDEPENDENT VARIABLE,
 X(1), ..., X(N) ARE THE DEPENDENT VARIABLES;
 EXIT: THE SOLUTION AT B=0;

XA: <ARRAY IDENTIFIER>;
 "ARRAY" XA(0:N);
 ENTRY: THE INITIAL VALUES OF X(0), ..., X(N);

B: <ARITHMETIC EXPRESSION>;
 B DEPENDS ON X(0), ..., X(N);
 IF THE EQUATION B=0 IS SATISFIED WITHIN A
 CERTAIN TOLERANCE, THE INTEGRATION IS TERMINATED;
 B IS EVALUATED AND TESTED FOR CHANGE OF SIGN AT THE
 END OF EACH STEP;
 FOR THE TOLERANCE SEE PARAMETER E.

FXJ: <ARITHMETIC EXPRESSION>;
 FXJ DEPENDS ON X(0), ..., X(N) AND J, DEFINING THE RIGHT
 HAND SIDE OF THE DIFFERENTIAL EQUATION;
 AT EACH CALL IT DELIVERS :DX(J)/DX(0);

J: <VARIABLE>;
 J IS USED AS A JENSEN PARAMETER FOR FXJ;

E: <ARRAY IDENTIFIER>;
 "ARRAY" E(0:2*N+3);
 ENTRY: E(2*J) AND E(2*J+1), 0<=J<=N,
 ARE THE RELATIVE AND THE ABSOLUTE TOLERANCE,
 RESPECTIVELY, ASSOCIATED WITH X(J);
 E(2*N+2) AND E(2*N+3) ARE THE RELATIVE AND ABSOLUTE
 TOLERANCE USED IN THE DETERMINATION OF THE ZERO OF B;

D: <ARRAY IDENTIFIER>;
 "ARRAY" D(0:N+3);
 AFTER COMPLETION OF EACH STEP WE HAVE :
 ENTIER(D(0)+.5) IS THE NUMBER OF STEPS SKIPPED;
 D(2) IS THE STEP LENGTH;
 D(J+3) IS THE LAST VALUE OF X(J), J=0, ..., J=N;

FI: <BOOLEAN EXPRESSION>
IF FI="TRUE" THEN THE INTEGRATION IS STARTED WITH INITIAL
CONDITIONS X[J]=XA[J]; IF FI="FALSE" THEN THE INTEGRATION IS
CONTINUED WITH X[J]=D[J+3];
N: <ARITHMETIC EXPRESSION>;
THE NUMBER OF EQUATIONS;
L: <ARITHMETIC EXPRESSION>;
AN INTEGER TO BE SUPPLIED BY THE USER, $0 \leq L \leq N$ (SEE POS);
POS: <BOOLEAN EXPRESSION>
IF FI="TRUE" THEN THE INTEGRATION STARTS IN SUCH A WAY
THAT X[L] INCREASES IF POS="TRUE" AND X[L] DECREASES IF
POS="FALSE";
IF FI="FALSE" THEN POS IS OF NO SIGNIFICANCE.

PROCEDURES USED : ZEROIN = CP34150.

REQUIRED CENTRAL MEMORY : CIRCA $9 * N$ MEMORY PLACES.

METHOD AND PERFORMANCE :

RK4NA IS BASED ON AN EXPLICIT, 5-TH ORDER RUNGE-KUTTA METHOD
AND INTERCHANGES VARIABLES. THE PROCEDURE IS PROVIDED WITH STEPSIZE
AND ERROR CONTROL. FOR DETAILS, E.G. HOW TO USE ARRAY E, HOW TO
SPECIFY THE ENDPOINT, HOW TO USE L AND POS, SEE [1] (RK4NA IS A
SLIGHTLY ADAPTED VERSION OF RK4N).

REFERENCES:

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

EXAMPLE OF USE:

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

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

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

  "INTEGER" J, K; "BOOLEAN" FIRST;
  "ARRAY" E[0:7], XA, X[0:2], D[0:5];
  "FOR" K:=0, 1, 2, 3, 4, 5 "DO" E[K]:=0.1**6; E[6]:=E[7]:=0**8 ;
  OUTPUT(61, "("("VAN DER POL")", /BB("EPS=")D.10D,/BB
  "("THE VALUES OF X[0], X[1], X[2], P, RESPECTIVELY")", E[0]);
  X0:=XA[0]; XA[2]:=0; XA[1]:=2; J:=0; PRINT(XA);
  FIRST:="TRUE";
  "FOR" J:=1, 2, 3, 4 "DO"
  "BEGIN" RK4NA(X, XA, X[2], "IF" K=1 "THEN" X[2] "ELSE"
  10*(1-X[1]**2)*X[2]-X[1], K, E, D, FIRST, 2, 0, "TRUE");
  X0:=X[0]-X0; PRINT(X); FIRST:="FALSE"; X0:=X[0]
  "END"
"END"
"EQP"
```

THE PROGRAM PRINTS THE FOLLOWING RESULTS:

VAN DER POL

EPS=0.0000001000

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

+0.00000000	+2.00000000	+0.00000000	+0.00000000
+9.32386570	-2.01428560	+0.00000000	+9.32386570
+18.86305411	+2.01428557	+0.00000000	+9.53918840
+28.40224194	-2.01428858	-0.00000000	+9.53918783
+37.94143003	+2.01428558	+0.00000000	+9.53918809

SOURCE TEXT(S):

```

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

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

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

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

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

```

```

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

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

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

```



```

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

```


SECTION : 5.2.1.1.1.1.E (FEBRUARY 1979)

PAGE 1

PROCEDURE : RK5NA.

AUTHOR: J.A.ZONNEVELD.

CONTRIBUTORS: M.BAKKER AND I.BRINK.

INSTITUTE: MATHEMATICAL CENTRE.

RECEIVED: 730715.

BRIEF DESCRIPTION:

RK5NA SOLVES AN INITIAL VALUE PROBLEM FOR A SYSTEM OF FIRST ORDER ORDINARY DIFFERENTIAL EQUATIONS $DY / DX = F(X, Y)$, OF WHICH THE DERIVATIVE COMPONENTS ARE SUPPOSED TO BECOME LARGE, E.G. IN THE NEIGHBOURHOOD OF SINGULARITIES. RK5NA INTRODUCES THE ARC LENGTH AS INTEGRATION VARIABLE. THE SYSTEM IS SUPPOSED TO BE NON-STIFF.

KEYWORDS:

INITIAL VALUE PROBLEM.
SYSTEM OF FIRST ORDER ORDINARY DIFFERENTIAL EQUATIONS.
LARGE DERIVATIVE COMPONENTS.

CALLING SEQUENCE:

THE HEADING OF THE PROCEDURE READS:
 "PROCEDURE" RK5NA(X, XA, B, FXJ, J, E, D, FI, N, L, POS);
 "VALUE" FI, N, L, POS; "INTEGER" J, N, L; "BOOLEAN" FI, POS;
 "REAL" B, FXJ; "ARRAY" X, XA, E, D;
 "CODE" 33018;

THE MEANING OF THE FORMAL PARAMETERS IS:

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

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

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

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

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

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

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

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

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

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

L: <ARITHMETIC EXPRESSION>;
1 <= L <= N; SEE THE EXPLANATION OF POS;
POS: <BOOLEAN EXPRESSION>;
IF FI = "TRUE" THEN THE INTEGRATION STARTS IN SUCH A WAY THAT
X[I] INCREASES IF POS = "TRUE" AND DECREASES IF POS = "FALSE".
IF FI = "FALSE" THEN POS IS IGNORED.

PROCEDURES USED : ZEROIN = CP34150.

REQUIRED CENTRAL MEMORY : CIRCA 9 * N MEMORY PLACES.

METHOD AND PERFORMANCE :

RK5NA IS BASED ON A 5-TH ORDER RUNGE-KUTTA METHOD AND INTEGRATES
THE SYSTEM $DX[J] / DX[0] = F(J, X[0], \dots, X[N]) / F(0, X[0], \dots, X[N])$.
THE ARC LENGTH IS INTRODUCED AS INTEGRATION VARIABLE.
THE INTEGRATION PROCESS IS TERMINATED IF SOME CONDITION ON
 $X[0], \dots, X[N]$, TO BE SUPPLIED BY THE USER, IS SATISFIED.
RK5NA USES STEPLENGTH AND ERROR CONTROL. DETAILS ABOUT THE
PROCEDURE AND THE UNDERLYING THEORY ARE GIVEN IN [1] (RK5NA IS
A SLIGHTLY ADAPTED VERSION OF RK5N).

REFERENCES :

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

EXAMPLE OF USE:

THE VAN DER POL EQUATION IN THE PHASE PLANE

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

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

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

RESULTS:

X[0]=+2.0000000000	X[1]=+0.0000000000	S=000.0000000000
X[0]=-2.0142853657	X[1]=-0.0000000012	S=029.3873834087
X[0]=+2.0142853659	X[1]=+0.0000000001	S=058.7884331939
X[0]=-2.0142853659	X[1]=-0.0000000000	S=088.1894829781
X[0]=+2.0142853659	X[1]=+0.0000000000	S=117.5905327623

SOURCE TEXT(S):

```

"CODE" 33018 ;
"PROCEDURE" RK5NA(X, XA, B, FXJ, J, E, D, FI, N, L, POS);
"VALUE" FI, N, L, POS; "INTEGER" J, N, L; "BOOLEAN" FI, POS;
"REAL" B, FXJ; "ARRAY" X, XA, E, D;
"BEGIN" "INTEGER" I;
  "BOOLEAN" FIRST, FIR, REJ;
  "REAL" FHM, S, SO, CONDO, S1, CONDI, H, ABSH, TOL, FH,
  HL, MU, MUL;
  "ARRAY" Y, XL, DISCR[0:N], K[0:5,0:N], E[1:2];
  "REAL" "PROCEDURE" SUM(J, A, B, XJ); "INTEGER" J, A, B; "REAL" XJ;
  "BEGIN" "REAL" S; S := 0;
    "FOR" J := A "STEP" 1 "UNTIL" B "DO" S := S + XJ; SUM := S
  "END" SUM;
  "BOOLEAN" "PROCEDURE" ZEROIN(X, Y, FX, EPS); "REAL" X, Y, FX, EPS;
  "CODE" 34150;
  "PROCEDURE" RKSTEP(H, D); "VALUE" H, D; "INTEGER" D; "REAL" H;
  "BEGIN" "INTEGER" I;
    "PROCEDURE" F(T); "VALUE" T; "INTEGER" T;
    "BEGIN" "INTEGER" I;
      "REAL" P;
      "FOR" J := 0 "STEP" 1 "UNTIL" N "DO" Y[J] := FXJ;
      P := H / SQRT(SUM(I, 0, N, Y[I] ** 2));
      "FOR" I := 0 "STEP" 1 "UNTIL" N "DO" K[I, I] := Y[I] * P
    "END" F;
    "IF" D = 2 "THEN" "GOTO" INTEGRATE; "IF" D = 1 "THEN"
    "BEGIN" "FOR" I := 0 "STEP" 1 "UNTIL" N "DO" K[I, I] := K[I, I]
      * MU; "GOTO" A
    "END";
    "FOR" I := 0 "STEP" 1 "UNTIL" N "DO" X[I] := XL[I]; F(0);
  A: "FOR" I := 0 "STEP" 1 "UNTIL" N "DO" X[I] := XL[I] +
    K[I, I] / 4.5; F(1);
    "FOR" I := 0 "STEP" 1 "UNTIL" N "DO" X[I] := XL[I] + (K[I, I]
      + K[1, I] * 3) / 12; F(2);
    "FOR" I := 0 "STEP" 1 "UNTIL" N "DO" X[I] := XL[I] + (K[I, I]
      + K[2, I] * 3) / 8; F(3);
    "FOR" I := 0 "STEP" 1 "UNTIL" N "DO" X[I] := XL[I] + (K[I, I]
      * 53 - K[1, I] * 135 + K[2, I] * 126 + K[3, I] * 56)
      / 125; F(4); "IF" D <= 1 "THEN"
    "BEGIN" "FOR" I := 0 "STEP" 1 "UNTIL" N "DO" X[I] := XL[I] +
      (K[I, I] * 133 - K[1, I] * 378 + K[2, I] * 276 +
      K[3, I] * 112 + K[4, I] * 25) / 168; F(5);
    "FOR" I := 0 "STEP" 1 "UNTIL" N "DO" DISCR[I] :=
      ABS(K[I, I] * 21 - K[2, I] * 162 + K[3, I] * 224
      - K[4, I] * 125 + K[5, I] * 42) / 14; "GOTO" END
    "END";
  INTEGRATE: "FOR" I := 0 "STEP" 1 "UNTIL" N "DO" X[I] := XL[I]
    + (- K[I, I] * 63 + K[1, I] * 189 - K[2, I] * 36 -
      K[3, I] * 112 + K[4, I] * 50) / 28; F(5);
    "FOR" I := 0 "STEP" 1 "UNTIL" N "DO" X[I] := XL[I] + (K[I, I]
      * 35 + K[2, I] * 162 + K[4, I] * 125 + K[5, I] * 14)
      / 336;
  END;
"END" RKSTEP;
"COMMENT"

```

```

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

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

```


SECTION : 5.2.1.1.1.1.F

(FEBRUARY 1979)

PAGE 1

PROCEDURE : MULTISTEP.

AUTHOR : P.W. HEMKER.

INSTITUTE : MATHEMATICAL CENTRE.

RECEIVED : 730515.

BRIEF DESCRIPTION :

MULTISTEP SOLVES AN INITIAL VALUE PROBLEM, FOR A SYSTEM OF
FIRST ORDER ORDINARY DIFFERENTIAL EQUATIONS $dy = f(y)$.
IN PARTICULAR THIS PROCEDURE IS SUITABLE FOR THE INTEGRATION OF
STIFF DIFFERENTIAL EQUATIONS. IT CAN ALSO BE USED FOR
NON-STIFF PROBLEMS.

KEYWORDS :

INITIAL VALUE PROBLEM,
SYSTEM OF FIRST ORDER ORDINARY DIFFERENTIAL EQUATIONS.
STIFF EQUATIONS,

CALLING SEQUENCE:

THE HEADING OF THE PROCEDURE READS:
 "BOOLEAN" "PROCEDURE" MULTISTEP(X,XEND,Y,HMIN,HMAX,YMAX,EPS,
 FIRST,SAVE,DERIV,AVAILABLE,JACOBIAN,STIFF,N,OUT);
 "VALUE" HMIN,HMAX,EPS,XEND,N,STIFF;
 "BOOLEAN" FIRST,AVAILABLE,STIFF;
 "INTEGER" N;
 "REAL" X,XEND,HMIN,HMAX,EPS;
 "ARRAY" Y,YMAX,SAVE,JACOBIAN;
 "PROCEDURE" DERIV,OUT;
 "CODE" 33080;

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

THE MEANING OF THE FORMAL PARAMETERS IS:

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

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

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

HMIN,HMAX: <ARITHMETIC EXPRESSION>;
 ENTRY : MINIMAL RESP. MAXIMAL STEPLENGTH ALLOWED;

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

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

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

SAVE: <ARRAY IDENTIFIER>;
 "ARRAY" SAVE[-38:6*N];
 IN THIS ARRAY THE PROCEDURE STORES INFORMATION WHICH CAN BE
 USED IN A CONTINUING CALL WITH FIRST="FALSE";
 BESIDES THE FOLLOWING MESSAGES ARE DELIVERED:
 SAVE[0]=0 : AN ADAMS METHOD HAS BEEN USED;
 1 : THE PROCEDURE SWITCHED TO GEARS METHOD;
 SAVE[-1]=0 : NO ERROR MESSAGE;
 1 : WITH THE HMIN SPECIFIED THE PROCEDURE CANNOT
 HANDLE THE NONLINEARITY (DECREASE HMIN!);
 SAVE[-2] NUMBER OF TIMES THAT THE REQUESTED LOCAL ERROR
 BOUND WAS EXCEEDED;
 SAVE[-3] IF SAVE[-2] IS NONZERO THEN SAVE[-3] GIVES AN
 ESTIMATE OF THE MAXIMAL LOCAL ERROR BOUND,
 OTHERWISE SAVE[-3]=0;

DERIV: <PROCEDURE IDENTIFIER>;
 THE HEADING OF THIS PROCEDURE READS:
 "PROCEDURE" DERIV(DF); "ARRAY" DF;
 THIS PROCEDURE SHOULD DELIVER DY[I]/DX IN DF[I];

AVAILABLE: <BOOLEAN EXPRESSION>;
 IF AN ANALYTICAL EXPRESSION OF THE JACOBIAN MATRIX IS NOT
 AVAILABLE THIS EXPRESSION IS SET TO "FALSE";
 OTHERWISE THIS EXPRESSION IS SET TO "TRUE" AND THE
 EVALUATION OF THIS BOOLEAN EXPRESSION MUST EFFECT THE
 FOLLOWING SIDE-EFFECT:
 THE ENTRIES OF THE JACOBIAN MATRIX $D(DY[I]/DX)/DY[J]$ ARE
 DELIVERED IN THE ARRAY ELEMENTS JACOBIAN[I,J];

JACOBIAN: <ARRAY IDENTIFIER>;
 "ARRAY" JACOBIAN[1:N,1:N];
 AT EACH EVALUATION OF THE BOOLEAN EXPRESSION AVAILABLE WITH
 THE RESULT AVAILABLE="TRUE", THE JACOBIAN MATRIX HAS TO BE
 ASSIGNED TO THIS ARRAY (SEE THE EXAMPLE OF USE);

STIFF: <BOOLEAN EXPRESSION>;
 IF STIFF = "TRUE" THE PROCEDURE SKIPS AN ATTEMPT TO SOLVE
 THE PROBLEM WITH ADAMS-BASHFORTH- OR ADAMS-MOULTON
 METHODS, DIRECTLY USING GEARS METHOD;

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

OUT: <PROCEDURE IDENTIFIER>;
 THE HEADING OF THIS PROCEDURE READS:
 "PROCEDURE" OUT(H,K); "VALUE" H,K; "REAL" H; "INTEGER" K;
 AT THE END OF EACH ACCEPTED STEP OF THE INTEGRATION PROCESS
 THIS PROCEDURE IS CALLED. THE LAST STEPLENGTH USED (H) AND
 THE ORDER OF THE METHOD (K) ARE DELIVERED.
 AT EACH CALL OF THE PROCEDURE OUT, THE CURRENT VALUES OF
 THE INDEPENDENT VARIABLE (X) AND OF THE SOLUTION (Y[I](X))
 ARE AVAILABLE FOR USE. MOREOVER, IN THE NEIGHBOURHOOD OF
 THE CURRENT VALUE OF X, ANY VALUE OF Y[I](X SPECIFIED) CAN
 BE COMPUTED BY MEANS OF THE FOLLOWING INTERPOLATION FORMULA
 $Y[I](X \text{ SPECIFIED}) =$
 $SUM(J,0,K, Y[I+J*N] * ((X \text{ SPECIFIED} - X)/H) ** J).$

PROCEDURES USED:

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

REQUIRED CENTRAL MEMORY: CIRCA $N * (2 * N + 5)$ MEMORY PLACES.

METHOD AND PERFORMANCE :

MULTISTEP IS BASED ON TWO LINEAR MULTISTEP METHODS. FOR STIFF PROBLEMS IT USES THE BACKWARD DIFFERENTIATION METHODS, FOR FOR NON-STIFF PROBLEMS THE ADAMS-BASHFORTH-MOULTON METHODS. MULTISTEP IS PROVIDED WITH ORDER, STEPSIZE AND ERROR CONTROL.

REFERENCES:

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

EXAMPLE OF USE:

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

```

"BEGIN"
  "BOOLEAN" "PROCEDURE" MULTISTEP(X,XEND,Y,HMIN,HMAX,YMAX,EPS,
    FIRST,SAVE,DERIV,AVAILABLE,JACOBIAN,STIFF,N,OUT);
  "CODE" 33080;

  "BOOLEAN" FIRST;
  "INTEGER" I,J,CF,CJ,CA;
  "REAL" X,XEND,HMIN,EPS,R;
  "ARRAY" Y[1:12],YMAX[1:2],D[-40:12],JAC[1:2,1:2];

  "PROCEDURE" DER (F); "ARRAY" F;
  "BEGIN" "REAL" R; CF:=CF+1;
    F[2]:= R:= 3"7*Y[1]*Y[1];
    F[1]:= 0.04*(1-Y[1]-Y[2]) - "4*Y[1]*Y[2] - R;
  "END" F;

  "BOOLEAN" "PROCEDURE" AVAIL;
  "BEGIN" "REAL" R; CJ:= CJ+1;
    AVAIL:= "TRUE";
    JAC[2,1]:= R:= 6"7*Y[1];
    JAC[1,1]:= -0.04 - "4*Y[2] - R;
    JAC[1,2]:= -0.04 - "4*Y[1];
    JAC[2,2]:= 0
  "END" JAC AVAIL;

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

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

  FIRST:= "TRUE"; CA:=CF=CJ=0;
  X:=0; Y[1]:= Y[2]:= 0;
  YMAX[1]:= 0.0001; YMAX[2]:= 1;

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

  "GOTO" LABEL;
  ESCAPE:
"END"

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

```

SOURCE TEXT(S):

```

"CODE" 33080;
"BOOLEAN" "PROCEDURE" MULTISTEP(X, XEND, Y, HMIN, HMAX, YMAX, EPS,
    FIRST, SAVE, DERIV, AVAILABLE, JACOBIAN, STIFF, N, OUT);
"VALUE" HMIN, HMAX, EPS, XEND, N, STIFF;
"BOOLEAN" FIRST, AVAILABLE, STIFF;
"INTEGER" N;
"REAL" X, XEND, HMIN, HMAX, EPS;
"ARRAY" Y, YMAX, SAVE, JACOBIAN;
"PROCEDURE" DERIV, OUT;
"BEGIN" "OWN" "BOOLEAN" ADAMS, WITH JACOBIAN;
    "OWN" "INTEGER" M, SAME, KOLD;
    "OWN" "REAL" XOLD, HOLD, AO, TOLUP, TOL, TOLDWN, TOLCONV;
    "BOOLEAN" EVALUATE, EVALUATED, DECOMPOSE, DECOMPOSED, CONV;
    "INTEGER" I, J, L, K, KNEW, FAILS;
    "REAL" H, CH, CHNEW, ERROR, DFI, C;
    "ARRAY" ACC(5), DELTA, LAST DELTA, DF[1:N], JAC[1:N, 1:N], AUX[1:3];
    "INTEGER" "ARRAY" P[1:N];

"REAL" "PROCEDURE" MATVEC(L, U, I, A, B); "CODE" 34011;
"REAL" "PROCEDURE" DEC(A, N, AUX, P); "CODE" 34300;
"PROCEDURE" SOL(A, N, P, B); "CODE" 34051;

"REAL" "PROCEDURE" NORM2(AI); "REAL" AI;
"BEGIN" "REAL" S, A; S := 1.0E-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" M "UNTIL" K*M "DO"
        "BEGIN" "FOR" I := 1 "STEP" 1 "UNTIL" N "DO"
            Y[J+I] := SAVE[J+I] * C;
            C := C * CH
        "END";
    DECOMPOSED := "FALSE"
"END" RESET;

```

"COMMENT"

```

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

    "BEGIN" "FOR" C:= 1,1,9,4,0,2/3,1,1/3,36,20.25,1,6/11,
      1,6/11,1/11,84.028,53.778,0.25,.48,1,.7,.2,.02,
      156.25, 108.51, .027778, 120/274, 1, 225/274,
      85/274, 15/274, 1/274, 0, 187.69, .0047361
      "DO" "BEGIN" I:= I + 1; SAVE[I]:= C "END"
    "END"
"END" METHOD;

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

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

```

```

"PROCEDURE" DECOMPOSE JACOBIAN;
"BEGIN" DECOMPOSE:= "FALSE";
      DECOMPOSED:= "TRUE"; C:= -A0 * H;
      "FOR" J:= 1 "STEP" 1 "UNTIL" N "DO"
"BEGIN" "FOR" I:= 1 "STEP" 1 "UNTIL" N "DO"
      JAC[I,J]:= JACOBIAN[I,J] * C;
      JAC[J,J]:= JAC[J,J] + 1
"END";
      AUX[2]:=1.0**-12;
      DEC(JAC,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*M+I])) ** (0.5/K);
      A2:= 0.80 * (TOL/ERROR) ** (0.5/(K + 1));
      A3:= "IF" K >= 5 "OR" FAILS ^= 0
           "THEN" 0 "ELSE"
           0.70 * (TOLUP/NORM2(DELTA[I] - LAST DELTA[I])) **
           (0.5/(K+2));

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

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

```

"COMMENT"

;

```

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

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

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

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

```

"COMMENT"

;

```

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

  RESET
"END" "ELSE" CONVERGENCE:

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

```

"COMMENT"

```

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

SAME:= SAME-1;
"IF" SAME= 1 "THEN"
"BEGIN" "FOR" I:= 1 "STEP" 1 "UNTIL" N "DO"
      LAST DELTA[I]:= DELTA[I]
"END" "ELSE"
"IF" SAME= 0 "THEN"
"BEGIN" CALCULATE STEP AND ORDER;
      "IF" CHNEW > 1.1 "THEN"
      "BEGIN" DECOMPOSED:= "FALSE";
            "IF" K ^= KNEW "THEN"
            "BEGIN" "IF" KNEW > K "THEN"
                    "BEGIN" "FOR" I:= 1 "STEP" 1
                            "UNTIL" N "DO" Y[KNEW*M+I]
                            := DELTA[I] * A[K]/KNEW
                    "END";
            K:= KNEW; ORDER
            "END";
            SAME:= K+1;
            "IF" CHNEW * H > HMAX
            "THEN" CHNEW:= HMAX/H;
            H:= 4 * CHNEW; C:= 1;
            "FOR" J:= M "STEP" M "UNTIL" K*M "DO"
            "BEGIN" C:= C * CHNEW;
                    "FOR" I:= J+1 "STEP" 1 "UNTIL"
                    J+N "DO" Y[I]:= Y[I] * C
            "END"
            "END"
      "ELSE" SAME:= 10
"END";
"IF" X ^= XEND "THEN"
"BEGIN" XOLD:= X; HOLD:= H; KOLD:= K; CH:= 1;
      "FOR" I:= K * M + N "STEP" -1 "UNTIL" 1 "DO"
      SAVE[I]:= Y[I];
      OUT(H,K)
"END"
"END" CORRECTION AND ESTIMATION LOCAL ERROR;
"END" STEP;

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

```


SECTION : 5.2.1.1.1.1.G

(FEBRUARY 1979)

PAGE 1

PROCEDURE : DIFFSYS.

AUTHORS : R.BULIRSCH AND J.STOER.

CONTRIBUTOR: K.DEKKER.

INSTITUTE: MATHEMATICAL CENTRE.

RECEIVED: 731231.

BRIEF DESCRIPTION:

DIFFSYS SOLVES AN INITIAL VALUE PROBLEM ,FOR A SYSTEM OF FIRST
ORDER ORDINARY DIFFERENTIAL EQUATIONS $dy / dx = F(x,y)$.
THE METHOD IS RECOMMENDED IF HIGH ACCURACY IS DESIRED.
DIFFSYS IS NOT SUITED FOR STIFF EQUATIONS.

KEYWORDS:

INITIAL VALUE PROBLEMS,
SYSTEM OF FIRST ORDER ORDINARY DIFFERENTIAL EQUATIONS.

CALLING SEQUENCE:

THE HEADING OF THE PROCEDURE DIFFSYS READS:
 "PROCEDURE" DIFFSYS(X,XE,N,Y,DERIVATIVE,AETA,RETA,S,H0,OUTPUT);
 "VALUE" N;
 "INTEGER" N;
 "REAL" X,XE,AETA,RETA,H0;
 "ARRAY" Y,S;
 "PROCEDURE" DERIVATIVE,OUTPUT;
 "CODE" 33180;

THE MEANING OF THE FORMAL PARAMETERS IS:

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

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

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

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

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

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

RETA: <ARITHMETIC EXPRESSION>;
 REQUIRED RELATIVE PRECISION IN THE INTEGRATION PROCESS;

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

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

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

PROCEDURES USED: NONE.

REQUIRED CENTRAL MEMORY : CIRCA 28* N MEMORY PLACES.

METHOD AND PERFORMANCE:

THE PROCEDURE DIFFSYS IS A SLIGHT MODIFICATION OF THE ALGORITHM PUBLISHED BY BULIRSCH AND STOER (SEE REF[1]). BY THIS MODIFICATION INTEGRATION FROM x_0 UNTIL x_e CAN BE PERFORMED BY ONE CALL OF DIFFSYS. A NUMBER OF INTEGRATION STEPS ARE TAKEN, STARTING WITH THE INITIAL STEP h_0 . IN EACH INTEGRATION STEP A NUMBER OF SOLUTIONS ARE COMPUTED BY MEANS OF THE MODIFIED MIDPOINT RULE. EXTRAPOLATION IS USED TO IMPROVE THESE SOLUTIONS, UNTIL THE REQUIRED ACCURACY IS MET. AN INTEGRATION STEP IS REJECTED, IF THE ACCURACY REQUIREMENTS ARE NOT FULFILLED AFTER NINE EXTRAPOLATION STEPS. IN THESE CASES THE INTEGRATION STEP IS REJECTED, AND INTEGRATION IS TRIED AGAIN WITH THE INTEGRATION STEP HALVED. THE ALGORITHM IS FOR EACH STEP A VARIABLE ORDER METHOD (THE HIGHEST ORDER IS 14), AND USES A VARIABLE NUMBER OF FUNCTION EVALUATIONS, DEPENDING ON THE ORDER (MINIMUM IS 3, MAXIMUM IS 217). THE ALGORITHM IS LESS SENSITIVE TO TOO SMALL VALUES OF THE INITIAL STEPSIZE THAN THE ORIGINAL ALGORITHM (SEE REF [2]); HOWEVER BAD GUESSES REQUIRE STILL SOME MORE COMPUTATIONS.

REFERENCES:

- [1]. R. BULIRSCH AND J. STOER.
NUMERICAL TREATMENT OF ORDINARY DIFFERENTIAL EQUATIONS BY
EXTRAPOLATION METHODS.
NUMERISCHE MATHEMATIK, VOLUME 8, PAGE 1-13, 1965.
- [2]. PHYLLIS FOX.
A COMPARATIVE STUDY OF COMPUTER PROGRAMS FOR INTEGRATING
DIFFERENTIAL EQUATIONS.
COMMUNICATIONS OF THE A.C.M., VOLUME 15, PAGE 941-948, 1972.
- [3]. T. E. HULL, W. H. ENRIGHT, B. M. FELLEN AND A. E. SEDGWICK.
COMPARING NUMERICAL METHODS FOR ORDINARY DIFFERENTIAL
EQUATIONS.
SIAM JOURNAL ON NUMERICAL ANALYSIS, VOLUME 9, PAGE 603-635, 1972.

EXAMPLE OF USE:

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

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

  "PROCEDURE" DER(X,Y,DY); "REAL" X; "ARRAY" Y,DY;
  "BEGIN" "REAL" MU,MU1,Y1,Y2,Y3,Y4,S1,S2;
  MU:=1/92.45; MU1:=1-MU;
  PASSES:=PASSES+1;
  Y1:=Y[1]; Y2:=DY[1]:=Y[2]; Y3:=Y[3]; Y4:=DY[3]:=Y[4];
  S1:=(Y1+MU)**2+Y3**2; S2:=(Y1-MU1)**2+Y3**2;
  S1:=S1*SQRT(S1); S2:=S2*SQRT(S2);
  DY[2]:=Y1+2*Y4-MU1*(Y1+MU)/S1-MU*(Y1-MU1)/S2;
  DY[4]:=Y3-2*Y2-MU1*Y3/S1-MU*Y3/S2
  "END";

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

  OUTPUT(61,"("(" THIS LINE AND THE FOLLOWING TEXT IS ""
  "(PRINTED BY THIS PROGRAM)",//,
  "( THE RESULTS WITH DIFFSYS - H0=.2 - ARE: )"//,
  "( K DER.EV. Y[1] Y[3] )"//,
  "( TIME)"//,/"");
  "FOR" TOL:=4,6,8,10,12 "DO"
  "BEGIN" PASSES:=K:=0; X:=0; XE:=6.192169331396;
  Y[1]:=1.2; Y[2]:=Y[3]:=0; Y[4]:=-1.04935750983;
  S[1]:=S[2]:=S[3]:=S[4]:=0; H0:=.2; TIME:=CLOCK;
  DIFFSYS(X,XE,4,Y,DER,TOL,TOL,S,H0,OUT);
  "END"
"END"
```

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

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

K	DER.EV.	Y[1]	Y[3]	TIME
30	2591	+1.320 357 347 741	-.032 645 454 836	5.686
33	3414	+1.200 078 037 879	-.000 053 906 067	7.455
37	4213	+1.200 003 282 801	-.000 002 363 741	9.267
44	4618	+1.199 999 999 711	-.000 000 000 095	10.242
56	6299	+1.200 000 000 003	-.000 000 000 090	13.827

SOURCE TEXT:

```

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

```

```

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

```

PROCEDURE : ARK.

AUTHOR: P.A. BEFNTJES.

INSTITUTE: MATHEMATICAL CENTRE.

RECEIVED: 740510.

BRIEF DESCRIPTION:

ARK SOLVES AN INITIAL VALUE PROBLEM, FOR A SYSTEM OF FIRST ORDER ORDINARY DIFFERENTIAL EQUATIONS. ARK IS RECOMMENDED FOR THE INTEGRATION OF SEMI-DISCRETE PARABOLIC AND HYPERBOLIC INITIAL-BOUNDARY PROBLEMS.

KEYWORDS:

INITIAL VALUE PROBLEM,
SEMI-DISCRETE PARABOLIC AND HYPERBOLIC PROBLEM.

CALLING SEQUENCE:

THE HEADING OF THE PROCEDURE READS:
"PROCEDURE" ARK (T, TE, MO, M, U, DERIVATIVE, DATA, OUT);
"INTEGER" MO, M; "REAL" T, TE; "ARRAY" U, DATA;
"PROCEDURE" DERIVATIVE, OUT;
"CODE" 33061;

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

THE MEANING OF THE FORMAL PARAMETERS IS:

T: <VARIABLE>;
THE INDEPENDENT VARIABLE T;
ENTRY: THE INITIAL VALUE T_0 ;
EXIT : THE FINAL VALUE T_E ;
TE: <ARITHMETIC EXPRESSION>;
THE FINAL VALUE OF T ($T_E \geq T$);
MO, M: <ARITHMETIC EXPRESSION>;
INDICES OF THE FIRST AND LAST EQUATION OF THE SYSTEM;
U: <ARRAY IDENTIFIER>;
"ARRAY" U[MO : M];
ENTRY: THE INITIAL VALUES OF THE SOLUTION OF THE SYSTEM OF
DIFFERENTIAL EQUATIONS AT $T = T_0$;
EXIT : THE VALUES OF THE SOLUTION AT $T = T_E$;

DERIVATIVE: <PROCEDURE IDENTIFIER>;
 THE HEADING OF THIS PROCEDURE READS:
 "PROCEDURE" DERIVATIVE(T, V); "REAL" T; "ARRAY" V;
 THIS PROCEDURE PERFORMS AN EVALUATION OF THE RIGHT HAND
 SIDE OF THE SYSTEM WITH DEPENDENT VARIABLES UC(MO : MJ) AND
 INDEPENDENT VARIABLE T; UPON COMPLETION OF DERIVATIVE, THE
 RIGHT HAND SIDE SHOULD BE OVERWRITTEN ON UC(MO : MJ);

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

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

DATA AND RESULTS:

FOR THE INDICES M_0 AND M THE FOLLOWING REMARKS CAN BE MADE:
WHEN THE METHOD OF LINES IS APPLIED TO HYPERBOLIC DIFFERENTIAL
EQUATIONS THE NUMBER OF RELEVANT ORDINARY DIFFERENTIAL EQUATIONS
DECREASES DURING THE INTEGRATION PROCESS; IN PROCEDURE ARK
THIS MAY BE REALIZED BY INTEGERS M_0 AND M , WHICH ARE
DEFINED AS FUNCTIONS OF THE NUMBER OF RIGHT HAND SIDE EVALUATIONS.
A SELECTION OF POSSIBLE ENTRIES FOR ARRAY DATA (DEPENDENT ON THE
KIND OF INITIAL VALUE PROBLEM) IS GIVEN IN REFERENCE [4], SECTION 8.

PROCEDURES USED:

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

REQUIRED CENTRAL MEMORY : CIRCA $75 + 2 * (M - M_0)$ MEMORY PLACES.

METHOD AND PERFORMANCE:

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

REFERENCES:

- [1]. P.J. VAN DER HOUWEN.
STABILIZED RUNGE KUTTA METHOD WITH LIMITED
STORAGE REQUIREMENTS.
MATH. CENTR. REPORT TW 124/71;
- [2]. P.A. BEENTJES.
AN ALGOL 60 VERSION OF STABILIZED RUNGE KUTTA
METHODS (DUTCH).
MATH. CENTR. REPORT NR 23/72;

- [3]. P.J. VAN DER HOUWEN, J. KOK.
NUMERICAL SOLUTION OF A MINIMAX PROBLEM.
MATH. CENTR. REPORT TW 123/71;
- [4]. P.J. VAN DER HOUWEN ET AL.
ONE STEP METHODS FOR LINEAR INITIAL VALUE PROBLEMS, I.I.I.
NUMERICAL EXAMPLES,
MATH. CENTR. REPORT TW 130/71.

EXAMPLE OF USE:

THE VALUES OF

1. Y(1) AND Y(2) OF THE INITIAL VALUE PROBLEM
 $dy / dx = y - 2 * x / y, \quad y(0) = 1$

AND

2. U(.6, 0) OF THE CAUCHY PROBLEM (SEE REFERENCE [2]):
 $du / dt = .5 * du / dx, \quad u(0, x) = \exp(-x * x)$

MAY BE OBTAINED BY THE FOLLOWING PROGRAM:

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

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

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

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

```

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

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

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

```

THIS PROGRAM DELIVERS:

PROBLEM 1

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

PROBLEM 2

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

SOURCE TEXT(S):

```

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

"BEGIN" "INTEGER" P, N, Q;
"DOWN" "REAL" ECO, EC1, EC2, TAU0, TAU1, TAU2, TAUS, T2;
"REAL" THETANM1, TAU, BETAN, QINV, ETA;
"ARRAY" MU, LAMBDA[1:DATA[1]], THETA[0:DATA[1]], RO, R[MO:M];
"BOOLEAN" START, STEP1, LAST;

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

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

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

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

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

"COMMENT"

```



```

N:= DATA[1]; P:= DATA[2]; EC1:= EC2 := 0;
BETAN:= DATA[3];
THETANM1:= "IF" P=3 "THEN" .75 "ELSE" 1;
THETAO:= 1 - THETANM1; S:= 1;
"FOR" J:= N - 1 "STEP" - 1 "UNTIL" 1 "DO"
"BEGIN" S:= - S * THETAO + DATA[N + 10 - J];
      MUI[J]:= DATA[N + 11 - J] / S;
      LAMBDA[J]:= MUI[J] - THETAO
"END";
"FOR" I:= 1 "STEP" 1 "UNTIL" 8 "DO"
"FOR" J:= 0 "STEP" 1 "UNTIL" N "DO"
ALFA[I, J + 1]:= "IF" I = 1 "THEN" 1 "ELSE"
  "IF" J = 0 "THEN" 0 "ELSE" "IF" I = 2 ! I = 4 ! I = 8 "THEN"
  MUI(J) ** ENTIER((I + 2) / 3) "ELSE"
  "IF" (I = 3 ! I = 6) & J > 1 "THEN" SUM(L, 1, J-1,
  LABDA(J, L) * MUI(L) ** ENTIER(I / 3)) "ELSE"
  "IF" I = 5 & J > 2 "THEN" SUM(L, 2, J - 1, LABDA(J, L) *
  SUM(K, 1, L - 1, LABDA(L, K) * MUI(K))) "ELSE"
  "IF" I = 7 & J > 1 "THEN" SUM(L, 1, J - 1, LABDA(J, L) *
  MUI(L)) * MUI(J) "ELSE" 0;
N1:= "IF" N < 4 "THEN" N + 1 "ELSE" "IF" N < 7 "THEN" 4
"ELSE" 8;
I:= 1;
"FOR" S:= 1, .5, 1 / 6, 1 / 3, 1 / 24, 1 / 12, .125, .25 "DO"
"BEGIN" TH[I]:= S; I:= I + 1 "END";
"IF" P = 3 & N < 7 "THEN" TH[1]:= TH[2]:= 0;
AUX[2]:= N - 14; DECSOL(ALFA, N1, AUX, TH);
INIVEC(O, N, THETA, 0);
DUPVEC(O, N1 - 1, 1, THETA, TH);
"IF" ^ (P = 3 & N < 7) "THEN"
"BEGIN" THETA[0]:= THETA[0] - THETAO;
      THETA[N - 1]:= THETA[N - 1] - THETANM1; Q:= P + 1
"END" "ELSE" Q:= 3;
QINV:= 1 / Q;
START:= DATA[8] = 0; DATA[10]:= 0; LAST:= "FALSE";
DUPVEC(M, M, O, R, U); DERIVATIVE(T, R)
"END" INITIALIZE

```

```

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

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

```

```
"PROCEDURE" DIFFERENCE SCHEME;  
"BEGIN" "INTEGER" I, J;  
  "REAL" MT, LT;  
  MULVEC(MO, M, O, RO, R, THETA(O));  
  "IF" P = 3 "THEN" ELMVEC(MO, M, O, U, R, .25 * TAU);  
  "FOR" I:= 1 "STEP" 1 "UNTIL" N - 1 "DO"  
  "BEGIN" MT:= MUC(I) * TAU; LT:= LAMBDA(I) * TAU;  
    "FOR" J:= MO "STEP" 1 "UNTIL" M "DO"  
      RI(J):= LT * RI(J) + U(J);  
      DERIVATIVE(T + MT, R); LOCAL ERROR CONSTRUCTION(I)  
    "END";  
    ELMVEC(MO, M, O, U, R, THETA(N) * TAU);  
    DUPVEC(MO, M, O, R, U); DERIVATIVE(T + TAU, R);  
    LOCAL ERROR CONSTRUCTION(N); T2:= T;  
    "IF" LAST "THEN"  
      "BEGIN" LAST:= "FALSE"; T:= TE "END" "ELSE" T:= T + TAU;  
      DATA(B):= DATA(B)+1  
    "END" DIFSCH;  
  
  INITIALIZE;  
  
NEXT STEP:  
  STEPSIZE; DIFFERENCE SCHEME; OUT;  
  "IF" T ^= TE "THEN" "GOTO" NEXT STEP;  
  
ENDARK:  
"END" ARK;  
  "EOP"
```


PROCEDURE : EFRK.

AUTHOR: K. DEKKER.

INSTITUTE: MATHEMATICAL CENTRE.

RECEIVED: 740710.

BRIEF DESCRIPTION:

EFRK SOLVES AN INITIAL VALUE PROBLEM FOR A SYSTEM OF FIRST ORDER ORDINARY DIFFERENTIAL EQUATIONS $DU / DT = H(T,U)$. EFRK IS A SPECIAL PURPOSE PROCEDURE FOR STIFF EQUATIONS WITH A KNOWN, CLUSTERED EIGENVALUE SPECTRUM.

KEYWORDS:

INITIAL VALUE PROBLEM,
SYSTEM OF FIRST ORDER ORDINARY DIFFERENTIAL EQUATIONS,
STIFF EQUATION.

CALLING SEQUENCE:

THE HEADING OF THE PROCEDURE EFRK READS:
"PROCEDURE" EFRK (T,TE, MO,M, U, SIGMA, PHI, DIAMETER, DERIVATIVE,
K, STEP, R, L, BETA, THIRDDORDER, TOL, OUTPUT);
"VALUE" R,L;
"INTEGER" MO,M,K,R,L;
"REAL" T,TE,SIGMA,PHI,DIAMETER,STEP,TOL;
"ARRAY" U,BETA;
"BOOLEAN" THIRDDORDER;
"PROCEDURE" DERIVATIVE,OUTPUT;
"CODE" 33070;

THE MEANING OF THE FORMAL PARAMETERS IS:

T: <VARIABLE>;
THE INDEPENDENT VARIABLE T;
ENTRY: THE INITIAL VALUE T0;
EXIT: THE FINAL VALUE TE;
TE: <ARITHMETIC EXPRESSION>;
THE FINAL VALUE OF T (TE>=T);
MO: <ARITHMETIC EXPRESSION>;
THE INDEX OF THE FIRST EQUATION;
M: <ARITHMETIC EXPRESSION>;
THE INDEX OF THE LAST EQUATION;

U: <ARRAY IDENTIFIER>;
 "REAL" "ARRAY" U[M0:M1];
 THE DEPENDENT VARIABLE;
 ENTRY: THE INITIAL VALUES OF THE SOLUTION OF THE SYSTEM OF
 DIFFERENTIAL EQUATIONS AT $T = T_0$;
 EXIT : THE VALUES OF THE SOLUTION AT $T = T_E$;
 SIGMA: <ARITHMETIC EXPRESSION>;
 THE MODULUS OF THE POINT AT WHICH EXPONENTIAL FITTING IS
 DESIRED , FOR EXAMPLE AN APPROXIMATION OF THE CENTRE OF THE
 LEFT HAND CLUSTER;
 PHI: <ARITHMETIC EXPRESSION>;
 THE ARGUMENT OF THE CENTRE OF THE LEFT HAND CLUSTER; IN THE
 CASE OF TWO COMPLEX CONJUGATED CLUSTERS , THE ARGUMENT OF
 THE CENTRE IN THE SECOND QUADRANT SHOULD BE TAKEN;
 DIAMETER: <ARITHMETIC EXPRESSION>;
 THE DIAMETER OF THE LEFT HAND CLUSTER OF EIGENVALUES OF THE
 JACOBIAN MATRIX OF THE SYSTEM OF DIFFERENTIAL EQUATIONS;
 IN CASE OF NON-LINEAR EQUATIONS DIAMETER SHOULD HAVE SUCH A
 VALUE THAT THE VARIATION OF THE EIGENVALUES IN THIS CLUSTER
 IN THE PERIOD ($T, T+STEP$) IS LESS THAN HALF THE DIAMETER;
 DERIVATIVE: <PROCEDURE IDENTIFIER>;
 THE HEADING OF THIS PROCEDURE READS:
 "PROCEDURE" DERIVATIVE(T,U); "REAL" T; "ARRAY" U;
 THIS PROCEDURE SHOULD DELIVER THE VALUE OF $H(T,U)$ IN THE
 POINT (T,U) IN THE ARRAY U;
 K: <VARIABLE>;
 COUNTS THE NUMBER OF INTEGRATION STEPS TAKEN;
 FOR EXAMPLE, K MAY BE USED IN THE EXPRESSION FOR T_E ;
 ENTRY: AN (ARBITRARY) CHOSEN VALUE K_0 , E.G. $K_0=0$;
 EXIT : $K_0 +$ THE NUMBER OF INTEGRATION STEPS PERFORMED;
 STEP: <ARITHMETIC EXPRESSION>;
 THE STEPSIZE CHOSEN WILL BE AT MOST EQUAL TO STEP ;
 THIS STEPSIZE MAY BE REDUCED BY STABILITY CONSTRAINTS,
 IMPOSED BY A POSITIVE DIAMETER , OR BY CONSIDERATIONS OF
 INTERNAL STABILITY (SEE REF[1], PAGE 11);
 R: <ARITHMETIC EXPRESSION>;
 $R + L$: THE NUMBER OF EVALUATIONS OF $H(T, U)$ ON WHICH THE
 RUNGE-KUTTA SCHEME IS BASED;
 FOR $R=1,2,>=3$ FIRST, SECOND AND THIRD ORDER ACCURACY MAY BE
 OBTAINED BY AN APPROPRIATE CHOICE OF THE ARRAY BETA;
 L: <ARITHMETIC EXPRESSION>;
 ENTRY:
 IF $PHI = 4 * ARCTAN(1)$: THE ORDER OF THE EXPONENTIAL FITTING,
 ELSE TWICE THE ORDER OF THE EXPONENTIAL FITTING;
 NOTE THAT L SHOULD BE EVEN IN THE LATTER CASE;
 BETA: <ARRAY IDENTIFIER>;
 "REAL" "ARRAY" BETA[0:R+L];
 ENTRY: THE ELEMENTS $BETA[I]$, $I=0, \dots, R$ SHOULD HAVE THE
 VALUE OF THE $R+1$ FIRST COEFFICIENTS OF THE STABILITY
 POLYNOMIAL;

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

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

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

PROCEDURES USED:

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

REQUIRED CENTRAL MEMORY : CIRCA 30 + (M - M0) + L * (5 + L).

METHOD AND PERFORMANCE:

EFRK IS BASED ON EXPLICIT RUNGE-KUTTA METHODS OF ORDER 1, 2 AND 3,
WHICH MAKE USE OF EXPONENTIAL FITTING. AUTOMATIC ERROR CONTROL
IS NOT PROVIDED.
A DETAILED DESCRIPTION OF THE METHOD AND SOME NUMERICAL EXAMPLES
ARE GIVEN IN REF [1], REF [3], PAGE 170 REPRESENTS A BRIEF SURVEY. A
COMPARATIVE TEST OVER A LARGE CLASS OF DIFFERENTIAL EQUATIONS IS
GIVEN IN REF [4].
FROM THESE RESULTS IT APPEARS THAT CALLS WITH THIRDORDER = "TRUE"
ARE LESS ADVISABLE.

REFERENCES:

- [1]. K. DEKKER.
AN ALGOL 60 VERSION OF EXPONENTIALLY FITTED RUNGE-KUTTA
METHODS (DUTCH).
NR 25 (1972), MATHEMATICAL CENTRE.
- [2]. T. J. DEKKER, P. W. HEMKER AND P. J. VAN DER HOUWEN.
COLLOQUIUM STIFF DIFFERENTIAL EQUATIONS 1 (DUTCH).
MC SYLLABUS 15.1, (1972) MATHEMATICAL CENTRE.
- [3]. P. A. BEENTJES, K. DEKKER, H. C. HEMKER, S.P.N. VAN KAMPEN AND
G. M. WILLEMS.
COLLOQUIUM STIFF DIFFERENTIAL EQUATIONS 2 (DUTCH).
MC SYLLABUS 15.2, (1973) MATHEMATICAL CENTRE.
- [4]. P. A. BEENTJES, K. DEKKER, H. C. HEMKER AND M. V. VELDHUIZEN.
COLLOQUIUM STIFF DIFFERENTIAL EQUATIONS 3 (DUTCH).
MC SYLLABUS 15.3, (1974) MATHEMATICAL CENTRE.

EXAMPLE OF USE:

CONSIDER THE SYSTEM OF DIFFERENTIAL EQUATIONS:
 $dy[1]/dx = -y[1] + y[1] * y[2] + .99 * y[2]$
 $dy[2]/dx = -1000 * (-y[1] + y[1] * y[2] + y[2])$
WITH THE INITIAL CONDITIONS AT $x = 0$:
 $y[1] = 1$ AND $y[2] = 0$. (SEE REF[2], PAGE 11).
THE SOLUTION AT $x = 50$ IS APPROXIMATELY:
 $y[1] = .765\ 878\ 320\ 487$ AND $y[2] = .433\ 710\ 353\ 5768$.
THE FOLLOWING PROGRAM SHOWS SOME DIFFERENT CALLS OF THE PROCEDURE
EFRK:


```

"BEGIN"
  "PROCEDURE" EFRK(X,XE,MO,M,Y,SIGMA,PHI,DIAMETER,DERIVATIVE,K,
                STEP,R,L,BETA,THIRDOORDER,TOL,OUTPUT);
  "CODE" 33070;

  "INTEGER" K,R,L,PASSES;
  "REAL" X,SIGMA,PHI,TIME,STEP,DIAMETER;
  "REAL" "ARRAY" Y[1:2],BETA[0:6];

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

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

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

```

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

THE RESULTS WITH EFRK ARE:

R	L	K	DER.EV.	Y[1]	Y[2]	TIME
1	1	237	474	+ .765 812 555	+ .433 689 306	1.395
1	1	501	1002	+ .765 847 870	+ .433 700 619	3.381
1	2	52	156	+ .765 570 874	+ .433 615 119	0.465
1	2	501	1503	+ .765 848 220	+ .433 700 709	4.200
1	3	52	208	+ .765 571 278	+ .433 615 202	0.531
1	3	500	2000	+ .765 848 512	+ .433 700 827	4.879
2	1	3317	9951	+ .765 878 320	+ .433 710 353	21.808
2	1	1050	3150	+ .765 878 321	+ .433 710 330	7.153
2	2	174	696	+ .765 878 335	+ .433 710 335	1.385
2	2	501	2004	+ .765 878 323	+ .433 709 211	4.915
2	3	57	285	+ .765 881 339	+ .433 817 185	0.642
2	3	501	2505	+ .765 878 323	+ .433 709 725	5.756
3	1	7010	28040	+ .765 878 320	+ .433 710 354	55.298
3	1	3255	13020	+ .765 878 320	+ .433 710 374	25.772
3	2	949	4745	+ .765 878 319	+ .433 711 893	8.499
3	2	1384	6920	+ .765 862 498	+ .449 724 830	13.452
3	3	917	5502	+ .765 878 018	+ .434 105 184	9.143
3	3	1166	6996	+ .765 861 696	+ .433 705 641	15.512

SOURCE TEXT(S):

```

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

```

```

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

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

"PROCEDURE" SOLUTION OF COMPLEX EQUATIONS;
"BEGIN" "INTEGER" I, J, C1, C3;
  "REAL" C2, E, B1, ZI, COSIPH1, SINIPH1, COSPH1L;
  "REAL" "ARRAY" D[1:L];

```

"COMMENT"

```

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

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

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

```

```

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

```

```

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

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

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

```

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: Minimamethode, 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, *Randwaardproblemen, deel 1*, 1967. ISBN 90 6196 024 X.
- MCS 3.2 H.A. LAUWERIER, *Randwaardproblemen, deel 2*, 1968. ISBN 90 6196 036 3.
- MCS 3.3 H.A. LAUWERIER, *Randwaardproblemen, 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 Approximatiethorie*, 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.