A Facility for Executing Concurrent Processes in Pascal

Essebo, Tommy; Johansson, Rolf; Lenells, Matz; Nielsen, Lars

1980

CODEN:LUTFD2/(TFRT-7194)/1-061/(1980)

A FACILITY FOR EXECUTING CONCURRENT PROCESSES IN PASCAL

TOMMY ESSEBO
ROLF JOHANSSON
MATZ LENELLS
LARS NIELSEN

INSTITUTIONEN FÖR REGLERTEKNIK
LUNDS TEKNISKA HÖGSKOLA
JUNI 1980

Organization
LUND INSTITUTE OF TECHNOLOGY
Department of Automatic Control
Box 725
S-220 07 LUND 7, SWEDEN

Document name
REPORT

Date of issue
JUNE 1980

CODEN: LUTFD2/(TFRT-7194)/1-061/(1980)

Author(s)
Tommy Essebo, Rolf Johansson, Matz Lenells, and Lars Nielsen

Sponsoring organization

Title and subtitle
A Facility for Executing Concurrent Processes in Pascal.

A4 A5

Abstract
This paper deals with additional facilities for executing concurrent processes in extended sequential Pascal on a PDP 11 digital computer. Queue features are implemented according to ideas outlined by P Brinch-Hansen. A direct digital control package using process-scheduling was implemented as an application.

Key words
A4 A5
Concurrent processes. Queue. Direct digital control. Process-scheduling.

Classification system and/or index terms (if any)

Supplementary bibliographical information

Language
English

ISSN and key title

ISBN

Recipient's notes

Number of pages
61

Price

Security classification

Distribution by (name and address)

Z

A Facility for Executing Concurrent Processes in Pascal

Course Project of
"Modern Languages
for
Process Computers"

80-05-28

Participants:

Tommy Essebo
Rolf Johansson
Matz Lenells
Lars Nielsen

Adviser:

Leif Andersson
Hilding Elmqvist
Sven Erik Mattsson

Department of Automatic Control
Lund Institute of Technology

CONTENTS

CODEN: LUTFD2/(TFRT-7150)/1-092/(1978)

EVALUATION OF A SUBROUTINE FOR NELDER AND MEAD SEARCH

SVEN ERIK MATTSSON

Department of Automatic Control
Lund Institute of Technology
September 1978

Dokumentutgivare
Lund Institute of Technology
Handläggare Dept of Automatic Control
Sven Erik Mattsson
Författare
Sven Erik Mattsson

Dokumentnamn
REPORT
Dokumentbeteckning
LUTFD2/(TFRT-7150)/1-92/(1978)
Utgivningsdatum
Sept 1978
Ärendebeteckning
06T6

10T4

Dokumenttitel och undertitel
18T0
Evaluation of a Subroutine for Nelder and Mead Search

Referat (sammandrag)
24T0
This paper presents a Fortran IV subroutine called NELME that is a straightforward implementation of the simplex method for function minimization proposed by Nelder and Mead. NELME is written and documented according to the rules of the Scandinavian Control Library.

NELME has been tested and compared with a quasi-Newton algorithm without derivatives and with the Powell-Brent algorithm.

The result of the tests shows that it is hard to rank the algorithms.

Referat skrivet av
Author

Förslag till ytterligare nyckelord
44T0
Nelder-Mead search, nonlinear programming, nonlinear optimization, parametric optimization, simplex method

Klassifikationssystem och -klass(er)
50T0

Indextermer (ange källa)
52T0

Omfång
92 pages

Övriga bibliografiska uppgifter
56T2

Språk
English

Sekretessuppgifter
60T0

ISSN
60T4

ISBN
60T6

Dokumentet kan erhållas från
62T0
Department of Automatic Control
Lund Institute of Technology
PO Box 725, S-220 07 Lund 7, SWEDEN

Mottagarens uppgifter
62T4

Pris
66T0

DOKUMENTDATABLAD enligt SIS 62 10 12

Blankett LU 11:25 1976-07

Abstract

This paper presents a Fortran IV subroutine called NELME that is a straightforward implementation of the simplex method for function minimization proposed by Nelder and Mead. NELME is written and documented according to the rules of the Scandinavian Control Library.

NELME has been tested and compared with a quasi-Newton algorithm without derivatives and with the Powell-Brent algorithm.

The result of the tests shows that it is hard to rank the algorithms.

1. INTRODUCTION

This paper presents a Fortran IV subroutine called NELME that is a straightforward implementation of the Nelder and Mead search. NELME has been tested and compared with a quasi-Newton algorithm without derivatives and with the Powell-Brent algorithm.

Chapter 2 describes NELME and the computer codes can be found in Appendix A. Chapter 3 treats the problem of evaluating NELME and test results are presented. Appendix B contains additional test results.

2. NELME

NELME is a Fortran IV subroutine that is a straightforward implementation of a simplex method for function minimization proposed by Nelder and Mead (1964). In this implementation the reflection coefficient is chosen to 1, the contraction coefficient to 0.5 and the expansion coefficient to 2.

NELME is written and documented according to the rules of the Scandinavian Control Library and consequently the subroutine within itself contains all adequate information how it is used. A listing of the subroutine NELME can be found in Appendix A.

3. EVALUATION

In Himmelblau (1972) chapter 5 and in Brent (1973) section 7.7 the problem of evaluate algorithms for unconstrained nonlinear programming is discussed and a number of algorithms are evaluated. The inventors have of course evaluated their method and Himmelblau (1972) discusses explicitly the method of Nelder and Mead.

Five different functions were used to test NELME. The functions, whose minimum value is zero, were:

(1) Rosenbrock's parabolic valley (Rosenbrock (1960))

$$y = 100(x_2-x_1^2)^2 + (1-x_1)^2,$$

starting point (-1.2,1),
minimum point (1,1)

(2) Wood's function (Colville (1968))

$$y = 100(x_2-x_1^2)^2 + (1-x_1)^2 + 90(x_4-x_3^2)^2 + (1-x_3)^2$$

$$+ 10.1((x_2-1)^2 + (x_4-1)^2) + 19.8(x_2-1)(x_4-1),$$

starting point (-3,-1,-3,-1),
minimum point (1,1,1,1)

(3) Fletcher and Powell's helical valley (Fletcher and Powell (1963))

$$y = 100(x_3-10\theta(x_1,x_2))^2 + (\sqrt{(x_1^2+x_2^2)} - 1)^2 + x_3^2$$

$$\text{where } 2\pi\theta(x_1,x_2) = \begin{cases} \arctan(x_2/x_1), & x_1 > 0 \\ \pi + \arctan(x_2/x_1), & x_1 < 0 \end{cases}$$

starting point (-1,0,0)
minimum point (1,0,0)

(4) Powell's quartic function (Powell (1962))

$$y = (x_1 + 10x_2)^2 + 5(x_3-x_4)^2 + (x_2-2x_3)^4 + 10(x_1-x_4)^4,$$

starting point (3,-1,0,1)
minimum point (0,0,0,0)

(5) A quadratic function with truncated linear terms

$$y = x_1^2 + e(|x_1|) + 5\ x_2^2 + e(x_2),$$

where $e(z) = \text{sign}(z)\cdot n$, n is the largest integer $\leq |z|$,

starting points (±2, ±2),
minimum point (0,0)

The properties of functions 1-4 are discussed in Brent (1973).

The progress of NELME on function 1-5 has been studied for different sizes of the initial simplex. The number of iterations, the number of function evaluations and the value of the test quantity (TESTQ) were noted when the function value had been reduced to 10^{-j} for $j = 1, 3, 5, 7$. TESTQ (to be compared in the algorithm with the desired accuracy in the minimum value) is given by

$$\text{TESTQ} = \sqrt{\frac{1}{n+1}\sum_{i=1}^{n+1} (f(x_i)-f(x_c))^2}$$

where n is the dimension of the optimization problem, the x_i:s are the vertices and x_c the centroid of the polyhedron and $f(\cdot)$ the function to be minimized.

The results can be found in Appendix B. The maximum number of times the loss function could be evaluated was set to 2000. If a number in the column "number of evaluations required" is 2000 or greater, it means that the polyhedron has degenerated (to a point) before reaching the minimum point.

In order to get a feeling for the relative efficiency of NELME, the problems 1-5 were solved with the help of NUFLET and POWBRE. NUFLET is based on a quasi-Newton method without derivatives. The method is described in Fletcher (1971). POWBRE is an implementation of a version of Powell´s algorithm, modified by Brent (1973). Powell´s algorithm is a conjugate direction method without derivatives. The computer codes (in Fortran) can be found in Källström (1978). NUFLET and POWBRE have some additional parameters that have to be set. The values of these parameters have been chosen according to Källström (1978),
i.e.

in NUFLET	DFN = -0.2	(estimate of the likely reduction to be obtained in f(x) is $0.2\|f(x_o)\|$. DFN is only used on the first iteration so an order of magnitude estimate suffices)
	XM=[1,...,1]	(no scaling wanted)
	$HH=10^{-3}$	(the step length used when calculating the gradient is 10^{-3})
	$EPS=10^{-5}$	(the accuracy required in x_i is 10^{-5})
	MODE=1	(the initial estimate of the Hessian matrix is set to the unit matrix)

and

in POWBRE	DIST=1	(estimated distance from initial approximation to minimum)
	SCALE=1	(no scaling wanted)
	$TOL=10^{-6}$	(wanted relative accuracy in x)
	MODE=1	(the algorithm is started with the coordinate axes as search directions)
	ILLCO=.TRUE.	(the problem is supposed to be illconditioned)
	NSTOP=1	(number of iterations without progress before termination)

The results for NELME with VDIST=1 (initial size of the simplex), NUFLET and POWBRE on functions 1-4 are given in Table 1-4.

Table 1

Number of iterations (n_i) and number of function evaluations (n_f) to reduce Rosenbrock's function to < 10^{-j}.

j	NELME		NUFLET		POWBRE	
	n_i	n_f	n_i	n_f	n_i	n_f
1	59	185	23	99	37	102
2	75	231	29	140	47	127
3	82	253	31	152	52	140
4	83	256	32	158	52	140
5	86	268	33	164	52	140
6	88	276	34	170	57	152
7	89	279	35	176	57	152

Table 2

Number of iterations (n_i) and number of function evaluations (n_f) to reduce Wood's function to < 10^{-j}.

j	NELME		NUFLET		POWBRE	
	n_i	n_f	n_i	n_f	n_i	n_f
1	30	105	68	664	253	681
2	35	125	72	704	265	709
3	35	125	73	714	270	719
4	81	269	74	724	287	758
5	94	310	75	734	287	758
6	103	344	76	744	294	777
7	108	358	77	754	299	787

Table 3

Number of iterations (n_i) and number of function evaluations (n_f) to reduce Fletcher and Powell's helical valley to < 10^{-j}.

j	NELME		NUFLET		POWBRE	
	n_i	n_f	n_i	n_f	n_i	n_f
1	154	480	18	102	43	110
2	172	533	24	158	57	143
3	185	576	28	191	63	158
4	189	590	29	199	63	158
5	193	606	29	199	67	166
6	196	616	30	207	67	166
7	200	632	31	215	73	180

Table 4

Number of iterations (n_i) and number of function evaluations (n_f) to reduce Powell´s quartic function to < 10^{-j}.

j	NELME		NUFLET		POWBRE	
	n_i	n_f	n_i	n_f	n_i	n_f
1	31	99	9	61	38	91
2	56	177	10	67	43	101
3	68	206	12	79	48	111
4	70	218	22	161	55	130
5	76	238	23	171	65	152
6	78	246	25	191	72	171
7	85	276	27	211	77	182

The results speak for themselves and only a few comments will be given here. As seen from the results in Appendix B the size of the initial simplex has a significant effect on the speed of the convergence. Wood's function is an illustrative example. The result shown in Table 2 (the size of the initial simplex is 1) is very good, but NELME fails for some sizes of the initial simplex. POWBRE too had difficulties and stopped at (-0.988, 0.986, -0.949, 0.913) and declared that this point is a minimum. But when NSTOP was increased to 5, POWBRE found the minimum point and it is these results that are given in Table 2. Brent (1973) reports that with DIST=10 POWBRE after 191 linear searches and 452 function evaluations had reduced the function value to $6 \cdot 10^{-14}$.

This dependence, which is not *a priori* known, makes it difficult to rank NELME, NUFLET and POWBRE. Function 5 is discontinuous when x_1 or x_2 is an integer and one may suspect that NUFLET and POWBRE might fail. As seen from the results in Table 5 this is the case.

If no special information is available that can confirm the result of a "minimization algorithm" to be a close approximation to the solution, how can it then be decided whether the solution of the problem is found or not? This is a hard question. This problem is discussed in Murray (1972). He gives the following advice:

1) Check the rate of convergence
2) Restart the routine
3) Try other starting points, input parameters, rescale the variables
4) Try a different method

Table 5

Number of iterations (n_i) and number of function evaluations (n_f) to reduce function 5 to 10^{-7} for different starting points.

starting point	NELME		NUFLET		POWBRE	
	n_i	n_f	n_i	n_f	n_i	n_f
(2, 2)	28	94	8	44	12	34
(2,-2)	24	86	failed 1)		failed 2)	
(-2, 2)	31	104	8	69	12	34
(-2, -2)	26	90	12	87	failed 3)	

1) stopped at (1.9996, -2.0098)
2) stopped at $(-1.53\cdot10^{-3}, -2)$
3) stopped at $(1.01\cdot10^{-3}, -2)$

4. REFERENCES

Brent, R.P. (1978): Algorithms for minimization without derivatives, Prentice-Hall, Inc. Englewood Cliffs, N.Y. (7.7).

Colville, A.R. (1968): A comparative study of nonlinear programming codes, IBM New-York Scientific Center, Report 320-2949.

Elmqvist, Tyssö, Wieslander (1976): Programming and documentation rules for subroutine libraries - designed for the Scandinavian Control Library, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden. CODEN:LUTFD2/(TFRT-3139)/1-041/(1976).

Fletcher, R., and Powell, M.Y.D. (1963): A rapidly convergent descent method for minimization, The Computer Journal, Vol. 6, p. 163,

Fletcher. R. (1971): Fortran subroutines for minimization by quasi-Newton methods, Report AERE-R7125, Harwell.

Himmelblau, D.M. (1972): Applied nonlinear programming, Mc Gray-Hill Book Company, New York (p. 148-157, 190-217).

Källström, C.G. (1978): LISPID-user´s manual, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden. CODEN: LUTFD2/(TFRT-7147)/1-147/(1978)

Murray, W. (1972): Numerical methods for unconstrained optimization, Academic Press, London (p. 119-122).

Nelder, Y.A. and Mead, R. (1964): A simplex method for function minimization, The Computer Journal, Vol. 7, p. 308.

Powell, M.Y.D. (1962): An iterative method for finding stationary value of a function of several variables, The Computer Journal, Vol. 5. p. 147.

Rosenbrock, H. (1960): An automatic method for finding the greatest or least value of a function, The Computer Journal, Vol.3, p.175.

APPENDIX A

Computer Codes

```
C      NAME: NELME                                        NUMBER:
C      -----                                              -------
C
C      SUBTITLE: THE FLEXIBLE POLYHEDRON SEARCH OF NELDER AND MEAD
C      ---------
C
C      LANGUAGE: FORTRAN IV
C      ---------
C
C      KEYWORDS: NONLINEAR PROGRAMMING, NONLINEAR OPTIMIZATION,
C      ---------  PARAMETRIC OPTIMIZATION, NELDER-MEAD SEARCH
C
C      IMPLEMENTOR: SVEN ERIK MATTSSON                    DATE: 1978-06-19
C      ------------                                       -----
C
C      INSTITUTE: DEPARTMENT OF AUTOMATIC CONTROL
C      ----------  LUND INSTITUTE OF TECHNOLOGY, SWEDEN
C
C      ACCEPTED:                                          VERSION:
C      ---------                                          --------
C
C=====================================================================
C
C      PURPOSE
C      =======
C
C      TO FIND THE MINIMUM OF A FUNCTION F(X) OF SEVERAL VARIABLES
C      WITHOUT CALCULATING DERIVATIVES.
C
C      USAGE
C      =====
C
C      PROGRAM TYPE: SUBROUTINE
C      -------------
C
C      ARGUMENTS:
C      ----------
C
C      NELME(FUNC,X,FMIN,N,MODE,VDIST,MAXFN,EPS,IERR,IPRINT,PRINT,WX,WF)
C
C      FUNC    - NAME OF SUBROUTINE WHICH COMPUTES THE LOSS FUNCTION
C                TO BE MINIMIZED. (I)
C      X       - REAL VECTOR, SIZE(N). (I/O)
C                SEE NOTE 1 AND 2!
C      FMIN    - REAL CONTAINING, IF SUCCESS, THE MINIMUM VALUE. (O)
C                SEE NOTE 2!
C      N       - ACTUAL DIMENSION OF THE OPTIMIZATION PROBLEM. (I)
C                N>1.
C      MODE    - INDICATOR TO CONTROL THE CALCULATION OF THE STARTING
C                POLYHEDRON. (I)
C                HAS TO BE 0 OR 1. SEE NOTE 1!
C      VDIST   - DISTANCE BETWEEN TWO VERTICES IN THE STARTING
C                POLYHEDRON. (I)
C                SEE NOTE 1!
C      MAXFN   - MAXIMUM NUMBER OF TIMES FUNC CAN BE CALLED. (I)
C                MAXFN>N+1
C      EPS     - STOPPING CRITERION. (I)
C                SEE NOTE 2 AND 3!
C      IERR    - ERROR INDICATOR. (O)
C                -1: THE SEARCH IS TO BE CONTINUED. USED ONLY IN PRINT.
C                 0: SUCCESS
C                 1: INCORRECT CALL OF NELME
C                    N<2
C                    INVALID MODE-VALUE
```

```
C                 MAXFN(N+1
C              2: FUNC HAS BEEN CALLED WITH AN ILLEGAL X AND THIS
C                 X CAN BE FOUND IN NELME'S WX(.,N+2). SEE NOTE 2!
C              3: FUNC CALLED MAXFN TIMES. SEE NOTE 2!
C     IPRINT - PRINT INDICATOR, (I)
C              PRINTING OCCURS ON ENTRY, EVERY ABS(IPRINT) ITERATIONS
C              AND ON EXIT.
C              ALL INTERMEDIATE PRINTING CAN BE SUPPRESSED BY
C              SETTING IPRINT>MAXFN.
C              ALL PRINTING CAN BE SUPPRESSED BY SETTING IPRINT=0.
C     PRINT  - NAME OF SUBROUTINE WHICH PERFORMS THE PRINTING, (I)
C              IF IPRINT=0, PRINT IS NOT CALLED.
C     WX     - WORK AREA, SIZE(N,N+4)
C              SEE NOTE 1!
C     WF     - WORK AREA, SIZE(N+4)
C
C     NOTE:
C     -----
C     1) THE INITIAL VALUES CAN BE GIVEN IN TWO DIFFERENT WAYS:
C        MODE=0: NELME CALCULATES AN INITIAL POLYHEDRON WITH X AS THE
C                CENTROID AND WITH THE DISTANCE BETWEEN TWO VERTICES
C                EQUAL TO VDIST AND STORES THESE VERTICES IN THE
C                FIRST N+1 COLUMNS OF WX. THIS IS THE NORMAL MODE.
C        MODE=1: THE INITIAL POLYHEDRON IS GIVEN IN THE N+1 FIRST
C                COLUMS OF WX. THIS MODE CAN BE USED IF THE ROUTINE IS
C                TO BE RESTARTED. VDIST IS NOT USED IN THIS MODE.
C     2) AFTER THE EXIT FROM NELME AND IF IERR IS NOT EQUAL TO 1, THE
C        RESULTS OF THE SEARCH ARE IN X AND FMIN.
C     3) THE STOPPING CRITERION USED FOR HALTING THE ROUTINE IS
C        SQRT(SUM((F(X(I))-F(XC))**2)/(N+1)) < EPS,
C        WHERE THE X(I):S ARE THE VERTICES AND XC THE CENTROID OF THE
C        POLYHEDRON AND F(.) THE FUNCTION TO BE MINIMIZED.
C        THE SUCCESS OF THE CRITERION DEPENDS ON THE POLYHEDRON NOT
C        BECOMING TOO SMALL IN RELATION TO THE CURVATURE OF THE
C        SURFACE UNTIL THE FINAL MINIMUM IS REACHED.
C
C
C     USER SUPPLIED SUBROUTINES:
C     --------------------------
C
C     SUBROUTINE FUNC(X,F,FX,FXX,N,ICONT,IERR1,IFXX)
C
C     DEFINES THE FUNCTION F(X) TO BE MINIMIZED IN NELME.
C     X      - ARGUMENT, SIZE(N),(I)
C     F      - RETURNED FUNCTION VALUE, (O)
C     FX     - GRADIENT VECTOR, SIZE(N),(O)
C              NOT USED IN NELME
C     FXX    - HESSIAN MATRIX, SIZE(N,N), DIMENSIONED(IFX,.), (O)
C              NOT USED IN NELME
C     N      - DIMENSION OF X, (I)
C     ICONT  - CONTROL VARIABLE, (I)
C               0: COMPUTE F
C               1: COMPUTE F, FX
C               2: COMPUTE F, FX, FXX
C              ICON IS SET TO ZERO AT CALLING FROM NELME
C     IERR1  - ERROR INDICATOR OF FUNC, (O)
C               0: SUCCESS
C               1: DIFFICULTIES
C              -1: ILLEGAL CALL OF FUNC, FX OR FXX WANTED (ICONT= 1,2)
C                  BUT THE THE COMPUTATION OF FX OR FXX IS NOT
C                  IMPLEMENTED IN FUNC
C     IFXX   - DIMENSION PARAMETER OF IFXX, (I)
C
C     THE ACTUAL SUBROUTINE NAME MUST BE DECLARED EXTERNAL IN THE
```

```
C     PROGRAM THAT CALLS NELME. IF ICONT IS 0 (1) THEN FX AND FXX
C     (FXX) MUST NOT BE USED IN FUNC IN ORDER TO MAKE IT POSSIBLE
C     TO CALL FUNC WITH DUMMIES.
C
C
C     SUBROUTINE PRINT(WX,WF,N,NITE,NFN,TESTQ,IPRINT,IERR)
C
C     AT THE BEGINNING OF THE FIRST ITERATION, ON EVERY SUBSEQUENT
C     ABS(IPRINT) ITERATIONS AND ON EXIT OF NELME PRINT IS CALLED BY
C     NELME IN ORDER TO PRINT A MESSAGE. IF IPRINT=0, PRINT IS NOT
C     CALLED AND PRINT MAY BE A DUMMY.
C
C     WX     - DEFINES THE POLYHEDRON, SIZE(N,N+4),DIMENSIONED(N,.),
C              (I)
C              THE FIRST N+1 COLUMNS OF WX CONTAIN THE VERTICES OF
C              THE POLYHEDRON.
C     WF     - VECTOR CONTAINING THE VALUES OF THE OBJECTIVE FUNCTION
C              AT THE VERTICES, SIZE(N+4), (I)
C     N      - ACTUAL DIMENSION OF THE OPTIMIZATION PROBLEM, (I)
C     NITE   - ITERATION NUMBER, (I)
C     NFN    - NUMBER OF CALLS OF FUNC, (I)
C     TESTQ  - OBTAINED STOPPING CRITERION, (I)
C     IPRINT - PRINT INDICATOR, (I)
C              SEE THE ARGUMENT LIST OF NELMF.
C     IERR   - ERROR INDICATOR OF NELME, (I)
C              SEE THE ARGUMENT LIST OF NELME.
C
C     THE ACTUAL SUBROUTINE NAME MUST BE DECLARED EXTERNAL IN THE
C     PROGRAM THAT CALLS NELME. IF IPRINT IS SET TO ZERO AND A DUMMY
C     IS GOING TO BE USED AS PRINT AND IF THIS DUMMY IS NOT DECLARED
C     EXTERNAL, IT NEED NOT BE INCLUDED AT LOADING.
C     THE SUBROUTINE PNELME IN THE LIBRARY CAN BE USED AS THE ACTUAL
C     SUBROUTINE PRINT.
C
C
C     METHOD
C     ======
C
C     THIS ROUTINE IS A STRAIGHTFORWARD IMPLEMENTATION OF A FLEXIBLE
C     POLYHEDRON SEARCH DESCRIBED IN NELDER,J.A. AND MEAD,R. (1964).
C     THE REFLECTION COEFFICIENT IS CHOSEN TO 1, THE CONTRACTION
C     COEFFICIENT TO 1/2 AND THE EXPANSION COEFFICIENT TO 2.
C     THE METHOD OF CALCULATING A STARTING POLYHEDRON CAN BE FOUND IN
C     HIMMELBLAU,D.M. (1972).
C
C
C     REFERENCES:
C     -----------
C     1. NELDER,J.A. AND MEAD,R. (1964) 'A SIMPLEX METHOD FOR FUNCTION
C        MINIMIZATION', THE COMPUTER JOURNAL, VOL. 7, P. 308
C     2. HIMMELBLAU,D.M. (1972) 'APPLIED NONLINEAR PROGRAMMING',
C        MC GRAW HILL BOOK COMPANY, NEW YORK, P. 148-157
C
C
C     CHARACTERISTICS
C     ===============
C
C     SIZE:
C     -----
C     PDP 15: 1716
C     PDP 11: 1570
C
C     LIBRARY SUBROUTINES REQUIRED: RMACON, MMOVE
C     -------------------------------------------
```

```
C
C     INTERNAL SUBROUTINES REQUIRED: NEWX
C     ------------------------------------
C
C---------------------------------------------------------------------------
C
C
      SUBROUTINE NELME(FUNC,X,FMIN,N,MODE,VDIST,MAXFN,EPS,IERR,
     *                 IPRINT,PRINT,WX,WF)
C
      DIMENSION X(1),WX(N,1),WF(1)
C
C      ALFA : REFLECTION COEFFICIENT
C     -BETA : CONTRACTION COEFFICIENT
C     -GAMMA: EXPANSION COEFFICIENT
C
      DATA ALFA,BETA,GAMMA/1.,-0.5,-2./
      DATA ICONT /0/
C
C     VALIDITY TEST AND CALCULATION OF INITIAL VALUES
C
      NPLUS1=N+1
      NPLUS2=N+2
      NPLUS3=N+3
      NPLUS4=N+4
      FLN=FLOAT(N)
      FLNPL1=FLOAT(NPLUS1)
      A=100.0*RMACON(3)
      IF(EPS.LT.A) EPS=A
      IERR=1
      IF(N.LT.2 .OR. MAXFN.LE.NPLUS1) RETURN
C
C     IS THE INITIAL POLYHEDRON GIVEN OR HAS IT TO BE CALCULATED?
C
      IF(MODE.EQ.1) GO TO 50
      IF(MODE.NE.0) RETURN
C
C     CALCULATE THE INITIAL POLYHEDRON. LET X BE THE CENTROID AND
C     VDIST THE DISTANCE BETWEEN TWO VERTICES AND STORE THE VERTICES
C     IN THE FIRST N+1 COLUMNS OF WX
C
      VDIST1=VDIST
      IF(ABS(VDIST1).LE.A) VDIST1=A
      A=SQRT(2.0)
      B=SQRT(FLNPL1)
      D3=-VDIST1/(A*B)
      C=VDIST1/(A*FLN)
      B=B-1.
      D2=C*B+D3
      D1=C*(B+FLN)+D3
      DO 40 I=1,N
      A=X(I)
      B=A+D2
      WX(I,NPLUS1)=A+D3
      DO 30 J=1,N
      WX(I,J)=B
   30 CONTINUE
      WX(I,I)=A+D1
   40 CONTINUE
C
C     NOW THE INITIAL POLYHEDRON EXISTS. CALCULATE THE VALUE OF THE
C     OBJECTIVE FUNCTION AT THE VERTICES OF THE INITIAL POLYHEDRON.
C
   50 CONTINUE
```

```
      IERR=-1
      DO 60 IX=1,NPLUS1
      CALL FUNC(WX(1,IX),WF(IX),DUM1,DUM2,N,ICONT,IERR1,IDUM2)
      IF(IERR1.NE.0) GO TO 400
   60 CONTINUE
C
C     NFN-NUMBER OF CALLS OF FUNC
C     NITE-ITERATION NUMBER
C     NPRINT-COUNTER TO CONTROL PRINT
C     IABSPR-ABS(IPRINT)
C
      NFN=NPLUS1
      NITE=0
      IABSPR=IABS(IPRINT)
C
C     PRINT START MESSAGE.
C
      TESTQ=0.
      DO 65 J=NPLUS2,NPLUS4
      WF(J)=0.
      DO 65 I=1,N
      WX(I,J)=0.
   65 CONTINUE
      IF(IPRINT.NE.0) CALL PRINT(WX,WF,N,NITE,NFN,TESTQ,IPRINT,IERR)
      NPRINT=IABSPR
C
C     END OF INITIALIZATION
C
C-----------------------------------------------------------------------
C
C     START OF AN ITERATION LOOP
C
   70 CONTINUE
      NITE=NITE+1
C
C     FIND THE VERTICES WITH THE HIGHEST AND LOWEST VALUE OF
C     THE OBJECTIVE FUNCTION AND LET WX(.,NPLUS2) BE THE CENTROID OF
C     ALL THE VERTICES EXCLUDING THAT WITH THE HIGHEST VALUE.
C
      IHIGH=NPLUS1
      ILOW=NPLUS1
      DO 80 I=1,N
      IF(WF(I).GT.WF(IHIGH)) IHIGH=I
      IF(WF(I).LT.WF(ILOW)) ILOW=I
   80 CONTINUE
      DO 100 I=1,N
      A=-WX(I,IHIGH)
      DO 90 J=1,NPLUS1
      A=A+WX(I,J)
   90 CONTINUE
      WX(I,NPLUS2)=A/FLN
  100 CONTINUE
C
C     CALCULATE THE FUNCTION VALUE AT WX(.,NPLUS2).
C
      NFN=NFN+1
      IX=NPLUS2
      CALL FUNC(WX(1,NPLUS2),WF(NPLUS2),DUM1,DUM2,N,ICONT,IERR1,IDUM2)
      IF(IERR1.NE.0) GO TO 500
C
C     CALCULATE THE REFLECTION POINT
C     WX(.,NPLUS3)=WX(.,NPLUS2)+ ALFA*(WX(.,NPLUS2)-WX(.,IHIGH))
C     AND VALUE (WF(NPLUS3)) AT THAT POINT.
C
```

```
      CALL NEWX(WX(1,NPLUS2),WX(1,IHIGH),ALFA,N,WX(1,NPLUS3))
      NFN=NFN+1
      IX=NPLUS3
      CALL FUNC(WX(1,NPLUS3),WF(NPLUS3),DUM1,DUM2,N,ICONT,IERR1,IDUM2)
      IF(IERR1.NE.0) GO TO 500
C
C     TEST FUNCTION VALUE AT REFLECTION POINT
C
      IF(WF(NPLUS3).GE.WF(ILOW)) GO TO 120
C
C     WF(NPLUS3) < WF(ILOW).
C     TRY EXPANSION! CALCULATE EXPANSION POINT
C     WX(.,NPLUS4)= WX(.,NPLUS2)+GAMMA*(WX(.,NPLUS2)-WX(.,NPLUS3))
C     AND VALUE (WF(NPLUS4)) AT THAT POINT.
C
      CALL NEWX(WX(1,NPLUS2),WX(1,NPLUS3),GAMMA,N,WX(1,NPLUS4))
      NFN=NFN+1
      IX=NPLUS4
      CALL FUNC(WX(1,NPLUS4),WF(NPLUS4),DUM1,DUM2,N,ICONT,IERR1,IDUM2)
      IF(IERR1.NE.0) GO TO 500
C
C     TEST FUNCTION VALUE AT EXPANSION POINT.
C
      IF(WF(NPLUS4).GE.WF(ILOW)) GO TO 110
C
C     ACCEPT WX(.,NPLUS4) AS A NEW VERTEX OF THE POLYHEDRON.
C     REPLACE WX(.,IHIGH) BY WX(.,NPLUS4).
C
      CALL MMOVE(WX(1,NPLUS4),WX(1,IHIGH),N,1,0,N,N)
      WF(IHIGH)=WF(NPLUS4)
      GO TO 170
C
C     ACCEPT WX(.,NPLUS3) AS A NEW VERTEX OF THE POLYHEDRON.
C     REPLACE WX(.,IHIGH) BY WX(.,NPLUS3).
C
  110 CONTINUE
      CALL MMOVE(WX(1,NPLUS3),WX(1,IHIGH),N,1,0,N,N)
      WF(IHIGH)=WF(NPLUS3)
      GO TO 170
C
C     AT THIS POINT OF THE PROGRAM IT IS KNOWN THAT WF(NPLUS3) >
C     WF(ILOW), BUT IS WF(NPLUS3) > WF(I) FOR ALL I=1,...,N+1 AND
C     I NOT EQUAL TO IHIGH ?
C
  120 CONTINUE
      DO 130 I=1,NPLUS1
      IF(WF(NPLUS3).GT.WF(I)) GO TO 130
      IF(I.EQ.IHIGH) GO TO 130
      GO TO 110
  130 CONTINUE
C
C     YES, WF(NPLUS3) > WF(I) FOR ALL I=1,...,N+1 AND I=/IHIGH, BUT
C     IS WF(NPLUS3) > WF(IHIGH) ?
C
      IF(WF(NPLUS3).GE.WF(IHIGH)) GO TO 150
C
C     NO, WF(NPLUS3) < WF(IHIGH). ACCEPT WX(.,NPLUS3) AS A NEW VERTEX
C     OF THE POLYHEDRON.
C     REPLACE WX(.,IHIGH) BY WX(.,NPLUS3).
C
      CALL MMOVE(WX(1,NPLUS3),WX(1,IHIGH),N,1,0,N,N)
      WF(IHIGH)=WF(NPLUS3)
      IHIGH=NPLUS1
      DO 140 I=1,N
```

```
      IF(WF(I).GT.WF(IHIGH)) IHIGH=I
  140 CONTINUE
C
C     TRY CONTRACTION! CALCULATE CONTRACTION POINT
C     WX(.,NPLUS4)=WX(.,NPLUS2)+BETA*(WX(.,NPLUS2)+WX(.,IHIGH))
C     AND VALUE AT (WF(NPLUS4) THAT POINT.
C
      CALL NEWX(WX(1,NPLUS2),WX(1,IHIGH),BETA,N,WX(1,NPLUS4))
      NFN=NFN+1
      IX=NPLUS4
      CALL FUNC(WX(1,NPLUS4),WF(NPLUS4),DUM1,DUM2,N,ICONT,IERR1,IDUM2)
      IF(IERR1.NE.0) GO TO 500
C
C     TEST FUNCTION VALUE AT CONTRACTION POINT.
C
      IF(WF(NPLUS4).GT.WF(IHIGH)) GO TO 150
C
C     WF(NPLUS4) < WF(IHIGH). ACCEPT WX(.,NPLUS4) AS A NEW VERTEX OF
C     THE POLYHEDRON.
C     REPLACE WX(.,IHIGH) BY WX(.,NPLUS3).
C
      CALL MMOVE(WX(1,NPLUS4),WX(1,IHIGH),N,1,0,N,N)
      WF(IHIGH)=WF(NPLUS4)
      GO TO 170
C
C     REDUCTION: REPLACE ALL WX(.,I) BY WX(.,ILOW)-0.5*(WX(.,ILOW)-
C                WX(.,I))
C
  150 CONTINUE
      NFN=NFN+N
      DO 160 IX=1,NPLUS1
      IF(IX.EQ.ILOW) GO TO 160
      CALL NEWX(WX(1,ILOW),WX(1,IX),-0.5,N,WX(1,IX))
      CALL FUNC(WX(1,IX),WF(IX),DUM1,DUM2,N,ICONT,IERR1,IDUM2)
      IF(IERR1.NE.0) GO TO 500
  160 CONTINUE
C
C     TEST OF CONVERGENCE
C
C     CALCULATE THE TEST QUANTITY
C     TESTQ=SQRT(SUM((WF(I)-WF(NPLUS2))**2/(N+1)))
C
  170 CONTINUE
      A=0
      DO 180 I=1,NPLUS1
      B=WF(I)-WF(NPLUS2)
      A=A+B*B
  180 CONTINUE
      TESTQ=SQRT(A/FLNPL1)
      IF(TESTQ.LT.EPS) GO TO 200
C
C     THE CONVERGENCE CRITERION IS NOT FULFILLED.
C     TEST NUMBER OF FUNCTION EVALUATIONS
C
      IF(NFN.GE.MAXFN) GO TO 190
C
C     PRINT A MESSAGE AND DO A NEW ITERATION.
C
      NPRINT=NPRINT-1
      IF(NPRINT.GT.0 .OR. IPRINT.EQ.0) GO TO 70
      CALL PRINT(WX,WF,N,NITE,NFN,TESTQ,IPRINT,IERR)
      NPRINT=IABSPR
      GO TO 70
C
```

```
C       END OF AN ITERATION LOOP
C------------------------------------------------------------------------------
C
C       FUNC HAS BEEN CALLED MORE THAN MAXFN TIMES.
C
  190 CONTINUE
      TERR=3
      GO TO 205
C
C       THE CONVERGENCE CRITERION IS FULFILLED, BUT THIS DOES NOT
C       CERTAINLY MEAN THAT A LOCAL MINIMUM IS FOUND.
C
  200 CONTINUE
      IERR=0
C
C       OUTPUT THE RESULT, PRINT A MESSAGE AND RETURN.
C
  205 CONTINUE
      DO 210 I=1,NPLUS1
      IF(WF(I).LT.WF(ILOW)) ILOW=I
  210 CONTINUE
      CALL MMOVE(WX(1,ILOW),X,N,1,0,N,N)
      FMIN=WF(ILOW)
      IF(IPRINT.NE.0) CALL PRINT(WX,WF,N,NITE,NFN,TEST0,IPRINT,IERR)
      RETURN
C
C       FUNC HAS BEEN CALLED WITH AN ILLEGAL X (WX(.,IX)) IN THE
C       INITIALIZATION PART.
C
  400 CONTINUE
      TEST0=0.
      DO 410 J=IX,NPLUS1
      WF(J)=0.
  410 CONTINUE
      DO 420 J=NPLUS2,NPLUS4
      WF(J)=0.
      DO 420 I=1,N
      WX(I,J)=0.
  420 CONTINUE
C
C       FUNC HAS BEEN CALLED WITH AN ILLEGAL X (WX(.,IX)).
C       MOVE THIS X TO WX(.,NPLUS2).
C
  500 CONTINUE
      IERR=2
      IF(IX.NE.NPLUS2) CALL MMOVE(WX(1,IX),WX(1,NPLUS2),N,1,0,N,N)
      GO TO 205
      END
```

```
      SUBROUTINE NEWX(X1,X2,ALFA,N,X3)
C
C     CALCULATES X3=X1+ALFA*(X1-X2)
C
      DIMENSION X1(1),X2(1),X3(1)
      DO 10 I=1,N
      X3(I)=X1(I)+ALFA*(X1(I)-X2(I))
   10 CONTINUE
      RETURN
      END
```

```
C     NAME: PNELME                                        NUMBER:
C     -----                                               -------
C
C     SUBTITLE: I/O ROUTINE OF THE SUBROUTINE NELME
C     ---------
C
C     LANGUAGE: FORTRAN IV
C     ---------
C
C     KEYWORDS:
C     ---------
C
C     IMPLEMENTOR: SVEN ERIK MATTSSON                     DATE: 1978-06-19
C     ------------                                        -----
C
C     INSTITUTE: DEPARTMENT OF AUTOMATIC CONTROL
C     ----------  LUND INSTITUTE OF TECHNOLOGY, SWEDEN
C
C     ACCEPTED:                                           VERSION:
C     ---------                                           --------
C
C=========================================================================
C
C     PURPOSE
C     =======
C
C     TO BE USED AS THE ACTUAL SUBROUTINE PRINT IN THE SUBROUTINE NELME.
C     SEE THE DOCUMENTATION OF THE SUBROUTINE NELME.
C
C     USAGE
C     =====
C
C     PROGRAMTYPE: SUBROUTINE
C     ------------
C
C     ARGUMENTS: SEE THE DOCUMENTATION OF THE SUBROUTINE NELME.
C     ----------
C
C     NOTE:
C     -----
C
C     IN THE DOCUMENTATION OF NELME IT IS DESCRIBED HOW THE CALLING OF
C     PRINT IN NELME IS CONTROLLED.
C     THE PRINTING STARTS ON A NEW PAGE WITH THE TEXT
C     'PRINT FROM NELME'.
C     THE PRINTING IS IN THE FORM
C
C     ITERATION NO.=I6 NO. OF CALLS OF FUNC=I6 STOPPING CRITERION=G6.3
C     F(X)      X
C     'F(X1)' 'X1(1)' 'X1(2)' ...' X1(7)'
C             'X1(8)' 'X1(9)' ...  X1(14)'
C              ................... ........
C              ..............'X1(N+1)'
C     'F(X2)' 'X2(1)' ......................
C              ............................
C     .....................................
C     'F(XNPLUS1)'..........................
C                           ................
C
C     THE FUNCTION VALUES ('FX(XI)' ) HAVE THE FORMAT G14.7 AND THE
C     COORDINATES OF THE VERTICES (XI(J)) HAVE THE FORMATS G16.7.
C     IF IPRINT<0, THE PRINTING OF ALL VERTICES AND THEIR FUNCTION
C     VALUES IS SUPPRESSED AND ONLY THE VERTEX WITH THE LOWEST VALUE
C     AND ITS FUNCTION VALUE IS PRINTED.
```

```
C       ON EXIT FROM NELME A SHORT MESSAGE IS PRINTED THAT TELLS WHY
C       NELME EXITS.
C
C-----------------------------------------------------------------------
C
C
        SUBROUTINE FNELME(WX,WF,N,NITE,NFN,TESTQ,IPRINT,IERR)
C
        DIMENSION WX(N,1),WF(1)
        DATA LOUT/6/
C
        NPLUS1=N+1
C
C       ON ENTRY OF NELME, START ON A NEW PAGE.
C
        IF(NITE.EQ.0) WRITE(LOUT,500)
  500   FORMAT(1H1,16HPRINT FROM NELME/1X,16(1H*)//)
C
C       PRINT NITE, NFN AND TESTQ.
C
        WRITE(LOUT,510) NITE,NFN,TESTQ
  510   FORMAT(/1X,14HITERATION NO.=,I6,2X,21HNO. OF CALLS OF FUNC=,I6,
     *        2X,19HSTOPPING CRITERION=,G10.3)
        WRITE(LOUT,520)
  520   FORMAT(1X,4HF(X),11X,1HX)
C
C       IS THE PRINTING OF ALL VERTICES SUPPRESSED?
C
        IF(IPRINT.LT.0) GO TO 10
C
C       NO, PRINT ALL VERTICES AND THEIR VALUES.
C
        ISTART=1
        IEND=NPLUS1
        GO TO 30
C
C       YES, PRINT ONLY THE VERTEX WITH THE LOWEST VALUE.
C       FIND THIS VERTEX.
C
   10   CONTINUE
        ILOW=NPLUS1
        DO 20 I=1,N
        IF(WF(I).LT.WF(ILOW)) ILOW=I
   20   CONTINUE
        ISTART=ILOW
        IEND=ILOW
   30   CONTINUE
        DO 40 I=ISTART,IEND
   40   WRITE(LOUT,530) WF(I),(WX(J,I),J=1,N)
  530   FORMAT(1X,G14.7,7G16.7/(15X,7G16.7))
C
C       IF IERR=1, THE SEARCH OF BETTER VALUES IS GOING TO BE CONTINUED.
C
        IF(IERR.EQ.-1) RETURN
C
C       IF IERR=0, THE STOPPING CRITERION IS SATISFIED.
C
        IF(IERR.NE.0) GO TO 50
        WRITE(LOUT,540)
  540   FORMAT(//1X,28HSTOPPING CRITERION SATISFIED/1H1)
        RETURN
C
C       IF IERR=2, FUNC HAS BEEN CALLED WITH AN ILLEGAL X THAT CAN BE
C       FOUND IN WX(.,NPLUS2).
```

```
C
   50 CONTINUE
      IF(IERR.NE.2) GO TO 60
      NPLUS2=N+2
      WRITE(LOUT,550) (WX(I,NPLUS2),I=1,N)
  550 FORMAT(//1X,40HFUNC HAS BEEN CALLED WITH THE ILLEGAL X=
     *          /(1X,8G16.7))
      WRITE(LOUT,560)
  560 FORMAT(1H1)
      RETURN
C
C      IERR=3.
C
   60 WRITE(LOUT,570)
  570 FORMAT(//1X,23HFUNC CALLED MAXFN TIMES/1H1)
      RETURN
      END
```

APPENDIX B

Results from the minimization of five functions using NELME.

If a number in the column "number of evaluations required" is 2000 or greater, it means that NELME failed. The polyhedron has degenerated before reaching the minimum point.

```
PRINT FROM NELME
****************

MINIMIZATION OF F(X)=100*(X(2)-X(1)**2)**2+(1-X(1))**2
UNTIL F(X) < 1.E-1 FOR DIFFERENT SIZES OF THE INITIAL POLYHEDRON.
STARTING POINT IS (-1.2, 1.0).
```

SIZE OF THE INITIAL POLYHEDRON	NUMBER OF ITERATIONS REQUIRED	NUMBER OF EVALUATIONS REQUIRED	TESTQ OF NELME
0.1	64	204	0.141E-01
0.2	51	159	0.213E-01
0.3	70	214	0.294E-01
0.4	60	191	0.406E-01
0.5	66	209	0.288E-01
0.6	59	191	0.184E-01
0.7	72	220	0.150E-01
0.8	47	151	0.443E-01
0.9	70	220	0.285E-02
1.0	59	185	0.195E-01
1.1	63	200	0.547E-02
1.2	63	198	0.536E-02
1.3	51	164	0.712E-01
1.4	64	202	0.651E-01
1.5	48	149	0.207E-01
1.6	58	187	0.181E-01
1.7	63	198	0.498E-01
1.8	63	192	0.155E-01
1.9	55	175	0.359E-02
2.0	72	229	0.925E-02
2.1	59	183	0.338E-01
2.2	53	171	0.342E-01
2.3	69	214	0.133E-01
2.4	58	187	0.249E-01
2.5	64	202	0.464E-01
2.6	86	271	0.161E-01
2.7	89	272	0.242E-01
2.8	80	252	0.668E-01
2.9	81	255	0.538E-01
3.0	92	284	0.457E-01
3.1	8	32	0.110
3.2	10	38	0.743E-01
3.3	6	26	0.877
3.4	1	8	145.
3.5	6	28	0.311
3.6	5	24	2.38
3.7	102	321	0.115E-01
3.8	99	309	0.990E-02
3.9	30	98	0.171E-01
4.0	43	143	0.345E-01

PRINT FROM NELME
****

MINIMIZATION OF F(X)=100*(X(2)-X(1)**2)**2+(1-X(1))**2
UNTIL F(X) < 1.E-1 FOR DIFFERENT SIZES OF THE INITIAL POLYHEDRON.
STARTING POINT IS (-1.2, 1.0).

SIZE OF THE INITIAL POLYHEDRON	NUMBER OF ITERATIONS REQUIRED	NUMBER OF EVALUATIONS REQUIRED	TEST0 OF NELME
4.1	113	354	0.371E-01
4.2	95	297	0.495E-01
4.3	111	353	0.411E-02
4.4	96	306	0.295E-01
4.5	84	271	0.198E-01
4.6	88	280	0.274E-01
4.7	93	296	0.261E-01
4.8	112	344	0.133E-01
4.9	92	288	0.162E-01
5.0	92	286	0.133E-01
5.1	87	277	0.122E-01
5.2	94	295	0.205E-01
5.3	105	321	0.241E-01
5.4	87	275	0.733E-01
5.5	105	330	0.861E-02
5.6	101	316	0.413E-01
5.7	111	356	0.354E-02
5.8	95	296	0.304E-01
5.9	96	309	0.720E-02
6.0	90	285	0.113E-01
6.1	105	326	0.506E-01
6.2	87	276	0.620E-01
6.3	67	213	0.215E-01
6.4	57	184	0.335E-01
6.5	73	236	0.302E-01
6.6	113	354	0.409E-01
6.7	95	296	0.129E-01
6.8	108	343	0.258E-01
6.9	102	325	0.744E-01
7.0	103	327	0.737E-01
7.1	92	283	0.412E-01
7.2	96	305	0.264E-01
7.3	109	341	0.313E-01
7.4	122	387	0.666E-02
7.5	95	306	0.259E-01
7.6	102	316	0.391E-01
7.7	116	361	0.232E-01
7.8	111	346	0.116E-01
7.9	110	343	0.153E-01
8.0	100	307	0.174E-01

PRINT FROM NELME
****

MINIMIZATION OF F(X)=100*(X(2)-X(1)**2)**2+(1-X(1))**2
UNTIL F(X) < 1.E-3 FOR DIFFERENT SIZES OF THE INITIAL POLYHEDRON.
STARTING POINT IS (-1.2, 1.0).

SIZE OF THE INITIAL POLYHEDRON	NUMBER OF ITERATIONS REQUIRED	NUMBER OF EVALUATIONS REQUIRED	TESTQ OF NELME
0.1	83	263	0.105E-02
0.2	70	216	0.201E-03
0.3	85	261	0.859E-03
0.4	77	246	0.191E-02
0.5	81	255	0.786E-03
0.6	72	233	0.179E-02
0.7	85	259	0.219E-02
0.8	67	212	0.149E-02
0.9	89	273	0.519E-03
1.0	82	253	0.366E-03
1.1	82	261	0.734E-03
1.2	80	251	0.192E-02
1.3	77	242	0.506E-03
1.4	80	252	0.117E-02
1.5	61	188	0.357E-02
1.6	75	240	0.933E-03
1.7	79	247	0.310E-03
1.8	74	226	0.112E-01
1.9	73	231	0.131E-02
2.0	84	266	0.766E-03
2.1	89	274	0.672E-03
2.2	68	218	0.845E-03
2.3	84	257	0.199E-02
2.4	68	217	0.633E-02
2.5	73	231	0.334E-02
2.6	102	321	0.656E-02
2.7	101	311	0.198E-02
2.8	94	298	0.455E-03
2.9	90	284	0.354E-02
3.0	117	363	0.448E-03
3.1	29	98	0.114E-02
3.2	50	164	0.117E-02
3.3	38	123	0.993E-03
3.4	32	111	0.881E-03
3.5	33	111	0.787E-04
3.6	28	96	0.606E-03
3.7	115	360	0.320E-02
3.8	121	375	0.332E-03
3.9	51	162	0.149E-03
4.0	65	210	0.191E-02

PRINT FROM NELME
****

MINIMIZATION OF F(X)=100*(X(2)-X(1)**2)**2+(1-X(1))**2
UNTIL F(X) < 1.E-3 FOR DIFFERENT SIZES OF THE INITIAL POLYHEDRON.
STARTING POINT IS (-1.2, 1.0).

SIZE OF THE INITIAL POLYHEDRON	NUMBER OF ITERATIONS REQUIRED	NUMBER OF EVALUATIONS REQUIRED	TESTQ OF NELME
4.1	121	381	0.487E-02
4.2	111	350	0.260E-03
4.3	141	444	0.679E-03
4.4	110	349	0.258E-02
4.5	102	326	0.839E-03
4.6	101	320	0.202E-02
4.7	109	343	0.846E-03
4.8	121	371	0.350E-02
4.9	101	317	0.131E-02
5.0	109	340	0.592E-03
5.1	100	317	0.120E-02
5.2	113	351	0.132E-02
5.3	118	364	0.134E-02
5.4	103	326	0.387E-03
5.5	120	379	0.201E-02
5.6	116	364	0.441E-03
5.7	134	422	0.341E-03
5.8	111	346	0.893E-03
5.9	113	362	0.515E-03
6.0	105	331	0.161E-02
6.1	129	400	0.601E-03
6.2	101	321	0.152E-02
6.3	78	248	0.140E-02
6.4	76	243	0.160E-02
6.5	101	322	0.675E-03
6.6	133	415	0.102E-02
6.7	112	346	0.696E-03
6.8	129	409	0.164E-02
6.9	111	356	0.976E-03
7.0	119	381	0.156E-02
7.1	112	349	0.173E-02
7.2	109	346	0.996E-03
7.3	137	430	0.869E-04
7.4	148	467	0.107E-02
7.5	111	355	0.175E-02
7.6	119	373	0.560E-03
7.7	130	407	0.433E-03
7.8	129	402	0.965E-03
7.9	126	390	0.153E-02
8.0	123	380	0.151E-02

PRINT FROM NELME
****

MINIMIZATION OF F(X)=100*(X(2)-X(1)**2)**2+(1-X(1))**2
UNTIL F(X) < 1.E-5 FOR DIFFERENT SIZES OF THE INITIAL POLYHEDRON.
STARTING POINT IS (-1.2, 1.0).

SIZE OF THE INITIAL POLYHEDRON	NUMBER OF ITERATIONS REQUIRED	NUMBER OF EVALUATIONS REQUIRED	TESTQ OF NELME
0.1	90	286	0.168E-04
0.2	82	254	0.691E-05
0.3	93	285	0.591E-05
0.4	85	271	0.176E-04
0.5	85	270	0.270E-04
0.6	76	248	0.684E-04
0.7	90	276	0.308E-04
0.8	72	230	0.103E-04
0.9	99	304	0.579E-04
1.0	86	268	0.726E-05
1.1	87	279	0.135E-04
1.2	84	266	0.114E-04
1.3	83	259	0.109E-04
1.4	86	273	0.762E-05
1.5	69	214	0.159E-04
1.6	80	257	0.192E-04
1.7	86	271	0.121E-04
1.8	80	248	0.555E-05
1.9	78	249	0.191E-04
2.0	90	285	0.184E-04
2.1	102	316	0.683E-04
2.2	77	249	0.127E-04
2.3	90	277	0.499E-04
2.4	75	240	0.278E-04
2.5	80	255	0.408E-04
2.6	109	345	0.212E-04
2.7	105	326	0.530E-04
2.8	100	318	0.236E-04
2.9	95	303	0.638E-05
3.0	123	382	0.164E-04
3.1	35	118	0.472E-05
3.2	55	180	0.138E-03
3.3	43	139	0.823E-04
3.4	41	139	0.193E-04
3.5	42	140	0.164E-04
3.6	36	122	0.913E-05
3.7	123	386	0.198E-04
3.8	123	382	0.624E-03
3.9	63	201	0.672E-05
4.0	[illegible]	234	0.104E-04

PRINT FROM NELME
****

MINIMIZATION OF F(X)=100*(X(2)-X(1)**2)**2+(1-X(1))**2
UNTIL F(X) < 1.E-5 FOR DIFFERENT SIZES OF THE INITIAL POLYHEDRON.
STARTING POINT IS (-1.2, 1.0).

SIZE OF THE INITIAL POLYHEDRON	NUMBER OF ITERATIONS REQUIRED	NUMBER OF EVALUATIONS REQUIRED	TESTQ OF NELME
4.1	131	415	0.222E-05
4.2	117	370	0.121E-04
4.3	145	459	0.404E-04
4.4	119	379	0.232E-05
4.5	107	344	0.159E-04
4.6	106	337	0.375E-04
4.7	115	364	0.123E-04
4.8	128	395	0.561E-04
4.9	107	337	0.306E-04
5.0	113	355	0.249E-04
5.1	104	331	0.205E-03
5.2	119	372	0.840E-05
5.3	123	380	0.549E-05
5.4	116	367	0.523E-04
5.5	123	389	0.170E-03
5.6	121	381	0.912E-05
5.7	144	455	0.734E-05
5.8	117	367	0.877E-05
5.9	116	369	0.378E-03
6.0	110	347	0.443E-04
6.1	137	427	0.196E-04
6.2	107	341	0.142E-04
6.3	85	270	0.243E-04
6.4	76	243	0.160E-02
6.5	106	336	0.149E-03
6.6	140	439	0.483E-05
6.7	119	368	0.922E-05
6.8	137	436	0.580E-05
6.9	117	377	0.806E-05
7.0	121	389	0.807E-04
7.1	122	382	0.396E-05
7.2	117	373	0.954E-05
7.3	145	454	0.652E-04
7.4	153	484	0.299E-05
7.5	118	375	0.119E-03
7.6	126	395	0.333E-04
7.7	138	433	0.599E-05
7.8	132	414	0.154E-04
7.9	133	411	0.810E-05
8.0	131	405	0.950E-05

PRINT FROM NELME
****

MINIMIZATION OF F(X)=100*(X(2)-X(1)**2)**2+(1-X(1))**2
UNTIL F(X) < 1.E-7 FOR DIFFERENT SIZES OF THE INITIAL POLYHEDRON.
STARTING POINT IS (-1.2, 1.0).

SIZE OF THE INITIAL POLYHEDRON	NUMBER OF ITERATIONS REQUIRED	NUMBER OF EVALUATIONS REQUIRED	TESTQ OF NELME
0.1	97	308	0.273E-06
0.2	89	276	0.731E-07
0.3	96	296	0.138E-06
0.4	94	303	0.113E-06
0.5	88	281	0.289E-05
0.6	81	267	0.574E-07
0.7	95	293	0.611E-06
0.8	76	244	0.808E-07
0.9	105	324	0.347E-06
1.0	89	279	0.259E-06
1.1	90	291	0.289E-06
1.2	88	280	0.889E-06
1.3	87	274	0.211E-06
1.4	89	285	0.472E-07
1.5	74	233	0.463E-07
1.6	83	269	0.447E-06
1.7	89	281	0.130E-05
1.8	88	274	0.748E-07
1.9	83	265	0.924E-07
2.0	94	300	0.262E-06
2.1	109	341	0.133E-06
2.2	82	266	0.239E-06
2.3	96	299	0.131E-06
2.4	86	276	0.432E-07
2.5	86	276	0.108E-06
2.6	113	360	0.767E-07
2.7	109	342	0.341E-06
2.8	105	336	0.147E-06
2.9	98	315	0.145E-06
3.0	128	401	0.979E-07
3.1	38	129	0.223E-06
3.2	60	200	0.554E-07
3.3	46	151	0.244E-05
3.4	45	153	0.114E-06
3.5	47	158	0.199E-06
3.6	40	137	0.708E-07
3.7	126	398	0.354E-06
3.8	129	404	0.297E-06
3.9	65	209	0.420E-06
4.0	77	252	0.434E-06

PRINT FROM NELME
****

MINIMIZATION OF F(X)=100*(X(2)-X(1)**2)**2+(1-X(1))**2
UNTIL F(X) < 1.E-7 FOR DIFFERENT SIZES OF THE INITIAL POLYHEDRON.
STARTING POINT IS (-1.2, 1.0).

SIZE OF THE INITIAL POLYHEDRON	NUMBER OF ITERATIONS REQUIRED	NUMBER OF EVALUATIONS REQUIRED	TESTQ OF NELME
4.1	137	435	0.206E-06
4.2	123	392	0.790E-07
4.3	151	480	0.205E-07
4.4	124	396	0.214E-06
4.5	121	390	0.233E-07
4.6	112	358	0.843E-07
4.7	118	375	0.727E-06
4.8	133	413	0.128E-06
4.9	113	358	0.997E-07
5.0	118	373	0.451E-07
5.1	110	353	0.624E-07
5.2	129	404	0.838E-07
5.3	125	388	0.127E-05
5.4	122	387	0.798E-07
5.5	141	449	0.466E-07
5.6	126	400	0.302E-07
5.7	151	478	0.162E-06
5.8	121	382	0.843E-07
5.9	123	395	0.264E-07
6.0	116	367	0.209E-06
6.1	142	445	0.272E-06
6.2	114	364	0.113E-06
6.3	91	290	0.113E-06
6.4	83	271	0.813E-07
6.5	114	364	0.100E-06
6.6	143	451	0.210E-06
6.7	124	385	0.773E-06
6.8	142	453	0.978E-06
6.9	118	381	0.223E-05
7.0	126	408	0.290E-07
7.1	125	394	0.582E-07
7.2	122	390	0.670E-07
7.3	149	468	0.831E-06
7.4	158	502	0.139E-06
7.5	122	390	0.975E-06
7.6	131	415	0.481E-07
7.7	143	451	0.100E-06
7.8	137	432	0.230E-06
7.9	138	429	0.200E-06
8.0	136	423	0.186E-06

PRINT FROM NELME
****

MINIMIZATION OF F(X)=100*(X(2)-X(1)**2)**2+(1-X(1))**2+
90*(X(4)-X(3)**2)**2+(1-X(3))**2+10.1((X(2)-1)**2+(X(4)-1)**2)+
19.8*(X(2)-1)*(X(4)-1)
UNTIL F(X) < 1.E-1 FOR DIFFERENT SIZES OF THE INITIAL POLYHEDRON,
STARTING POINT IS (-3,-1,-3,-1).

SIZE OF THE INITIAL POLYHEDRON	NUMBER OF ITERATIONS REQUIRED	NUMBER OF EVALUATIONS REQUIRED	TESTG OF NELME
0.1	82	249	0.488E-02
0.2	183	541	0.951E-02
0.3	48	146	0.355E-01
0.4	50	161	0.592E-01
0.5	46	150	0.926E-02
0.6	86	276	0.153E-01
0.7	34	119	0.176
0.8	49	152	0.142
0.9	32	107	0.209
1.0	30	105	0.397E-01
1.1	40	135	0.404E-01
1.2	33	108	0.677E-01
1.3	33	125	0.259E-01
1.4	670	2001	0.105E-05
1.5	863	2000	0.148E-05
1.6	951	2000	0.477E-06
1.7	565	1656	0.197E-01
1.8	452	1324	0.190E-03
1.9	316	944	0.877E-02
2.0	250	748	0.244E-01
2.1	264	764	0.160E-01
2.2	954	2000	0.369E-06
2.3	950	2000	0.369E-06
2.4	956	2001	0.674E-06
2.5	950	2000	0.640E-06
2.6	947	2000	0.302E-06
2.7	286	821	0.100E-01
2.8	101	305	0.190E-01
2.9	70	226	0.207E-01
3.0	59	192	0.368E-01
3.1	72	233	0.729E-02
3.2	346	1018	0.161E-01
3.3	250	733	0.145E-01
3.4	123	398	0.252E-01
3.5	136	415	0.414E-01
3.6	157	477	0.109E-01
3.7	105	337	0.844E-02
3.8	101	329	0.148E-01
3.9	57	184	0.860E-02
4.0	37	131	0.930E-01

```
PRINT FROM NELME
****************

MINIMIZATION OF F(X)=100*(X(2)-X(1)**2)**2+(1-X(1))**2+
90*(X(4)-X(3))**2+(1-X(3))**2+10.1((X(2)-1)**2)+(X(4)-1)**2)
19.8*(X(2)-1)*(X(4)-1)
UNTIL F(X) < 1.E-1 FOR DIFFERENT SIZES OF THE INITIAL POLYHEDRON.
STARTING POINT IS (-3,-1,-3,-1).
```

SIZE OF THE INITIAL POLYHEDRON	NUMBER OF ITERATIONS REQUIRED	NUMBER OF EVALUATIONS REQUIRED	TESTQ OF NELME
4.1	196	604	0.224E-01
4.2	267	804	0.953E-02
4.3	24	94	0.650E-01
4.4	656	2001	0.506E-04
4.5	349	1021	0.248E-01
4.6	325	965	0.541E-02
4.7	384	1149	0.584E-02
4.8	292	877	0.862E-02
4.9	291	858	0.582E-01
5.0	268	799	0.212E-01
5.1	363	1081	0.264E-02
5.2	369	1106	0.127E-01
5.3	245	726	0.492E-01
5.4	325	961	0.161E-01
5.5	252	755	0.624E-02
5.6	895	2001	0.475E-05
5.7	320	968	0.467E-02
5.8	253	772	0.552E-01
5.9	130	409	0.118E-01
6.0	133	412	0.295E-01
6.1	138	438	0.648E-02
6.2	221	670	0.503E-01
6.3	244	716	0.336E-01
6.4	71	226	0.154E-01
6.5	83	267	0.225E-01
6.6	222	652	0.352E-01
6.7	283	854	0.230E-01
6.8	271	794	0.725E-01
6.9	67	222	0.134E-01
7.0	33	123	0.185E-01
7.1	24	91	0.332
7.2	17	69	1.31
7.3	932	2001	0.302E-06
7.4	64	219	0.175E-02
7.5	58	199	0.189E-01
7.6	67	221	0.122E-01
7.7	66	216	0.430E-01
7.8	55	193	0.217E-01
7.9	75	241	0.391E-01
8.0	152	469	0.186E-01

PRINT FROM NELME
****

MINIMIZATION OF F(X)=100*(X(2)-X(1)**2)**2+(1-X(1))**2+
90*(X(4)-X(3))**2+(1-X(3))**2+10.1((X(2)-1)**2)+(X(4)-1)**2)
19.8*(X(2)-1)*(X(4)-1)
UNTIL F(X) < 1.E-3 FOR DIFFERENT SIZES OF THE INITIAL POLYHEDRON.
STARTING POINT IS (-3,-1,-3,-1).

SIZE OF THE INITIAL POLYHEDRON	NUMBER OF ITERATIONS REQUIRED	NUMBER OF EVALUATIONS REQUIRED	TESTQ OF NELME
0.1	211	613	0.321E-04
0.2	217	644	0.171E-02
0.3	101	306	0.673E-03
0.4	115	358	0.150E-04
0.5	129	385	0.998E-02
0.6	127	394	0.427E-03
0.7	129	395	0.600E-04
0.8	134	397	0.283E-03
0.9	101	317	0.292E-03
1.0	35	125	0.616E-02
1.1	126	393	0.412E-03
1.2	92	293	0.613E-03
1.3	128	397	0.159E-02
1.4	670	2001	0.105E-05
1.5	863	2000	0.148E-05
1.6	951	2000	0.477E-06
1.7	658	1918	0.282E-03
1.8	588	1713	0.228E-03
1.9	378	1125	0.583E-03
2.0	324	967	0.479E-03
2.1	302	884	0.304E-03
2.2	954	2000	0.369E-06
2.3	950	2000	0.369E-06
2.4	956	2001	0.674E-06
2.5	950	2000	0.640E-06
2.6	947	2000	0.302E-06
2.7	335	964	0.881E-03
2.8	146	445	0.106E-02
2.9	134	421	0.924E-04
3.0	81	254	0.851E-03
3.1	101	311	0.156E-02
3.2	389	1146	0.794E-03
3.3	320	941	0.135E-02
3.4	263	804	0.185E-03
3.5	149	458	0.897E-03
3.6	204	608	0.103E-02
3.7	161	499	0.238E-03
3.8	127	415	0.816E-03
3.9	80	256	0.138E-03
4.0	75	250	0.147E-02

PRINT FROM NELME
****

MINIMIZATION OF F(X)=100*(X(2)-X(1)**2)**2+(1-X(1))**2+
90*(X(4)-X(3))**2+(1-X(3))**2+10.1((X(2)-1)**2)+(X(4)-1)**2)
19.8*(X(2)-1)*(X(4)-1)
UNTIL F(X) < 1.E-3 FOR DIFFERENT SIZES OF THE INITIAL POLYHEDRON.
STARTING POINT IS (-3,-1,-3,-1).

SIZE OF THE INITIAL POLYHEDRON	NUMBER OF ITERATIONS REQUIRED	NUMBER OF EVALUATIONS REQUIRED	TEST0 OF NELME
4.1	217	668	0.243E-02
4.2	309	925	0.623E-03
4.3	108	349	0.563E-03
4.4	656	2001	0.506E-04
4.5	390	1139	0.452E-03
4.6	412	1214	0.563E-03
4.7	422	1255	0.934E-03
4.8	355	1058	0.156E-03
4.9	303	897	0.241E-02
5.0	339	1016	0.463E-03
5.1	458	1351	0.184E-03
5.2	391	1173	0.455E-03
5.3	284	852	0.242E-04
5.4	350	1039	0.408E-03
5.5	354	1054	0.945E-03
5.6	895	2001	0.475E-05
5.7	373	1125	0.384E-03
5.8	413	1242	0.560E-03
5.9	204	635	0.770E-04
6.0	160	495	0.623E-03
6.1	179	557	0.331E-02
6.2	282	840	0.229E-02
6.3	320	949	0.425E-03
6.4	124	389	0.129E-02
6.5	141	436	0.210E-02
6.6	332	973	0.304E-03
6.7	347	1037	0.657E-02
6.8	308	900	0.247E-02
6.9	104	334	0.647E-03
7.0	102	328	0.667E-03
7.1	86	277	0.346E-03
7.2	77	260	0.566E-03
7.3	932	2001	0.302E-06
7.4	105	332	0.110E-02
7.5	128	409	0.736E-04
7.6	116	362	0.213E-03
7.7	88	284	0.101E-02
7.8	78	263	0.733E-03
7.9	98	316	0.793E-03
8.0	186	574	0.467E-03

```
PRINT FROM NELME
***************

MINIMIZATION OF F(X)=100*(X(2)-X(1)**2)**2+(1-X(1))**2+
90*(X(4)-X(3))**2+(1-X(3))**2+10.1((X(2)-1)**2)+(X(4)-1)**2)
19.8*(X(2)-1)*(X(4)-1)
UNTIL F(X) < 1.E-5 FOR DIFFERENT SIZES OF THE INITIAL POLYHEDRON.
STARTING POINT IS (-3,-1,-3,-1).
```

SIZE OF THE INITIAL POLYHEDRON	NUMBER OF ITERATIONS REQUIRED	NUMBER OF EVALUATIONS REQUIRED	TESTQ OF NELME
0.1	259	748	0.790E-05
0.2	232	695	0.436E-05
0.3	127	380	0.569E-05
0.4	146	441	0.499E-05
0.5	156	483	0.825E-06
0.6	140	437	0.108E-04
0.7	154	471	0.309E-05
0.8	154	457	0.624E-05
0.9	132	412	0.141E-04
1.0	94	310	0.789E-05
1.1	139	436	0.453E-05
1.2	104	332	0.667E-05
1.3	141	438	0.180E-04
1.4	670	2001	0.105E-05
1.5	863	2000	0.148E-05
1.6	951	2000	0.477E-06
1.7	677	1985	0.804E-06
1.8	614	1787	0.145E-04
1.9	386	1151	0.274E-04
2.0	338	1009	0.592E-05
2.1	315	923	0.200E-04
2.2	954	2000	0.369E-06
2.3	950	2000	0.369E-06
2.4	956	2001	0.674E-06
2.5	950	2000	0.640E-06
2.6	947	2000	0.302E-06
2.7	376	1086	0.533E-05
2.8	163	499	0.208E-04
2.9	154	477	0.827E-05
3.0	115	358	0.724E-05
3.1	123	383	0.438E-05
3.2	444	1313	0.275E-05
3.3	328	975	0.639E-05
3.4	279	855	0.563E-05
3.5	179	553	0.474E-05
3.6	220	654	0.585E-05
3.7	173	536	0.683E-05
3.8	139	456	0.219E-04
3.9	112	354	0.201E-05
4.0	90	304	0.299E-05

PRINT FROM NELME
****

MINIMIZATION OF F(X)=100*(X(2)-X(1)**2)**2+(1-X(1))**2+
90*(X(4)-X(3))**2+(1-X(3))**2+10.1((X(2)-1)**2)+(X(4)-1)**2)
19.8*(X(2)-1)*(X(4)-1)
UNTIL F(X) < 1.E-5 FOR DIFFERENT SIZES OF THE INITIAL POLYHEDRON.
STARTING POINT IS (-3,-1,-3,-1).

SIZE OF THE INITIAL POLYHEDRON	NUMBER OF ITERATIONS REQUIRED	NUMBER OF EVALUATIONS REQUIRED	TESTO OF NELME
4.1	502	2000	0.817E-08
4.2	320	959	0.130E-04
4.3	125	400	0.684E-05
4.4	656	2001	0.506E-04
4.5	404	1184	0.108E-04
4.6	424	1254	0.615E-05
4.7	434	1296	0.197E-04
4.8	368	1100	0.231E-04
4.9	498	2001	0.274E-08
5.0	360	1080	0.582E-05
5.1	470	1383	0.522E-04
5.2	413	1244	0.363E-05
5.3	409	1190	0.202E-04
5.4	378	1122	0.600E-05
5.5	369	1108	0.501E-05
5.6	895	2001	0.475E-05
5.7	379	1147	0.532E-04
5.8	445	1346	0.147E-05
5.9	224	690	0.964E-05
6.0	198	609	0.499E-05
6.1	233	713	0.637E-05
6.2	310	930	0.170E-04
6.3	336	999	0.930E-05
6.4	153	478	0.448E-05
6.5	149	466	0.889E-05
6.6	370	1093	0.146E-05
6.7	360	1086	0.682E-05
6.8	317	933	0.437E-04
6.9	119	388	0.782E-05
7.0	134	430	0.322E-05
7.1	104	336	0.262E-05
7.2	96	318	0.487E-05
7.3	932	2001	0.302E-06
7.4	118	376	0.860E-05
7.5	145	454	0.242E-04
7.6	129	403	0.130E-04
7.7	95	312	0.751E-05
7.8	117	376	0.668E-05
7.9	118	375	0.139E-04
8.0	199	615	0.947E-05

PRINT FROM NELME
****

MINIMIZATION OF F(X)=100*(X(2)-X(1)**2)**2+(1-X(1))**2+
90*(X(4)-X(3))**2+(1-X(3))**2+10.1((X(2)-1)**2)+(X(4)-1)**2)
19.8*(X(2)-1)*(X(4)-1)
UNTIL F(X) < 1.E-7 FOR DIFFERENT SIZES OF THE INITIAL POLYHEDRON.
STARTING POINT IS (-3,-1,-3,-1).

SIZE OF THE INITIAL POLYHEDRON	NUMBER OF ITERATIONS REQUIRED	NUMBER OF EVALUATIONS REQUIRED	TESTQ OF NELME
0.1	272	789	0.796E-07
0.2	242	729	0.127E-06
0.3	142	428	0.495E-07
0.4	149	456	0.314E-06
0.5	175	541	0.345E-07
0.6	148	467	0.135E-06
0.7	170	521	0.142E-06
0.8	169	504	0.111E-06
0.9	155	485	0.128E-06
1.0	108	358	0.288E-06
1.1	151	475	0.374E-07
1.2	120	385	0.512E-07
1.3	151	475	0.175E-06
1.4	670	2001	0.105E-05
1.5	863	2000	0.148E-05
1.6	951	2000	0.477E-06
1.7	682	2001	0.882E-06
1.8	626	1829	0.396E-07
1.9	399	1199	0.295E-07
2.0	352	1059	0.540E-07
2.1	324	953	0.306E-06
2.2	954	2000	0.369E-06
2.3	950	2000	0.369E-06
2.4	956	2001	0.674E-06
2.5	950	2000	0.640E-06
2.6	947	2000	0.302E-06
2.7	391	1133	0.685E-07
2.8	168	517	0.132E-05
2.9	170	535	0.198E-07
3.0	124	390	0.562E-07
3.1	146	453	0.784E-07
3.2	456	1352	0.105E-06
3.3	338	1011	0.841E-07
3.4	297	915	0.383E-07
3.5	192	594	0.559E-07
3.6	228	679	0.431E-06
3.7	181	567	0.126E-06
3.8	151	499	0.124E-06
3.9	146	460	0.315E-07
4.0	110	362	0.313E-06

PRINT FROM NELME
****

MINIMIZATION OF F(X)=100*(X(2)-X(1)**2)**2+(1-X(1))**2+
90*(X(4)-X(3))**2+(1-X(3))**2+10.1((X(2)-1)**2)+(X(4)-1)**2)
19.8*(X(2)-1)*(X(4)-1)
UNTIL F(X) < 1.E-7 FOR DIFFERENT SIZES OF THE INITIAL POLYHEDRON.
STARTING POINT IS (-3,-1,-3,-1).

SIZE OF THE INITIAL POLYHEDRON	NUMBER OF ITERATIONS REQUIRED	NUMBER OF EVALUATIONS REQUIRED	TESTQ OF NELME
4.1	502	2000	0.817E-08
4.2	334	1014	0.469E-07
4.3	136	438	0.106E-06
4.4	656	2001	0.506E-04
4.5	418	1232	0.386E-07
4.6	445	1322	0.743E-07
4.7	444	1332	0.170E-06
4.8	376	1132	0.133E-06
4.9	498	2001	0.274E-08
5.0	373	1126	0.388E-07
5.1	485	1434	0.116E-06
5.2	429	1295	0.126E-06
5.3	420	1226	0.240E-06
5.4	391	1165	0.134E-06
5.5	377	1139	0.717E-07
5.6	895	2001	0.475E-05
5.7	388	1182	0.112E-06
5.8	477	1438	0.874E-07
5.9	237	735	0.957E-07
6.0	224	694	0.590E-07
6.1	247	758	0.116E-06
6.2	337	1014	0.348E-07
6.3	342	1021	0.680E-06
6.4	172	539	0.180E-06
6.5	163	516	0.732E-07
6.6	386	1143	0.785E-07
6.7	374	1140	0.526E-07
6.8	324	966	0.253E-06
6.9	132	429	0.102E-06
7.0	143	459	0.327E-06
7.1	118	382	0.894E-07
7.2	110	369	0.670E-07
7.3	932	2001	0.302E-06
7.4	128	414	0.124E-06
7.5	166	522	0.675E-07
7.6	142	450	0.813E-07
7.7	104	348	0.820E-07
7.8	122	396	0.323E-06
7.9	133	423	0.792E-07
8.0	207	644	0.134E-06

PRINT FROM NELME
****

MINIMIZATION OF F(X)=100*(X(3)-10*ATAN2(X(2),X(1))/(2*PI))**2+
(SQRT(X(1)**2+X(2)**2)-1)**2+X(3)**2
UNTIL F(X) < 1.E-1 FOR DIFFERENT SIZES OF THE INITIAL POLYHEDRON.
STARTING POINT IS (-1,0,0).

SIZE OF THE INITIAL POLYHEDRON	NUMBER OF ITERATIONS REQUIRED	NUMBER OF EVALUATIONS REQUIRED	TESTQ OF NELME
0.1	203	630	0.819E-01
0.2	198	609	0.278E-01
0.3	142	428	0.144E-01
0.4	153	471	0.553E-01
0.5	141	438	0.107E-01
0.6	114	364	0.983E-01
0.7	866	2001	0.119E-06
0.8	797	2001	0.000
0.9	146	436	0.681E-01
1.0	154	480	0.603E-02
1.1	184	566	0.143
1.2	148	441	0.147E-01
1.3	90	286	0.406E-02
1.4	81	258	0.471E-01
1.5	88	277	0.102
1.6	205	640	0.520E-01
1.7	117	361	0.348E-01
1.8	23	80	0.353E-01
1.9	23	80	0.443E-01
2.0	12	48	0.465
2.1	12	48	0.279
2.2	10	38	2.45
2.3	10	38	2.50
2.4	24	83	0.707E-01
2.5	98	314	0.307E-01
2.6	31	105	0.693E-02
2.7	8	30	4.04
2.8	8	30	4.26
2.9	18	69	0.619E-02
3.0	21	77	0.127E-01
3.1	26	92	0.122E-01
3.2	42	136	0.204E-01
3.3	31	112	0.574E-02
3.4	47	158	0.913E-03
3.5	29	103	0.396E-02
3.6	31	112	0.250E-01
3.7	31	107	0.407E-01
3.8	40	136	0.557E-02
3.9	48	160	0.234E-01
4.0	54	169	0.135E-01

PRINT FROM NELME
****

MINIMIZATION OF F(X)=100*(X(3)-10*ATAN2(X(2),X(1))/(2*PI))**2+
(SQRT(X(1)**2+X(2)**2)-1)**2+X(3)**2
UNTIL F(X) < 1.E-1 FOR DIFFERENT SIZES OF THE INITIAL POLYHEDRON.
STARTING POINT IS (-1,0,0).

SIZE OF THE INITIAL POLYHEDRON	NUMBER OF ITERATIONS REQUIRED	NUMBER OF EVALUATIONS REQUIRED	TESTQ OF NELME
4.1	42	137	0.267E-01
4.2	50	159	0.351E-01
4.3	26	89	0.101
4.4	33	107	0.364E-01
4.5	49	163	0.130E-01
4.6	30	103	0.373E-01
4.7	52	167	0.908E-01
4.8	48	158	0.497E-01
4.9	38	122	0.189E-01
5.0	82	260	0.184E-01
5.1	38	129	0.694E-02
5.2	70	231	0.198E-01
5.3	32	116	0.442E-01
5.4	48	161	0.205E-01
5.5	42	138	0.142E-01
5.6	23	86	0.167
5.7	31	109	0.571E-01
5.8	38	133	0.616E-01
5.9	109	333	0.116E-01
6.0	36	118	0.556E-01
6.1	35	121	0.589E-01
6.2	64	202	0.342E-01
6.3	37	134	0.208E-01
6.4	44	150	0.203E-01
6.5	53	176	0.169E-01
6.6	51	171	0.149E-01
6.7	62	199	0.324E-01
6.8	84	262	0.665E-01
6.9	111	355	0.656E-01
7.0	68	224	0.449E-01
7.1	96	300	0.885E-02
7.2	65	212	0.102
7.3	56	192	0.733E-02
7.4	85	267	0.486E-01
7.5	40	141	0.229E-01
7.6	56	179	0.367E-01
7.7	50	168	0.126E-01
7.8	99	313	0.231E-01
7.9	59	187	0.351E-01
8.0	81	253	0.329E-01

PRINT FROM NELME
****

MINIMIZATION OF F(X)=100*(X(3)-10*ATAN2(X(2),X(1))/(2*PI))**2+
(SQRT(X(1)**2+X(2)**2)-1)**2+X(3)**2
UNTIL F(X) < 1.E-3 FOR DIFFERENT SIZES OF THE INITIAL POLYHEDRON.
STARTING POINT IS (-1,0,0).

SIZE OF THE INITIAL POLYHEDRON	NUMBER OF ITERATIONS REQUIRED	NUMBER OF EVALUATIONS REQUIRED	TESTQ OF NELME
0.1	228	708	0.455E-03
0.2	235	719	0.686E-03
0.3	186	561	0.496E-03
0.4	194	591	0.968E-03
0.5	194	593	0.128E-03
0.6	191	601	0.138E-03
0.7	866	2001	0.119E-06
0.8	797	2001	0.000
0.9	194	579	0.754E-03
1.0	185	576	0.555E-03
1.1	248	758	0.648E-03
1.2	174	525	0.302E-03
1.3	148	467	0.288E-03
1.4	90	287	0.184E-02
1.5	124	387	0.126E-02
1.6	251	784	0.673E-03
1.7	142	442	0.188E-02
1.8	30	103	0.365E-02
1.9	42	139	0.117E-02
2.0	56	185	0.646E-03
2.1	77	251	0.603E-03
2.2	75	243	0.199E-03
2.3	84	270	0.118E-03
2.4	65	212	0.171E-03
2.5	134	425	0.797E-03
2.6	59	195	0.428E-03
2.7	49	165	0.183E-02
2.8	64	204	0.940E-03
2.9	51	164	0.192E-02
3.0	51	162	0.110E-02
3.1	57	179	0.117E-02
3.2	79	246	0.569E-03
3.3	74	236	0.451E-02
3.4	80	261	0.346E-03
3.5	88	285	0.264E-03
3.6	68	222	0.281E-02
3.7	83	274	0.164E-02
3.8	89	282	0.370E-03
3.9	84	267	0.444E-03
4.0	95	296	0.197E-03

PRINT FROM NELME
****

MINIMIZATION OF F(X)=100*(X(3)-10*ATAN2(X(2),X(1))/(2*PI))**2+
(SQRT(X(1)**2+X(2)**2)-1)**2+X(3)**2
UNTIL F(X) < 1.E-3 FOR DIFFERENT SIZES OF THE INITIAL POLYHEDRON.
STARTING POINT IS (-1,0,0).

SIZE OF THE INITIAL POLYHEDRON	NUMBER OF ITERATIONS REQUIRED	NUMBER OF EVALUATIONS REQUIRED	TESTQ OF NELME
4.1	54	176	0.432E-02
4.2	92	287	0.199E-02
4.3	68	216	0.252E-02
4.4	101	313	0.534E-03
4.5	90	288	0.694E-03
4.6	56	189	0.516E-03
4.7	90	282	0.197E-02
4.8	84	273	0.409E-04
4.9	73	233	0.442E-03
5.0	126	391	0.108E-02
5.1	69	219	0.749E-03
5.2	127	403	0.624E-03
5.3	64	217	0.586E-03
5.4	126	397	0.396E-03
5.5	118	380	0.114E-03
5.6	79	255	0.429E-03
5.7	88	272	0.292E-03
5.8	118	369	0.115E-02
5.9	121	371	0.645E-03
6.0	74	228	0.150E-02
6.1	58	197	0.812E-03
6.2	108	344	0.655E-03
6.3	80	266	0.191E-02
6.4	111	353	0.633E-03
6.5	110	346	0.638E-03
6.6	113	358	0.370E-03
6.7	111	351	0.401E-03
6.8	114	355	0.784E-04
6.9	124	401	0.340E-03
7.0	130	404	0.352E-03
7.1	156	478	0.729E-03
7.2	113	363	0.299E-03
7.3	87	284	0.100E-02
7.4	136	421	0.377E-03
7.5	109	353	0.187E-02
7.6	84	272	0.499E-03
7.7	102	329	0.312E-03
7.8	110	349	0.330E-02
7.9	135	417	0.325E-03
8.0	115	352	0.757E-03

PRINT FROM NELME
****

MINIMIZATION OF F(X)=100*(X(3)-10*ATAN2(X(2),X(1))/(2*PI))**2+
(SQRT(X(1)**2+X(2)**2)-1)**2+X(3)**2
UNTIL F(X) < 1.E-5 FOR DIFFERENT SIZES OF THE INITIAL POLYHEDRON.
STARTING POINT IS (-1,0,0).

SIZE OF THE INITIAL POLYHEDRON	NUMBER OF ITERATIONS REQUIRED	NUMBER OF EVALUATIONS REQUIRED	TESTQ OF NELME
0.1	235	732	0.623E-04
0.2	254	774	0.213E-04
0.3	201	604	0.140E-04
0.4	206	632	0.908E-05
0.5	203	622	0.148E-04
0.6	209	654	0.176E-04
0.7	866	2001	0.119E-06
0.8	797	2001	0.000
0.9	201	604	0.209E-04
1.0	193	606	0.672E-05
1.1	269	822	0.181E-04
1.2	186	560	0.620E-05
1.3	166	522	0.132E-04
1.4	161	498	0.469E-05
1.5	140	433	0.173E-04
1.6	260	815	0.880E-05
1.7	150	473	0.648E-05
1.8	60	196	0.138E-04
1.9	75	240	0.154E-04
2.0	79	261	0.719E-05
2.1	99	319	0.202E-04
2.2	87	280	0.315E-04
2.3	93	299	0.116E-04
2.4	80	255	0.366E-05
2.5	147	467	0.636E-05
2.6	67	221	0.733E-04
2.7	57	196	0.811E-05
2.8	71	229	0.146E-04
2.9	60	195	0.533E-05
3.0	60	191	0.189E-04
3.1	90	280	0.225E-04
3.2	96	303	0.409E-05
3.3	116	367	0.276E-05
3.4	112	356	0.402E-05
3.5	107	346	0.778E-05
3.6	76	251	0.620E-05
3.7	109	315	0.143E-04
3.8	103	327	0.128E-04
3.9	100	313	0.468E-05
4.0	107	335	0.876E-05

PRINT FROM NELME
****

MINIMIZATION OF F(X)=100*(X(3)-10*ATAN2(X(2),X(1))/(2*PI))**2+
(SQRT(X(1)**2+X(2)**2)-1)**2+X(3)**2
UNTIL F(X) < 1.E-5 FOR DIFFERENT SIZES OF THE INITIAL POLYHEDRON.
STARTING POINT IS (-1,0,0).

SIZE OF THE INITIAL POLYHEDRON	NUMBER OF ITERATIONS REQUIRED	NUMBER OF EVALUATIONS REQUIRED	TESTQ OF NELME
4.1	91	287	0.942E-05
4.2	103	324	0.836E-05
4.3	87	275	0.387E-05
4.4	108	339	0.157E-04
4.5	105	337	0.426E-05
4.6	65	220	0.637E-05
4.7	100	318	0.380E-05
4.8	109	349	0.104E-04
4.9	90	285	0.471E-05
5.0	136	424	0.795E-05
5.1	77	252	0.223E-05
5.2	135	430	0.622E-05
5.3	67	230	0.239E-04
5.4	140	440	0.767E-05
5.5	132	418	0.424E-04
5.6	86	280	0.854E-05
5.7	100	307	0.128E-04
5.8	126	397	0.821E-05
5.9	145	447	0.967E-05
6.0	83	255	0.860E-04
6.1	96	312	0.477E-05
6.2	121	384	0.424E-05
6.3	111	361	0.180E-04
6.4	132	422	0.186E-04
6.5	120	379	0.133E-04
6.6	133	422	0.443E-05
6.7	117	370	0.548E-04
6.8	129	399	0.445E-05
6.9	165	522	0.240E-05
7.0	141	435	0.981E-05
7.1	168	518	0.640E-05
7.2	145	462	0.220E-04
7.3	112	364	0.185E-05
7.4	148	460	0.569E-05
7.5	133	435	0.745E-05
7.6	93	301	0.510E-04
7.7	110	357	0.709E-05
7.8	134	428	0.899E-05
7.9	151	466	0.176E-04
8.0	122	380	0.688E-05

PRINT FROM NELME
****

MINIMIZATION OF F(X)=100*(X(3)-10*ATAN2(X(2),X(1))/(2*PI))**2+
(SQRT(X(1)**2+X(2)**2)-1)**2+X(3)**2
UNTIL F(X) < 1.E-7 FOR DIFFERENT SIZES OF THE INITIAL POLYHEDRON.
STARTING POINT IS (-1,0,0).

SIZE OF THE INITIAL POLYHEDRON	NUMBER OF ITERATIONS REQUIRED	NUMBER OF EVALUATIONS REQUIRED	TESTQ OF NELME
0.1	242	756	0.106E-05
0.2	265	815	0.671E-07
0.3	207	625	0.101E-06
0.4	215	662	0.881E-07
0.5	207	639	0.204E-06
0.6	214	674	0.216E-06
0.7	866	2001	0.119E-06
0.8	797	2001	0.000
0.9	210	634	0.774E-07
1.0	200	632	0.154E-06
1.1	279	857	0.473E-07
1.2	192	581	0.564E-07
1.3	176	557	0.314E-07
1.4	167	522	0.109E-06
1.5	143	445	0.257E-05
1.6	266	837	0.268E-06
1.7	157	497	0.791E-07
1.8	72	235	0.719E-07
1.9	82	269	0.350E-07
2.0	90	297	0.688E-07
2.1	119	384	0.183E-06
2.2	96	314	0.408E-07
2.3	99	322	0.133E-07
2.4	98	312	0.968E-07
2.5	151	485	0.138E-06
2.6	79	262	0.271E-06
2.7	66	227	0.107E-04
2.8	77	251	0.732E-06
2.9	68	226	0.322E-07
3.0	65	212	0.194E-06
3.1	100	317	0.389E-07
3.2	106	337	0.903E-07
3.3	127	401	0.105E-06
3.4	120	381	0.111E-06
3.5	115	376	0.461E-06
3.6	83	277	0.210E-07
3.7	105	335	0.234E-06
3.8	114	364	0.832E-07
3.9	110	347	0.105E-06
4.0	114	359	0.481E-07

PRINT FROM VOLME
****

MINIMIZATION OF F(X)=100*(X(3)-10*[illegible]
+(SQRT(X(1)**2+X(2)**2)-1)**2+X(3)**2
UNTIL F(X) < 1.E-7 FOR DIFFERENT SIZES OF THE INITIAL POLYHEDRON
STARTING POINT IS (-1,0,0).

SIZE OF THE INITIAL POLYHEDRON	NUMBER OF ITERATIONS REQUIRED	NUMBER OF EVALUATIONS REQUIRED	TESTS OF VOLME
4.1	97	308	0.122E-06
4.2	114	359	0.497E-07
4.3	94	301	0.437E-07
4.4	118	372	0.216E-06
4.5	110	356	0.149E-06
4.6	70	240	0.302E-06
4.7	109	347	0.432E-06
4.8	119	382	0.190E-06
4.9	106	334	0.173E-06
5.0	149	467	0.396E-07
5.1	87	282	0.141E-06
5.2	145	465	0.439E-07
5.3	73	254	0.470E-07
5.4	146	463	0.107E-06
5.5	139	442	0.370E-06
5.6	93	309	0.380E-07
5.7	112	349	0.492E-07
5.8	139	436	0.126E-06
5.9	152	472	0.842E-07
6.0	92	290	0.388E-07
6.1	106	348	0.349E-07
6.2	135	429	0.108E-06
6.3	120	389	0.518E-06
6.4	141	453	0.483E-07
6.5	129	408	0.185E-06
6.6	144	461	0.747E-07
6.7	123	396	0.668E-07
6.8	137	428	0.201E-06
6.9	175	557	0.364E-07
7.0	143	443	0.361E-05
7.1	174	541	0.151E-06
7.2	150	485	0.104E-06
7.3	121	395	0.371E-07
7.4	159	498	0.195E-06
7.5	139	455	0.461E-06
7.6	107	349	0.334E-07
7.7	119	388	0.451E-06
7.8	144	461	0.103E-06
7.9	157	491	0.570E-07
8.0	128	402	0.263E-06

PRINT FROM NELME
****

MINIMIZATION OF F(X)=(X(1)+10*X(2))**2+5*(X(3)-X(4))**2+
(X(2)-2*X(3))**4+10*(X(1)-X(4))**4
UNTIL F(X) < 1.E-1 FOR DIFFERENT SIZES OF THE INITIAL POLYHEDRON.
STARTING POINT IS (3,-1,0,1).

SIZE OF THE INITIAL POLYHEDRON	NUMBER OF ITERATIONS REQUIRED	NUMBER OF EVALUATIONS REQUIRED	TESTQ OF NELME
0.1	97	279	0.108
0.2	83	240	0.886E-01
0.3	43	134	0.248
0.4	55	170	0.841E-01
0.5	45	142	0.707E-01
0.6	43	133	0.381E-01
0.7	38	126	0.432E-01
0.8	48	148	0.151
0.9	42	134	0.993E-01
1.0	31	99	0.366
1.1	27	90	0.885E-01
1.2	41	127	0.712E-01
1.3	31	101	0.268
1.4	30	97	0.645E-01
1.5	28	100	0.569E-01
1.6	36	118	0.196E-01
1.7	43	128	0.102
1.8	28	93	0.241
1.9	48	150	0.715E-01
2.0	33	103	0.206
2.1	32	109	0.112
2.2	42	128	0.118
2.3	31	103	0.200
2.4	31	106	0.259
2.5	43	140	0.639E-01
2.6	48	146	0.712E-01
2.7	35	115	0.168
2.8	36	109	0.178
2.9	39	124	0.110
3.0	46	157	0.141E-01
3.1	37	129	0.629E-01
3.2	29	102	0.120
3.3	22	83	0.204E-01
3.4	36	127	0.808E-02
3.5	51	162	0.616E-01
3.6	27	101	0.703E-01
3.7	36	120	0.543E-01
3.8	35	116	0.357E-01
3.9	20	77	0.682E-01
4.0	32	111	0.250

PRINT FROM NELME
****

MINIMIZATION OF F(X)=(X(1)+10*X(2))**2+5*(X(3)-X(4))**2+
(X(2)-2*X(3))**4+10*(X(1)-X(4))**4
UNTIL F(X) < 1.E-1 FOR DIFFERENT SIZES OF THE INITIAL POLYHEDRON.
STARTING POINT IS (3,-1,0,1).

SIZE OF THE INITIAL POLYHEDRON	NUMBER OF ITERATIONS REQUIRED	NUMBER OF EVALUATIONS REQUIRED	TESTQ OF NELME
4.1	31	111	0.250E-01
4.2	30	100	0.254
4.3	43	145	0.429E-01
4.4	37	123.	0.203
4.5	31	106	0.473E-01
4.6	34	113	0.699E-01
4.7	40	126	0.561E-01
4.8	42	132	0.699E-01
4.9	36	125	0.741E-01
5.0	27	90	0.325
5.1	57	179	0.432E-01
5.2	46	152	0.475E-01
5.3	48	150	0.489E-01
5.4	62	196	0.274E-01
5.5	49	149	0.714E-01
5.6	45	145	0.630E-01
5.7	49	155	0.288E-01
5.8	31	105	0.737
5.9	41	142	0.750E-01
6.0	40	138	0.754E-01
6.1	45	151	0.566E-01
6.2	32	110	0.481E-01
6.3	39	135	0.731E-01
6.4	39	137	0.114E-01
6.5	57	178	0.120E-01
6.6	46	143	0.934E-01
6.7	55	165	0.814E-01
6.8	41	141	0.493E-01
6.9	35	117	0.243E-01
7.0	41	142	0.584E-01
7.1	60	195	0.187E-01
7.2	49	158	0.991E-02
7.3	45	154	0.432E-01
7.4	32	114	0.438E-01
7.5	20	80	0.428E-01
7.6	18	74	0.699E-01
7.7	18	74	0.531E-01
7.8	14	57	0.398
7.9	28	104	0.375E-01
8.0	26	98	0.309E-01

PRINT FROM NELME
****

MINIMIZATION OF F(X)=(X(1)+10*X(2))**2+5*(X(3)-X(4))**2+
(X(2)-2*X(3))**4+10*(X(1)-X(4))**4
UNTIL F(X) < 1.E-3 FOR DIFFERENT SIZES OF THE INITIAL POLYHEDRON.
STARTING POINT IS (3,-1,0,1).

SIZE OF THE INITIAL POLYHEDRON	NUMBER OF ITERATIONS REQUIRED	NUMBER OF EVALUATIONS REQUIRED	TESTQ OF NELME
0.1	106	314	0.407E-03
0.2	88	259	0.768E-02
0.3	56	179	0.113E-02
0.4	67	210	0.117E-02
0.5	107	324	0.115E-02
0.6	59	187	0.183E-02
0.7	54	174	0.500E-03
0.8	66	205	0.718E-03
0.9	63	200	0.300E-03
1.0	68	206	0.116E-02
1.1	64	213	0.172E-03
1.2	54	170	0.293E-03
1.3	42	141	0.216E-03
1.4	47	152	0.909E-03
1.5	61	199	0.777E-03
1.6	52	165	0.756E-03
1.7	58	184	0.102E-03
1.8	40	136	0.133E-02
1.9	70	220	0.297E-03
2.0	42	139	0.643E-03
2.1	59	201	0.494E-03
2.2	66	205	0.359E-03
2.3	73	240	0.590E-04
2.4	65	216	0.170E-03
2.5	78	247	0.148E-02
2.6	69	211	0.657E-03
2.7	54	184	0.151E-03
2.8	50	159	0.121E-02
2.9	49	158	0.115E-02
3.0	78	249	0.952E-03
3.1	59	199	0.244E-03
3.2	46	164	0.163E-03
3.3	58	188	0.586E-02
3.4	65	215	0.285E-03
3.5	76	240	0.396E-03
3.6	89	295	0.572E-04
3.7	86	271	0.189E-03
3.8	77	243	0.896E-03
3.9	63	208	0.134E-03
4.0	55	187	0.534E-04

PRINT FROM NELME
****

MINIMIZATION OF F(X)=(X(1)+10*X(2))**2+5*(X(3)-X(4))**2+
(X(2)-2*X(3))**4+10*(X(1)-X(4))**4
UNTIL F(X) < 1.E-3 FOR DIFFERENT SIZES OF THE INITIAL POLYHEDRON.
STARTING POINT IS (3,-1,0,1).

SIZE OF THE INITIAL POLYHEDRON	NUMBER OF ITERATIONS REQUIRED	NUMBER OF EVALUATIONS REQUIRED	TESTQ OF NELME
4.1	59	185	0.110E-02
4.2	62	200	0.632E-03
4.3	80	259	0.585E-03
4.4	53	183	0.741E-03
4.5	78	249	0.126E-02
4.6	64	200	0.125E-02
4.7	54	182	0.190E-03
4.8	52	171	0.651E-03
4.9	56	190	0.514E-03
5.0	57	180	0.449E-03
5.1	75	236	0.864E-03
5.2	63	202	0.813E-03
5.3	64	203	0.643E-03
5.4	78	247	0.118E-02
5.5	75	230	0.123E-02
5.6	77	245	0.697E-03
5.7	70	222	0.385E-03
5.8	39	143	0.364E-03
5.9	52	177	0.325E-02
6.0	60	200	0.192E-02
6.1	60	195	0.261E-02
6.2	82	256	0.116E-02
6.3	48	169	0.986E-03
6.4	65	220	0.915E-03
6.5	71	223	0.491E-03
6.6	101	302	0.590E-03
6.7	76	229	0.125E-02
6.8	65	214	0.168E-02
6.9	48	157	0.157E-02
7.0	54	185	0.749E-03
7.1	88	282	0.932E-03
7.2	93	286	0.160E-03
7.3	59	195	0.482E-02
7.4	49	170	0.504E-03
7.5	60	201	0.153E-03
7.6	36	130	0.444E-02
7.7	76	245	0.361E-03
7.8	85	277	0.786E-04
7.9	46	164	0.388E-03
8.0	76	234	0.209E-03

PRINT FROM NELME
****

MINIMIZATION OF F(X)=(X(1)+10*X(2))**2+5*(X(3)-X(4))**2+
(X(2)-2*X(3))**4+10*(X(1)-X(4))**4
UNTIL F(X) < 1.E-5 FOR DIFFERENT SIZES OF THE INITIAL POLYHEDRON.
STARTING POINT IS (3,-1,0,1).

SIZE OF THE INITIAL POLYHEDRON	NUMBER OF ITERATIONS REQUIRED	NUMBER OF EVALUATIONS REQUIRED	TESTQ OF NELME
0.1	149	442	0.427E-06
0.2	127	386	0.111E-05
0.3	82	260	0.181E-05
0.4	87	279	0.967E-05
0.5	128	392	0.221E-05
0.6	72	235	0.332E-05
0.7	107	333	0.223E-05
0.8	79	252	0.313E-05
0.9	77	251	0.630E-05
1.0	76	238	0.684E-05
1.1	85	276	0.531E-05
1.2	90	276	0.241E-04
1.3	64	210	0.771E-06
1.4	69	225	0.924E-05
1.5	81	267	0.675E-05
1.6	76	242	0.432E-05
1.7	76	237	0.383E-04
1.8	74	228	0.212E-04
1.9	93	293	0.359E-05
2.0	96	299	0.119E-05
2.1	71	241	0.796E-05
2.2	92	286	0.343E-05
2.3	99	310	0.135E-04
2.4	87	284	0.490E-05
2.5	96	311	0.943E-06
2.6	83	255	0.518E-05
2.7	111	352	0.347E-05
2.8	85	265	0.141E-05
2.9	91	293	0.114E-05
3.0	116	369	0.432E-05
3.1	103	322	0.157E-04
3.2	87	280	0.866E-05
3.3	63	214	0.283E-04
3.4	88	289	0.224E-04
3.5	92	289	0.125E-05
3.6	106	347	0.109E-04
3.7	108	339	0.274E-05
3.8	111	355	0.429E-05
3.9	121	374	0.172E-04
4.0	85	267	0.685E-05

PRINT FROM NELME
****

MINIMIZATION OF F(X)=(X(1)+10*X(2))**2+5*(X(3)-X(4))**2+
(X(2)-2*X(3))**4+10*(X(1)-X(4))**4
UNTIL F(X) < 1.E-5 FOR DIFFERENT SIZES OF THE INITIAL POLYHEDRON.
STARTING POINT IS (3,-1,0,1).

SIZE OF THE INITIAL POLYHEDRON	NUMBER OF ITERATIONS REQUIRED	NUMBER OF EVALUATIONS REQUIRED	TESTQ OF NELME
4.1	77	248	0.352E-05
4.2	77	249	0.336E-04
4.3	89	294	0.277E-05
4.4	92	304	0.188E-06
4.5	90	289	0.554E-05
4.6	69	223	0.877E-05
4.7	144	437	0.539E-05
4.8	98	305	0.403E-05
4.9	101	327	0.122E-05
5.0	79	252	0.105E-04
5.1	95	299	0.341E-04
5.2	88	277	0.124E-04
5.3	93	284	0.627E-05
5.4	97	308	0.305E-04
5.5	93	294	0.297E-05
5.6	93	290	0.871E-05
5.7	80	262	0.415E-05
5.8	121	379	0.281E-05
5.9	99	317	0.264E-05
6.0	106	340	0.192E-05
6.1	100	326	0.126E-05
6.2	89	283	0.170E-04
6.3	158	502	0.323E-05
6.4	81	275	0.527E-05
6.5	133	401	0.651E-05
6.6	115	350	0.401E-05
6.7	93	286	0.664E-05
6.8	81	271	0.196E-05
6.9	64	207	0.220E-04
7.0	74	251	0.865E-05
7.1	99	322	0.238E-05
7.2	108	332	0.128E-04
7.3	131	421	0.207E-05
7.4	73	241	0.167E-05
7.5	93	302	0.138E-05
7.6	150	465	0.868E-06
7.7	101	326	0.353E-05
7.8	117	365	0.319E-05
7.9	76	256	0.175E-04
8.0	85	259	0.770E-04

PRINT FROM NELME
****

MINIMIZATION OF F(X)=(X(1)+10*X(2))**2+5*(X(3)-X(4))**2+
(X(2)-2*X(3))**4+10*(X(1)-X(4))**4
UNTIL F(X) < 1.E-7 FOR DIFFERENT SIZES OF THE INITIAL POLYHEDRON.
STARTING POINT IS (3,-1,0,1).

SIZE OF THE INITIAL POLYHEDRON	NUMBER OF ITERATIONS REQUIRED	NUMBER OF EVALUATIONS REQUIRED	TESTQ OF NELME
0.1	172	512	0.169E-07
0.2	163	489	0.410E-07
0.3	134	405	0.275E-06
0.4	104	333	0.530E-07
0.5	140	435	0.191E-07
0.6	136	421	0.178E-06
0.7	125	388	0.333E-07
0.8	135	402	0.178E-06
0.9	105	333	0.569E-06
1.0	85	276	0.401E-08
1.1	99	322	0.549E-07
1.2	124	377	0.657E-07
1.3	126	380	0.203E-06
1.4	144	442	0.423E-07
1.5	111	357	0.218E-07
1.6	127	396	0.214E-06
1.7	157	470	0.177E-06
1.8	95	303	0.244E-07
1.9	110	345	0.843E-07
2.0	118	363	0.239E-07
2.1	121	396	0.682E-07
2.2	145	432	0.767E-07
2.3	105	335	0.335E-06
2.4	93	309	0.658E-07
2.5	116	369	0.115E-06
2.6	135	406	0.169E-07
2.7	142	445	0.256E-07
2.8	144	434	0.699E-07
2.9	115	366	0.385E-07
3.0	130	410	0.640E-07
3.1	126	397	0.169E-06
3.2	100	325	0.244E-06
3.3	94	314	0.766E-07
3.4	112	369	0.621E-07
3.5	124	379	0.188E-06
3.6	125	410	0.103E-06
3.7	151	467	0.696E-07
3.8	145	457	0.205E-06
3.9	144	452	0.542E-07
4.0	100	314	0.166E-06

PRINT FROM NELME
****

MINIMIZATION OF F(X)=(X(1)+10*X(2))**2+5*(X(3)-X(4))**2+
(X(2)-2*X(3))**4+10*(X(1)-X(4))**4
UNTIL F(X) < 1.E-7 FOR DIFFERENT SIZES OF THE INITIAL POLYHEDRON.
STARTING POINT IS (3,-1,0,1).

SIZE OF THE INITIAL POLYHEDRON	NUMBER OF ITERATIONS REQUIRED	NUMBER OF EVALUATIONS REQUIRED	TESTQ OF NELME
4.1	110	347	0.971E-07
4.2	145	452	0.119E-07
4.3	127	406	0.197E-06
4.4	116	373	0.376E-07
4.5	109	351	0.224E-07
4.6	84	274	0.630E-07
4.7	154	473	0.103E-06
4.8	120	378	0.875E-07
4.9	138	437	0.202E-06
5.0	123	387	0.116E-06
5.1	129	406	0.109E-07
5.2	113	355	0.456E-07
5.3	127	392	0.365E-07
5.4	119	375	0.600E-07
5.5	150	465	0.219E-07
5.6	109	350	0.306E-07
5.7	98	318	0.177E-06
5.8	133	423	0.440E-07
5.9	116	371	0.359E-07
6.0	128	405	0.658E-07
6.1	117	381	0.351E-07
6.2	114	366	0.339E-07
6.3	173	551	0.299E-07
6.4	93	312	0.100E-06
6.5	162	492	0.492E-07
6.6	128	394	0.109E-06
6.7	155	474	0.528E-07
6.8	95	313	0.140E-06
6.9	88	287	0.405E-08
7.0	133	416	0.463E-07
7.1	162	511	0.664E-07
7.2	133	409	0.611E-07
7.3	143	456	0.931E-06
7.4	96	308	0.646E-07
7.5	117	375	0.338E-07
7.6	175	533	0.246E-07
7.7	120	386	0.571E-07
7.8	140	436	0.472E-07
7.9	112	367	0.209E-07
8.0	88	274	0.226E-05

PRINT FROM NELME
****

MINIMIZATION OF F(X)=X(1)**2+AINT(ABS(X(1)))+5*X(2)**2+AINT(X(2))
UNTIL F(X) < 1.E-1 FOR DIFFERENT SIZES OF THE INITIAL POLYHEDRON.
STARTING POINT IS (2.0,2.0).

SIZE OF THE INITIAL POLYHEDRON	NUMBER OF ITERATIONS REQUIRED	NUMBER OF EVALUATIONS REQUIRED	TESTQ OF NELME
0.1	19	61	0.133
0.2	15	50	0.840
0.3	12	41	0.152
0.4	12	40	0.875E-01
0.5	10	32	0.118
0.6	12	38	0.679E-01
0.7	13	43	0.105
0.8	9	32	0.922
0.9	10	34	0.176
1.0	11	36	0.219
1.1	10	36	0.594E-01
1.2	6	23	0.936
1.3	7	27	0.148
1.4	12	43	0.469E-01
1.5	6	24	0.202
1.6	6	24	0.193
1.7	7	26	0.284
1.8	8	30	0.933E-01
1.9	7	27	0.539E-01
2.0	6	24	0.233
2.1	7	27	0.709E-01
2.2	7	25	0.597E-01
2.3	8	29	0.958E-01
2.4	8	29	0.126
2.5	7	25	0.912E-01
2.6	9	31	0.694E-01
2.7	6	21	1.15
2.8	5	17	1.94
2.9	6	22	0.300
3.0	9	31	0.111
3.1	7	25	0.132
3.2	8	28	0.161
3.3	8	29	0.450E-01
3.4	7	25	0.324
3.5	7	24	0.141
3.6	6	21	0.592
3.7	5	20	0.170
3.8	5	20	0.122
3.9	7	27	0.662E-01
4.0	5	20	0.175

PRINT FROM NELME
****

MINIMIZATION OF F(X)=X(1)**2+AINT(ABS(X(1)))+5*X(2)**2+AINT(X(2))
UNTIL F(X) < 1.E-2 FOR DIFFERENT SIZES OF THE INITIAL POLYHEDRON.
STARTING POINT IS (2.0,2.0).

SIZE OF THE INITIAL POLYHEDRON	NUMBER OF ITERATIONS REQUIRED	NUMBER OF EVALUATIONS REQUIRED	TESTQ OF NELME
4.1	6	24	0.445E-01
4.2	6	24	0.599E-01
4.3	5	20	0.162
4.4	5	20	0.136
4.5	5	20	0.145
4.6	1	6	15.7
4.7	1	6	16.0
4.8	1	6	16.5
4.9	1	6	16.9
5.0	1	6	17.4
5.1	1	6	17.9
5.2	1	6	18.4
5.3	6	24	0.957E-01
5.4	6	25	0.579E-01
5.5	5	20	0.171
5.6	5	20	0.272
5.7	5	20	0.376
5.8	7	27	0.135
5.9	6	23	0.357
6.0	6	23	0.410
6.1	7	[illegible]	[illegible]
6.2	7	[illegible]	[illegible]
6.3	6	24	[illegible]
6.4	6	24	[illegible]
6.5	5	[illegible]	[illegible]
6.6	6	24	[illegible]
6.7	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	0.942E-01
[illegible]	[illegible]	[illegible]	0.359E-01
[illegible]	8	[illegible]	[illegible]
[illegible]	5	[illegible]	0.197
7.2	5	23	0.552E-01
7.3	6	26	0.580E-01
7.4	3	15	2.07
7.5	3	15	2.30
7.6	3	15	2.53
7.7	3	15	2.77
7.8	3	15	3.01
7.9	6	26	0.265
8.0	6	24	0.233

PRINT FROM NELME
****

MINIMIZATION OF F(X)=X(1)**2+AINT(ABS(X(1)))+5*X(2)**2+AINT(X(2))
UNTIL F(X) < 1.E-3 FOR DIFFERENT SIZES OF THE INITIAL POLYHEDRON.
STARTING POINT IS (2.0,2.0).

SIZE OF THE INITIAL POLYHEDRON	NUMBER OF ITERATIONS REQUIRED	NUMBER OF EVALUATIONS REQUIRED	TESTQ OF NELME
0.1	23	75	0.181E-02
0.2	21	70	0.158E-02
0.3	17	60	0.918E-03
0.4	18	61	0.138E-02
0.5	14	48	0.141E-02
0.6	17	56	0.147E-02
0.7	18	62	0.457E-03
0.8	14	50	0.775E-03
0.9	14	49	0.207E-02
1.0	17	56	0.820E-03
1.1	19	65	0.325E-02
1.2	12	44	0.319E-02
1.3	11	43	0.665E-03
1.4	17	61	0.126E-02
1.5	17	59	0.478E-03
1.6	11	39	0.885E-02
1.7	13	48	0.107E-02
1.8	13	47	0.465E-02
1.9	10	38	0.667E-02
2.0	11	42	0.379E-03
2.1	11	41	0.118E-02
2.2	11	40	0.190E-02
2.3	13	47	0.186E-02
2.4	11	40	0.949E-02
2.5	13	47	0.655E-03
2.6	19	66	0.189E-03
2.7	13	44	0.224E-02
2.8	5	17	1.94
2.9	9	32	0.276E-01
3.0	18	62	0.896E-03
3.1	14	50	0.744E-03
3.2	15	52	0.214E-02
3.3	15	54	0.578E-03
3.4	16	56	0.308E-03
3.5	13	46	0.543E-03
3.6	12	41	0.145E-02
3.7	11	39	0.504E-02
3.8	10	38	0.189E-02
3.9	10	37	0.673E-02
4.0	11	41	0.342E-03

PRINT FROM NELME
****

MINIMIZATION OF F(X)=X(1)**2+AINT(ABS(X(1)))+5*X(2)**2+AINT(X(2))
UNTIL F(X) < 1.E-3 FOR DIFFERENT SIZES OF THE INITIAL POLYHEDRON.
STARTING POINT IS (2.0,2.0).

SIZE OF THE INITIAL POLYHEDRON	NUMBER OF ITERATIONS REQUIRED	NUMBER OF EVALUATIONS REQUIRED	TESTQ OF NELME
4.1	9	35	0.256E-02
4.2	10	35	0.138E-01
4.3	10	39	0.749E-03
4.4	11	41	0.113E-02
4.5	10	39	0.181E-02
4.6	11	41	0.238E-03
4.7	10	37	0.132E-02
4.8	9	34	0.981E-02
4.9	1	6	16.9
5.0	10	39	0.127E-02
5.1	11	41	0.115E-02
5.2	13	47	0.210E-02
5.3	9	35	0.142E-02
5.4	9	35	0.771E-02
5.5	8	32	0.401E-02
5.6	11	42	0.610E-03
5.7	10	39	0.494E-03
5.8	17	61	0.581E-03
5.9	17	59	0.598E-03
6.0	11	41	0.339E-02
6.1	12	42	0.244E-02
6.2	13	47	0.175E-02
6.3	9	36	0.126E-02
6.4	11	41	0.603E-03
6.5	12	44	0.134E-02
6.6	10	37	0.564E-02
6.7	10	38	0.974E-03
6.8	11	42	0.134E-02
6.9	9	37	0.214E-02
7.0	10	41	0.224E-03
7.1	11	43	0.209E-02
7.2	12	48	0.577E-03
7.3	11	44	0.344E-02
7.4	12	47	0.161E-02
7.5	12	46	0.398E-03
7.6	13	47	0.112E-02
7.7	11	41	0.337E-02
7.8	3	15	3.01
7.9	11	44	0.530E-03
8.0	11	44	0.379E-03

PRINT FROM NELME
****

MINIMIZATION OF F(X)=X(1)**2+AINT(ABS(X(1)))+5*X(2)**2+AINT(X(2))
UNTIL F(X) < 1.E-5 FOR DIFFERENT SIZES OF THE INITIAL POLYHEDRON.
STARTING POINT IS (2.0,2.0).

SIZE OF THE INITIAL POLYHEDRON	NUMBER OF ITERATIONS REQUIRED	NUMBER OF EVALUATIONS REQUIRED	TESTO OF NELME
0.1	29	96	0.376E-05
0.2	25	85	0.119E-04
0.3	22	78	0.137E-04
0.4	23	80	0.111E-04
0.5	14	48	0.141E-02
0.6	22	72	0.229E-04
0.7	24	83	0.554E-05
0.8	20	70	0.335E-04
0.9	19	68	0.150E-04
1.0	24	78	0.280E-04
1.1	21	72	0.578E-03
1.2	17	62	0.214E-04
1.3	15	57	0.198E-04
1.4	20	73	0.174E-04
1.5	18	62	0.607E-03
1.6	19	67	0.874E-05
1.7	19	70	0.992E-05
1.8	17	60	0.173E-03
1.9	17	61	0.136E-03
2.0	14	53	0.189E-04
2.1	15	55	0.617E-05
2.2	16	58	0.232E-04
2.3	18	65	0.407E-04
2.4	14	51	0.343E-03
2.5	18	65	0.815E-05
2.6	25	86	0.704E-04
2.7	20	69	0.104E-04
2.8	5	17	1.94
2.9	19	65	0.480E-05
3.0	20	69	0.635E-03
3.1	18	65	0.168E-04
3.2	24	82	0.125E-04
3.3	19	67	0.197E-04
3.4	25	84	0.104E-04
3.5	19	65	0.859E-05
3.6	19	66	0.830E-05
3.7	17	61	0.321E-05
3.8	16	59	0.686E-05
3.9	15	54	0.222E-04
4.0	15	55	0.788E-04

PRINT FROM NELME
****

MINIMIZATION OF F(X)=X(1)**2+AINT(ABS(X(1)))+5*X(2)**2+AINT(X(2))
UNTIL F(X) < 1.E-5 FOR DIFFERENT SIZES OF THE INITIAL POLYHEDRON.
STARTING POINT IS (2.0,2.0).

SIZE OF THE INITIAL POLYHEDRON	NUMBER OF ITERATIONS REQUIRED	NUMBER OF EVALUATIONS REQUIRED	TESTQ OF NELME
4.1	13	51	0.128E-04
4.2	16	56	0.107E-04
4.3	17	63	0.194E-04
4.4	16	58	0.114E-04
4.5	14	54	0.121E-04
4.6	15	56	0.543E-05
4.7	14	53	0.726E-05
4.8	14	52	0.142E-04
4.9	1	6	16.9
5.0	16	58	0.279E-04
5.1	20	69	0.935E-05
5.2	20	70	0.137E-04
5.3	13	50	0.670E-05
5.4	15	56	0.203E-04
5.5	14	54	0.103E-04
5.6	17	63	0.210E-04
5.7	16	60	0.119E-04
5.8	22	76	0.372E-04
5.9	24	82	0.109E-04
6.0	15	56	0.276E-04
6.1	16	57	0.365E-04
6.2	15	55	0.331E-03
6.3	14	54	0.201E-04
6.4	17	62	0.168E-04
6.5	17	62	0.194E-04
6.6	19	69	0.452E-05
6.7	16	60	0.512E-05
6.8	15	56	0.628E-04
6.9	13	52	0.128E-04
7.0	14	56	0.219E-04
7.1	16	61	0.606E-05
7.2	18	66	0.491E-05
7.3	15	58	0.237E-04
7.4	18	68	0.165E-04
7.5	20	74	0.939E-05
7.6	21	75	0.914E-05
7.7	15	57	0.476E-05
7.8	18	67	0.402E-05
7.9	12	48	0.194E-03
8.0	14	55	0.189E-04

PRINT FROM NELME
****

MINIMIZATION OF F(X)=X(1)**2+AINT(ABS(X(1)))+5*X(2)**2+AINT(X(2))
UNTIL F(X) < 1.E-7 FOR DIFFERENT SIZES OF THE INITIAL POLYHEDRON.
STARTING POINT IS (2.0,2.0).

SIZE OF THE INITIAL POLYHEDRON	NUMBER OF ITERATIONS REQUIRED	NUMBER OF EVALUATIONS REQUIRED	TESTQ OF NELME
0.1	34	115	0.335E-07
0.2	30	103	0.156E-06
0.3	27	94	0.236E-06
0.4	27	95	0.668E-07
0.5	23	79	0.229E-06
0.6	25	84	0.522E-06
0.7	28	98	0.133E-06
0.8	26	91	0.585E-07
0.9	23	82	0.181E-06
1.0	28	94	0.787E-07
1.1	26	91	0.182E-05
1.2	20	72	0.203E-05
1.3	21	78	0.151E-06
1.4	26	93	0.108E-06
1.5	30	100	0.606E-07
1.6	22	78	0.111E-05
1.7	25	90	0.113E-06
1.8	23	83	0.126E-06
1.9	36	118	0.791E-06
2.0	21	78	0.107E-06
2.1	21	77	0.703E-07
2.2	23	83	0.110E-06
2.3	26	93	0.983E-07
2.4	18	66	0.267E-05
2.5	23	83	0.404E-07
2.6	32	112	0.695E-07
2.7	28	97	0.385E-07
2.8	17	64	0.460E-06
2.9	24	82	0.889E-06
3.0	27	95	0.371E-07
3.1	22	79	0.399E-06
3.2	29	100	0.912E-07
3.3	23	82	0.116E-06
3.4	26	87	0.105E-04
3.5	23	80	0.297E-06
3.6	26	90	0.988E-07
3.7	22	78	0.273E-06
3.8	18	67	0.110E-05
3.9	20	73	0.744E-07
4.0	23	81	0.838E-07

PRINT FROM NELME
****

MINIMIZATION OF F(X)=X(1)**2+AINT(ABS(X(1)))+5*X(2)**2+AINT(X(2))
UNTIL F(X) < 1.E-7 FOR DIFFERENT SIZES OF THE INITIAL POLYHEDRON.
STARTING POINT IS (2.0,2.0).

SIZE OF THE INITIAL POLYHEDRON	NUMBER OF ITERATIONS REQUIRED	NUMBER OF EVALUATIONS REQUIRED	TESTO OF NELME
4.1	20	73	0.810E-07
4.2	20	72	0.536E-07
4.3	21	77	0.407E-06
4.4	28	99	0.123E-06
4.5	19	73	0.943E-07
4.6	20	73	0.128E-06
4.7	16	61	0.481E-06
4.8	20	70	0.528E-06
4.9	17	63	0.657E-06
5.0	25	89	0.150E-06
5.1	26	87	0.312E-06
5.2	25	88	0.432E-07
5.3	18	67	0.202E-06
5.4	19	70	0.197E-06
5.5	19	71	0.381E-06
5.6	23	83	0.467E-07
5.7	21	78	0.326E-06
5.8	30	104	0.632E-07
5.9	29	101	0.583E-07
6.0	21	75	0.375E-06
6.1	24	83	0.262E-06
6.2	22	80	0.497E-07
6.3	19	72	0.763E-07
6.4	24	86	0.985E-07
6.5	21	76	0.860E-06
6.6	23	83	0.303E-06
6.7	19	70	0.699E-06
6.8	18	68	0.115E-05
6.9	19	73	0.187E-06
7.0	18	70	0.632E-06
7.1	21	79	0.207E-06
7.2	21	76	0.800E-06
[illegible]	[illegible]	77	0.419E-07
[illegible]	[illegible]	101	0.498E-07
[illegible]	[illegible]	[illegible]	0.245E-06
[illegible]	[illegible]	[illegible]	0.325E-05
[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]

PRINT FROM NELME
****

MINIMIZATION OF F(X)=X(1)**2+AINT(ABS(X(1)))+5*X(2)**2+AINT(X(2))
UNTIL F(X) < 1.E-1 FOR DIFFERENT SIZES OF THE INITIAL POLYHEDRON.
STARTING POINT IS (2.0,-2.0).

SIZE OF THE INITIAL POLYHEDRON	NUMBER OF ITERATIONS REQUIRED	NUMBER OF EVALUATIONS REQUIRED	TESTQ OF NELME
0.1	16	46	0.464
0.2	16	51	0.640E-01
0.3	11	36	0.506E-01
0.4	25	79	0.988E-01
0.5	20	65	0.501E-01
0.6	11	37	0.128
0.7	6	20	0.505
0.8	8	27	0.163
0.9	7	26	0.136
1.0	10	35	0.106
1.1	12	41	0.323E-01
1.2	9	31	0.570E-01
1.3	9	33	0.224E-01
1.4	6	23	0.440E-01
1.5	5	19	0.202
1.6	5	19	0.580
1.7	6	23	0.566E-01
1.8	8	26	0.153
1.9	5	16	2.39
2.0	17	57	0.615E-01
2.1	10	32	0.404E-01
2.2	6	20	1.37
2.3	5	19	0.316
2.4	5	19	0.469
2.5	5	19	0.488
2.6	7	25	0.268E-01
2.7	7	25	0.656E-01
2.8	5	18	0.449
2.9	5	18	0.634
3.0	5	18	0.984
3.1	7	24	0.124
3.2	7	26	0.155
3.3	8	29	0.123
3.4	6	21	1.56
3.5	6	21	1.63
3.6	9	30	0.152
3.7	9	29	0.476
3.8	7	26	0.339E-01
3.9	7	25	0.990E-01
4.0	7	25	0.660E-01

PRINT FROM NELME
****

MINIMIZATION OF F(X)=X(1)**2+AINT(ABS(X(1)))+5*X(2)**2+AINT(X(2))
UNTIL F(X) < 1.E-1 FOR DIFFERENT SIZES OF THE INITIAL POLYHEDRON.
STARTING POINT IS (2.0,-2.0).

SIZE OF THE INITIAL POLYHEDRON	NUMBER OF ITERATIONS REQUIRED	NUMBER OF EVALUATIONS REQUIRED	TESTQ OF NELME
4.1	7	25	0.465E-01
4.2	7	25	0.134
4.3	7	25	0.209
4.4	6	22	0.362
4.5	7	26	0.116E-01
4.6	7	26	0.775E-01
4.7	7	26	0.208
4.8	7	26	0.989E-01
4.9	8	30	0.362
5.0	7	27	0.688
5.1	7	27	0.558
5.2	7	26	0.413
5.3	8	30	0.228
5.4	8	30	0.302
5.5	10	38	0.785E-01
5.6	9	33	0.772E-01
5.7	9	34	0.465E-01
5.8	10	36	0.795E-01
5.9	6	25	0.236
6.0	6	25	0.183
6.1	6	25	0.131
6.2	6	25	0.833E-01
6.3	7	29	0.924E-01
6.4	6	25	0.878E-01
6.5	6	25	0.142
6.6	7	26	0.274
6.7	7	26	0.227
6.8	7	26	0.384
6.9	9	32	0.104
7.0	8	31	0.271
7.1	7	28	0.219
7.2	8	31	0.137
7.3	6	26	0.781E-01
7.4	6	26	0.111
7.5	7	29	0.323E-01
7.6	6	25	0.234
7.7	6	25	0.259
7.8	6	25	0.294
7.9	7	29	0.589E-01
8.0	4	19	0.422

PRINT FROM NELME
****

MINIMIZATION OF F(X)=X(1)**2+AINT(ABS(X(1)))+5*X(2)**2+AINT(X(2))
UNTIL F(X) < 1.E-3 FOR DIFFERENT SIZES OF THE INITIAL POLYHEDRON.
STARTING POINT IS (2.0,-2.0).

SIZE OF THE INITIAL POLYHEDRON	NUMBER OF ITERATIONS REQUIRED	NUMBER OF EVALUATIONS REQUIRED	TESTQ OF NELME
0.1	23	69	0.109E-02
0.2	24	76	0.163E-02
0.3	18	58	0.759E-03
0.4	29	93	0.159E-02
0.5	23	76	0.432E-02
0.6	11	37	0.128
0.7	13	44	0.929E-03
0.8	11	39	0.514E-02
0.9	12	44	0.468E-02
1.0	14	50	0.277E-02
1.1	19	65	0.228E-02
1.2	22	74	0.710E-03
1.3	15	54	0.204E-02
1.4	10	36	0.479E-02
1.5	10	37	0.888E-03
1.6	11	40	0.195E-02
1.7	9	35	0.479E-03
1.8	13	42	0.198E-02
1.9	11	37	0.673E-02
2.0	20	68	0.232E-02
2.1	19	59	0.204E-02
2.2	16	53	0.198E-02
2.3	9	34	0.679E-02
2.4	11	41	0.921E-03
2.5	9	35	0.342E-02
2.6	12	42	0.280E-02
2.7	13	44	0.159E-02
2.8	12	44	0.610E-03
2.9	10	36	0.188E-02
3.0	12	41	0.117E-02
3.1	10	38	0.889E-03
3.2	11	41	0.301E-02
3.3	11	40	0.118E-01
3.4	12	44	0.863E-03
3.5	12	43	0.150E-02
3.6	20	68	0.779E-03
3.7	16	53	0.115E-02
3.8	12	43	0.130E-02
3.9	17	59	0.970E-03
4.0	12	42	0.124E-02

PRINT FROM NELME
****

MINIMIZATION OF F(X)=X(1)**2+AINT(ABS(X(1)))+5*X(2)**2+AINT(X(2))
UNTIL F(X) < 1.E-3 FOR DIFFERENT SIZES OF THE INITIAL POLYHEDRON.
STARTING POINT IS (2.0,-2.0).

SIZE OF THE INITIAL POLYHEDRON	NUMBER OF ITERATIONS REQUIRED	NUMBER OF EVALUATIONS REQUIRED	TESTQ OF NELME
4.1	10	37	0.147E-02
4.2	12	44	0.699E-03
4.3	11	39	0.123E-02
4.4	10	38	0.224E-02
4.5	12	44	0.163E-02
4.6	11	41	0.830E-03
4.7	15	52	0.419E-02
4.8	12	45	0.660E-03
4.9	23	77	0.142E-02
5.0	11	41	0.906E-02
5.1	16	55	0.885E-03
5.2	17	60	0.910E-03
5.3	12	46	0.777E-03
5.4	13	48	0.113E-02
5.5	15	55	0.490E-03
5.6	14	51	0.133E-02
5.7	15	54	0.169E-02
5.8	20	70	0.172E-02
5.9	13	48	0.216E-02
6.0	14	52	0.888E-03
6.1	12	45	0.158E-02
6.2	9	37	0.403E-02
6.3	15	55	0.176E-02
6.4	10	40	0.235E-02
6.5	12	47	0.537E-03
6.6	12	45	0.124E-02
6.7	9	34	0.367E-01
6.8	12	44	0.896E-03
6.9	12	42	0.838E-02
7.0	16	59	0.766E-03
7.1	21	74	0.183E-03
7.2	17	61	0.108E-02
7.3	12	45	0.312E-03
7.4	15	52	0.177E-02
7.5	10	40	0.662E-02
7.6	12	47	0.435E-03
7.7	13	49	0.120E-02
7.8	9	37	0.610E-02
7.9	11	43	0.629E-02
8.0	12	45	0.190E-02

PRINT FROM NELME
****

MINIMIZATION OF F(X)=X(1)**2+AINT(ABS(X(1)))+5*X(2)**2+AINT(X(2))
UNTIL F(X) < 1.E-5 FOR DIFFERENT SIZES OF THE INITIAL POLYHEDRON.
STARTING POINT IS (2.0,-2.0).

SIZE OF THE INITIAL POLYHEDRON	NUMBER OF ITERATIONS REQUIRED	NUMBER OF EVALUATIONS REQUIRED	TESTQ OF NELME
0.1	28	85	0.519E-05
0.2	29	94	0.172E-04
0.3	25	80	0.640E-05
0.4	37	120	0.126E-04
0.5	26	86	0.548E-03
0.6	20	67	0.986E-05
0.7	18	60	0.351E-05
0.8	15	55	0.759E-05
0.9	18	66	0.116E-04
1.0	20	71	0.124E-04
1.1	24	83	0.160E-04
1.2	24	82	0.487E-04
1.3	21	76	0.679E-05
1.4	15	55	0.128E-04
1.5	13	49	0.630E-05
1.6	15	56	0.818E-05
1.7	14	53	0.362E-04
1.8	18	61	0.744E-05
1.9	15	52	0.269E-04
2.0	24	83	0.283E-04
2.1	23	75	0.105E-04
2.2	22	72	0.369E-04
2.3	18	63	0.105E-04
2.4	15	56	0.641E-05
2.5	9	35	0.342E-02
2.6	17	59	0.108E-04
2.7	16	56	0.348E-04
2.8	15	54	0.512E-04
2.9	13	48	0.573E-04
3.0	19	65	0.642E-05
3.1	16	58	0.161E-04
3.2	15	57	0.456E-05
3.3	18	61	0.101E-03
3.4	20	72	0.835E-05
3.5	15	55	0.179E-04
3.6	23	80	0.297E-04
3.7	18	60	0.180E-03
3.8	17	61	0.228E-04
3.9	17	59	0.970E-03
4.0	16	57	0.704E-05

PRINT FROM NELME
****

MINIMIZATION OF F(X)=X(1)**2+AINT(ABS(X(1)))+5*X(2)**2+AINT(X(2))
UNTIL F(X) < 1.E-5 FOR DIFFERENT SIZES OF THE INITIAL POLYHEDRON.
STARTING POINT IS (2.0,-2.0).

SIZE OF THE INITIAL POLYHEDRON	NUMBER OF ITERATIONS REQUIRED	NUMBER OF EVALUATIONS REQUIRED	TESTQ OF NELME
4.1	17	61	0.217E-05
4.2	16	56	0.282E-03
4.3	16	58	0.823E-05
4.4	17	61	0.742E-05
4.5	19	69	0.116E-04
4.6	14	53	0.324E-04
4.7	20	71	0.144E-04
4.8	15	55	0.434E-04
4.9	30	99	0.142E-04
5.0	20	71	0.220E-04
5.1	19	66	0.142E-03
5.2	24	83	0.123E-04
5.3	15	58	0.237E-04
5.4	15	56	0.993E-04
5.5	18	67	0.120E-04
5.6	18	67	0.113E-04
5.7	20	71	0.218E-05
5.8	24	84	0.169E-04
5.9	22	78	0.843E-05
6.0	17	64	0.630E-05
6.1	18	66	0.105E-04
6.2	13	53	0.184E-04
6.3	18	67	0.198E-04
6.4	17	63	0.173E-04
6.5	14	55	0.584E-04
6.6	19	68	0.571E-05
6.7	15	55	0.941E-04
6.8	15	55	0.497E-04
6.9	18	64	0.383E-05
7.0	20	73	0.463E-05
7.1	24	83	0.179E-03
7.2	20	71	0.792E-04
7.3	19	68	0.107E-04
7.4	20	70	0.926E-05
7.5	18	67	0.523E-05
7.6	15	59	0.202E-04
7.7	23	83	0.248E-05
7.8	13	53	0.124E-04
7.9	19	69	0.194E-04
8.0	20	73	0.239E-04

PRINT FROM NELME
****

MINIMIZATION OF F(X)=X(1)**2+AINT(ABS(X(1)))+5*X(2)**2+AINT(X(2))
UNTIL F(X) < 1.E-7 FOR DIFFERENT SIZES OF THE INITIAL POLYHEDRON.
STARTING POINT IS (2.0,-2.0).

SIZE OF THE INITIAL POLYHEDRON	NUMBER OF ITERATIONS REQUIRED	NUMBER OF EVALUATIONS REQUIRED	TESTQ OF NELME
0.1	33	104	0.498E-07
0.2	33	108	0.209E-06
0.3	29	94	0.308E-06
0.4	42	135	0.906E-06
0.5	38	126	0.526E-07
0.6	24	80	0.944E-06
0.7	25	82	0.110E-06
0.8	19	69	0.423E-06
0.9	22	80	0.108E-06
1.0	24	86	0.333E-06
1.1	24	83	0.160E-04
1.2	31	107	0.914E-07
1.3	25	91	0.166E-06
1.4	18	67	0.451E-07
1.5	17	63	0.218E-06
1.6	18	67	0.492E-06
1.7	20	74	0.138E-06
1.8	21	72	0.461E-06
1.9	21	74	0.122E-06
2.0	28	99	0.179E-06
2.1	31	100	0.949E-06
2.2	29	95	0.497E-07
2.3	24	83	0.291E-07
2.4	20	75	0.494E-07
2.5	20	74	0.465E-07
2.6	23	80	0.418E-07
2.7	22	78	0.408E-07
2.8	21	74	0.953E-07
2.9	18	65	0.121E-06
3.0	21	72	0.246E-05
3.1	20	71	0.887E-06
3.2	20	73	0.207E-06
3.3	26	91	0.437E-07
3.4	25	89	0.903E-07
3.5	22	79	0.207E-06
3.6	27	96	0.874E-07
3.7	25	85	0.668E-07
3.8	21	76	0.338E-06
3.9	25	88	0.685E-07
4.0	22	78	0.689E-07

PRINT FROM NELME
****

MINIMIZATION OF F(X)=X(1)**2+4INT(ABS(X(1)))+X(2)**2+4INT(X(2))
UNTIL F(X) < 1.E-7 FOR DIFFERENT SIZES OF THE INITIAL POLYHEDRON.
STARTING POINT IS (2.0,-2.0).

SIZE OF THE INITIAL POLYHEDRON	NUMBER OF ITERATIONS REQUIRED	NUMBER OF EVALUATIONS REQUIRED	TEST? OF NELME
4.1	23	81	[illegible]
4.2	27	[illegible]	0.109E-06
4.3	[illegible]	88	0.152E-06
4.4	22	82	0.160E-07
4.5	26	94	0.660E-07
4.6	22	80	0.597E-07
4.7	25	88	0.578E-07
4.8	22	77	0.414E-06
4.9	39	129	0.102E-06
5.0	28	98	0.179E-07
5.1	27	94	0.454E-07
5.2	33	113	0.651E-07
5.3	20	76	0.666E-07
5.4	24	83	0.270E-06
5.5	25	91	0.103E-06
5.6	24	86	0.453E-07
5.7	24	85	0.639E-06
5.8	31	108	0.181E-06
5.9	26	93	0.151E-06
6.0	21	78	0.219E-06
6.1	20	74	0.416E-06
6.2	19	74	0.830E-07
6.3	22	83	0.703E-07
6.4	20	75	0.206E-06
6.5	21	78	0.145E-06
6.6	22	78	0.423E-06
6.7	20	73	0.441E-07
6.8	22	81	0.673E-07
6.9	23	82	0.908E-07
7.0	24	87	0.127E-06
7.1	33	116	0.539E-07
7.2	30	103	0.416E-07
7.3	23	81	0.157E-06
7.4	23	82	0.198E-06
7.5	20	75	0.346E-06
7.6	19	74	0.285E-06
7.7	30	106	0.805E-07
7.8	17	69	0.114E-06
7.9	21	77	0.317E-05
8.0	26	95	0.917E-07

PRINT FROM NELME
****

MINIMIZATION OF F(X)=X(1)**2+AINT(ABS(X(1)))+5*X(2)**2+AINT(X(2))
UNTIL F(X) < 1.E-1 FOR DIFFERENT SIZES OF THE INITIAL POLYHEDRON.
STARTING POINT IS (-2.0,2.0).

SIZE OF THE INITIAL POLYHEDRON	NUMBER OF ITERATIONS REQUIRED	NUMBER OF EVALUATIONS REQUIRED	TESTR OF NELME
0.1	21	65	0.187
0.2	21	67	0.363
0.3	26	77	0.522E-01
0.4	11	31	0.377
0.5	7	26	0.997E-01
0.6	7	25	0.607E-01
0.7	6	22	0.131
0.8	2	9	9.79
0.9	2	9	9.54
1.0	7	26	0.593E-01
1.1	8	31	0.541E-01
1.2	2	9	9.19
1.3	5	21	0.183
1.4	6	23	0.106
1.5	4	17	1.45
1.6	12	43	0.867E-02
1.7	7	25	0.203
1.8	6	23	0.580
1.9	6	23	0.268
2.0	5	20	0.660E-01
2.1	6	23	0.485E-01
2.2	4	17	0.362
2.3	5	21	0.775E-01
2.4	5	21	0.989E-01
2.5	7	27	0.624E-01
2.6	7	27	0.123
2.7	6	23	0.557
2.8	8	30	0.124
2.9	7	27	0.200
3.0	6	24	0.183
3.1	6	24	0.833E-01
3.2	6	23	0.878E-01
3.3	6	23	0.204
3.4	7	26	0.102
3.5	6	22	0.547
3.6	8	28	0.292E-01
3.7	5	20	0.634E-01
3.8	4	16	0.548
3.9	4	16	0.755
4.0	4	16	0.740

PRINT FROM NELME
****

MINIMIZATION OF F(X)=X(1)**2+AINT(ABS(X(1)))+5*X(2)**2+AINT(X(2))
UNTIL F(X) < 1.E-1 FOR DIFFERENT SIZES OF THE INITIAL POLYHEDRON.
STARTING POINT IS (-2.0,2.0).

SIZE OF THE INITIAL POLYHEDRON	NUMBER OF ITERATIONS REQUIRED	NUMBER OF EVALUATIONS REQUIRED	TESTQ OF NELME
4.1	4	16	0.734
4.2	5	19	0.221
4.3	7	28	0.531E-01
4.4	6	24	0.139
4.5	6	24	0.161
4.6	6	24	0.199
4.7	8	29	0.122
4.8	9	33	0.144
4.9	9	34	0.174
5.0	7	26	0.834E-01
5.1	8	29	0.142
5.2	9	32	0.658E-01
5.3	8	31	0.383E-01
5.4	5	22	0.682E-01
5.5	5	21	0.134
5.6	6	25	0.372E-01
5.7	6	25	0.513E-01
5.8	6	24	0.371E-01
5.9	4	18	0.330
6.0	4	18	0.398
6.1	4	18	0.472
6.2	4	18	0.551
6.3	5	21	0.390
6.4	5	21	0.371
6.5	5	20	0.298
6.6	6	24	0.154
6.7	5	21	0.330
6.8	6	25	0.335E-01
6.9	6	25	0.565E-01
7.0	6	25	0.826E-01
7.1	6	25	0.110
7.2	6	25	0.148
7.3	9	34	0.489E-01
7.4	4	18	0.593
7.5	4	18	0.522
7.6	5	22	0.481E-01
7.7	5	22	0.799E-01
7.8	5	22	0.990E-01
7.9	5	22	0.106
8.0	5	22	0.660E-01

PRINT FROM NELME
****

MINIMIZATION OF F(X)=X(1)**2+AINT(ABS(X(1)))+5*X(2)**2+AINT(X(2))
UNTIL F(X) < 1.E-3 FOR DIFFERENT SIZES OF THE INITIAL POLYHEDRON.
STARTING POINT IS (-2.0,2.0).

SIZE OF THE INITIAL POLYHEDRON	NUMBER OF ITERATIONS REQUIRED	NUMBER OF EVALUATIONS REQUIRED	TESTQ OF NELME
0.1	25	80	0.238E-02
0.2	28	90	0.109E-02
0.3	33	100	0.130E-02
0.4	16	48	0.277E-02
0.5	11	42	0.645E-03
0.6	20	68	0.710E-03
0.7	13	44	0.386E-02
0.8	14	48	0.904E-03
0.9	11	41	0.167E-02
1.0	15	53	0.452E-03
1.1	14	52	0.101E-02
1.2	15	53	0.241E-02
1.3	12	46	0.150E-02
1.4	14	50	0.716E-03
1.5	13	46	0.995E-03
1.6	18	62	0.207E-01
1.7	14	46	0.195E-02
1.8	11	41	0.550E-03
1.9	14	49	0.876E-03
2.0	10	37	0.124E-02
2.1	12	44	0.532E-03
2.2	8	33	0.224E-02
2.3	9	36	0.830E-03
2.4	10	40	0.660E-03
2.5	16	58	0.113E-03
2.6	11	43	0.171E-02
2.7	10	39	0.157E-02
2.8	12	45	0.133E-02
2.9	11	42	0.333E-02
3.0	14	51	0.988E-03
3.1	9	36	0.403E-02
3.2	10	38	0.235E-02
3.3	10	39	0.124E-02
3.4	11	41	0.604E-03
3.5	15	52	0.881E-03
3.6	10	36	0.114E-01
3.7	10	39	0.510E-03
3.8	10	38	0.134E-02
3.9	7	29	0.165E-01
4.0	9	35	0.798E-03

PRINT FROM NELME
****

MINIMIZATION OF F(X)=X(1)**2+AINT(ABS(X(1)))+5*X(2)**2+AINT(X(2))
UNTIL F(X) < 1.E-3 FOR DIFFERENT SIZES OF THE INITIAL POLYHEDRON.
STARTING POINT IS (-2.0,2.0).

SIZE OF THE INITIAL POLYHEDRON	NUMBER OF ITERATIONS REQUIRED	NUMBER OF EVALUATIONS REQUIRED	TESTQ OF NELME
4.1	8	32	0.240E-02
4.2	10	37	0.485E-03
4.3	10	37	0.121E-01
4.4	10	39	0.206E-02
4.5	10	37	0.889E-02
4.6	10	36	0.180E-01
4.7	14	51	0.684E-03
4.8	16	55	0.910E-03
4.9	16	57	0.570E-03
5.0	10	38	0.407E-02
5.1	17	56	0.188E-02
5.2	12	43	0.280E-02
5.3	10	38	0.132E-01
5.4	8	34	0.117E-02
5.5	9	36	0.171E-02
5.6	12	45	0.108E-02
5.7	11	42	0.109E-02
5.8	9	36	0.273E-03
5.9	9	35	0.155E-02
6.0	12	45	0.261E-02
6.1	11	42	0.133E-02
6.2	9	36	0.386E-03
6.3	13	47	0.865E-02
6.4	12	45	0.801E-02
6.5	11	42	0.759E-03
6.6	10	38	0.796E-03
6.7	12	46	0.677E-03
6.8	12	44	0.197E-02
6.9	8	32	0.216E-01
7.0	12	44	0.237E-02
7.1	13	48	0.872E-03
7.2	15	52	0.158E-02
7.3	16	56	0.543E-03
7.4	8	34	0.127E-02
7.5	10	39	0.564E-03
7.6	11	42	0.919E-03
7.7	10	40	0.147E-02
7.8	15	56	0.970E-03
7.9	8	32	0.116E-01
8.0	10	39	0.124E-02

PRINT FROM NELME
****

MINIMIZATION OF F(X)=X(1)**2+AINT(ABS(X(1)))+5*X(2)**2+AINT(X(2))
UNTIL F(X) < 1.E-5 FOR DIFFERENT SIZES OF THE INITIAL POLYHEDRON.
STARTING POINT IS (-2.0,2.0).

SIZE OF THE INITIAL POLYHEDRON	NUMBER OF ITERATIONS REQUIRED	NUMBER OF EVALUATIONS REQUIRED	TESTQ OF NELME
0.1	31	102	0.122E-04
0.2	34	111	0.958E-05
0.3	39	118	0.147E-04
0.4	22	69	0.395E-05
0.5	14	53	0.257E-04
0.6	22	76	0.487E-04
0.7	22	75	0.105E-04
0.8	20	68	0.246E-04
0.9	17	59	0.494E-04
1.0	23	78	0.327E-05
1.1	17	63	0.769E-04
1.2	19	67	0.796E-04
1.3	19	67	0.157E-04
1.4	18	63	0.107E-03
1.5	20	69	0.714E-05
1.6	24	83	0.415E-04
1.7	28	92	0.229E-04
1.8	18	64	0.356E-04
1.9	23	79	0.514E-05
2.0	14	52	0.704E-05
2.1	22	77	0.816E-05
2.2	15	56	0.742E-05
2.3	12	48	0.324E-04
2.4	13	50	0.434E-04
2.5	25	87	0.254E-04
2.6	15	57	0.175E-04
2.7	16	61	0.375E-05
2.8	16	61	0.113E-04
2.9	15	58	0.452E-05
3.0	17	63	0.630E-05
3.1	13	52	0.184E-04
3.2	17	61	0.172E-04
3.3	17	62	0.571E-05
3.4	16	60	0.408E-05
3.5	18	62	0.158E-03
3.6	17	60	0.675E-05
3.7	17	60	0.294E-04
3.8	16	57	0.273E-04
3.9	13	48	0.351E-04
4.0	14	54	0.459E-05

PRINT FROM NELME
****

MINIMIZATION OF F(X)=X(1)**2+AINT(ABS(X(1)))+5*X(2)**2+AINT(X(2))
UNTIL F(X) < 1.E-5 FOR DIFFERENT SIZES OF THE INITIAL POLYHEDRON.
STARTING POINT IS (-2.0,2.0).

SIZE OF THE INITIAL POLYHEDRON	NUMBER OF ITERATIONS REQUIRED	NUMBER OF EVALUATIONS REQUIRED	TESTQ OF NELME
4.1	12	46	0.459E-04
4.2	14	51	0.146E-04
4.3	15	57	0.229E-05
4.4	14	55	0.177E-05
4.5	17	61	0.692E-05
4.6	17	60	0.275E-04
4.7	18	64	0.381E-04
4.8	22	75	0.156E-04
4.9	19	68	0.536E-04
5.0	17	62	0.242E-04
5.1	22	74	0.109E-04
5.2	17	60	0.108E-04
5.3	16	58	0.751E-04
5.4	13	52	0.732E-05
5.5	16	59	0.281E-04
5.6	16	59	0.941E-05
5.7	14	54	0.170E-04
5.8	15	56	0.352E-04
5.9	15	55	0.395E-04
6.0	12	45	0.261E-02
6.1	16	59	0.147E-04
6.2	16	59	0.139E-04
6.3	19	69	0.892E-05
6.4	16	59	0.216E-03
6.5	16	59	0.138E-04
6.6	14	52	0.309E-04
6.7	19	67	0.143E-04
6.8	16	60	0.634E-05
6.9	16	61	0.218E-04
7.0	16	58	0.656E-04
7.1	15	55	0.139E-03
7.2	20	70	0.496E-05
7.3	21	74	0.117E-04
7.4	10	42	0.174E-03
7.5	12	47	0.164E-03
7.6	17	64	0.550E-05
7.7	14	55	0.326E-04
7.8	15	56	0.970E-03
7.9	13	51	0.617E-05
8.0	14	54	0.704E-05

PRINT FROM NELME
****

MINIMIZATION OF F(X)=X(1)**2+AINT(ABS(X(1)))+5*X(2)**2+AINT(X(2))
UNTIL F(X) < 1.E-7 FOR DIFFERENT SIZES OF THE INITIAL POLYHEDRON.
STARTING POINT IS (-2.0,2.0).

SIZE OF THE INITIAL POLYHEDRON	NUMBER OF ITERATIONS REQUIRED	NUMBER OF EVALUATIONS REQUIRED	TEST0 OF NELME
0.1	34	120	0.107E-06
0.2	36	119	0.914E-06
0.3	42	130	0.558E-06
0.4	24	84	0.241E-06
0.5	22	81	0.692E-07
0.6	29	101	0.911E-07
0.7	28	96	0.860E-07
0.8	28	94	0.141E-06
0.9	20	71	0.949E-06
1.0	31	104	0.402E-06
1.1	[illegible]	82	0.239E-06
1.2	22	88	0.112E-06
1.3	22	79	[illegible]
1.4	23	81	[illegible]
1.5	26	[illegible]	[illegible]
1.6	[illegible]	[illegible]	[illegible]
1.7	[illegible]	[illegible]	0.102E-06
1.8	[illegible]	[illegible]	0.155E-06
1.9	29?	104	0.127E-06
2.0	20	73	0.689E-07
2.1	25	89	0.264E-06
2.2	21	77	0.160E-07
2.3	20	75	0.600E-07
2.4	20	72	0.415E-06
2.5	30	106	0.584E-07
2.6	22	80	0.507E-07
2.7	19	73	0.246E-06
2.8	22	80	0.452E-07
2.9	21	79	0.181E-06
3.0	21	77	0.218E-06
3.1	19	73	0.831E-07
3.2	20	73	0.205E-06
3.3	20	72	0.424E-06
3.4	20	74	0.722E-06
3.5	24	84	0.274E-06
3.6	20	72	0.406E-07
3.7	21	76	0.791E-07
3.8	19	69	0.796E-06
3.9	19	69	0.252E-06
4.0	20	74	0.231E-06

PRINT FROM NELME
****

MINIMIZATION OF F(X)=X(1)**2+AINT(ABS(X(1)))+5*X(2)**2+AINT(X(2))
UNTIL F(X) (1.E-7 FOR DIFFERENT SIZES OF THE INITIAL POLYHEDRON.
STARTING POINT IS (-1.0,2.0)

SIZE OF THE INITIAL POLYHEDRON	NUMBER OF ITERATIONS REQUIRED	NUMBER OF EVALUATIONS REQUIRED	TESTQ OF NELME
4.1	17	64	0.149E-06
4.2	17	63	0.251E-06
4.3	24	84	0.142E-06
4.4	18	69	0.310E-06
4.5	21	74	0.490E-06
4.6	22	79	0.640E-07
4.7	23	80	0.888E-06
4.8	27	93	0.174E-06
4.9	26	92	0.127E-05
5.0	23	83	0.465E-07
5.1	26	96	0.833E-06
5.2	23	81	0.418E-07
5.3	20	74	0.284E-06
5.4	18	69	0.173E-06
5.5	20	69	0.519E-05
5.6	22	81	0.817E-07
5.7	18	68	0.262E-06
5.8	22	80	0.120E-06
5.9	20	72	0.182E-06
6.0	23	82	0.727E-07
6.1	21	76	0.530E-07
6.2	20	73	0.424E-06
6.3	22	80	0.936E-06
6.4	22	82	0.220E-06
6.5	22	79	0.396E-06
6.6	17	64	0.396E-06
6.7	24	85	0.156E-06
6.8	20	74	0.160E-06
6.9	22	82	0.539E-07
7.0	19	70	0.112E-05
7.1	22	80	0.637E-07
7.2	24	85	0.211E-06
7.3	26	90	0.165E-06
7.4	17	67	0.747E-07
7.5	18	70	0.827E-07
7.6	21	78	0.483E-07
7.7	18	69	0.820E-06
7.8	23	85	0.478E-07
7.9	17	67	0.717E-07
8.0	20	75	0.689E-07

PRINT FROM NELME
****

MINIMIZATION OF F(X)=X(1)**2+AINT(ABS(X(1)))+5*X(2)**2+AINT(X(2))
UNTIL F(X) < 1.E-1 FOR DIFFERENT SIZES OF THE INITIAL POLYHEDRON.
STARTING POINT IS (-2.0,-2.0).

SIZE OF THE INITIAL POLYHEDRON	NUMBER OF ITERATIONS REQUIRED	NUMBER OF EVALUATIONS REQUIRED	TESTQ OF NELME
0.1	16	51	0.661E-01
0.2	19	60	0.365E-01
0.3	6	20	0.929
0.4	14	48	0.882E-01
0.5	6	21	0.831
0.6	8	29	0.694E-01
0.7	9	31	0.414E-01
0.8	9	31	0.727E-01
0.9	8	31	0.334E-01
1.0	5	18	0.670
1.1	7	25	0.597E-01
1.2	6	21	0.126
1.3	7	25	0.694E-01
1.4	1	6	11.3
1.5	5	20	0.213
1.6	5	19	0.251
1.7	5	20	0.821
1.8	7	26	0.760E-01
1.9	6	24	0.155
2.0	5	21	0.203
2.1	6	25	0.107
2.2	5	22	0.315E-01
2.3	1	6	10.8
2.4	1	6	11.0
2.5	1	6	11.1
2.6	1	6	11.1
2.7	4	18	0.124
2.8	5	21	0.124
2.9	4	18	0.240
3.0	6	25	0.502E-01
3.1	5	21	0.261
3.2	5	21	0.152
3.3	4	18	0.215
3.4	5	22	0.671E-01
3.5	5	21	0.190
3.6	5	22	0.552E-01
3.7	3	14	2.07
3.8	7	27	0.539E-01
3.9	4	15	1.77
4.0	4	16	2.96

PRINT FROM NELME
****

MINIMIZATION OF F(X)=X(1)**2+AINT(ABS(X(1)))+5*X(2)**2+AINT(X(2))
UNTIL F(X) < 1.E-1 FOR DIFFERENT SIZES OF THE INITIAL POLYHEDRON.
STARTING POINT IS (-2.0,-2.0).

SIZE OF THE INITIAL POLYHEDRON	NUMBER OF ITERATIONS REQUIRED	NUMBER OF EVALUATIONS REQUIRED	TESTQ OF NELME
4.1	4	16	2.76
4.2	8	30	0.423E-01
4.3	6	24	0.786E-01
4.4	7	26	0.597E-01
4.5	7	26	0.165
4.6	8	30	0.958E-01
4.7	7	27	0.164
4.8	8	30	0.126
4.9	8	30	0.312E-01
5.0	7	27	0.912E-01
5.1	7	27	0.476E-01
5.2	9	33	0.694E-01
5.3	6	24	0.225
5.4	6	23	1.15
5.5	4	17	1.76
5.6	4	17	1.94
5.7	4	17	2.13
5.8	10	35	0.469E-01
5.9	3	14	2.88
6.0	6	26	0.538E-01
6.1	7	28	0.143
6.2	7	29	0.695E-01
6.3	7	29	0.556E-01
6.4	6	25	0.168
6.5	8	30	0.153
6.6	8	31	0.120
6.7	5	21	0.697
6.8	5	21	0.805
6.9	5	21	0.918
7.0	8	32	0.103
7.1	6	24	0.795
7.2	6	24	0.759
7.3	6	24	0.726
7.4	6	24	0.696
7.5	9	34	0.506E-01
7.6	9	34	0.154
7.7	9	34	0.103
7.8	7	27	0.873
7.9	6	25	0.218
8.0	6	25	0.266

PRINT FROM NELME
****

MINIMIZATION OF F(X)=X(1)**2+AINT(ABS(X(1)))+5*X(2)**2+AINT(X(2))
UNTIL F(X) < 1.E-3 FOR DIFFERENT SIZES OF THE INITIAL POLYHEDRON.
STARTING POINT IS (-2.0,-2.0).

SIZE OF THE INITIAL POLYHEDRON	NUMBER OF ITERATIONS REQUIRED	NUMBER OF EVALUATIONS REQUIRED	TESTQ OF NELME
0.1	22	72	0.725E-03
0.2	23	74	0.120E-02
0.3	15	49	0.233E-02
0.4	18	63	0.178E-02
0.5	11	39	0.199E-02
0.6	9	32	0.400E-01
0.7	15	52	0.259E-02
0.8	14	52	0.339E-02
0.9	[illegible]	50	0.137E-02
1.0	11	38	[illegible]
1.1	11	40	[illegible]
1.2	9	30	[illegible]
1.3	17	60	[illegible]
1.4	[illegible]	[illegible]	[illegible]
1.5	10	37	[illegible]
1.6	[illegible]	[illegible]	[illegible]
1.7	[illegible]	44	[illegible]
1.8	[illegible]	[illegible]	[illegible]
1.9	10	39	[illegible]
2.0	10	39	0.154E-02
2.1	12	44	0.391E-03
2.2	9	33	0.725E-02
2.3	11	40	0.237E-03
2.4	10	36	0.533E-03
2.5	10	38	0.127E-02
2.6	13	46	0.210E-02
2.7	11	43	0.629E-03
2.8	11	42	0.192E-02
2.9	12	45	0.463E-04
3.0	8	32	0.128E-01
3.1	8	32	0.506E-02
3.2	11	43	0.647E-03
3.3	11	43	0.993E-03
3.4	11	43	0.107E-02
3.5	10	40	0.224E-03
3.6	12	47	0.577E-03
3.7	12	44	0.161E-02
3.8	10	38	0.667E-02
3.9	4	15	1.77
4.0	11	40	0.321E-02

```
PRINT FROM NELME
****************

MINIMIZATION OF F(X)=X(1)**2+AINT(ABS(X(1)))+5*X(2)**2+AINT(X(2))
UNTIL F(X) < 1.E-3 FOR DIFFERENT SIZES OF THE INITIAL POLYHEDRON.
STARTING POINT IS (-2.0,-2.0).
```

SIZE OF THE INITIAL POLYHEDRON	NUMBER OF ITERATIONS REQUIRED	NUMBER OF EVALUATIONS REQUIRED	TESTQ OF NELME
4.1	14	52	0.272E-03
4.2	13	48	0.119E-02
4.3	13	49	0.838E-03
4.4	11	41	0.190E-02
4.5	14	51	0.582E-03
4.6	13	48	0.186E-02
4.7	11	41	0.600E-02
4.8	11	41	0.949E-02
4.9	15	54	0.150E-02
5.0	13	49	0.655E-03
5.1	12	45	0.688E-03
5.2	19	68	0.189E-03
5.3	13	49	0.408E-03
5.4	13	46	0.224E-02
5.5	9	37	0.143E-02
5.6	4	17	1.94
5.7	13	48	0.668E-03
5.8	16	56	0.148E-02
5.9	11	44	0.178E-02
6.0	12	46	0.353E-02
6.1	11	43	0.349E-02
6.2	11	43	0.180E-02
6.3	11	43	0.296E-02
6.4	13	50	0.217E-02
6.5	20	71	0.644E-03
6.6	16	58	0.262E-02
6.7	10	39	0.169E-01
6.8	14	53	0.309E-03
6.9	12	47	0.133E-02
7.0	13	48	0.230E-02
7.1	6	24	0.795
7.2	11	44	0.825E-03
7.3	12	47	0.535E-03
7.4	12	47	0.151E-02
7.5	14	52	0.879E-03
7.6	14	52	0.509E-03
7.7	18	64	0.589E-03
7.8	16	59	0.173E-02
7.9	12	47	0.188E-02
8.0	13	50	0.979E-03

PRINT FROM NELME
****

MINIMIZATION OF F(X)=X(1)**2+AINT(ABS(X(1)))+5*X(2)**2+AINT(X(2))
UNTIL F(X) < 1.E-5 FOR DIFFERENT SIZES OF THE INITIAL POLYHEDRON.
STARTING POINT IS (-2.0,-2.0).

SIZE OF THE INITIAL POLYHEDRON	NUMBER OF ITERATIONS REQUIRED	NUMBER OF EVALUATIONS REQUIRED	TESTQ OF NELME
0.1	26	87	0.401E-05
0.2	27	88	0.485E-04
0.3	20	68	0.139E-04
0.4	21	74	0.472E-04
0.5	17	59	0.379E-05
0.6	16	59	0.699E-05
0.7	19	67	0.108E-04
0.8	21	71	0.160E-04
0.9	18	66	0.850E-05
1.0	20	68	0.242E-04
1.1	16	58	0.232E-04
1.2	12	43	0.343E-03
1.3	23	80	0.704E-04
1.4	1	6	11.3
1.5	15	56	0.193E-04
1.6	14	60	0.657E-05
1.7	20	72	0.743E-05
1.8	16	56	0.129E-03
1.9	17	62	0.622E-05
2.0	13	49	0.125E-03
2.1	16	58	0.932E-05
2.2	16	59	0.111E-04
2.3	15	55	0.543E-05
2.4	14	51	0.111E-04
2.5	16	57	0.279E-04
2.6	20	69	0.137E-04
2.7	17	63	0.136E-04
2.8	17	64	0.638E-05
2.9	23	81	0.854E-05
3.0	18	65	0.676E-05
3.1	13	50	0.270E-04
3.2	17	64	0.141E-04
3.3	19	71	0.375E-05
3.4	17	65	0.992E-05
3.5	14	55	0.219E-04
3.6	18	65	0.491E-05
3.7	18	67	0.165E-04
3.8	17	61	0.136E-03
3.9	18	65	0.403E-05
4.0	20	70	0.242E-04

PRINT FROM NELME
****

MINIMIZATION OF F(X)=X(1)**2+AINT(ABS(X(1)))+5*X(2)**2+AINT(X(2))
UNTIL F(X) < 1.E-5 FOR DIFFERENT SIZES OF THE INITIAL POLYHEDRON.
STARTING POINT IS (-2.0,-2.0).

SIZE OF THE INITIAL POLYHEDRON	NUMBER OF ITERATIONS REQUIRED	NUMBER OF EVALUATIONS REQUIRED	TESTO OF NELME
4.1	20	72	0.724E-05
4.2	17	62	0.617E-05
4.3	19	68	0.863E-05
4.4	16	59	0.232E-04
4.5	18	65	0.299E-04
4.6	18	66	0.407E-04
4.7	16	59	0.884E-05
4.8	14	52	0.343E-03
4.9	21	76	0.580E-05
5.0	18	67	0.815E-05
5.1	16	59	0.170E-04
5.2	25	88	0.703E-04
5.3	16	60	0.514E-04
5.4	20	71	0.104E-04
5.5	17	65	0.613E-05
5.6	4	17	1.94
5.7	19	67	0.111E-04
5.8	24	83	0.313E-04
5.9	16	60	0.176E-04
6.0	15	58	0.155E-03
6.1	15	58	0.118E-04
6.2	13	51	0.826E-04
6.3	19	70	0.272E-04
6.4	19	71	0.850E-05
6.5	26	90	0.193E-04
6.6	23	83	0.103E-04
6.7	16	61	0.932E-05
6.8	23	81	0.104E-04
6.9	16	61	0.310E-04
7.0	23	81	0.186E-04
7.1	14	54	0.136E-03
7.2	16	62	0.235E-04
7.3	17	65	0.202E-04
7.4	18	67	0.396E-04
7.5	18	67	0.141E-04
7.6	18	66	0.461E-05
7.7	29	99	0.478E-05
7.8	24	84	0.722E-05
7.9	19	72	0.394E-05
8.0	20	74	0.195E-04

PRINT FROM NELME
****

MINIMIZATION OF F(X)=X(1)**2+AINT(ABS(X(1)))+5*X(2)**2+AINT(X(2))
UNTIL F(X) < 1.E-7 FOR DIFFERENT SIZES OF THE INITIAL POLYHEDRON.
STARTING POINT IS (-2.0,-2.0).

SIZE OF THE INITIAL POLYHEDRON	NUMBER OF ITERATIONS REQUIRED	NUMBER OF EVALUATIONS REQUIRED	TESTQ OF NELME
0.1	31	104	0.599E-07
0.2	31	104	0.688E-07
0.3	24	81	0.117E-05
0.4	28	99	0.685E-07
0.5	22	75	0.157E-06
0.6	21	77	0.942E-07
0.7	24	84	0.163E-06
0.8	25	85	0.637E-06
0.9	23	82	0.143E-06
1.0	26	90	0.840E-07
1.1	23	83	0.111E-06
1.2	16	58	0.267E-05
1.3	30	106	0.694E-07
1.4	15	59	0.217E-06
1.5	21	76	0.158E-06
1.6	18	67	0.164E-05
1.7	27	94	0.999E-07
1.8	21	76	0.750E-07
1.9	20	73	0.502E-06
2.0	19	70	0.169E-06
2.1	20	72	0.902E-07
2.2	21	76	0.198E-06
2.3	20	72	0.128E-06
2.4	20	72	0.929E-07
2.5	25	88	0.150E-06
2.6	25	87	0.429E-07
2.7	25	91	0.624E-07
2.8	23	85	0.118E-06
2.9	28	99	0.125E-06
3.0	22	80	0.217E-06
3.1	18	69	0.300E-07
3.2	22	82	0.271E-06
3.3	23	86	0.766E-07
3.4	23	85	0.113E-06
3.5	18	69	0.630E-06
3.6	21	75	0.880E-06
3.7	28	100	0.498E-07
3.8	36	118	0.791E-06
3.9	26	92	0.162E-06
4.0	26	92	0.841E-07

PRINT FROM NELME
****

MINIMIZATION OF F(X)=X(1)**2+AINT(ABS(X(1)))+5*X(2)**2+AINT(X(2))
UNTIL F(X) < 1.E-7 FOR DIFFERENT SIZES OF THE INITIAL POLYHEDRON.
STARTING POINT IS (-2.0,-2.0).

SIZE OF THE INITIAL POLYHEDRON	NUMBER OF ITERATIONS REQUIRED	NUMBER OF EVALUATIONS REQUIRED	TESTQ OF NELME
4.1	24	85	0.374E-06
4.2	23	84	0.703E-07
4.3	24	86	0.521E-07
4.4	23	84	0.110E-06
4.5	22	80	0.796E-07
4.6	26	94	0.976E-07
4.7	18	67	0.735E-06
4.8	18	67	0.247E-05
4.9	26	94	0.230E-06
5.0	23	85	0.406E-07
5.1	20	74	0.634E-07
5.2	32	114	0.694E-07
5.3	24	87	0.187E-06
5.4	28	99	0.386E-07
5.5	28	102	0.806E-07
5.6	18	69	0.278E-06
5.7	27	92	0.255E-06
5.8	28	97	0.145E-05
5.9	19	72	0.545E-06
6.0	21	80	0.378E-07
6.1	20	76	0.178E-06
6.2	19	73	0.210E-06
6.3	24	87	0.409E-07
6.4	23	85	0.663E-07
6.5	30	105	0.107E-06
6.6	27	98	0.278E-06
6.7	21	80	0.670E-07
6.8	24	84	0.105E-04
6.9	22	83	0.934E-07
7.0	31	106	0.143E-06
7.1	20	77	0.554E-07
7.2	23	85	0.465E-06
7.3	22	83	0.197E-06
7.4	29	105	0.145E-06
7.5	25	92	0.189E-06
7.6	23	84	0.145E-06
7.7	36	122	0.177E-06
7.8	27	96	0.637E-07
7.9	25	93	0.820E-07
8.0	28	101	0.246E-06

1. INTRODUCTION

Z is a real-time monitor which makes it possible to run procedures in Pascal in parallel and provides facilities for interaction and synchronization between different procedures.
This work was done as a projekt in a graduate-course held at the department. The title of the course was:

Modern Languages
for
Process Computers

Responsibily for the course were

Leif Andersson
Hilding Elmqvist
Sven Erik Mattsson

Z is implemented on an LSI-11. The relevant software is

1. RT-11

2. Pascal-compiler:

   OMS1 PASCAL-1
   Version 1.1 for RT-11

   May 20, 1978
   Copyright 1978, Oregon Minicomputer Software, Inc.

2. USER'S MANUAL

When studying this chapter the reader is adviced to now and then look at the example in chapter 3.

A user, who wants to run different processes in parallel, writes the different processes as procedures in a superior Pascal program. In the procedures the user can call for the external procedures which are used for making interaction and sychronization between different processes. In the main body of the program an initialization is done. Two sections follow which are written in order to explain in more detail how the superior program shall be written.

2.1 Comments on the superior Pascal program

Comments are made on three parts of the program:

declarations of types
declarations of processes
main body

2.1.1 type declaration:

Some of the formal parameters which have to be used are of types which are not standard in Pascal. Therefore the user must declare these non-standard types. The following type declaration shall be done: (Observe what is written about the type semaphore).

```
type  procid=integer;
      unsignedint=0..65535;
      semaphore=(teletype,lineprinter);
      time = record
               min,tick:integer;
             end;
```

Comments

procid:
declared in the same way in Z. It is used in initproc.

semaphore:
may be any user defined ordered set of enumeration type, so the set above is only an example.

time:
must be declared as above, because time is declared in this way in Kernel.

A variable of type time tells how many minutes and ticks (tick=20 ms) have elapsed since the internal clock was started. Kernel has a variable "clock" of type time. clock.tick is incremented by one after each interupt from the internal clock. This occurs every tick (= 20 ms). Every time clock.tick is incremented it is also normalized, which means that if clock.tick is greater than 3000 then clock.min is incremented by 1 and clock.tick is decremented by 3000 (3000 tick = 3000*20 ms = 1 minute).

All integers are ranging between 0 and 32767 because an integer is represented by 16 bits. So the internal time is counting modulo 32768 minutes (= 546 hours = 22.7 days). Therefore a user must use variables of type time carefully.

The user does not need to normalize time because if any external procedure is called which has time as a parameter, the time will be normalized by Kernel. So the only problem with the variable time.tick is the limit 32767.

2.1.2 process declaration and initialization:

A procedure can serve as a process <u>type</u>. A process instance is created by a call to the standard procedure initproc followed by a call to the procedure itself. The first statement of the procedure body must be a call to the procedure ready.

Observe that there are no objections for a process to initiate another process.

Processes may communicate with each other using non-local variables. Mutual exclusion should be ensured by the programmer using semaphores and the procedures signal and wait.

From the procedures it is possible to call for the following external procedures:

```
initproc
ready
wait
signal
await
cause
sleep
clocktime
setpri
awake
```

Example:

```
initproc(proc,3,1000,PROCESS);
PROCESS(i,j);
```

2.1.3 main body:

The mainprogram is treated as a process with two exceptions:

1. A call for initkernel must be done as the first statement in main program.

2. When the mainprogram pass its end all executing stop.

2.2 How to use the different external procedures

initkernel(var freememory:unsignedint; kernelrequest, mainrequest:unsignedint);

This procedure must be called at the first line in the mainprogram and starts Z and main.

freememory:

returns available memoryspace (in bytes) for processes

kernelrequest:

memoryrequest for kernel(in bytes).In most cases 100+100*(number of processes) will be enough.

mainrequest:

memoryrequest for stack and dynamic heap by the user's main program.

Example: initkernel(free,1000,1000);

initproc(var ident:procid;prio:integer;memoryrequest:unsignedint; procedure process);

A process is initialized by first a call for initproc then a call of the corresponding procedure.

ident:

ident is the identifier of the process returned by Z. The processes get values 2,3 etc.

Prio:

Prio defines the priority. Observe that the implementation is such that low numbers correspond to high priorities.

memoryrequest:

memoryrequest in bytes for parameters, local variables, stack and heap for process

procedure process:

Included to ensure that process has only var parameters. (otherwise the compiler will protest)

```
Example:  initproc(proc,3,1000,PROCESS);
  PROCESS( var parameters );
```

Notice that this procedure must be called immediately after initproc.

<u>ready;</u>

Should be the first statement in every procedure that defines a process.

```
Example: PROCESS( var parameters );
         local variables;
         begin
         ready;
         .
         .
         end;
```

<u>wait(x:semaphore);</u>

Before using a common resource x, a call for wait should be done.

Example: wait(teletype);

<u>signal(x:semaphore);</u>

After a common resource x has been used the resource must be released so other processes can get access to the resource.

Example: signal(teletype);

<u>await(x:semaphore);</u>

A call for await will put calling process in delayed queue of semaphore x and will remove calling process from running state. A process delayed by procedure await will not reactivate unless procedure cause called by some other process releases the delayed procedure from the queue of delayed. Procedure await may be used when checking some external condition before proceeding execution. Busy waiting may be prevented this way, since another process reaches running state.

Example: await(teletype);

<u>cause(x:semaphore);</u>

A call for cause will transfer all processes of the delayed queue of semaphore x to the semaphore queue of x. When cause has been called the transferred processes will reach running state after a sufficient number of calls for signal.

Example: cause(teletype);

<u>sleep(var x:time);</u>

A call for sleep will delay the process until time x.

Example: sleep(time1);

<u>clocktime(var x:time);</u>

Returns absolute time in x.

Example: clocktime(time1);

<u>setpri(ident:procid; prio:integer);</u>

Sets the priority of process with identifier ident to prio. Observe that this command can be used anywhere in the user program provided the process has been initialized.

Example: setpri(proc1, 5);

Kernel

```
program kernel;
{ A set of procedures for running concurrent processes in Pascal}

{$T-}
{kerneldata}

const idleprocsize=100;
const kernsize=100; {This includes kernel global variables+
stack area for idleproc}

type processref=^processrec;
type semref=^semaphore;
type procid=integer;
type unsignedint=0..65535;

type time=record
min,tick:integer
end;

type processrec=record
succ:processref;
pred:processref;
child:processref;
nextproc:processref;
stacktop:integer;
identifier:procid;
priority:integer;
starttime:time
end;

type semaphore=record
succ:semref;
number:integer;
flag:integer;
queue:processref;
delayed:processref
end;

var run,rdy,clk,all,p1,p2,p3:processref;
sem,s1:semref;
procsize,allocstart,size:unsignedint;
kernsp,kernheap,sp,parentsp,childsp:unsignedint;
procnr:procid;
qtype,i:integer;
clock:time;
a,b:boolean;

{$C
 ; KERNEL DATA ALLOCATED DIRECTLY IN THE PROGRAM CODE
 ; ACCESSIBLE ONLY FROM MACRO CODE WITHOUT USING R5
PSUSR:       .BLKW
PCUSR:       .BLKW
NPARAM:      .BLKW
PCMON:       .BLKW
R0SAV:       .BLKW
```

```
R5USR:       .BLKW

USERR5:      .BLKW
KERNR5:      .BLKW

PCSAVE:      .BLKW
SAVEKORE:    .BLKW
VAREND:      .BLKW
}

{-----------------------------------------------------------------------
 Internal procedures:
-----------------------------------------------------------------------}

procedure enterkernel;
{Called when a process enters the kernel via a procedure call to
 simulate an interrupt by insertion of the PSW before the PC value
 on the local stack}
begin
{$C
        MOV     (%6)+,%0        ; POP STACK TO RO
        MTPS    #^0340          ; TURN OFF INTERRUPTS
        MOV     (%6),-(%6)      ; PUSH PC
        CLR     2(%6)           ; USER PS VALUE
        MOV     %0,-(%6)        ; RESTORE STACK FOR RETURN OP.
}
end;
{-----------------------------------------------------------------------}
procedure savestatus(nparambytes:integer);
{Called when a process enters the kernel to save the status of the
 process on the local stack and switch to kernel mode.
 nparambytes is number of parameters of the kernel entry call in bytes}
begin
{$C
        MOV     %0,R0SAV        ; SAVE R0
        MOV     %5,R5USR        ; SWITCH R5
        MOV     KERNR5,%5
        MOV     (%6)+,PCMON     ; POP 4 ELEMENTS FROM LOCAL STACK
        MOV     (%6)+,NPARAM
        MOV     (%6)+,PCUSR
        MOV     (%6)+,PSUSR
        MOV     KERNSP(%5),%0   ; R0 IS NOW KERNEL STACK
        ADD     NPARAM,%6       ; R6 POINTS TO CELL BEFORE PARAMETERS
        MOV     %6,SP(%5)       ; SAVE THIS VALUE FOR LATER USE
        TST     NPARAM          ; SKIP LOOP IF NO PARAMETERS
        BEQ     LOS1
LOS0:   MOV     -(%6),-(%0)     ; MOVE PARAMETERS TO KERNEL STACK
        SUB     #2,NPARAM       ; 2 BYTES
        BNE     LOS0
LOS1:   MOV     SP(%5),%6       ; RESET R6
        MOV     PSUSR,-(%6)     ; STORE PROCESS RESTART INFO: FIRST F
        MOV     PCUSR,-(%6)     ; RESTART ADDRESS
        MOV     R0SAV,-(%6)     ; REGISTERS R0 - R5 + $KORE
        MOV     %1,-(%6)%
        MOV     %2,-(%6)%
```

```
        MOV     %3,-(%6)%
        MOV     %4,-(%6)%
        MOV     R5USR,-(%6)     ; LOCAL VALUE STORED ABOVE
        MOV     $KORE,-(%6)
        MOV     %6,SP(%5)       ; SAVE PROCESS STACK POINTER IN SP
        MOV     PCUSR,-(%0)     ; MAKE KERNEL STACK LOOK LIKE LOCAL S
        CLR     -(%0)           ; THIS AND NEXT IS USED BY RETURN COD
        MOV     PCMON,-(%0)
        MOV     %0,%6           ; SWITCH TO KERNEL STACK
        MOV     KERNHEAP(%5),$KORE ; SWITCH TO KERNEL HEAP
}
run^.stacktop:=sp
end;
{-----------------------------------------------------------------------}
procedure swkernel;
begin
{$C
        MTPS    #^0340          ; TURN OFF INTERRUPTS
        MOV     %5,USERR5       ; SAVE PROCESS R5
        MOV     KERNR5,%5       ; SWITCH TO KERNEL R5
}
end;
{-----------------------------------------------------------------------}
procedure swuser;
{To be called after swkernel}
begin
{$C
        MOV     USERR5,%5       ; RESTORE USER R5
        MTPS    #0              ; TURN ON INTERRUPTS
}
end;
{-----------------------------------------------------------------------}
procedure startrun;
{Exits from kernel by starting process pointed out by run}
begin
sp:=run^.stacktop;
{$C
        MOV     $KORE,KERNHEAP(%5)   ; SAVE KERNEL HEAPTOP
        MOV     SP(%5),%6    ; PROCESS SP
        MOV     (%6)+,$KORE  ; RESTORE PROCESS STATUS
        MOV     (%6)+,%5
        MOV     (%6)+,%4
        MOV     (%6)+,%3
        MOV     (%6)+,%2
        MOV     (%6)+,%1
        MOV     (%6)+,%0
        RTI                     ; START PROCESS
}
end;
{-----------------------------------------------------------------------}
procedure normalize(var x:time);
{Normalizes processtime x}
begin
while x.tick >= 3000 do
  begin
```

```
  x.min:=x.min+1;
  x.tick:=x.tick-3000
  end
end;
{----------------------------------------------------------------------}
procedure put(p;q:processref);
{Inserts processvariable p before q in q's list}
begin
p^.succ:=q;
p^.pred:=q^.pred;
q^.pred^.succ:=p;
q^.pred:=p
end;
{----------------------------------------------------------------------}
procedure remove(p:processref; var q:processref);
{Removes processvariable p from it's list and sets q:=p}
begin
q:=p;
p^.pred^.succ:=p^.succ;
p^.succ^.pred:=p^.pred;
end;
{----------------------------------------------------------------------}
procedure putpriority(p;q:processref);
{Inserts processvariable p in queue q  according to priority}
begin
p1:=q^.succ;
while (p1 <> q) and (p^.priority >= p1^.priority) do p1:=p1^.succ;
put(p;p1)
end;
{----------------------------------------------------------------------}
procedure clkstore(var p:processref);
{Stores p in clk-queue if starttime > current time. In this case
p:=nil, otherwise p is not changed}
begin
if (p^.starttime.min > clock.min) or (p^.starttime.min = clock.min)
and (p^.starttime.tick > clock.tick) then
  begin
  p1:=clk^.succ;
  {Put process in clk-queue according to starttime}
  while (p1 <> clk) and ((p1^.starttime.min < p^.starttime.min) or
  (p1^.starttime.min = p^.starttime.min) and
  (p1^.starttime.tick < p^.starttime.tick)) do p1:=p1^.succ;
  put(p;p1);
  p:=nil
  end
end;
{----------------------------------------------------------------------}
procedure semqueue(nr:integer; var s:semref);
{Returns a reference to the semaphore variable with number nr in
 the semaphore list. If nr non-existent a new entry in the list
 is created}
begin
s:=sem;
b:=true;
while (s <> nil) and b do
```

```
if s^.number = nr then b:=false
else s:=s^.succ;
if b then^
  begin {Create new semaphore}
  new(s);
  s^.succ:=sem;
  sem:=s;
  sem^.number:=nr;
  sem^.flag:=1;
  new(sem^.queue);
  sem^.queue^.succ:=sem^.queue;
  sem^.queue^.pred:=sem^.queue;
  sem^.queue^.identifier:=0;
  new(sem^.delayed);
  sem^.delayed^.succ:=sem^.delayed;
  sem^.delayed^.pred:=sem^.delayed;
  sem^.delayed^.identifier:=0
  end^
end;
{----------------------------------------------------------------------}
procedure schedule;
{Decides which process to be started by looking at run and the first
 element in rdy-queue and starts it}
begin
if rdy^.succ <> rdy then
  begin {rdy-queue not empty}
  if run <> nil then
    begin {run-queue not empty}
    if rdy^.succ^.priority <= run^.priority then
      begin {switch process}
      remove(rdy^.succ,p2);
      putpriority(run,rdy);
      run:=p2;
      end
    end
  else
    begin {run-queue empty, then rdy-queue cannot be empty}
    remove(rdy^.succ,run);
    end
  end;
startrun
end;
{----------------------------------------------------------------------}
procedure clkint;
{Clock interrupt routine}
begin
savestatus(0);
clock.tick:=clock.tick+1;
normalize(clock);
a:=true;
{Move all processes in clk-queue with starttime <= clock to rdy-queue}
while (clk^.succ <> clk) and a and (clk^.succ^.starttime.min >= 0) do
if (clk^.succ^.starttime.min < clock.min) or
(clk^.succ^.starttime.min = clock.min) and
(clk^.succ^.starttime.tick <= clock.tick) then
```

```
  begin
  remove(clk^.succ,p2);
  putpriority(p2,rdy)
  end
else a:=false;
schedule
end;
{-----------------------------------------------------------------------}
procedure exitproc;
begin
swkernel;
run:=nil;
schedule
end;
{-----------------------------------------------------------------------}
procedure semerror;
{Called if semaphore reservation error detected in wait}
begin
writeln(' Process:',run^.identifier,' terminated: wait requests in',
' wrong order');
run:=nil;
schedule
end;
{-----------------------------------------------------------------------}
procedure memoryerror;
{Called if memory reservation error detected}
begin
writeln('mmmmmmmmmmmmmm')
end;
{-----------------------------------------------------------------------}
procedure idleproc;
{Idle process - always last in rdy-queue or running}
begin
repeat
until false
end;

{-----------------------------------------------------------------------
 User interface to kernel
 ----------------------------------------------------------------------}

{$E+}
procedure initkernel(var freememory:unsignedint;kernelrequest,
                                    mainrequest:unsignedint);
begin
{$C
    MOV   (%6),PCSAVE       ;SAVE PC FOR MAIN
    MOV   %5,USERR5
    MOV   $KORE,%5
    MOV   %5,KERNR5
    MOV   %6,KERNSP(%5)
    MOV   %6,%0
    SUB   %5,%0
    SUB   #KERNSIZE,%0
    MOV   %0,SIZE(%5)
```

```
}
procsize:=kernelrequest;
size:=size-procsize;
allocstart:=kernsp-procsize;
kernheap:=allocstart+2;
{$C
    MOV  KERNHEAP(%5),$KORE
}
run:=nil;
sem:=nil;
new(rdy);
rdy^.succ:=rdy;rdy^.pred:=rdy;
new(clk);
clk^.succ:=clk;clk^.pred:=clk;
{start idleproc}
{$C
    MOV ALLOCSTART(%5),%0
    CLR  (%0)
    MOV  #IDLEPROC,-(%0)
    SUB  #10.,%0
    MOV  USERR5,-(%0)%
    MOV  ALLOCSTART(%5),%1
    SUB  #IDLEPROCSIZE,%1
    ADD  #2,%1
    MOV  %1,-(%0)
}
new(p2);
with p2^ do begin
  stacktop:=allocstart-16;
  allocstart:=allocstart-idleprocsize;
  identifier:=1;
  priority:=maxint;
  starttime.min:=0;
  starttime.tick:=0
end;
putpriority(p2,rdy);
{stack and heap for main}
procsize:=mainrequest;        {tester}
{$C
    MOV  ALLOCSTART(%5),%0
    CLR  (%0)
    MOV  PCSAVE,-(%0)
    SUB  #10.,%0
    MOV  USERR5,-(%0)
    MOV  ALLOCSTART(%5),%1
    SUB  PROCSIZE(%5),%1%
    ADD  #2,%1
    MOV  %1,-(%0)
}
size:=size-procsize-idleprocsize;
freememory:=size;
new(p3);
with p3^ do begin
  stacktop:=allocstart-16;
  allocstart:=allocstart-procsize;
```

```
  identifier:=2;
  procnr:=2;
  priority:=maxint-1;
  starttime.min:=0;
  starttime.tick:=0
end;
all^.nextproc:=p3;
putpriority(p3,rdy);
clock.min:=0;
clock.tick:=0;
{$C
    MTPS   #^0340
    MOV    #^0340,@#^0102
    MOV    #CLKINT,@#^0100
}
schedule
end;
{-------------------------------------------------------------}
procedure initproc(var ident:procid;prio:integer;
                   memoryrequest:unsignedint;procedure p);
{memoryrequest in bytes}
begin
enterkernel;
savestatus(10);
if memoryrequest > size
then memoryerror
else begin
     procsize:=memoryrequest;
     size:=size-procsize;
     sp:=run^.stacktop;
     {$C
         MOV   ALLOCSTART(%5),%1    ;DEFINE LOCAL PROCESS STACK
         MOV   SP(%5),%0
         ADD   #18.,%0%
         MOV   %0,(%1)             ;STACK VAREND
         MOV   ALLOCSTART(%5),%0
         SUB   PROCSIZE(%5),%0%
         ADD   #2,%0
         MOV   %0,-(%1)             ;STACK SAVEKORE
     }
     new(p2);
     with p2^ do begin
       stacktop:=allocstart-2;
       allocstart:=allocstart-procsize;
       procnr:=procnr+1;
       identifier:=procnr;
       ident:=procnr;
       priority:=prio
     end;
     run^.child:=p2;
     p1:=all^.nextproc;
     for i:=1 to procnr-3 do p1:=p1^.nextproc;
     p1^.nextproc:=p2
     end;
schedule
```

```
end;
{-----------------------------------------------------------------------}
procedure ready;
begin
enterkernel;
savestatus(0);
childsp:=run^.child^.stacktop;
parentsp:=run^.stacktop;
{$C
    MOV  CHILDSP(%5),%1                    ;1
    MOV  (%1)+,SAVEKORE
    MOV  (%1)+,VAREND
    MOV  PARENTSP(%5),%2
    ADD  #18.,%2
    MOV  VAREND,%0                         ;2
 1$:CMP  -(%0),USERR5
    BHIS 1$%
    MOV  (%0),PCSAVE
    MOV  #EXITPROC,(%0)
    MOV  VAREND,%0                         ;3
 2$:MOV  -(%0),-(%1)
    CMP  %0,%2
    BNE  2$
    MOV  VAREND,%0                         ;4
    MOV  #18.,%3%
 3$:MOV  -(%2),-(%0)
    SUB  #2,%3
    BNE  3$
    MOV  %0,PARENTSP(%5)
    CLR  -(%1)                             ;5
    MOV  14.(%0),-(%1)                     ;STACK SAVEPC FOR CHILD
    MOV  PCSAVE,14.(%0)          ;STACK PCSAVE FOR PARENT
    SUB  #10.,%1
    MOV  USERR5,-(%1)
    MOV  SAVEKORE,-(%1)
    MOV  %1,CHILDSP(%5)
}
run^.stacktop:=parentsp;
run^.child^.stacktop:=childsp;
run^.child^.starttime:=run^.starttime;
putpriority(run^.child,rdy);
schedule
end;
{-----------------------------------------------------------------------}
procedure wait(nr:integer);
{Performs the P operation on semaphore with value nr}
begin
enterkernel;
savestatus(2);
{Check for semaphore reservation error}
s1:=sem;
while s1 <> nil do
if (s1^.number > nr) and (s1^.queue^.identifier = run^.identifier)
then semerror else s1:=s1^.succ;
semqueue(nr,s1);
```

```
if s1^.flag = 1 then
  begin
  s1^.flag:=0;
  s1^.queue^.identifier:=run^.identifier
  end
else
  begin
  putpriority(run,s1^.queue);
  run:=nil
  end;
schedule
end;
{------------------------------------------------------------------------}
procedure signal(nr:integer);
{Performs the V operation on semaphore with value nr}
begin
enterkernel;
savestatus(2);
semqueue(nr,s1);
if s1^.flag = 0 then
  begin
  s1^.queue^.identifier:=0;
  if s1^.queue^.succ = s1^.queue then s1^.flag:=1
  else
    begin {Release a process from semaphore queue}
    remove(s1^.queue^.succ,p2);
    putpriority(p2,rdy);
    s1^.queue^.identifier:=p2^.identifier
    end
  end;
schedule
end;
{------------------------------------------------------------------------}
procedure await(nr:integer);
begin
enterkernel;
savestatus(2);
s1:=sem;
while s1<>nil do
if (s1^.number>nr) and
   (s1^.queue^.identifier=run^.identifier)
   then semerror else s1:=s1^.succ;
semqueue(nr,s1);
putpriority(run,s1^.delayed);
if s1^.flag=0 then begin
  s1^.queue^.identifier:=0;
  if s1^.queue^.succ=s1^.queue then s1^.flag:=1
  else begin{Release a process from semaphore queue}
    remove(s1^.queue^.succ,p2);
    putpriority(p2,rdy);
    s1^.queue^.identifier:=p2^.identifier;
  end;{else}
end;
run:=nil;
schedule;
```

```
end;
{------------------------------------------------------------------------}
procedure cause(nr:integer);
begin
enterkernel;
savestatus(2);
while s1<>nil do
  if (s1^.number>nr) and
     (s1^.queue^.identifier=run^.identifier)
     then semerror else s1:=s1^.succ;
semqueue(nr,s1);
while s1^.delayed^.succ<>s1^.delayed do begin
  remove(s1^.delayed^.succ,p2);
  putpriority(p2,s1^.queue);
end;{while}
schedule;
end;
{------------------------------------------------------------------------}
procedure sleep(var x:time);
{Removes calling process from running state for a restart at
 absolute time x}
begin
enterkernel;
savestatus(2);
normalize(x);
run^.starttime:=x;
{if 1 < ident < procnr+1 }
clkstore(run);
schedule
end;
{------------------------------------------------------------------------}
procedure clocktime(var x:time);
{Returns current absolute time in x}
begin
swkernel;
x:=clock;
swuser
end;
{------------------------------------------------------------------------}
procedure setpri(ident:procid;prio:integer);
{Sets the priority of process with name procname to prio}
begin
enterkernel;
savestatus(4);
p1:=all^.nextproc;
while (p1^.identifier < ident) do p1:=p1^.nextproc;
p1^.priority:=prio;
schedule
end;
{------------------------------------------------------------------------}
procedure awake(ident:procid;x:time);
{Awakes sleeping process (in clk-queue) with name procname for start
 at time x}
begin
enterkernel;
```

```
savestatus(6);
{Look for process in clk-queue}
a:=true;
p2:=clk^.succ;
while (p2 <> clk) and a do
if p2^.identifier = ident then a:=false
else p2:=p2^.succ;
if not a then
  begin {Have found process: remove and re-schedule}
  remove(p2,p2);
  p2^.starttime:=x;
  if x.min >= 0 then
    begin
    clkstore(p2);
    if p2 <> nil then putpriority(p2,rdy);
    end
  else put(p2,clk)
  end;
schedule
end;
{------------------------------------------------------------------------}
begin
end.
```

APPENDIX 2: DDCPAC

Programlistings

DDCPAC

```
{---------------------------------------------------------------------}
{---------------------------------------------------------------------}
program DDCPAC;

type unsignedint=0..65535;
     procid=integer;
     time=record
       min,tick:integer
     end;
     semaphore=(s0,s1,s2,s3,s4,s5,s6,s7,s8,s9);

function adin(chan:integer):real; external;
procedure daout(chan:integer; value:real); external;

procedure initkernel(var freememory:unsignedint;
                     kernelrequest,mainrequest:unsignedint); external;
procedure initproc(var ident:procid; prio:integer;
                   memoryrequest:unsignedint; procedure p); external;
procedure ready; external;
procedure wait(nr:semaphore); external;
procedure signal(nr:semaphore); external;
procedure await(nr:semaphore); external;
procedure cause(nr:semaphore); external;
procedure sleep(var x:time); external;
procedure clocktime(var x:time); external;
procedure setpri(ident:procid; prio:integer); external;
procedure awake(ident:procid; x:time); external;

const AD='AD        ';
      DA='DA        ';
      tickspersecond=50.0;
      blanks='          ';
      nodelistsize=20;

type nametype=array[1..10] of char;
     address=record
       name:nametype;
       number:integer
     end;{address}
     nodetype=(innode,PIDnode,outnode);
     paramtype=record
       case tag:nodetype of
         innode:(insig:address;
                 scale,filter:real);
         PIDnode:(PIDin,PIDref,PIDout:address;
                  k,ti,td,alfa,beta,limit:real);
         outnode:(insig1,insig2,insig3,insig4,output:address;
                  scal1,scal2,scal3,scal4,level:real)
       end;{paramtype}
     statetype=record
       filterstate,yold,ipart,dpart:real;
     end;{statetype}
     ddcnodetype=record
       fwd,bwd:integer;
       name:nametype;
```

```
        priority:integer;
        period,counter:integer;
        outvalue:real;
        parampart:paramtype;
        state:statetype
      end;{ddcnodetype}
      commandtype=array[1..6] of char;
      opindex=(openx,showx,linkx,deletex,lastopindex);
      variableindex=(periodx,insigx,scalex,filterx,
                     PIDinx,PIDrefx,PIDoutx,kx,tix,
                     tdx,limitx,insig1x,insig2x,
                     insig3x,insig4x,outputx,scal1x,
                     scal2x,scal3x,scal4x,
                     levelx,lastvarindex);
      errors=(fewarg,toomanyarg,illname,nospace,
              notopen,blankaddr,nonode,referr,notin,
              noname,prierr,noop,tagerr);
      nodemonitortype=record
        nodelist:array[1..nodelistsize] of ddcnodetype;
        i,regindex:integer;
        entrygate:semaphore;
      end;{nodemonitortype}

var regulnode,opcomnode:ddcnodetype;
    name:nametype;
    clock:time;
    op:array[opindex] of commandtype;
    vars:array[variableindex] of commandtype;
    opx:opindex;
    varindex:variableindex;
    command:commandtype;
    opened,anyfound,notfound,notspace:boolean;
    nodemonitor:nodemonitortype;
    allocationarea:unsignedint;
    identity:integer;
    opqueue,opcomqueue,TTYqueue:semaphore;
{---------------------------------------------------------------------}
procedure setsemaphores;
{Initializes all semaphores}
begin
  nodemonitor.entrygate:=s0;
  opqueue:=s1;
  opcomqueue:=s2;
  TTYqueue:=s3;
end;
{---------------------------------------------------------------------}
procedure regputgetnode;
{Copies a node of the nodelist into a regulator's copy or copies back
statevalues from regulator's copy into nodelist}
var done:boolean;
begin
  wait(nodemonitor.entrygate);
  with nodemonitor do begin
    if regindex<>1 then begin
      with nodelist[regindex] do begin
```

```
          outvalue:=regulnode.outvalue;
          state:=regulnode.state
        end {with}
      end;
      done:=false; anyfound:=false;
        repeat
        regindex:=nodelist[regindex].fwd;
        if regindex=1 then done:=true
        else begin
          with nodelist[regindex] do begin
            if period>0 then begin
              counter:=counter-1;
              if counter<=0 then begin
                counter:=period;
                anyfound:=true;
                regulnode:=nodelist[regindex];
                done:=true
              end
            end
          end {with}
        end
      until done;
      {continue(opqueue)}
      cause(opqueue);
      signal(opqueue);
    end;{with}
  signal(nodemonitor.entrygate)
  end;{regputgetnode}
{----------------------------------------------------------------------}
procedure getoutvalue(var number:integer;
                      var outvalue:real);
begin
  outvalue:=nodemonitor.nodelist[number].outvalue
end;{getoutvalue}
{----------------------------------------------------------------------}
procedure lookup(var name:nametype; var ptr:integer);
{Scans nodelist for node name}
begin
  with nodemonitor do begin
    nodelist[1].name:=name;
    ptr:=1;
    repeat
      ptr:=nodelist[ptr].fwd
    until nodelist[ptr].name=name;
  end {with}
end;{lookup}
{----------------------------------------------------------------------}
procedure opgetnode(var node:ddcnodetype;
                    var notfound,notspace:boolean);
{Checks nodelist for a node called (node.name). When found node gives
a reference}
var ptr:integer;
begin
  wait(nodemonitor.entrygate);
  lookup(node.name,ptr);
```

```
  if ptr<>1 then begin
    notfound:=false;
    node:=nodemonitor.nodelist[ptr] end
  else begin
    notfound:=true;
    notspace:=nodemonitor.nodelist[2].fwd=2
  end;
  signal(nodemonitor.entrygate);
end;{opgetnode}
{---------------------------------------------------------------------}
procedure deletenode(this:integer);
{Deletes the node (this) from the nodelist}
begin
  wait(nodemonitor.entrygate);
  with nodemonitor do
    with nodelist[this] do begin
      nodelist[fwd].bwd:=bwd;
      nodelist[bwd].fwd:=fwd;
      bwd:=2;
      fwd:=nodelist[2].fwd;
      nodelist[2].fwd:=this;
      nodelist[fwd].bwd:=this
    end;{with}
  signal(nodemonitor.entrygate);
end;{deletenode}
{---------------------------------------------------------------------}
procedure linknode;
var this,ptr:integer;
{Links a node to the nodelist}
begin
  wait(nodemonitor.entrygate);
  lookup(opcomnode.name,this);
  with nodemonitor do begin
    if this<>1 then begin
        with nodelist[this] do begin
          period:=opcomnode.period;
          parampart:=opcomnode.parampart
      end {with} end
    else begin {a new node}
      this:=nodelist[2].fwd;
      ptr:=nodelist[this].fwd;
      nodelist[ptr].bwd:=2;
      nodelist[2].fwd:=ptr;
      nodelist[this]:=opcomnode;
      ptr:=1;
      with nodelist[this] do begin
        repeat
          ptr:=nodelist[ptr].fwd
        until priority<nodelist[ptr].priority;
        fwd:=ptr;
        bwd:=nodelist[ptr].bwd;
        nodelist[ptr].bwd:=this;
        nodelist[bwd].fwd:=this
      end {with}
  end {with}
```

```
  end;
signal(nodemonitor.entrygate);
end;{linknode}
{----------------------------------------------------------------------}
procedure checkname(var name:nametype;
                    var ptr,pri:integer);
{Returns a reference to a node and its priority}
begin
  wait(nodemonitor.entrygate);
  lookup(name,ptr);
  if ptr<>1 then pri:=nodemonitor.nodelist[ptr].priority;
  signal(nodemonitor.entrygate)
end;{checkname}
{----------------------------------------------------------------------}
procedure initnodemonitor;
{Initializes nodelist}
var k:integer;
begin
  wait(nodemonitor.entrygate);
  with nodemonitor do begin
    regindex:=1;
    with nodelist[1] do begin
      fwd:=1; bwd:=1;
      priority:=32767;
    end;{with}
    for k:=2 to nodelistsize do begin
      with nodelist[k] do begin
        fwd:=k+1; bwd:=k-1
      end {with}
    end;{for}
    nodelist[2].bwd:=nodelistsize;
    nodelist[nodelistsize].fwd:=2;
  end;{with}
  signal(nodemonitor.entrygate);
end;{initnodemonitor}
{----------------------------------------------------------------------}
{----------------------------------------------------------------------}
procedure opcom;
label 999;

procedure error(err:errors);
{Makes print-outs of errors}
begin
  wait(TTYqueue);
  case err of
    fewarg:      writeln('missing arguments');
    toomanyarg:  writeln('too many arguments');
    illname:     writeln('illegal name');
    nospace:     writeln('the nodelist is full');
    notopen:     writeln('no node is open');
    blankaddr:   writeln('undefined signal');
    nonode:      writeln('the node didn''t exist');
    referr:      writeln('other nodes use this node');
    notin:       writeln('the signal has wrong ',
                         'direction');
```

```
    noname:     writeln('undefined signal');
    prierr:     writeln('the value is not ',
                        'available here');
    noop:       writeln('illegal operation');
    tagerr:     writeln('wrong nodetype')
  end;{case}
  signal(TTYqueue);
  goto 999
end;{error}
{...................................................................}
procedure writeaddress(addr:address);
begin
  wait(TTYqueue);
  with addr do begin
    if name=AD then writeln('AD ',number:2)
    else if name=DA then writeln('DA ',number:2)
      else writeln(name)
  end;{with}
  signal(TTYqueue);
end;{writeadress}
{...................................................................}
procedure readname;
{Reads names written by operator on TTY}
begin
  wait(TTYqueue);
  if eoln then error(fewarg);
  read(name);
  if not eoln then error(toomanyarg);
  if (name=AD) or (name=DA) or
    (name=blanks) then error(illname);
  signal(TTYqueue);
end;{readname}
{...................................................................}
procedure open;
{Initializes a node or makes a copy from the nodelist}
var done:boolean;
    nodestring:array[1..8] of char;
procedure initname(var addr:address);
begin
  addr.name:=blanks;
  addr.number:=0
end;{initname}
begin
  wait(opcomqueue);
  opened:=false;
  readname;
  opcomnode.name:=name;
  opgetnode(opcomnode,notfound,notspace);
  if notfound then begin
    if notspace then error(nospace)
    else begin
      with opcomnode do begin
        repeat
          write('* priority= ');
          readln(priority);
```

```
        until (priority>0) and (priority<32767);
        period:=0;
        counter:=1;
        outvalue:=0.0
      end;{with}
      with opcomnode.parampart do begin
        repeat
          done:=true;
          write('* nodetype= ');
          read(nodestring);
          if nodestring='INNODE  ' then
            tag:=innode
          else if nodestring='PIDNODE ' then
            tag:=PIDnode
            else if nodestring='OUTNODE ' then
              tag:=outnode
              else begin done:=false; readln end
        until done;
        case tag of
          innode: begin
                    initname(insig);
                    scale:=1.0;
                    filter:=1.0
                  end;
          PIDnode:begin
                    limit:=1.0;
                    k:=0.0;
                    ti:=0.0;
                    td:=0.0;
                    initname(PIDin);
                    initname(PIDref);
                    initname(PIDout)
                  end;
          outnode:begin
                    initname(insig1); scal1:=0.0;
                    initname(insig2); scal2:=0.0;
                    initname(insig3); scal3:=0.0;
                    initname(insig4); scal4:=0.0;
                    initname(output);
                    level:=0.0
                  end
        end {case}
      end;{with}
      with opcomnode.state do begin
        filterstate:=0.0;
        yold:=0.0;
        ipart:=0.0;
      end {with}
    end {else}
  end;{if}
  opened:=true;
  signal(opcomqueue);
end;{open}
{.....................................................................}
```

```
procedure show;
{Prints the contents of operator's nodecopy on TTY}
begin
  if not opened then error(notopen);
  with opcomnode do begin
    write('NODENAME: '); writeln(name);
    write('PERIOD   : '); writeln(period);
    write('PRIORITY: '); writeln(priority);
    write('OUTVALUE: '); writeln(outvalue)
  end;{with}
  with opcomnode.parampart do begin
    case tag of
      innode: begin
                 write('INSIG    : ');
                 writeaddress(insig);
                 write('FILTER   : ');
                 writeln(filter);
                 write('SCALE    : ');
                 writeln(scale);
              end;
      PIDnode:begin
                 write('PIDREF   : ');
                 writeaddress(PIDref);
                 write('PIDIN    : ');
                 writeaddress(PIDin);
                 write('PIDOUT   : ');
                 writeaddress(PIDout);
                 write('K        : ');
                 writeln(K);
                 write('TI       : ');
                 writeln(TI);
                 write('TD       : ');
                 writeln(TD);
                 write('LIMIT    : ');
                 writeln(limit);
              end;
      outnode:begin
                 write('INSIG1   : ');
                 writeaddress(insig1);
                 write('SCAL1    : ');
                 writeln(scal1);
                 write('INSIG2   : ');
                 writeaddress(insig2);
                 write('SCAL2    : ');
                 writeln(scal2);
                 write('INSIG3   : ');
                 writeaddress(insig3);
                 write('SCAL3    : ');
                 writeln(scal3);
                 write('INSIG4   : ');
                 writeaddress(insig4);
                 write('SCAL4    : ');
                 writeln(scal4);
                 write('OUTPUT   : ');
                 writeaddress(output);
```

```
                   write('LEVEL    : ');
                   writeln(level)
                 end;
      end;{case}
      with opcomnode.state do begin
        case tag of
          innode: begin
                    write('FILTERSTATE : ');
                    writeln(filterstate)
                  end;
          PIDnode:begin
                    write('YOLD       : ');
                    writeln(yold);
                    write('IPART      : ');
                    writeln(ipart);
                    write('DPART      : ');
                    writeln(dpart);
                  end;
        end;{case}
      end {with}
    end {with}
  end;{show}
{......................................................................}
procedure link;
{Administrates linkage of operator's nodecopy and hands over to
procedure linknode above}
var ts:real;
begin
  if not opened then error(notopen);
  with opcomnode.parampart do begin
    case tag of
      innode: if insig.name=blanks then
                error(blankaddr);
      PIDnode:begin
                if PIDref.name=blanks then
                  error(blankaddr);
                if PIDin.name=blanks then
                  error(blankaddr);
                ts:=opcomnode.period/tickspersecond;
                if ti>0.0 then alfa:=k*ts/ti
                else alfa:=0.0;
                if (ts>0.0) and (td>0.0) then
                  beta:=k*td/ts
                else beta:=0.0
              end;
      outnode:begin
                if (scal1<>0.0) and
                  (insig1.name=blanks) then
                  error(blankaddr);
                if (scal2<>0.0) and
                  (insig2.name=blanks) then
                  error(blankaddr);
                if (scal3<>0.0) and
                  (insig3.name=blanks) then
                  error(blankaddr);
```

```
                    if (scal4<>0.0) and
                      (insig4.name=blanks) then
                      error(blankaddr)
                end
        end {case}
      end;{with}
      linknode;
    end;{link}
    {..................................................................}
    procedure delete;
    {Administrates deletion of node and hands over to deletenode above}
    var this,ptr,result:integer;
        nodename:nametype;
    begin
      readname;
      nodename:=name;
      result:=0;
      lookup(nodename,this);
      if this=1 then result:=1
      else begin
        ptr:=nodemonitor.nodelist[this].fwd;
        while (ptr<>1) and (result<>2) do begin
          with nodemonitor.nodelist[ptr].parampart do begin
            if insig1.name=nodename then result:=2;
            if tag<>innode then begin
              if insig2.name=nodename then result:=2;
              if tag=outnode then begin
                if insig3.name=nodename then result:=2;
                if insig4.name=nodename then result:=2
              end
            end
          end;{with}
          ptr:=nodemonitor.nodelist[ptr].fwd
        end {while}
      end;
      if result=0 then begin
      { if this=nodemonitor.regindex then delay(opqueue);      }
        if this=nodemonitor.regindex then await(opqueue);
        deletenode(this);
      end;
      if result=1 then error(nonode);
      if result=2 then error(referr)
    end;{delete}
    {..................................................................}
    procedure readreal(var variable:real;
                           tag:nodetype);
    {Reads parameters etc. from TTY}
    var r:real;
    begin
      if opcomnode.parampart.tag<>tag then
        error(tagerr);
      if eoln then error(fewarg);
      read(r);
      if not eoln then error(toomanyarg);
      variable:=r
```

```
end;
{...............................................................}
procedure readaddr(var signal:address;
                       tag:nodetype;
                       outsignal:boolean);
{Reads address of signal-reference from TTY}
var nr,pri:integer;
begin
  if opcomnode.parampart.tag<>tag then
    error(tagerr);
  if eoln then error(fewarg);
  read(name);
  if name<>blanks then begin
    if outsignal <> (name=DA) then
      error(notin);
    if (name=DA) or (name=AD) then begin
      if eoln then error(fewarg);
      read(nr) end
    else begin
      checkname(name,nr,pri);
      if nr=1 then error(noname);
      if opcomnode.priority<=pri then error(prierr)
    end
  end;
  if not eoln then error(toomanyarg);
  signal.name:=name;
  signal.number:=nr
end;{readaddr}
{...............................................................}
procedure setvariable;
{Administrates setting af variables}
var i:integer;
begin
  with opcomnode.parampart do begin
    varindex:=periodx;
    vars[lastvarindex]:=command;
    while vars[varindex]<>command do
      varindex:=succ(varindex);
    if varindex=lastvarindex then error(noop);
    if not opened then error(notopen);
    case varindex of
      periodx: begin
                 if eoln then error(fewarg);
                 read(i);
                 if not eoln then
                   error(toomanyarg);
                 opcomnode.period:=i
               end;
      insigx:  readaddr(insig,innode,false);
      scalex:  readreal(scale,innode);
      filterx: readreal(filter,innode);
      PIDinx:  readaddr(PIDin,PIDnode,false);
      PIDrefx: readaddr(PIDref,PIDnode,false);
      PIDoutx: readaddr(PIDout,PIDnode,true);
      kx:      readreal(k,PIDnode);
```

```
        tix:      readreal(ti,PIDnode);
        tdx:      readreal(td,PIDnode);
        limitx:   readreal(limit,PIDnode);
        insig1x:  readaddr(insig1,outnode,false);
        insig2x:  readaddr(insig2,outnode,false);
        insig3x:  readaddr(insig3,outnode,false);
        insig4x:  readaddr(insig4,outnode,false);
        outputx:  readaddr(output,outnode,true);
        scal1x:   readreal(scal1,outnode);
        scal2x:   readreal(scal2,outnode);
        scal3x:   readreal(scal3,outnode);
        scal4x:   readreal(scal4,outnode);
        levelx:   readreal(level,outnode);
        lastvarindex: error(noop)
      end {case}
    end {with}
  end;{setvariable}
  {.............................................................}
  procedure initialize;
  {Initializes opcom}
  begin
    op[openx]:='OPEN  ';
    op[showx]:='SHOW  ';
    op[linkx]:='LINK  ';
    op[deletex]:='DELETE';
    vars[periodx]:= 'PERIOD';
    vars[insigx]:=  'INSIG ';
    vars[scalex]:=  'SCALE ';
    vars[filterx]:= 'FILTER';
    vars[PIDinx]:=  'PIDIN ';
    vars[PIDrefx]:= 'PIDREF';
    vars[PIDoutx]:= 'PIDOUT';
    vars[kx]:=      'K     ';
    vars[tix]:=     'TI    ';
    vars[tdx]:=     'TD    ';
    vars[limitx]:=  'LIMIT ';
    vars[insig1x]:= 'INSIG1';
    vars[insig2x]:= 'INSIG2';
    vars[insig3x]:= 'INSIG3';
    vars[insig4x]:= 'INSIG4';
    vars[outputx]:= 'OUTPUT';
    vars[scal1x]:=  'SCAL1 ';
    vars[scal2x]:=  'SCAL2 ';
    vars[scal3x]:=  'SCAL3 ';
    vars[scal4x]:=  'SCAL4 ';
    vars[levelx]:=  'LEVEL ';
    opened:=false
  end;{initialize}
{...............................................................}
begin {opcom}
  ready;
  initialize;
  repeat
    write('>');
    read(command);
```

```
      op[lastopindex]:=command;
      opx:=openx;
      while op[opx]<>command do
        opx:=succ(opx);
      case opx of
        openx       : open;
        showx       : show;
        linkx       : link;
        deletex     : delete;
        lastopindex: setvariable
      end;{case}
999:readln
  until false;
end;{opcom}
{---------------------------------------------------------------------}
{---------------------------------------------------------------------}
function ulim(u,lolim,hilim:real):real;
{Limits signal u}
begin
  ulim:=u;
  if u<lolim then ulim:= lolim;
  if u>hilim then ulim:= hilim
end;{ulim}
{---------------------------------------------------------------------}
function nodevalue(var addr:address):real;
{Fetches outvalues from AD-converter or node}
var value:real;
begin
  if addr.name=AD then value:=adin(addr.number)
  else getoutvalue(addr.number,value);
  nodevalue:=value;
end;{nodevalue}
{---------------------------------------------------------------------}
procedure inreg;
{Filters a signal}
var value,out:real;
begin
  with regulnode.parampart,regulnode.state do begin
    value:=scale*nodevalue(insig);
    value:=(1.0-filter)*filterstate+filter*value;
    filterstate:=value
  end;{with}
  regulnode.outvalue:=value
end;{inreg}
{---------------------------------------------------------------------}
Procedure PIDreg;
{Performs calculations for PID-control}
var e,yr,y,u:real;
begin
  with regulnode.parampart,regulnode.state do begin
    yr:=nodevalue(PIDref);
    y:=nodevalue(PIDin);
    e:=yr-y;
    ipart:=ipart+alfa*e;
    dpart:=beta*(yold-y);
```

```
      u:=k*e+ipart+dpart;
      u:=ulim(u,-limit,limit);
      yold:=y;
      if ti>0.0 then ipart:=u-dpart-k*e;
      {anti-reset-windup}
      if PIDout.name=DA then
        daout(PIDout.number,u);
      regulnode.outvalue:=u
    end {with}
  end;{PIDreg}
  {----------------------------------------------------------------------}
  Procedure outreg;
  {Calculates a weighted sum and performs limitation, when
  DA-conversion is expected}
  var out:real;
  begin
    with regulnode.parampart do begin
      out:=level;
      if scal1<>0.0 then
        out:=out+scal1*nodevalue(insig1);
      if scal2<>0.0 then
        out:=out+scal2*nodevalue(insig2);
      if scal3<>0.0 then
        out:=out+scal3*nodevalue(insig3);
      if scal4<>0.0 then
        out:=out+scal4*nodevalue(insig4);
      regulnode.outvalue:=out;
      if output.name=DA then begin
        out:=ulim(out,-1.0,1.0);
        daout(output.number,out)
      end
    end {with}
  end;{outreg}
  {----------------------------------------------------------------------}
  procedure regulator;
  {Administrates regulating facilities and keep record of time}
  var nextclock:time;
  begin
    ready;
    repeat
      nextclock.tick:=clock.tick+1;
      sleep(nextclock);
      clocktime(clock);
      regputgetnode;
      while anyfound do begin
        case regulnode.parampart.tag of
          innode:  inreg;
          PIDnode: PIDreg;
          outnode: outreg
        end;{case}
        regputgetnode;
      end {while}
    until false;
  end;{regulator}
  {----------------------------------------------------------------------}
```

```
{---------------------------------------------------------------------------}
begin {main}
  setsemaphores;
  initkernel(allocationarea,1000,1000);
  writeln('allocationarea=',allocationarea);
  initnodemonitor;
  initproc(identity,8,1000,regulator);
  writeln('#',identity);
  regulator;
  initproc(identity,10,1000,opcom);
  writeln('#',identity);
  opcom;
  writeln('#',identity);
  repeat until false;
end.
{---------------------------------------------------------------------------}
{---------------------------------------------------------------------------}
{---------------------------------------------------------------------------}
```

enterkernel
savestatus
swkernel
swuser
startrun
exitproc

The two first procedures in the list are commented below.

enterkernel:

The programcontrol can switch from the users code to the kernel's code in two ways, by procedure call (e.g. wait(x)) and by interrupt (for the time being the only interrupt is the clock-interrupt). See figure on next page. However, in the other direction (from the kernel's code to users code) the programcontrol can switch in only one way, by

RTI

(See startrun). Therefore the stack must be modified in the case of procedure call. This modification is done by enterkernel.

savestatus:

When PC goes into the savestatus it still uses the local stack (the stack belonging to the process which are calling a kernelprocess or being interrupt). On the top of this stack some information is kept which is moved to the kernelstack by savestatus. After that the values of the registers and $KORE are saved on the local stack. For details see the code.

9.2.2 MISCELLANEOUS PROCEDURES

The miscellaneous procedures are

The effect of interrupt

before interrupt

```
        --------------
PS      !    0000     !
        --------------                 stack

        --------------             ! / / / / / / !
R7      !     PC      !            !/ / / / / / /!
        --------------             ! / / / / / / !
                                   !/ / / / / / /!
        --------------             ! / / / / / / !
R6      !     A       !  --------> !/ / / / / / /!
        --------------             -------------
```

after interrupt

```
        --------------
PS      !    0340     !
        --------------                 stack

        --------------             !/ / / / / / /!
R7      !   CLKINT    !            ! / / / / / / !
        --------------             !/ / / / / / /!
                                   ! / / / / / / !
        --------------             !/ / / / / / /!
R6      !    A + 4    !  ---\      ! / / / / / / !
        --------------       \     !    0000     !
                              \--> !     PC      !
                                   -------------
```

The effect of jump to subroutine

```
            JSR    PC,SBR
```

before

```
        --------------
R7      !     PC      !                stack
        --------------
                                   !/ / / / / / /!
        --------------             ! / / / / / / !
R6      !     B       !  --------> !/ / / / / / /!
        --------------             -------------
```

after

```
        --------------                  stack
R7      !     SBR     !
        --------------             !/ / / / / / /!
                                   ! / / / / / / !
        --------------             !/ / / / / / /!
R6      !    B + 2    !  ---\      ! / / / / / / !
        --------------       \---> !   PC + 2    !
                                   -------------
```

.

normalize
put
remove
putpriority
clkstore
semqueue
schedule
clkint
memoryerror
semerror
idleproc

The program-code is in these cases easy to understand.

10. APPLICATIONS --- DDC-PACKAGE

An available Concurrent-Pascal-version of a program-package designed for direct digital control(DDC) by Wieslander, Mattson et al was modified such that Concurrent-Pascal-features were exchanged by tools offered by the kernel.

The program is structured as two concurrently executing procedures (regulator and opcom) sharing a common database (nodelist).

```
|------------------------------------------------------------------|
| main  |-------|  |-----|  |-----|          |-----|               |
|   |-->| node  |->|     |->|     |-       ->|     |---|           |
|   |   | list  |  |     |  |     |   ...    |     |   |           |
|   | |-|  (1)  |<-| (2) |<-| (3) |<-       -| (n) |<-||           |
|   | | |-------|  |-----|  |-----|          |-----|  ||           |
|   | |-----------------------------------------------||           |
|   |------------------------------------------------|            |
|                                                                  |
|                                                                  |
|    |----------------------|    |--------------------|           |
|    |   proc. regulator    |    |    proc. opcom     |           |
|    |                      |    |                    |           |
|    |  (performs calcu-    |    |  (Operator's com-  |           |
|    |   lations on data-   |    |   munication with  |           |
|    |   base nodelist)     |    |   database)        |           |
|    |----------------------|    |--------------------|           |
|------------------------------------------------------------------|
```

Figure 10.1: The main program takes care of initialization and starts up the procedures opcom and regulator by kernel-procedure initproc and lets these procedures work concurrently. Opcom and regulator perform operations on a common database called nodelist(cf.figure above), which thus should be protected - when worked upon - by appropriate kernelprocedures.

<u>Main program</u>

The main program has the following outline:

```
begin
  setsemaphores;           {initializes semaphores}
  initkernel;              {            kernel}
  initnodemonitor;         {            database}
  initproc(regulator);     {starts regulator}
  regulator;
  initproc(opcom);         {starts operators commmuni-
  opcom;                    cation with database}
  repeat until false       {prevents main from terminating}
end.
```

Main is not allowed to terminate, since this stops the execution of the procedures regulator and opcom.

Nodelist

The database nodelist is organised as an array of records(nodes), each of which can be chosen as INNODE, PIDNODE or OUTNODE.

Innode contains address INSIG to a suitable unit of input like AD-converter or some other node. Furthermore there are scalefactor SCALE and filterfactor FILTER for scaling and first-order filtering resp. of input. The result is stored in outvalue and later in filterstate.

PIDnode contains addresses to input-units of commandsignal (PIDREF), process-output (PIDIN) and output-unit (PIDOUT) like AD(or DA)-converter or some other node. There is also space for all parameters (K,TI,TD) and states necessary for PID-control and a saturation-limit LIMIT for the saturated controller.

Finally outnode contains addresses to four different input units (INSIG1, INSIG2, INSIG3, INSIG4) with corresponding weighting factors (SCAL1, SCAL2, SCAL3, SCAL4) and a constant term LEVEL in order to calculate the expression u=level + scal1*x1 +....+scal4*x4.

Regulator

The procedure regulator performs calculations on nodelist as indicated above. Regulator has higher priority than opcom and will thus start executing immediately after a clockinterrupt unless it isn't still running after a previous activation. When running, regulator scans the nodelist, which is sorted by priority, checks variable counter, which tells, whether the calculations should be performed at this time instant. When so expected, regulator makes a copy(regulnode) of the node and calculations are done. Regulator then updates states of the node and proceeds until all nodes in nodelist has been examined. When finished opcom starts running.

Opcom

Opcom facilitates operator's communication with nodelist. Opcom works interactively with the operator. When expecting a new command opcom writes '>' on TTY. The operator can choose between commands OPEN, SHOW, LINK, DELETE and parametersettings PERIOD, INSIG, SCALE, FILTER, PIDIN, PIDREF, PIDOUT, K, TI, TD, LIMIT, INSIG1, INSIG2, INSIG3, INSIG4, OUTPUT, SCAL1, SCAL2, SCAL3, SCAL4 and LEVEL. Furthermore opcom takes care of error messages and other references to external units except for AD- and DA-conversion.

<u>Kernel-reference</u>

Those procedures of DDCPAC, which make use of the protection facilities offered by the kernel, consist of the equivalents of the Concurrent-Pascal-feature entryprocedure of monitors. In our program certain procedures have received protection by calling kernel-procedures 'wait' and 'signal' in the following manner.

```
    procedure alfa(r:real);
    begin
      wait(semaphore1);
           .
           .
           .
           .
      signal(semaphore1)
    end;
```

When this simple-most way of declaration is used, clarity is gained, fewer semaphore errors are likely to occur and program structure adheres to that of Concurrent-Pascal. When some other procedure calls 'wait(semaphore1)', while execution of alfa has passed 'wait' but not 'signal', it will have to make a break at least until alfa has passed 'signal'. When a calling procedure calls 'wait' of a busy semaphore, it will be put into the semaphore-queue by priority and execution will make a break until a sufficient number of 'signal' has been executed. Each 'signal' releases one of the waiting procedures in the semaphore-queue and transfers it to the ready-queue i.e. the the queue of processes ready for start(or restart) of execution.

Kernelprocedures 'await' and 'cause' do not occur pairwise in a procedure as 'wait' and 'signal' do. Procedure 'await' may be compared with the Concurrent-Pascal-feature 'delay', which is used when the calling procedure should make a break, until some condition, which is external of the calling procedure, has been fulfilled. Then a call 'cause' transfers the delayed procedure to the semaphore-queue from which it will be released to execute after a sufficient number of signals in the same manner as above. These features are not used exclusively to prevent procedures, which compete about access to a common resource, from bad interaction - as is the case about 'wait' and 'signal' - but more as means to synchronize demands of certain resources with their actual capability. This might be useful in order to prevent errors for example when trying to put characters into a full buffer or when deleting a node of which the regulator has got a copy and works upon in order to write back into the node without actually occupying the node with a semaphore-request in the mean-time.

In DDCPAC the following protection facilities have been developed using the kernelprocedures.

To protect from bad interaction due to competition about database:

Semaphore	Procedure	Comment
nodemonitor. entrygate	regputgetnode opgetnode deletenode linknode checkname initnodemonitor	

To protect admission to external units:

Semaphore	Procedure	Comment
TTY	error writeaddress readname	

To protect admission for different opcoms(not necessary):

Semaphore	Procedure	Comment
opcomqueue	opcom	

To facilitate medium-term-scheduling of nodelist (deletion):

Semaphore	Procedure	Comment
opqueue	regputgetnode delete	{cause} {await}

Operator's interface to DDCPAC

The opcom-procedure works interactively with the operator and returns a '>' when expecting a new command, which may be chosen among commands (computer outputs underlined):

```
>OPEN <nodename:string of characters>
>SHOW
>LINK <nodename:string of characters>
>DELETE <nodename:string of characters>
```

or among parametersettings:

```
>PERIOD <number of ticks:integer>
>INSIG <name:string of characters>/<'AD ',number:integer>
>SCALE <scale:real>
>FILTER <filter:real>
>PIDIN <name:string of characters>/<'AD ',number:integer>
>PIDREF <name:string of characters>/<'AD ',number:integer>
>PIDOUT <name:string of characters>/<'DA ',number:integer>
>K <gain:real>
>TI <integrating time constant:real>
>TD <derivative time constant:real>
>LIMIT <limit:real>
>INSIG1 <name:string of characters>/<'AD ',number:integer>
>INSIG2 <name:string of characters>/<'AD ',number:integer>
>INSIG3 <name:string of characters>/<'AD ',number:integer>
>INSIG4 <name:string of characters>/<'AD ',number:integer>
>OUTPUT <name:string of characters>/<'DA ',number:integer>
>SCAL1 <scal1:real>
>SCAL2 <scal2:real>
>SCAL3 <scal3:real>
>SCAL4 <scal4:real>
>LEVEL <level:real>
```

Example 1: State feedback of second order system

```
AD 0 |------|                       DA 0|  |------|       |------|
---->| -2.5 |->+->+-----------------|->|  G2  |--|->|  G1  |--|-->
     |------|  |  |                 |  |------|  |    |------|  |
               |  |                 |              |              |
               |  | |------|AD 2|                  |              |
               |  |-| -1.3 |----|<-----------------|              |
               |    |------|    |                                 |
               |    |------|AD 1|                                 |
               |----| -2.5 |----|<--------------------------------|
                    |------|    |
               computer|process
                       |
```

```
>OPEN REG1
>* priority= 10
>* nodetype= OUTNODE
>INSIG1 AD 0
>INSIG2 AD 1
>INSIG3 AD 2
>OUTPUT DA 0
>SCAL1 2.5
>SCAL2 -2.5
>SCAL3 -1.3
>LINK
```

Example 2: Nested PID-loops

```
                                                  |
 AD 0       |------|      |------| DA 0 |  |------|  |------|
----->-+-| PID1 |->+-| PID2 |----->|->| G2  |-|-| G1  |--|-->
       |  |------|  |  |------|     |  |------| |  |------|  |
       |            |                |           |            |
       |            |     |---| AD 2 |           |            |
       |            |-----|-1 |------|<----------|            |
       |                  |---|      |                        |
       |                             |                        |
       |                  |---| AD 1 |                        |
       |------------------|-1 |------|<-----------------------|
                          |---|      |
                                     |
                             computer|process
                                     |
```

```
>OPEN PID1
*_priority= 15
*_nodetype= PIDNODE
>PIDREF AD 0
>PIDIN AD 1
>K 1.2
>TI 100.0
>LINK
>OPEN PID2
*_priority= 20
*_nodetype= PIDNODE
>PIDREF PID1
>PIDIN AD 2
>K 2.1
>TI 50.0
>LINK
```

APPENDIX 1: KERNEL

Programlistings

<u>awake(ident:procid; x:time);</u>

Awakes sleeping process with procid ident for start.

Example: awake(process1,time1);

3. EXAMPLE OF USAGE OF Z

A short, instructive but very artificial example will give an idea about how to use Z and how Z is working.

Assume that for some peculiar reason you want a program to write

Process 1 time is {actual time}

every two seconds and

Process 2 time is {actual time}

every third second on the terminal. How this is solved in Z is shown by the program Twoprocesses.

```
Program TwoProcesses;
{A simple example of usage of the program kernel}

type procid=integer;
     unsignedint=0..65535;
     semaphore=(teletype,nothing);
     time=record
          min,tick:integer;
          end;

var allocationarea:unsignedint;
    proc1,proc2:procid;

Procedure initkernel(var freememory:unsignedint;kernelrequest,
                              mainrequest:unsignedint);external;
Procedure initproc(var ident:procid; prio:integer;
             memoryrequest:unsignedint; procedure process); external;
Procedure ready;external;
Procedure wait(x:semaphore); external;
Procedure signal(x:semaphore); external;
Procedure sleep(var x:time); external;
Procedure clocktime(var x:time);  external;

Procedure process1;
var time1:time;
  begin
  ready;
  clocktime(time1);
  repeat
    wait(teletype);
    write('process 1 time is ',time1.min:4,'  minutes  ');
    writeln(time1.tick:4,' tick');
    signal(teletype);
    time1.tick:=time1.tick + 100;
    sleep(time1);
  until false
  end;
```

```
Procedure process2;
var time2:time;
  begin
  ready;
  clocktime(time2);
  repeat
    wait(teletype);
    write('process 2 time is ',time2.min:4,'  minutes  ');
    writeln(time2.tick:4,'  tick');
    signal(teletype);
    time2.tick:=time2.tick + 150;
    sleep(time2)
  until false
  end;

{=========================================================}
{MAIN PROGRAM    =========================================}
{=========================================================}

begin

initkernel(allocationarea,1000,1000);
writeln('allocationarea= ',allocationarea);

initproc(proc1,2,1000,process1);
process1;

initproc(proc2,2,1000,process2);
process2;

repeat
until false
end.
```

To run this example the objectfile is linked as

```
.R LINK
*DX1:KERNEX=DX1:KERNEX,KERNEL,DX0:PASSUP
```

(The file with Twoprocesses is called KERNEX)

4. HOW A NORMAL PASCALPROGRAM USES THE PRIMARY MEMORY

To make it possibly to explain how Z is implemented we will here explain how a normal Pascal program uses the primary memory.

There are two data areas in the primary memory which are of concern to us - the global data area and the program stack. The global area contains the values of all variables defined at level 0 within a PASCAL program, while the program stack contains all variables local to PASCAL procedures along with linkage information. Register 5 (R5) always points to the base of the global data area, while the stack pointer (SP, register 6) points to the top of the program stack. The dynamic variables are stored on the HEAP and the space is allocated when the procedures <u>new</u> is called.

```
TOP OF        -------------------
MEMORY ==>   ! MONITOR  (RT-11)  !
              -------------------
             !   USER STACK      !
             !   AND LOCAL       !
             !   VARIABLES       !
             !      \/           !
             !      \/           !
   SP ==>    !      \/           !
              -------------------
             !  FREE CORE        !
 $KORE ==>   !                   !
              -------------------
             !      ^            !
             !      ^            !
             !      ^            !
             !  HEAP             !
              -------------------
             !  GLOBAL           !
   R5 ==>    !    VARIABLES      !
              -------------------
             !  PASSUP CODE      !
              -------------------
             !  USER'S CODE      !
              -------------------
             !  INTERRUPT        !
             !  VECTORS          !
              -------------------
```

When a procedure is called, the PASCAL system saves the return address and allocates space for parameters and local variables on the stack. Each separate invocation of a procedure causes a new data area (called a *stack frame*) to be allocated. The following diagram shows the structure of a stack frame:

```
                           -------------------------
     not used in           !   Function return     !
     Z                     !        value          !
                           -------------------------
                           !      Parameters       !
                           !       (if any)        !
                           -------------------------
                           !      Return address   !
                           -------------------------
                           !     Local variables   !
                           !        (if any)       !
                           -------------------------
                    SP =>  !       Static link     !
                           -------------------------
```

The *static* link is simply a pointer to the stack frame of the latest invocation of the procedure lexically enclosing the one just entered. This pointer is used to compute the address of intermediate level variables - variables which are neither local nor global. All local variables (those which are defined by the current procedure) are indexed relative to the SP. Global variables are accessed by indexing from R5. To access an intermediate level variable it is necessary to traverse the static link list until the proper stack frame is reached, and indexing by that base value. To make variable addressing a little easier for the MACRO programmer, the compiler will replace the name of a PASCAL variable by its offset in a code insert, and will replace the name of a PASCAL function or procedure by its entry label. Note that it is the programmer's responsibility to address the PASCAL variable using the proper base register (R5 for globals, SP for locals). The compiler will give no error message if this is done improperly.

5. HOW Z IS IMPLEMENTED

Z is written as a Пascalprogram↓ (with inline assembly coding) without any main body. Thus the program, which is called kernel, is in fact only defining a number of procedures. Among these procedures there are the so called external procedures (Available to a user):

initkernel
initproc
ready
wait
signal
await
cause
sleep
clocktime
setpri
awake

When linking the users program with Passup, the user has to include the program kernel in the command string (see the command string in context with the example Twoprocesses).

The kernel administers the registers and the stacks to make it possible to switch the program control between different processes. When a process is running, the PC is pointing in the code of the process. The stackpointer is pointing at the local stack. The process continues to run until a significant event occurs, i.e. when the process calls a kernel procedure or a clock interrupt occurs. When a significant event happens the kernel decides which process to run next. To stop a process and start up another is a messy affair of fixing registers and the stacks of the actual processes. The picture is even more complicated depending on the fact that the kernel in itself is a program which use the registers and an own stack.

6. THE LIST STRUCTURES OF Z

Z keeps record over the different processes by keeping them in five different queues. The queues are organized as a list of nodes with one dummy node as a listhead pointed out by a variable which is used to identify the list. The list structures are described by a figure on the next page.

This figures shows the lists'structures.

```
                          --------------
run --------->          !   succ        !
                        !   pred        !
                        !   child       !
                        !   nextproc    !
                        !   stacktop    !
                        !   procid      !
                        !   priority    !
                        !   starttime   !
                          --------------

                          ---------          ---------          ---------
rdy ---------->         !         !  ------>  !         !  ------>  !         !  -->list
     last nod<--!         !  ^-------  !         !  ^-------  !         !     head
   list head ->  !xxxxxxx!            !         !            !         !
                 !xxxxxxx!            !         !            !         !
                          ---------          ---------          ---------

                          ---------          ---------          ---------
clk ---------->         !         !  ------>  !         !  ------>  !         !  -->list
     last nod<--!         !  ^-------  !         !  ^-------  !         !     head
   list head ->  !xxxxxxx!            !         !            !         !
                 !xxxxxxx!            !         !            !         !
                          ---------          ---------          ---------

                     ----------                            ----------
sem ----->         ! succ     !  ------------------->  ! succ     !
                   ! number   !                          ! number   !
                   ! flag     !                          ! flag     !
                   ! queue    !                          ! queue    !
                     ----------                            ----------
  list head          list head              list head           list head
  of queue           of delayed             of queue            of delayed
  ----------         ----------             ---------           ---------
  !        !         !        !             !       !           !       !
  !        !         !        !             !       !           !       !
  !xxxxxxxx!         !xxxxxxxx!             !xxxxxxx!           !xxxxxxx!
  !xxxxxxxx!         !xxxxxxxx!             !xxxxxxx!           !xxxxxxx!
  ----------         ----------             ---------           ---------

  ----------         ----------             ---------           ---------
  !        !         !        !             !       !           !       !
  !        !         !        !             !       !           !       !
  !        !         !        !             !       !           !       !
  !        !         !        !             !       !           !       !
  ----------         ----------             ---------           ---------

                ---------                                 ---------
all --->      !         !  ---->   ..........   ---->  !         !  -->list head
last nod<-!         !  ^----                ^----  !         !
              !         !                              !         !
              !         !                              !         !
                ---------                                 ---------
```

The figure describe a situation when 9 processes are defined.

The run-queue consists of only the currently running process.

The processes which are ready to run lies in the ready-queue in priority-order.

The clockqueue consists of processes waiting to be run in the future. They are ordered after indicated starttime. At each clockinterrupt the clockqueue is checked and if the leading processes' starttimes are equal to or less than the actual time the process will be moved to the ready-queue.

The semaphore-list consists of one queue for each resource (for example teletype, lineprinter, common data area etc.). In this example there are two semaphores.

The all-list is an single-linked list which consist of all processes. The processes are linked to the list in the order of initialization.

7. STATUS OF THE RESTING PROCESSES

The status of a not running process is saved on the local stack after the static link as in figur.

```
!        SL        !
--------------------
!        PSW       !
!        PC        !
!        R0        !
!        R1        !
!        R2        !
!        R3        !
!        R4        !
!        R5        !
!      $KORE       !
```

The stackpointer is stored in the processrecord, which is available for Z via the liststructures. For saving the status of an interrupted process the procedures enterkernel and savestatus are selfexplaining. For startup of a process chosen by Z to be running see procedure startrun.

8. MEMORY ALLOCATION

Initkernel initiates the lists described in chapter 5 and creates processrecords and startstatus for idleproc and main. After that the clock is started and the realtime scheduling starts.

Creating a new process (child) may be done from any process (parent).The memoryallocation is done by initproc.

On next page the memoryallocation is illustrated for the example in chapter 3.

9. PROCEDURES IN THE KERNEL

The procedures in the kernel are divided in two main groups. The second group is divided into two subgroups.

1. External procedures.

2. Internal procedures

   2.1 Interfacing procedures

   2.2 Miscellaneous procedures

9.1. EXTERNAL PROCEDURES

The external procedures are those procedures, which the user may use.

```
initkernel
initproc
wait
signal
await
cause
sleep
clocktime
setpri
awake
```

Comments on the first two procedures are found in chapter 8.

The next two procedures are commented below.

Figure illustrating the effect of initkernel and initproc.
In this case two calls of initproc.

```
before                                          after

          ---------------------------          ---------------------------------
          !  RT-11                  !          !  RT-11                        !
          ---------------------------          ---------------------------------
SP==>     !  USER'S STACK           !          ! KERNEL'S STACK                !
          ---------------------------          ---------------------------------
          !  FREE MEMORY            ! $KORE==> !  FREE MEMORY                  !
          !                         !          -------------------------------!
          !                         !          !  KERNEL'S HEAP
          !                         !          -------------------------------!
          !                         !          !  IDLEPROC'S STACK             !
          !                         !          ---------------------------------
          !                         !          !  MAIN'S STACK                 !
          !                         !          ---------------------------------
          !                         !          !  FREE MEMORY                  !
          !                         !          ---------------------------------
          !                         !          !  MAIN'S HEAP                  !
          !                         !          ---------------------------------
          !                         !          !  STACK OF PROCESS 1           !
          !                         !          ---------------------------------
          !                         !          !  FREE MEMORY                  !
          !                         !          ---------------------------------
          !                         !          !  HEAP OF PROCESS 1            !
          !                         !          ---------------------------------
          !                         !          !  STACK OF PROCESS 2           !
          !                         !          ---------------------------------
          !                         !          !  FREE MEMORY                  !
          !                         !          ---------------------------------
          !                         !          !  HEAP OF PROCESS 2            !
          !                         !          ---------------------------------
          !                         !          !                               !
          !                         !          !  FREE MEMORY                  !
          !                         !          !                               !
          !                         !          ---------------------------------
          !                         !          !  KERNEL'S GLOBAL              !
$KORE==>  !                         !    R5==> !  VARIABLES                    !
          ---------------------------          ---------------------------------
          !  USER'S GLOBAL          !          !  USER'S GLOBAL                !
R5==>     !  VARIABLES              !          !  VARIABLES                    !
          ---------------------------          ---------------------------------
          !  PASSUP CODE            !          !  PASSUP CODE                  !
          ---------------------------          ---------------------------------
          !  KERNEL'S CODE          !          !  KERNEL'S CODE                !
          ---------------------------          ---------------------------------
          !  USER'S CODE            !          !  USER'S CODE                  !
          ---------------------------          ---------------------------------
          !  INTERRUPT              !          !  INTERRUPT                    !
          !  VECTORS                !          !  VECTORS                      !
          ---------------------------          ---------------------------------
```

wait

wait(x) is a request for a resource x (x is for example a lineprinter). In kernel the declaration of wait begin with

Procedure wait(nr:integer)

But in the user's program wait is declared as

    Procedure wait(x:semaphore); external;

This is possible because an ordered set with n elements is by Pascal handed as integers from 0 to n-1. In that way the type semaphore is a dynamic field. There are in principle three different situations in which wait can be called:

1. The resource has not been requested before. In this case

   i) a new semaphore is created

   ii) The flag is set to zero to indicate that the resource is occupied.

   iii) the process is not moved from running position

   iv) schedule is called

2. The resource has been requested before but the flag is set , which means that the resource is free. In this case the moment ii), iii) and iv) above is done.

3. The resource har been requested earlier and the flag is zero. In this case

   i) the process is moved from the running position to the queue, where it is placed in priority-order.

   ii) schedule is called

signal(x);

This procedure is used when a resource x is not requested any more. There are in principle three different situations when signal can be called.

The third one is in a way not normal.

1. The flag is zero and the queue is empty. In this case

   1) the flag is set to one

   ii) schedule is called.

2. The flag is zero and the queue is not empty. In this case

   i) the first process in the queue is moved to the ready-queue

   ii) schedule is called

3. The flag is one. This means that a call of signal has already been done by the process without being followed by a wait(x). In this case

   i) schedule is called.

9.2 INTERNAL PROCEDURES

The procedures in the kernel which can not be (by some degree of success) called by the user are called the internal procedures. They are divided in two groups, the interfacing procedures and the miscellaneous procedures.

9.2.1 INTERFACING PROCEDURES

The interfacing procedures administer the registers and the stacks which must been done when the programcontrol is switching between the users code and the kernel's code. The interfacing procedures are: