Appendix B

Program Codes

This appendix contains the two FORTRAN programs EVOL and GRUP (with option REKO) used for the test series described in Chapter 6 plus the extension KORR as of 1976, which covers all features of GRUP/REKO as well as correlated mutations (Schwefel, 1974; see also Chap. 7) introduced shortly after the first German version of this work was finished in 1974 (and reproduced as monograph by Birkhäuser, Basle, Switzerland, in 1977). GRUP and REKO thus should no longer be used or imitated.

B.1 (1+1) Evolution Strategy EVOL

1. Purpose

The EVOL subroutine is a FORTRAN coding of the two membered evolution strategy. It is an iterative direct search strategy for a parameter optimization problem. A search is made for the minimum in a non-linear function of an arbitrary but finite number of continuously variable parameters. Derivatives of the objective function are not required. Constraints in the form of inequalities can be incorporated (right hand side ≥ 0). The user must supply initial values for the variables and for the appropriate step sizes. If the initial state is not feasible, a search is made for a feasible point by minimizing the sum of the negative values for the constraints that have been violated.

2. Subroutine parameter list

```
EVOL (N,M,LF,LR,LS,TM,EA,EB,EC,ED,SN,FB,XB,SM,X,F,G,T,Z,R)
```

All parameters apart from LF, FB, X, and Z must be assigned values or names either before or when the subroutine is called. The variables XB and SM do not retain the values initially assigned to them.

N (integer) Number of parameters (>0).
M (integer) Number of constraints (≥0).

LF (integer) Return code with the following meaning:
- LF=−2 Starting point not feasible and search for a feasible state unsuccessful. Feasible region probably empty.
- LF=−1 Starting point not feasible and search for a feasible state terminated because time limit was reached.
- LF=0 Starting point not feasible, search for a feasible state successful. The final values of XB can be used as starting point for a subsequent search for a minimum if EVOL is called again.
- LF=1 Search for minimum terminated because time limit was reached.
- LF=2 Search for minimum terminated in an orderly fashion. No further improvement in the value of the objective function could be achieved in the context of the given accuracy parameters. Probably the final state XB (extreme point) having FB (extreme value) lies near a local minimum, perhaps the global minimum.

LR (integer) Auxiliary quantity used in step size management. Normal value 1.0. The step sizes are adjusted so that on average one success (improvement in the value of the objective function) is obtained in $5 \cdot \mathrm{LR}$ trials (objective function calls). This is computed on the last $10 \cdot \mathrm{N} \cdot \mathrm{LR}$ trials.

LS (integer) Auxiliary quantity used in convergence testing. Minimum value 2.0. The search is terminated if the value of the objective function has improved by less than EC (absolute) or ED (relative) in the course of $10 \cdot \mathrm{N} \cdot \mathrm{LR} \cdot \mathrm{LS}$ trials. Note: the step sizes are reduced by at most a factor $\mathrm{SN}^{10 \cdot \mathrm{LS}}$ during this period. The factor is 0.2^{LS} if SN = 0.85 is selected.

TM (real) Parameter used in controlling the computation time, e.g., the maximum CPU time in seconds, depending on the function designated T (see below). The search is terminated if T > TM. This check is performed after each $\mathrm{N} \cdot \mathrm{LR}$ mutations (objective function calls).

EA (real) Lower bound to step sizes, absolute. EA > 0.0 must be chosen large enough to be treated as different from 0.0 within the accuracy of the computer used.

EB (real) Lower bound to step sizes relative to values of variables. EB > 0.0 must be chosen large enough for 1.0 + EB to be treated as different from 1.0 within the accuracy of the computer used.

EC (real) Parameter in convergence test, absolute. See under LS. (EC > 0.0, see EA).

ED (real) Parameter in convergence test, relative. See under LS. (1.0 + ED > 1.0, see EB). Convergence is assumed if the data pass one or both tests. If it is desired to suppress a test, it is possible either to set EC = 0.0 or to choose a value for ED such that 1.0 + ED = 1.0 but ED > 0.0 within the accuracy of the machine.

SN (real) Auxiliary variable for step size adjustment. Normal value 0.85. The step size can be kept constant during the trials by setting SN = 1.0. The success rate indicated by LR is used to adjust the step size by a factor SN or 1.0/SN after every $\mathrm{N} \cdot \mathrm{LR}$ trials.

FB	(real)	Best value of objective function obtained during the search.
XB	(one dimensional real array of length N)	On call: holds initial values of variables. On exit: holds best values of variables corresponding to FB.
SM	(one dimensional real array of length N)	On call: holds initial values of step sizes (more precisely, standard deviations of components of the mutation vector). On exit: holds current step sizes of the last (not necessarily successful) mutation. Optimum initialization: somewhere near the local radius of curvature of the objective function hypersurface divided by the number of variables. More practical suggestion: SM(I) = DX(I)/SQRT(N), where DX(I) is a crude estimate of either the distance between start and expected optimum or the maximum uncertainty range for the variable X(I). If the SM(I) are initially set too large, a certain time elapses before they are appropriately adjusted. This is advantageous as regards the probability of locating the global optimum in the presence of several local optima.
X	(one dimensional real array of length N)	Space for holding a variable vector.
F	(real function)	Name of the objective function, which is to be provided by the user.
G	(real function)	Name of the function used in calculating the values of the constraint functions; to be provided by the user.
T	(real function)	Name of the function used in controlling the computation time.
Z	(real function)	Name of the function used in transforming a uniform random number distribution to a normal distribution. If the nomenclature Z is retained, the function Z appended to the EVOL subroutine can be used for this purpose.
R	(real function)	Name of the function that generates a uniform random number distribution.

3. Method

See I. Rechenberg, Evolution Strategy: Optimization of Technical Systems in Accordance with the Principles of Biological Evolution (in German), vol. 15 of Problemata series, Verlag Frommann-Holzboog, Stuttgart, 1973; also H.-P. Schwefel, Numerical Optimization of Computer Models, Wiley, Chichester, 1981 (translated by M. W. Finnis from Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie, vol. 26 of Interdisciplinary Systems Research, Birkhäuser, Basle, Switzerland, 1977).

The method is based on a very much simplified simulation of biological evolution using the principles of mutation (random changes in variables, normal distribution for change vector) and selection (elimination of deteriorations and retention of improvements). The widths of the normal distribution (or step sizes) are controlled by reference to the ratio of the number of improvements to the number of mutations.

4. Convergence criterion

Based on the change in the value of the objective function (see under LS, EC, and ED).

5. Peripheral I/O: none.

6. Notes

If there are several (local) minima, only one is located. Which one actually is found depends on the initial values of variables and step sizes as well as on the random number sequence. In such cases it is recommended to repeat the search several times with different sets of initial values and/or random numbers. The approximation to the minimum is usually poor if the search terminates at the boundary of the feasible region defined by the constraints. Better results can then be obtained by setting LR > 1, LS > 2, and/or SN > 0.85 (maximum value 1.0). In addition, the bounds EA and EB should not be made too small. The same applies if the objective function has discontinuous first partial derivatives (e.g., in the case of Tchebycheff approximation).

7. Subroutines or functions used

The function names should be declared as external in the segment that calls EVOL.

7.1 Objective function

This is to be written by the user in the form:

```
------------------------------------------------------
      FUNCTION F(N,X)
      DIMENSION X(N)
      ...
      F=...
      RETURN
      END
------------------------------------------------------
```

N represents the number of parameters, and X represents the formal parameter vector. The function should be written on the basis that EVOL searches for a minimum; if a maximum is to be sought, F must be supplied with a negative sign.

7.2 Constraints function

This is to be written by the user in the general style:

```
------------------------------------------------------
      FUNCTION G(J,N,X)
      DIMENSION X(N)
      GOTO(1,2,3,...,(M)),J
1     G=...
      RETURN
2     G=...
```

```
      RETURN
      ...
(M)   G=...
      RETURN
      END
------------------------------------------------------
```

N and X have the meanings described for the objective function, while J (integer) is the serial number of the constraint. The statements should be written on the basis that EVOL will accept vector X as feasible if all the G values are larger than or equal to 0.0.

7.3 Function for controlling the computation time

This may be defined by the user or called from the subroutine library in the particular machine. The following structure is assumed:

```
REAL FUNCTION T(D)
```

where D is a dummy parameter. T should be assigned the monitored quantity, e.g., the CPU time in seconds limited by TM. Many computers are supplied with ready-made timing software. If this is given as a function, only its name needs to be supplied to EVOL instead of T as a parameter. If it is a subroutine, the user can program the required function. For example, the subroutine might be called SECOND(I), where parameter I is an integer representing the CPU time in microseconds, in which case one could program:

```
------------------------------------------------------
      FUNCTION T(D)
      CALL SECOND(I)
      T=1.E-6*FLOAT(I)
      RETURN
      END
------------------------------------------------------
```

7.4 Function for transforming a uniformly distributed random number to a normally distributed one

See under Section 8.

7.5 Function for generating a uniform random number distribution in the range (0,1]

The structure must be

```
REAL FUNCTION R(D)
```

where D is dummy. R is the value of the random number. Note: The smallest value of R must be large enough for the natural logarithm to be generated without floating-point overflow. The standard library usually includes a suitable program, in which case only the appropriate name has to be supplied to EVOL.

8. Function Z(S,R)

This function converts a uniform random number distribution to a normal distribution pairwise by means of the Box-Muller rules. The standard deviation is supplied as parameter S, while the expectation value for the mean is always 0.0. The quantity LZ is common to EVOL and Z by virtue of a COMMON block and acts as a switch to transmit only one of the two random numbers generated in response to each second call.

```
------------------------------------------------------------
      SUBROUTINE EVOL(N,M,LF,LR,LS,TM,EA,EB,EC,ED,SN,FB,
     1XB,SM,X,F,G,T,Z,R)
      DIMENSION XB(1),SM(1),X(1),L(10)
      COMMON/EVZ/LZ
      EXTERNAL R
      TN=TM+T(D)
      LZ=1
      IF(M)4,4,1
1     LF=-1
C
C   FEASIBILITY CHECK
C
      FB=0.
      DO 3 J=1,M
      FG=G(J,N,XB)
      IF(FG)2,3,3
2     FB=FB-FG
3     CONTINUE
      IF(FB)4,4,5
C
C   ALL CONSTRAINTS SATISFIED IF FB<=0
C
4     LF=1
      FB=F(N,XB)
C
C   INITIALIZATION OF SUCCESS COUNTER
C
5     DO 6 K=1,10
6     L(K)=N*K/5
      LE=N+N
      LM=0
      LC=0
      FC=FB
C
C   MUTATION, START OF MAIN LOOP
C
```

```
7     DO 8 I=1,N
8     X(I)=XB(I)+Z(SM(I),R)
      IF(LF)9,9,12
C
C   AUXILIARY OBJECTIVE
C
9     FF=0.
      DO 11 J=1,M
      FG=G(J,N,X)
      IF(FG)10,11,11
10    FF=FF-FG
11    CONTINUE
      IF(FF)32,32,16
C
C   ALL CONSTRAINTS SATISFIED IF FF<=0
C
12    IF(M)15,15,13
13    DO 14 J=1,M
      IF(G(J,N,X))19,14,14
14    CONTINUE
15    FF=F(N,X)
16    IF(FF-FB)17,17,19
C
C   SUCCESS
C
17    LE=LE+1
      FB=FF
      DO 18 I=1,N
18    XB(I)=X(I)
19    LM=LM+1
      IF(LM-N*LR)7,20,20
C
C   SUCCESS RATE CONTROL
C
20    K=1
      IF(LE-L(1)-N-N)23,22,21
21    K=K-1
22    K=K-1
C
C   ADAPTATION OF STEP SIZES
C
23    DO 24 I=1,N
24    SM(I)=AMAX1(SM(I)*SN**K,ABS(XB(I))*EB,EA)
      DO 25 K=1,9
```

```
25    L(K)=L(K+1)
      L(10)=LE
      LM=0
      LC=LC+1
      IF(LC-10*LS)31,26,26
C
C   CONVERGENCE CRITERION
C
26    IF(FC-FB-EC)28,28,27
27    IF((FC-FB)/ED-ABS(FC))28,28,30
28    LF=ISIGN(2,LF)
29    RETURN
30    LC=0
      FC=FB
C
C   TIME CONTROL
C
31    IF(T(D)-TN)7,29,29
32    DO 33 I=1,N
33    XB(I)=X(I)
      FB=F(N,XB)
      LF=0
      GOTO 29
      END
------------------------------------------------------------
      FUNCTION Z(S,R)
      COMMON/EVZ/LZ
      DATA ZP/6.28318531/
      GOTO(1,2),LZ
1     A=SQRT(-2.*ALOG(R(D)))
      B=ZP*R(D)
      Z=S*A*SIN(B)
      LZ=2
      RETURN
2     Z=S*A*COS(B)
      LZ=1
      RETURN
      END
------------------------------------------------------------
```

B.2 (μ , λ) Evolution Strategies GRUP and REKO

1. Purpose

The GRUP subroutine is a FORTRAN program to handle a multimembered (L,LL) evolution strategy with L parents and LL descendants per generation. It is an iterative direct search strategy for handling parameter optimization problems. A search is made for the minimum in a non-linear function of an arbitrary but finite number of continuously variable parameters. Derivatives of the objective function are not required. Constraints in the form of inequalities can be incorporated (right hand side ≥ 0). The user must supply initial values for the variables and for the appropriate step sizes. If the initial state is not feasible, a search is made for a feasible point that minimizes the sum of the negative values for the constraints that have been violated.

2. Subroutine parameter list

```
GRUP (REKO,L,LL,N,M,LF,TM,EA,EB,EC,ED,SN,FA,FB,XB,SM,X,FK,XK,SK,F,G,T,Z,R)
```

All parameters apart from LF, FA, FB, X, FK, XK, SK, and Z must be assigned values or names before or when the subroutine is called. The variables XB and SM do not retain the values initially assigned to them.

REKO	(logical)	Switch for alternative with/without recombination.
	REKO=.FALSE.	No recombination. The step sizes retain the relationship initially assigned to them.
	REKO=.TRUE.	Recombination occurs. The relationships between the step sizes alter during the search.
L	(integer)	Number of parents ($\geq$1). This parameter should not be chosen too small if recombination is to occur.
LL	(integer)	Number of descendants (>L). Recommended to choose a value $\geq 6 \cdot$ L.
N	(integer)	Number of parameters (>0).
M	(integer)	Number of constraints ($\geq$0).
LF	(integer)	Return code with the following meanings:
	LF=-2	Starting point not feasible and search for a feasible state unsuccessful. Feasible region probably empty.
	LF=-1	Starting point not feasible and search for a feasible state terminated because time limit was reached.
	LF=0	Starting point not feasible, search for a feasible state successful. The final values of XB can be used as starting point for the search for a minimum if GRUP is called again.
	LF=1	Search for minimum terminated because time limit was reached.
	LF=2	Search for minimum terminated in an orderly fashion.

	LF=2 (continued)	No further improvement in the value of the objective function could be achieved in the context of the framework of the given accuracy parameters. Probably the final state XB (extreme value) lies near a local minimum, perhaps the global minimum.
TM	(real)	Parameter used in monitoring the computation time, e.g., the maximum CPU time in seconds, depending on the function designated T (see below). The search is terminated if T > TM. This check is performed after every generation = LL objective function calls.
EA	(real)	Lower bound to step sizes, absolute. EA > 0.0 must be chosen large enough to be treated as different from 0.0 within the accuracy of the computer used.
EB	(real)	Lower bound to step sizes relative to values of variables. EB > 0.0 must be chosen large enough for 1.0 + EB to be treated as different from 1.0 within the accuracy of the computer used.
EC	(real)	Parameter in convergence test, absolute. The search is terminated if the difference between the best and worst values of the objective function within a generation is less than or equal to EC (EC > 0.0, see EA).
ED	(real)	Parameter in convergence test, relative. The search is terminated if the difference between the best and worst values of the objective function within a generation is less than or equal to ED multiplied by the absolute value of the mean of the objective function as taken over all L parents in a generation (1.0 + ED > 1.0, see EB). Convergence is assumed if the data pass one or both tests. If it is desired to delete a test, it is possible either to set EC = 0.0 or to choose a value for ED such that 1.0 + ED = 1.0 but ED > 0.0 within the accuracy of the machine.
SN	(real)	Auxiliary quantity used in step size adjustment. Normal value C/SQRT(N), with C > 0.0, e.g., C = 1.0 for L = 10 and LL = 100. C can be increased as LL increases, but it must be reduced as L increases. An approximation for L = 1 is LL proportional to SQRT(C) · EXP(C).
FA	(real)	Current best objective function value for population.
FB	(real)	Best value of objective function attained during the whole search. The minimum found may not be unique if FB differs from FA because: (1) there is a state with an even smaller value for the objective function (e.g., near a local minimum or even near the global minimum) that has been lost over the generations; or (2) the minimum consists of several quasi-singular peaks on account of the finite accuracy of the computer used. Usually, the difference between FA and FB is larger in the first case than in the second, if EC and ED have been assigned small values.

XB	(one dimensional real array of length N)	On call: holds initial values of variables. On exit: holds best values of variables corresponding to FB.
SM	(one dimensional real array of length N)	On call: holds initial values of step sizes (more precisely, standard deviations of components of the mutation vector). On exit: holds current step sizes of the last (not necessarily successful) mutation. Optimum initialization: somewhere near the local radius of curvature of the objective function hypersurface divided by the number of variables. More practical suggestion: SM(I) = DX(I)/SQRT(N), where DX(I) is a crude estimate of either the distance between start and expected optimum or the maximum uncertainty range for the variable X(I). If the SM(I) are initially set too large, it may happen that a good starting point is lost in the first generation. This is advantageous as regards the probability of locating the global optimum in the presence of several local optima.
X	(one dimensional real array of length N)	Space for holding a variable vector.
FK	(one dimensional real array of length $2 \cdot$ L)	Holds objective function values for the L best individuals in each of the last two generations.
XK	(one dimensional real array of length $2 \cdot$ L $\cdot$ N)	Holds the variable values for N components for each of the L parents in each of the last two generations. XK(1) to XK(N) hold the state vector X for the first individual, the next N locations do the same for the second, and so on.
SK	(one dimensional real array of length $2 \cdot$ L $\cdot$ N)	Holds the standard deviations, structure as for XK.
F	(real function)	Name of the objective function, which is to be programmed by the user.
G	(real function)	Name of the function for calculating the values of the constraints, to be programmed by the user.
T	(real function)	Name of function used in monitoring the computation time.
Z	(real function)	Name of function used in transforming a uniform random number distribution to a normal distribution. If the name Z is retained, the function Z listed after the GRUP subroutine can be used for this purpose.
R	(real function)	Name of the function that generates a uniform random number distribution.

3. Method

GRUP has been developed from EVOL. The method is based on a very much simplified simulation of biological evolution. See I. Rechenberg, Evolution Strategy: Optimization of Technical Systems in Accordance with the Principles of Biological Evolution (in Ger-

man), vol. 15 of Problemata series, Verlag Frommann-Holzboog, Stuttgart, 1973; also H.-P. Schwefel, Numerical Optimization of Computer Models, Wiley, Chichester, 1981 (translated by M. W. Finnis from Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie, vol. 26 of Interdisciplinary Systems Research, Birkhäuser, Basle, Switzerland, 1977).

The current L parameter vectors are used to generate LL new ones by means of small random changes.

The best L of these become the initial ones for the next generation (iteration). At the same time, the step sizes (standard deviations) for the changes in the variables (strategy parameters) are altered. The selection leads to adaptation to the local topology if LL/L is assigned a suitably large value, e.g., >6. The random changes in the parameters are produced by the addition of normally distributed random numbers, while those in the step sizes are produced from random numbers with a log-normal distribution by multiplication.

4. Convergence criterion

Based on the differences in value of the objective function (see under EC and ED).

5. Peripheral I/O: none.

6. Notes

The multimembered strategy represents an improvement in reliability over the two membered strategy. On the other hand, the run time is greater when an ordinary (serial) digital computer is used. The run time increases less rapidly than in proportion to LL (the number of descendants per generation), because increasing LL increases the convergence rate (over the generations). However, minima at a boundary of the feasible region or at a vertex are attained only slowly or inexactly. In any case, although the certainty of global convergence cannot be guaranteed, numerical tests have shown that the multimembered strategy is far better than other search procedures in this respect. It is capable of handling separated feasible regions provided that the number of parameters is not large and that the initial step sizes are set suitably large. In doubtful cases it is recommended to repeat the search each time with a different set of initial values and/or random numbers. If the optimum being sought lies at a boundary of the feasible region, it is probably better to choose a value for SN (the parameter governing the rates of change of the standard deviations) less than the (maximal) value suggested above.

7. Subroutines or functions used

The function names are to be declared as external in the segment that calls GRUP.

7.1 Objective function

To be written by the user in the form:

```
------------------------------------------------------
      FUNCTION F(N,X)
      DIMENSION X(N)
      ...
      ...
      F=...
      RETURN
      END
------------------------------------------------------
```

N represents the number of parameters, and X represents the formal parameter vector. GRUP supplies the actual values. The function should be written on the basis that GRUP searches for a minimum; if a maximum is to be sought, F must be supplied with a negative sign.

7.2 Constraints function

To be written by the user in the general style:

```
------------------------------------------------------
      FUNCTION G(J,N,X)
      DIMENSION X(N)
      GOTO(1,2,3,...,(M)),J
1     G=...
      RETURN
2     G=...
      RETURN
      ...
      ...
(M)   G=...
      RETURN
      END
------------------------------------------------------
```

N and X have the meanings described for the objective function, while J (integer) is the serial number of the constraint. The statements should be written on the basis that GRUP will accept vector X as feasible if all the G values are larger than or equal to zero.

7.3 Function for monitoring the computation time

This may be defined by the user or called from the subroutine library in the particular machine. The following structure is assumed:

```
REAL FUNCTION T(D)
```

where D is a dummy parameter. T should be assigned the monitored quantity, e.g., the CPU time in seconds limited by TM. Many computers are supplied with ready-made timing software. If this is given as a function only its name needs to be supplied to GRUP,

instead of T, as a parameter. If it is a subroutine, the user can program the required function. For example, the subroutine might be called SECOND(I), where parameter I is an integer representing the CPU time in microseconds, in which case one could program:

```
------------------------------------------------------
      FUNCTION T(D)
      CALL SECOND(I)
      T=1.E-6*FLOAT(I)
      RETURN
      END
------------------------------------------------------
```

7.4 Function for transforming a uniformly distributed random number to a normally distributed one

See under 8.

7.5 Function for generating a uniform random number distribution in the range (0,1]

The structure must be

```
REAL FUNCTION R(D)
```

where D is dummy. R is the value of the random number.

Note: The smallest value of R must be large enough for the natural logarithm to be generated without floating-point overflow. The standard library usually includes a suitable program, in which case only the appropriate name has to be supplied to GRUP.

8. Function Z(S,R)

This function converts a uniform random number distribution to a normal distribution pairwise by means of the Box-Muller rules. The standard deviation is supplied as parameter S, while the expectation value for the mean is always zero. The quantity LZ is common to GRUP and Z by virtue of a COMMON block and acts as a switch to transmit only one of the two random numbers generated in response to each second call.

```
------------------------------------------------------------
      SUBROUTINE GRUP(REKO,L,LL,N,M,LF,TM,EA,EB,EC,ED,SN,
     1FA,FB,XB,SM,X,FK,XK,SK,F,G,T,Z,R)
      LOGICAL REKO
      DIMENSION XB(1),SM(1),X(1),FK(1),XK(1),SK(1)
      COMMON/GRZ/LZ
      EXTERNAL R
      KK(RR)=(LA+IFIX(FLOAT(L)*RR))*N
C
C   THE PRECEDING LINE CONTAINS A STATEMENT FUNCTION
C
      TN=TM+T(D)
```

```
      LZ=1
      IF(M)4,4,1
1     LF=-1
C
C   FEASIBILITY CHECK
C
      FB=0.
      DO 3 J=1,M
      FG=G(J,N,XB)
      IF(FG)2,3,3
2     FB=FB-FG
3     CONTINUE
      IF(FB)4,4,5
C
C   ALL CONSTRAINTS SATISFIED IF FB<=0
C
4     LF=1
      FB=F(N,XB)
5     DO 6 I=1,N
      SK(I)=AMAX1(SM(I),ABS(XB(I))*EB,EA)
6     XK(I)=XB(I)
      FK(1)=FB
      KA=N
      KB=0
C
C   SETUP OF POPULATION
C
      DO 21 K=2,L
      SA=1.
7     DO 8 I=1,N
8     X(I)=XB(I)+Z(SM(I)*SA,R)
      IF(LF)9,9,12
9     FF=0.
      DO 11 J=1,M
      FG=G(J,N,X)
      IF(FG)10,11,11
10    FF=FF-FG
11    CONTINUE
      IF(FF)60,60,17
12    IF(M)16,16,13
13    DO 15 J=1,M
      IF(G(J,N,X))14,15,15
14    SA=SA*.5
      GOTO 7
```

```
15    CONTINUE
16    FF=F(N,X)
17    IF(FF-FB)18,19,19
C
C   STORING OF BEST INTERMEDIATE RESULT
C
18    FB=FF
      KB=K
19    DO 20 I=1,N
      KA=KA+1
      SK(KA)=AMAX1(SM(I)*SA,ABS(X(I))*EB,EA)
20    XK(KA)=X(I)
21    FK(K)=FF
      IF(KB)24,24,22
22    KB=(KB-1)*N
      DO 23 I=1,N
23    XB(I)=XK(KB+I)
C
C   START OF MAIN LOOP
C
24    LA=L
      LB=0
C
C   LA AND LB FORM A ROTATING COUNTER TO AVOID SHUFFLING
C   GENOTYPES WITHIN THE ARRAYS CONTAINING PARENTS AND
C   DESCENDANTS
C
25    LC=LB
      LB=LA
      LA=LC
      LC=0
      LD=0
26    SA=EXP(Z(SN,R))
C
C   LOG-NORMAL STEP SIZE FACTOR
C
      IF(REKO)GOTO 28
      KI=KK(R(D))
      DO 27 I=1,N
      KI=KI+1
      SM(I)=SK(KI)*SA
27    X(I)=XK(KI)+Z(SM(I),R)
C
C   MUTATION WITHOUT RECOMBINATION ABOVE
```

```
C
      GOTO 30
28    SA=SA*.5
C
C   MUTATION WITH RECOMBINATION BELOW
C
      DO 29 I=1,N
      SM(I)=(SK(KK(R(D))+I)+SK(KK(R(D))+I))*SA
29    X(I)=XK(KK(R(D))+I)+Z(SM(I),R)
30    IF(LF)31,31,34
C
C   AUXILIARY OBJECTIVE
C
31    FF=0.
      DO 33 J=1,M
      FG=G(J,N,X)
      IF(FG)32,33,33
32    FF=FF-FG
33    CONTINUE
      IF(FF)60,60,38
C
C   ALL CONSTRAINTS SATISFIED IF FF<=0
C
34    IF(M)37,37,35
35    DO 36 J=1,M
      IF(G(J,N,X))46,36,36
36    CONTINUE
37    FF=F(N,X)
C
C   LD COUNTS THE NUMBER OF DESCENDANTS PRODUCED
C
38    LD=LD+1
      IF(LD-L)40,40,39
C
C   WHEREAS THE FIRST L DESCENDANTS ARE STORED AUTOMATICALLY
C   FURTHER DESCENDANTS REPLACE THE CURRENT WORST IN CASE OF
C   BEING BETTER ONLY
C
39    IF(FF-FS)41,41,46
40    KS=LB+LD
41    FK(KS)=FF
      KS=(KS-1)*N
      DO 42 I=1,N
      KS=KS+1
```

```
      SK(KS)=AMAX1(SM(I),ABS(X(I))*EB,EA)
42    XK(KS)=X(I)
      IF(LD-L)46,43,43
C
C   DETERMINING THE CURRENT WORST
C
43    KS=LB+1
      FS=FK(KS)
      DO 45 K=2,L
      KA=LB+K
      FF=FK(KA)
      IF(FF-FS)45,45,44
44    FS=FF
      KS=KA
45    CONTINUE
46    LC=LC+1
      IF(LC-LL)26,47,47
47    IF(LD-L)26,48,48
C
C   END OF A GENERATION
C
48    KA=LB+1
      FA=FK(KA)
      FC=FA
C
C   DETERMINING THE CURRENT BEST AND SUM
C
      DO 50 K=2,L
      KB=LB+K
      FF=FK(KB)
      FC=FC+FF
      IF(FF-FA)49,50,50
49    FA=FF
      KA=KB
50    CONTINUE
      IF(FA-FB)51,51,53
C
C   DETERMINING WHETHER THE CURRENT BEST IS BETTER THAN
C   THE SO FAR OVERALL BEST
C
51    FB=FA
      KB=(KA-1)*N
      DO 52 I=1,N
52    XB(I)=XK(KB+I)
```

```
C
C   CONVERGENCE CRITERION
C
53    IF(FS-FA-EC)55,55,54
54    IF((FS-FA)*FLOAT(L)/ED-ABS(FC))55,55,59
55    LF=ISIGN(2,LF)
56    KB=(KA-1)*N
      DO 57 I=1,N
57    X(I)=XK(KB+I)
58    RETURN
C
C   TIME CONTROL
C
59    IF(T(D)-TN)25,56,56
60    DO 61 I=1,N
61    XB(I)=X(I)
      FB=F(N,XB)
      FA=FB
      LF=0
      GOTO 58
      END
-------------------------------------------------------------
      FUNCTION Z(S,R)
      COMMON/GRZ/LZ
      DATA ZP/6.28318531/
      GOTO(1,2),LZ
1     A=SQRT(-2.*ALOG(R(D)))
      B=ZP*R(D)
      Z=S*A*SIN(B)
      LZ=2
      RETURN
2     Z=S*A*COS(B)
      LZ=1
      RETURN
      END
-------------------------------------------------------------
```

B.3 $(\mu \dot{+} \lambda)$ Evolution Strategy KORR

Plus additional subroutines: PRUEFG, SPEICH, MUTATI, DREHNG, UMSPEI, MINMAX, GNPOOL, ABSCHA.
and functions: ZULASS, GAUSSN, BLETAL.

1. Purpose

The KORR subroutine is a FORTRAN coding of a multimembered evolution strategy. It is an iterative direct search strategy for a parameter optimization problem. A search is made for the minimum in a non-linear function of an arbitrary but finite number of continuously variable parameters. Derivatives of the objective function are not required. Constraints in the form of inequalities can be incorporated (right hand side ≥ 0). The user must supply initial values for the variables and for the appropriate step sizes. If the initial state is not feasible, a search is made for a feasible point by minimizing the sum of the negative values for the constraints that have been violated.

2. Parameter list for subroutine KORR

```
KORR (IELTER, BKOMMA, NACHKO, IREKOM, BKORRL, KONVKR, IFALLK, TGRENZ,
EPSILO, DELTAS, DELTAI, DELTAP, N, M, NS, NP, NY, ZSTERN, XSTERN,
ZBEST, X, S, P, Y, ZIELFU, RESTRI, GAUSSN, GLEICH, TKONTR, KANAL)
```

All parameters apart from IFALLK, ZSTERN, ZBEST, X, and Y must be assigned values or names before or when the subroutine is called. The variables XSTERN, S, and P do not retain the values initially assigned to them.

IELTER	(integer)	Number of parents of a generation. IELTER ≥ 1 if IREKOM = 111 IELTER > 1 IF IREKOM > 111
BKOMMA	(logical)	Switch for comma or plus version.
	BKOMMA=.FALSE.	Selection criterion applied to parents and descendants; (IELTER + NACHKO) evolution strategy.
	BKOMMA=.TRUE.	Selection criterion applied only to descendants; (IELTER, NACHKO) evolution strategy.
NACHKO	(integer)	Number of descendants in a generation. NACHKO ≥ 1 if BKOMMA = .FALSE. NACHKO $>$ IELTER if BKOMMA = .TRUE.
IREKOM	(integer)	Switch for recombination type consisting of three digits each of which has values between 1 and 5. The first digit applies to the object variables X, the second one to the step sizes S, and the third one to the correlation angles P. Thus $111 \leq$ IREKOM ≤ 555. Each digit controls the recombination in the following way:

	1	No recombination
	2	Discrete recombination of pairs of parents
	3	Intermediary recombination of pairs of parents
	4	Discrete recombination of all parents
	5	Intermediary recombination of all parents in pairs
BKORRL	(logical)	Switch for variability of the mutation hyperellipsoid (locus of equal probability density).
	BKORRL=.FALSE.	The ellipsoid cannot rotate.
	BKORRL=.TRUE.	The ellipsoid can extend and rotate.
KONVKR	(integer)	Switch for the convergence criterion:
	KONVKR = 1	The difference in the objective function values between the best and worst parents at the start of each generation is used to determine whether to terminate the search before the time limit is reached. It is assumed that IELTER > 1.
	KONVKR > 1	(best $\geq 2 \cdot$ N): The change in the mean of all the parental objective function values in KONVKR generations is used as the search termination criterion. In both cases EPSILO(3) serves as the absolute and EPSILO(4) as the relative bound for deciding to terminate the search.
IFALLK	(integer)	Return code with the following meaning:
	IFALLK $= -2$	Starting point not feasible, search terminated on finding a minimal value of the auxiliary objective function without satisfying all the constraints.
	IFALLK $= -1$	Starting point not feasible, search for a feasible parameter vector terminated because time limit was reached.
	IFALLK = 0	Starting point not feasible, search for a feasible XSTERN vector successful, search for a minimum can be restarted with this.
	IFALLK = 1	Search for a minimum terminated because time limit was reached.
	IFALLK = 2	Search for minimum terminated regularly. The convergence criterion was satisfied.
	IFALLK = 3	As for IFALLK = 1, but time limit reached not at the end of a generation but in an attempt to generate NACHKO viable descendants.
TGRENZ	(real)	Parameter used in monitoring the computation time, e.g., the maximum CPU time in seconds. Search terminated at the latest at the end of the generation for which TKONTR $\geq$ TGRENZ.

EPSILO	(one dimensional real array of length 4)	Holds parameters that affect the attainable accuracy of approximation. The lowest possible values are machine-dependent.
	EPSILO(1)	Lower bound to step sizes, absolute.
	EPSILO(2)	Lower bound to step sizes relative to values of variables (not implemented in this program).
	EPSILO(3)	Limit to absolute value of objective function differences for convergence test.
	EPSILO(4)	As EPSILO(3) but relative.
DELTAS	(real)	Factor used in step-size change. All standard deviations (=step sizes) S(I) are multiplied by a common random number EXP(GAUSSN(DELTAS)), where GAUSSN(DELTAS) is a normally distributed random number with zero mean and standard deviation DELTAS. EXP(DELTAS) $\geq$ 1.0.
DELTAI	(real)	As for DELTAS, but each S(I) is multiplied by its own random factor EXP(GAUSSN(DELTAI)). EXP(DELTAI) $\geq$ 1.0. The S(I) retain their initial values if DELTAS = 0.0 and DELTAI = 0.0. The variables can be scaled only by recombination (IREKOM > 1) if DELTAI = 0.0. The following rules are suggested to provide the most rapid convergence for sphere models: DELTAS = C/SQRT(2.0 $\cdot$ N). DELTAI = C/(SQRT(2.0 $\cdot$ N/SQRT(NS)). The constant C can increase sublinearly with NACHKO, but it must be reduced as IELTER increases. The empirical value C = 1.0 has been found applicable for IELTER = 10, NACHKO = 100, and BKOMMA = .TRUE., which is a (10 , 100) evolution strategy.
DELTAP	(real)	Standard deviation in random variation of the position angles P(I) for the mutation ellipsoid. DELTAP > 0.0 if BKORRL = .TRUE. Data in radians. A suitable value has been found to be DELTAP = 5.0 $\cdot$ 0.01745 (5 degrees) in certain cases.
N	(integer)	Number of parameters N > 0.
M	(integer)	Number of constraints M $\geq$ 0.
NS	(integer)	Field length in array S or number of distinct step-size parameters that can be used, 1$\leq$NS$\leq$N. The mutation ellipse becomes a hypersphere for NS = 1. All the principal axes of the ellipsoid may be different for NS = N, whereas 1<NS<N implies an ellipsoid of rotation, e.g., NS = 2 implies an ellipsoid in which only one principal axis is different from the other N-1, which are equal in length.

NP	(integer)	Field length of array P. NP = N · (NS − 1) − ((NS − 1) · NS)/2 if BKORRL = .TRUE.; NP = 1 if BKORRL = .FALSE.
NY	(integer)	Field length of array Y. NY = (N+NS+NP+1) · IELTER · 2 if BKORRL = .TRUE.; NY = (N+NS+1) · IELTER · 2 if BKORRL = .FALSE.
ZSTERN	(real)	Best value of objective function found during search for minimum.
XSTERN	(one dimensional real array of length N)	On call: initial parameter vector. At end of search: best values for parameters corresponding to ZSTERN, or feasible vector found for the special case IFALLK = 0.
ZBEST	(real)	Current best value of objective function for the population, may be different from ZSTERN if BKOMMA = .TRUE.
X	(one dimensional real array of length N)	Holds the variables for a descendant.
S	(one dimensional real array of length NS)	Holds the step sizes for a descendant. The user must supply initial values. Universally valid rules for selecting the best S(I) are not available. If the step sizes are too large, a very good starting point can be wasted (BKOMMA = .TRUE.) or the step size adjustment may be very much delayed (BKOMMA = .FALSE.). If the initial step sizes are too small, there is only a slight chance of locating the global optimum in the presence of several local ones. In general, the optimum overall step sizes vary with the number N of parameters as C/SQRT(N), so the individual standard deviations vary as C/N with C = const.
P	(one dimensional real array of length NP)	Holds the positional angles of the ellipsoid for a descendant. The user must supply initial values if BKORRL = .TRUE. has been selected. If no better values are known initially, one can set P(I) = ATAN(1.0) for all I = 1(1)NP.
Y	(one dimensional real array of length NY)	Holds the vectors X, S, P, and the objective function values for the parents of the current generation and the next generation as well.
ZIELFU	(real function)	Name of the objective function, to be programmed by the user.
RESTRI	(real function)	Name of the function for evaluating all constraints, to be programmed by the user.
GAUSSN	(real function)	Name of the function used in transforming a uniform random number distribution to a Gaussian one.

GLEICH	(real function)	Name of the function for generating uniformly distributed random numbers.
TKONTR	(real function)	Name of the run-time monitoring function.
KANAL	(integer)	Channel number for output, relates only to messages output by subroutine PRUEFG concerning formal errors detected in the parameter list of subroutine KORR when the latter is called.

3. Method

KORR is a development from EVOL, Rechenberg's two membered strategy, and GRUP, the older version of Schwefel's multimembered evolution strategy. The method is based on a very much simplified simulation of biological evolution. See I. Rechenberg, Evolution Strategy: Optimization of Technical Systems in Accordance with the Principles of Biological Evolution (in German), vol. 15 of Problemata series, Verlag Frommann-Holzboog, Stuttgart, 1973; also H.-P. Schwefel, Numerical Optimization of Computer Models, Wiley, Chichester, 1981 (translated by M. W. Finnis from Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie, vol. 26 of Interdisciplinary Systems Research, Birkhäuser, Basle, Switzerland, 1977).

The IELTER parameter vectors are used to generate NACHKO new ones by introducing small normally distributed random changes. The IELTER best of these are used as starting points for the next generation (iteration). At the same time the strategy parameters are altered. These are the parameters of the current normal distributions for the lengths of the principal axes (standard deviations = step sizes) and the angular position of the mutation ellipsoid in N-dimensional space. Selection results in adaptation to local topology if the ratio NACHKO/IELTER is set large enough, e.g., at least 6. The random variations in the angles are produced by the addition of normally distributed random numbers, while those in the step sizes are produced from random numbers with a log-normal distribution by multiplication.

4. Convergence criterion

The termination criterion is based on value differences in the objective function, see under KONVKR, EPSILO(3), and EPSILO(4).

5. Peripheral I/O

Input:	none.
Output:	via channel KANAL, but only if there are formal errors in the parameter list of KORR. See under KANAL.

6. Notes

The two membered strategy (EVOL) usually has the shortest run time of all these evolution strategies (the EVOL, GRUP, and KORR codings so far developed) because ordinary (serial) computers can test the descendants only one after another in a generation, whereas in nature they are tested in parallel. The run times of the multimembered strategies increase less rapidly than in proportion to NACHKO because the convergence rate taken

over the generations tends to increase with NACHKO. However, there are frequent instances where even the simpler multimembered scheme (GRUP) has a run time less than that of EVOL because GRUP and KORR in principle allow one to adapt the step sizes individually to the local topology, which is not possible with EVOL, and this permits one to scale the variables in a flexible fashion. For this reason, the reliability and attainable accuracy are appreciably better than those given by EVOL.

The new KORR program represents further improvements on GRUP in this respect on account of the increased flexibility in the mutation ellipsoid, which improves the variability of the object variables. In addition to the lengths of the principal axes (standard deviations = step sizes) the positions of the principal axes in N-dimensional space are strategy parameters that are adjustable within the population. This together with the scaling provides directional adaptation to any valleys or ridges in the objective function surface. The changes in the object variables are no longer independent but linearly correlated, and this improves the convergence rate (with respect to the number of generations) quite appreciably in many instances. In special cases, however, there may be an increase in the run time arising from the storage and modification of the positional angles, and also from coordinate transformation. KORR enables the user to test how many strategy parameters (=degrees of freedom in the mutation ellipsoid) may be adequate to solve his special problem. The correlation can be suppressed completely, in which case KORR becomes equivalent to GRUP. Intermediary stages can be implemented by means of the NS parameter, the number of mutually independent step sizes. For example, for NS = 2 < N we have a hyperellipsoid of rotation with N-NS rotation axes. KORR differs from the older EVOL and GRUP in being divided into numerous small subroutines. This modular structure is disadvantageous as regards core requirement and run time, but it provides insight into the mode of operation of the program as a whole, so that it is easier for the user to modify the algorithm.

Although KORR in general allows one to improve the reliability of the optimum search there are still two critical situations. Minima at the boundary of the feasible region or in a vertex are attained only slowly or inaccurately. In any case, certainty of global convergence cannot be guaranteed; however, numerical tests have shown that the multimembered strategy is far better than other search procedures in this respect. It is capable of handling separated feasible regions provided that the number of parameters is not large and that the initial step sizes are set suitably large. In doubtful cases it is recommended to repeat the search each time with a different set of initial values and/or random numbers.

7. Subroutines or functions used

```
------------------------------------------------------
     SUBROUTINES: PRUEFG, SPEICH, MUTATI, DREHNG, UMSPEI,
                  MINMAX, GNPOOL, ABSCHA
     FUNCTIONS  : ZIELFU, RESTRI, GAUSSN, GLEICH, TKONTR,
                  ZULASS, BLETAL
------------------------------------------------------
```

The segment that calls KORR should have the name of the functions ZIELFU, RESTRI,

GLEICH, and TKONTR declared as external. This applies also to the name of any function used instead of GAUSSN to convert a uniform distribution to a normal one.

ZIELFU Objective function, to be programmed by the user in the form:

```
------------------------------------------------
      FUNCTION ZIELFU(N,X)
      DIMENSION X(N)
      ...
      ZIELFU=...
      RETURN
      END
------------------------------------------------
```

N represents the number of parameters, and X represents the formal parameter vector. The actual values are supplied by KORR. The function should be written on the basis that KORR searches for a minimum; if a maximum is to be sought, F must be supplied with a negative sign.

RESTRI Constraints function, to be programmed by the user in the general style:

```
------------------------------------------------
      FUNCTION RESTRI(J,N,X)
      DIMENSION X(N)
      GOTO(1,2,3,...,(M)),J
1     RESTRI=...
      RETURN
2     RESTRI=...
      RETURN
      ...
      ...
(M)   RESTRI=...
      RETURN
      END
------------------------------------------------
```

N and X have the meanings described for the objective function, while J (integer) is the serial number of the constraint. The statements should be written on the basis that KORR will accept vector X as feasible if RESTRI ≥ 0.0 for all J = 1(1)M.

TKONTR The function for monitoring the computation time may be defined by the user or called from the subroutine library in the particular machine. The following structure is assumed:

```
REAL FUNCTION TKONTR(D)
```

where D is a dummy parameter. TKONTR should be assigned the monitored quantity, e.g., the CPU time in seconds limited by TGRENZ. Many computers are supplied with

ready-made timing software. If this is given as a function, only its name needs to be supplied to KORR instead of TKONTR as a parameter.

GLEICH Function for generating a uniform random number distribution in the range (0,1]. The structure must be:

```
REAL FUNCTION GLEICH(D)
```

where D is arbitrary. GLEICH is the value of the random number. The standard library usually includes a suitable program, in which case only the appropriate name has to be supplied to KORR. The other subroutines and functions are explained briefly in the program itself.

```
------------------------------------------------------------
      SUBROUTINE KORR
     1(IELTER,BKOMMA,NACHKO,IREKOM,BKORRL,KONVKR,IFALLK,
     2TGRENZ,EPSILO,DELTAS,DELTAI,DELTAP,N,M,NS,NP,NY,
     3ZSTERN,XSTERN,ZBEST,X,S,P,Y,ZIELFU,RESTRI,GAUSSN,
     4GLEICH,TKONTR,KANAL)
      LOGICAL BKOMMA,BKORRL,BFATAL,BKONVG,BLETAL
      DIMENSION EPSILO(4),XSTERN(N),X(N),S(NS),P(NP),
     1Y(NY)
      COMMON/PIDATA/PIHALB,PIEINS,PIZWEI
      EXTERNAL RESTRI,GAUSSN,GLEICH
      IREKOX =  IREKOM / 100
      IREKOS = (IREKOM - IREKOX*100) / 10
      IREKOP =  IREKOM - IREKOX*100  - IREKOS*10
      D = 0.
      CALL PRUEFG
     1(IELTER,BKOMMA,NACHKO,IREKOM,BKORRL,KONVKR,TGRENZ,
     2EPSILO,DELTAS,DELTAI,DELTAP,N,M,NS,NP,NY,KANAL,
     3BFATAL)
C
C   CHECK INPUT PARAMETERS FOR FORMAL ERRORS.
C
      IF(BFATAL) RETURN
C
C   PREPARE AUXILIARY  QUANTITIES. TIMING  MONITORED IN
C   ACCORDANCE  WITH  THE  TKONTR  FUNCTION  FROM  HERE
C   ONWARDS.
C
      TMAXIM=TGRENZ+TKONTR(D)
      IF(.NOT.BKORRL) GOTO 1
      PIHALB=2.*ATAN(1.)
      PIEINS=PIHALB+PIHALB
      PIZWEI=PIEINS+PIEINS
```

```
      NL=1+N-NS
      NM=N-1
  1   NZ=NY/(IELTER+IELTER)
      IF(M.EQ.0) GOTO 2
C
C   CHECK FEASIBILITY OF INITIAL VECTOR XSTERN.
C
      IFALLK=-1
      ZSTERN=ZULASS(N,M,XSTERN,RESTRI)
      IF(ZSTERN.GT.0.) GOTO 3
  2   IFALLK=1
      ZSTERN=ZIELFU(N,XSTERN)
  3   CALL SPEICH
     1(0,BKORRL,EPSILO,N,NS,NP,NY,ZSTERN,XSTERN,S,P,Y)
C
C   THE INITIAL VALUES SUPPLIED  BY THE USER ARE STORED
C   IN FIELD Y AS THE DATA OF THE FIRST PARENT.
C
      IF(KONVKR.GT.1) Z1=ZSTERN
      ZBEST=ZSTERN
      LBEST=0
      IF(IELTER.EQ.1) GOTO 16
      DSMAXI=DELTAS
      DPMAXI=AMIN1(DELTAP*10.,PIHALB)
      DO 14 L=2,IELTER
C
C   IF IELTER > 1, THE OTHER IELTER - 1 INITIAL VECTORS
C   ARE DERIVED FROM THE VECTOR FOR THE FIRST PARENT BY
C   MUTATION (WITHOUT SELECTION). THE STRATEGY
C   PARAMETERS ARE WIDELY SPREAD.
C
      DO 4 I=1,NS
  4   S(I)=Y(N+I)
  5   IF(TKONTR(D).LT.TMAXIM) GOTO 501
      IFALLK=-3
      GOTO 42
501   IF(.NOT.BKORRL) GOTO 7
      DO 6 I=1,NP
  6   P(I)=Y(N+NS+I)
  7   CALL MUTATI
     1(NL,NM,BKORRL,DSMAXI,DELTAI,DPMAXI,N,NS,NP,X,S,P,
     2GAUSSN,GLEICH)
C
C   MUTATION IN ALL OBJECT AND STRATEGY PARAMETERS.
```

```
C
      DO 8 I=1,N
  8   X(I)=X(I)+Y(I)
      IF(IFALLK.GT.0) GOTO 9
C
C   IF  THE  STARTING  POINT  IS  NOT  FEASIBLE,   EACH
C   MUTATION  IS  CHECKED  AT  ONCE  TO  SEE  WHETHER A
C   FEASIBLE VECTOR HAS  BEEN  FOUND. THE  SEARCH  ENDS
C   WITH IFALLK = 0 IF THIS IS SO.
C
      Z=ZULASS(N,M,X,RESTRI)
      IF(Z)40,40,12
  9   IF(M.EQ.0) GOTO 11
      IF(.NOT.BLETAL(N,M,X,RESTRI)) GOTO 11
C
C   IF  A  MUTATION  FROM  A  FEASIBLE  STARTING  POINT
C   RESULTS IN  A NON-FEASIBLE X VECTOR, THEN  THE STEP
C   SIZES ARE REDUCED (ON THE ASSUMPTION THAT THEY WERE
C   INITIALLY  TOO   LARGE)  IN  ORDER   TO  AVOID  THE
C   THE CONSUMPTION  OF EXCESSIVE TIME  IN DEFINING THE
C   THE FIRST PARENT GENERATION.
C
      DO 10 I=1,NS
 10   S(I)=S(I)*.5
      GOTO 5
 11   Z=ZIELFU(N,X)
 12   IF(Z.GT.ZBEST) GOTO 13
      ZBEST=Z
      LBEST=L-1
      DSMAXI=DSMAXI*ALOG(2.)
 13   CALL SPEICH
     1((L-1)*NZ,BKORRL,EPSILO,N,NS,NP,NY,Z,X,S,P,Y)
C
C   STORE PARENT DATA IN ARRAY Y.
C
      IF(KONVKR.GT.1) Z1=Z1+Z
 14   CONTINUE
C
C   THE  INITIAL  PARENT GENERATION  IS  NOW  COMPLETE.
C   ZSTERN AND XSTERN,  WHICH HOLD THE BEST VALUES, ARE
C   OVERWRITTEN  WHEN  AN IMPROVEMENT  OF  THE  INITIAL
C   SITUATION IS OBTAINED.
C
      IF(LBEST.EQ.0) GOTO 16
```

```
      ZSTERN=ZBEST
      K=LBEST*NZ
      DO 15 I=1,N
 15   XSTERN(I)=Y(K+I)
 16   L1=IELTER
      L2=0
      IF(KONVKR.GT.1) KONVZ=0
C
C   ALL INITIALIZATION STEPS COMPLETED AT THIS POINT.
C   EACH FRESH GENERATION NOW STARTS AT LABEL 17.
C
 17   L3=L2
      L2=L1
      L1=L3
      IF(M.GT.0) L3=0
      LMUTAT=0
C
C   LMUTAT IS THE MUTATION COUNTER WITHIN A GENERATION,
C   WHILE L3 IS  THE COUNTER FOR LETHAL  MUTATIONS WHEN
C   THE PROBLEM INVOLVES CONSTRAINTS.
C
      IF(BKOMMA) GOTO 18
C
C   IF BKOMMA=.FALSE.  HAS BEEN  SELECTED,  THE PARENTS
C   MUST BE INCORPORATED IN THE SELECTION. THE DATA FOR
C   THESE ARE  TRANSFERRED FROM  THE FIRST  (OR SECOND)
C   PART OF THE ARRAY Y TO  THE SECOND (OR FIRST) PART.
C   IN THIS  CASE  THE  WORST INDIVIDUAL  MUST  ALSO BE
C   KNOWN,  THIS  IS   REPLACED  BY  THE  FIRST  BETTER
C   DESCENDANT.
C
      CALL UMSPEI
     1(L1*NZ,L2*NZ,IELTER*NZ,NY,Y)
      CALL MINMAX
     1(-1.,L2,NZ,ZSCHL,LSCHL,IELTER,NY,Y)
C
C   THE GENERATION OF EACH DESCENDANT STARTS AT LABEL 18
C
 18   K1=L1+IELTER*GLEICH(D)
C
C   RANDOM CHOICE  OF A PARENT OR  OF A PAIR OF PARENTS
C   IN ACCORDANCE WITH THE  VALUE CHOSEN FOR IREKOM. IF
C   IREKOM=3 OR IREKOM=5, THE CHOICE OF PARENTS IS MADE
C   WITHIN GNPOOL.
```

```
C
      K2=L1+IELTER*GLEICH(D)
 19   CALL GNPOOL
     1(1,L1,K1,K2,NZ,N,IELTER,IREKOS,NS,NY,S,Y,GLEICH)
C
C   STEP SIZES  SUPPLIED  FOR  THE DESCENDANT  FROM THE
C   POOL OF GENES.
C
      IF(BKORRL) CALL GNPOOL
     1(2,L1,K1,K2,NZ,N+NS,IELTER,IREKOP,NP,NY,P,Y,GLEICH)
C
C   POSITIONAL  ANGLES OF  ELLIPSOID  SUPPLIED  FOR THE
C   DESCENDANT FROM THE POOL  OF GENES WHEN CORRELATION
C   IS REQUIRED.
C
      CALL MUTATI
     1(NL,NM,BKORRL,DELTAS,DELTAI,DELTAP,N,NS,NP,X,S,P,
     2GAUSSN,GLEICH)
C
C   CALL TO  MUTATION  SUBROUTINE  FOR  ALL  VARIABLES,
C   INCLUDING  POSSIBLY  COORDINATE  TRANSFORMATION.  S
C   (AND P)  ARE  ALREADY  THE  NEW  ATTRIBUTES  OF THE
C   DESCENDANT,  WHILE X REPRESENTS  THE CHANGES  TO BE
C   MADE IN THE OBJECT VARIABLES.
C
      CALL GNPOOL
     1(3,L1,K1,K2,NZ,0,IELTER,IREKOX,N,NY,X,Y,GLEICH)
C
C   OBJECT VARIABLES  SUPPLIED FOR  THE DESCENDANT FROM
C   THE POOL OF GENES  AND ADDITION OF THE MODIFICATION
C   VECTOR. X NOW  REPRESENTS  THE  NEW  STATE  OF  THE
C   DESCENDANT.
C
      LMUTAT=LMUTAT+1
      IF(IFALLK.GT.0) GOTO 20
C
C   EVALUATION OF THE AUXILIARY  OBJECTIVE FUNCTION FOR
C   THE SEARCH FOR A FEASIBLE VECTOR.
C
      Z=ZULASS(N,M,X,RESTRI)
      IF(Z)40,40,22
 20   IF(M.EQ.0) GOTO 21
C
C   CHECK FEASIBILITY  OF DESCENDANT.  IF THE RESULT IS
```

```
C   NEGATIVE  (LETHAL MUTATION),  THE  MUTATION  IS NOT
C   COUNTED AS REGARDS THE NACHKO PARAMETER.
C
      IF(.NOT.BLETAL(N,M,X,RESTRI)) GOTO 21
      IF(.NOT.BKOMMA) GOTO 25
      LMUTAT=LMUTAT-1
      L3=L3+1
      IF(L3.LT.NACHKO) GOTO 18
      L3=0
C
C   TIME CHECK MADE NOT ONLY  AFTER EACH GENERATION BUT
C   ALSO  AFTER  EVERY  NACHKO   LETHAL  MUTATIONS  FOR
C   CERTAINTY.
C
      IF(TKONTR(D).LT.TMAXIM) GOTO 18
      IFALLK=3
      GOTO 26
 21   Z=ZIELFU(N,X)
C
C   EVALUATION  OF  OBJECTIVE  FUNCTION  VALUE  FOR THE
C   DESCENDANT.
C
 22   IF(BKOMMA.AND.LMUTAT.LE.IELTER) GOTO 23
      IF(Z-ZSCHL)24,24,25
 23   LSCHL=L2+LMUTAT-1
 24   CALL SPEICH
     1(LSCHL*NZ,BKORRL,EPSILO,N,NS,NP,NY,Z,X,S,P,Y)
C
C   TRANSFER OF DATA  OF DESCENDANT  TO PART OF ARRAY Y
C   HOLDING THE PARENTS FOR THE NEXT GENERATION.
C
      IF(.NOT.BKOMMA.OR.LMUTAT.GE.IELTER) CALL MINMAX
     1(-1.,L2,NZ,ZSCHL,LSCHL,IELTER,NY,Y)
C
C   LOOK FOR THE  CURRENTLY WORST  INDIVIDUAL STORED IN
C   ARRAY Y WITHOUT CONSIDERING  THE PARENTS THAT STILL
C   CAN PRODUCE DESCENDANTS IN THIS GENERATION.
C
 25   IF(LMUTAT.LT.NACHKO) GOTO 18
C
C   END OF GENERATION.
C
 26   CALL MINMAX
     1(1.,L2,NZ,ZBEST,LBEST,IELTER,NY,Y)
```

```
C
C   LOOK FOR  THE  BEST  OF  THE  INDIVIDUALS  HELD  AS
C   PARENTS FOR THE NEXT  GENERATION. IF THIS IS BETTER
C   THAN ANY DESCENDANT  PREVIOUSLY GENERATED, THE DATA
C   ARE WRITTEN INTO ZSTERN AND XSTERN.
C
      IF(ZBEST.GT.ZSTERN) GOTO 28
      ZSTERN=ZBEST
      K=LBEST*NZ
      DO 27 I=1,N
 27   XSTERN(I)=Y(K+I)
 28   IF(IFALLK.EQ.3) GOTO 30
      Z2=0.
      K=L2*NZ
      DO 29 L=1,IELTER
      K=K+NZ
 29   Z2=Z2+Y(K)
      CALL ABSCHA
     1(IELTER,KONVKR,IFALLK,EPSILO,ZBEST,ZSCHL,Z1,Z2,
     2KONVZ,BKONVG)
C
C   TEST CONVERGENCE CRITERION.
C
      IF(BKONVG) GOTO 30
C
C   CHECK TIME ELAPSED.
C
      IF(TKONTR(D).LT.TMAXIM) GOTO 17
C
C   PREPARE  FINAL DATA  FOR  RETURN  FROM KORR  IF THE
C   STARTING POINT WAS FEASIBLE.
C
 30   K=LBEST*NZ
      DO 31 I=1,N
      K=K+1
 31   X(I)=Y(K)
      DO 32 I=1,NS
      K=K+1
 32   S(I)=Y(K)
      IF(.NOT.BKORRL) RETURN
      DO 33 I=1,NP
      K=K+1
 33   P(I)=Y(K)
      RETURN
```

```
C
C   PREPARE  FINAL DATA  FOR  RETURN  FROM KORR  IF THE
C   STARTING POINT WAS NOT FEASIBLE.
C
 40   DO 41 I=1,N
 41   XSTERN(I)=X(I)
      ZSTERN=ZIELFU(N,XSTERN)
      ZBEST=ZSTERN
      IFALLK=0
 42   RETURN
      END
```

Subroutine PRUEFG

PRUEFG checks the values given with the parameter list on calling KORR. If discrepancies are found, an attempt is made to eliminate them. If this is not possible, e.g., arrays required are not appropriately dimensioned, the search for the minimum is not initiated. Then PRUEFG outputs a message to the peripheral unit denoted by KANAL on the correction of the error or else a warning message. BFATAL supplies KORR with information on the outcome of the check as a Boolean value.

```
      SUBROUTINE PRUEFG
     1(IELTER,BKOMMA,NACHKO,IREKOM,BKORRL,KONVKR,TGRENZ,
     2EPSILO,DELTAS,DELTAI,DELTAP,N,M,NS,NP,NY,KANAL,
     3BFATAL)
      LOGICAL BKOMMA,BKORRL,BFATAL
      DIMENSION EPSILO(4)
      IREKOX =  IREKOM / 100
      IREKOS = (IREKOM - IREKOX*100) / 10
      IREKOP =  IREKOM - IREKOX*100  - IREKOS*10
100   FORMAT(1H ,' CORRECTION. IELTER > 0 . ASSUMED: 2 AND
     1 KONVKR = ',I5)
101   FORMAT(1H ,' CORRECTION. NACHKO > 0 . ASSUMED: ',I5)
102   FORMAT(1H ,' WARNING. BETTER VALUE NACHKO >= 6*IELTER')
103   FORMAT(1H ,' CORRECTION. IF BKOMMA = .TRUE., THEN
     1 NACHKO > IELTER . ASSUMED: ',I3)
1041  FORMAT(1H ,' CORRECTION. 0 < IREKOX < 6 . ASSUMED: 1')
1042  FORMAT(1H ,' CORRECTION. 0 < IREKOS < 6 . ASSUMED: 1')
1043  FORMAT(1H ,' CORRECTION. 0 < IREKOP < 6 . ASSUMED: 1')
105   FORMAT(1H ,' CORRECTION. IF IELTER = 1, THEN
     1 IREKOM = 111 . ASSUMED: 111')
106   FORMAT(1H ,' CORRECTION. IF N = 1 OR NS = 1, THEN
     1 BKORRL = .FALSE. . ASSUMED: .FALSE.')
107   FORMAT(1H ,' CORRECTION. KONVKR > 0 . ASSUMED: ',I5)
```

```
108   FORMAT(1H ,' CORRECTION. IF IELTER = 1, THEN
     1 KONVKR > 1 . ASSUMED: ',I5)
109   FORMAT(1H ,' CORRECTION. EPSILO(',I1,') > 0. .
     1 SIGN REVERSED')
110   FORMAT(1H ,' WARNING. EPSILO(',I1,') TOO SMALL.
     1 TREATED AS 0. .')
111   FORMAT(1H ,' CORRECTION. DELTAS >= 0. .
     1 SIGN REVERSED')
112   FORMAT(1H ,' WARNING. EXP(DELTAS) = 1.
     1 OVER-ALL STEP SIZE CONSTANT')
113   FORMAT(1H ,' CORRECTION. DELTAI >= 0. .
     1 SIGN REVERSED')
114   FORMAT(1H ,' WARNING. EXP(DELTAI) = 1.
     1 STEP-SIZE RELATIONS CONSTANT')
115   FORMAT(1H ,' CORRECTION. DELTAP >= 0. .
     1 SIGN REVERSED')
116   FORMAT(1H ,' WARNING. DELTAP = 0.
     1 CORRELATION REMAINS FIXED')
117   FORMAT(1H ,' WARNING. TGRENZ <= 0.
     1 ONE GENERATION TESTED')
118   FORMAT(1H ,' CORRECTION. M >= 0 . ASSUMED: 0')
119   FORMAT(1H ,' FATAL ERROR. N <= 0')
120   FORMAT(1H ,' FATAL ERROR. NS <= 0')
121   FORMAT(1H ,' FATAL ERROR. NP <= 0')
122   FORMAT(1H ,' CORRECTION. 1 <= NS <= N . ASSUMED: ',I5)
123   FORMAT(1H ,' CORRECTION. IF BKOORL = .FALSE., THEN
     1 NP = 1 . ASSUMED: 1')
124   FORMAT(1H ,' FATAL ERROR. NY < (N+NS+1)*IELTER*2')
125   FORMAT(1H ,' CORRECTION. NY = (N+NS+1)*IELTER*2 .
     1 ASSUMED: ',I5)
126   FORMAT(1H ,' FATAL ERROR. NP < N*(NS-1)-((NS-1)*NS)/2')
127   FORMAT(1H ,' CORRECTION. NP = N*(NS-1)-((NS-1)*NS)/2 .
     1 ASSUMED: ',I5)
128   FORMAT(1H ,' FATAL ERROR. NY < (N+NS+NP+1)*IELTER*2')
129   FORMAT(1H ,' CORRECTION. NY = (N+NS+NP+1)*IELTER*2 .
     1 ASSUMED: ',I5)
      BFATAL=.TRUE.
      IF(IELTER.GT.0) GOTO 1
      IELTER=2
      KONVKR=N+N
      WRITE(KANAL,100)KONVKR
1     IF(NACHKO.GT.0) GOTO 2
      NACHKO=6*IELTER
      WRITE(KANAL,101)NACHKO
```

```
2     IF(.NOT.BKOMMA.OR.NACHKO.GE.6*IELTER) GOTO 3
      WRITE(KANAL,102)
      IF(NACHKO.GT.IELTER) GOTO 3
      NACHKO=6*IELTER
      WRITE(KANAL,103)NACHKO
3     IF(IREKOX.GT.0.AND.IREKOX.LT.6) GOTO 301
      IREKOX=1
      WRITE(KANAL,1041)
301   IF(IREKOS.GT.0.AND.IREKOS.LT.6) GOTO 302
      IREKOS=1
      WRITE(KANAL,1042)
302   IF(IREKOP.GT.0.AND.IREKOP.LT.6) GOTO 4
      IREKOP=1
      WRITE(KANAL,1043)
4     IF(IREKOM.EQ.111.OR.IELTER.NE.1) GOTO 5
      IREKOM=111
      IREKOX=1
      IREKOS=1
      IREKOP=1
      WRITE(KANAL,105)
5     IF(.NOT.BKORRL.OR.(N.GT.1.AND.NS.GT.1)) GOTO 6
      BKORRL=.FALSE.
      WRITE(KANAL,106)
6     IF(KONVKR.GT.0) GOTO 7
      IF(IELTER.EQ.1) KONVKR=N+N
      IF(IELTER.GT.1) KONVKR=1
      WRITE(KANAL,107)KONVKR
      GOTO 8
7     IF(KONVKR.GT.1.OR.IELTER.GT.1) GOTO 8
      KONVKR=N+N
      WRITE(KANAL,108)KONVKR
8     DO 12 I=1,4
      IF(I.EQ.2.OR.I.EQ.4) GOTO 9
      IF(EPSILO(I))10,11,12
9     IF((1.+EPSILO(I))-1.)10,11,12
10    EPSILO(I)=-EPSILO(I)
      WRITE(KANAL,109)I
      GOTO 12
11    WRITE(KANAL,110)I
12    CONTINUE
      IF(EXP(DELTAS)-1.)13,14,15
13    DELTAS=-DELTAS
      WRITE(KANAL,111)
      GOTO 15
```

```
14    IF(EXP(DELTAI).NE.1.) GOTO 15
      WRITE(KANAL,112)
15    IF(EXP(DELTAI)-1.)16,17,18
16    DELTAI=-DELTAI
      WRITE(KANAL,113)
      GOTO 18
17    IF(IREKOS.GT.1.AND.EXP(DELTAS).GT.1.) GOTO 18
      WRITE(KANAL,114)
18    IF(.NOT.BKORRL) GOTO 21
      IF(DELTAP)19,20,21
19    DELTAP=-DELTAP
      WRITE(KANAL,115)
      GOTO 21
20    WRITE(KANAL,116)
21    IF(TGRENZ.GT.0.) GOTO 22
      WRITE(KANAL,117)
22    IF(M.GE.0) GOTO 23
      M=0
      WRITE(KANAL,118)
23    IF(N.GT.0) GOTO 24
      WRITE(KANAL,119)
      RETURN
24    IF(NS.GT.0) GOTO 25
      WRITE(KANAL,120)
      RETURN
25    IF(NP.GT.0) GOTO 26
      WRITE(KANAL,121)
      RETURN
26    IF(NS.LE.N) GOTO 27
      NS=N
      WRITE(KANAL,122)N
27    IF(BKORRL) GOTO 31
      IF(NP.EQ.1) GOTO 28
      NP=1
      WRITE(KANAL,123)
28    NYY=(N+NS+1)*IELTER*2
      IF(NY-NYY)29,37,30
29    WRITE(KANAL,124)
      RETURN
30    NY=NYY
      WRITE(KANAL,125)NY
      GOTO 37
31    NPP=N*(NS-1)-((NS-1)*NS)/2
      IF(NP-NPP)32,34,33
```

```
32    WRITE(KANAL,126)
      RETURN
33    NP=NPP
      WRITE(KANAL,127)NP
34    NYY=(N+NS+NP+1)*IELTER*2
      IF(NY-NYY)35,37,36
35    WRITE(KANAL,128)
      RETURN
36    NY=NYY
      WRITE(KANAL,129)NY
37    BFATAL=.FALSE.
      RETURN
      END
```

Function ZULASS

This function is required only if there are constraints. If the starting point does not lie in the feasible region, ZULASS generates an auxiliary objective function that is used to search for a feasible initial vector.

If ZULASS, the negative sum of the values for the functions representing constraints that have been violated, is zero, then X represents a feasible vector that can be used in restarting the search with KORR.

XX represents XSTERN or X.

```
      FUNCTION ZULASS
     1(N,M,XX,RESTRI)
      DIMENSION XX(N)
      ZULASS=0.
      DO 1 J=1,M
      R=RESTRI(J,N,XX)
      IF(R.LT.0.) ZULASS=ZULASS-R
1     CONTINUE
      RETURN
      END
```

Subroutine UMSPEI

UMSPEI is required only if BKOMMA = .FALSE., whereupon the parents in the source generation have to be subject to selection. UMSPEI transposes the data on the parents within array Y.

K1, K2, and KK are auxiliary quantities transmitted from KORR that define the number and addresses of the data to be transposed.

```
------------------------------------------------------
      SUBROUTINE UMSPEI
     1(K1,K2,KK,NY,Y)
      DIMENSION Y(NY)
      DO 1 K=1,KK
1     Y(K2+K)=Y(K1+K)
      RETURN
      END
------------------------------------------------------
```

Subroutine GNPOOL

GNPOOL supplies a set of variables for a descendant by drawing on the pool of parents taken together in accordance with the type of recombination selected. This subroutine is called once each for the object variables X, the strategy variables S, and possibly also P. To minimize storage demand, the changes in the object variables by mutation are added immediately (J = 3). In intermediary recombination for the positional angle (J = 2), a check must be made on the difference between the parental angles to establish suitable mean values. J = 1 denotes the case where step sizes are involved.

L1 denotes the part of the gene pool from which the parent data are to be drawn if IREKO = 3 or IREKO = 5. K1 denotes the parent selected by KORR whose data are to be used when IREKO = 1 (no recombination). K1 and K2 denote the two parents whose data are to be recombined if IREKO = 2 or IREKO = 4 has been selected.

NZ and NN are auxiliary quantities for deriving the addresses in array Y.

NX represents N or NS or NP, XX represents X or S or P, IREKO represents one of the digits of IREKOM, i.e., IREKOX or IREKOS or IREKOP.

```
------------------------------------------------------
      SUBROUTINE GNPOOL
     1(J,L1,K1,K2,NZ,NN,IELTER,IREKO,NX,NY,XX,Y,GLEICH)
      DIMENSION XX(NX),Y(NY)
      COMMON/PIDATA/PIHALB,PIEINS,PIZWEI
      EXTERNAL GLEICH
      IF(J.EQ.3) GOTO 11
      GOTO(1,1,1,7,9),IREKO
1     KI1=K1*NZ+NN
      IF(IREKO.GT.1) GOTO 3
      DO 2 I=1,NX
2     XX(I)=Y(KI1+I)
      RETURN
3     KI2=K2*NZ+NN
      IF(IREKO.EQ.3) GOTO 5
```

```
      DO 4 I=1,NX
      KI=KI1
      IF(GLEICH(D).GE..5) KI=KI2
4     XX(I)=Y(KI+I)
      RETURN
5     DO 6 I=1,NX
      XX1=Y(KI1+I)
      XX2=Y(KI2+I)
      XXI=(XX1+XX2)*.5
      IF(J.EQ.1) GOTO 6
      DXX=XX1-XX2
      IF(ABS(DXX).GT.PIEINS) XXI=XXI+SIGN(PIEINS,DXX)
6     XX(I)=XXI
      RETURN
7     DO 8 I=1,NX
8     XX(I)=Y((L1+IFIX(IELTER*GLEICH(D)))*NZ+NN+I)
      RETURN
9     DO 10 I=1,NX
      XX1=Y((L1+IFIX(IELTER*GLEICH(D)))*NZ+NN+I)
      XX2=Y((L1+IFIX(IELTER*GLEICH(D)))*NZ+NN+I)
      XXI=(XX1+XX2)*.5
      IF(J.EQ.1) GOTO 10
      DXX=XX1-XX2
      IF(ABS(DXX).GT.PIEINS) XXI=XXI+SIGN(PIEINS,DXX)
10    XX(I)=XXI
      RETURN
11    GOTO(12,12,12,18,20),IREKO
12    KI1=K1*NZ+NN
      IF(IREKO.GT.1) GOTO 14
      DO 13 I=1,NX
13    XX(I)=XX(I)+Y(KI1+I)
      RETURN
14    KI2=K2*NZ+NN
      IF(IREKO.EQ.3) GOTO 16
      DO 15 I=1,NX
      KI=KI1
      IF(GLEICH(D).GE..5) KI=KI2
15    XX(I)=XX(I)+Y(KI+I)
      RETURN
16    DO 17 I=1,NX
17    XX(I)=XX(I)+(Y(KI1+I)+Y(KI2+I))*.5
      RETURN
18    DO 19 I=1,NX
19    XX(I)=XX(I)+Y((L1+IFIX(IELTER*GLEICH(D)))*NZ+NN+I)
```

```
      RETURN
20    DO 21 I=1,NX
21    XX(I)=XX(I)+(Y((L1+IFIX(IELTER*GLEICH(D)))*NZ+NN+I)
     1+Y((L1+IFIX(IELTER*GLEICH(D)))*NZ+NN+I))*.5
      RETURN
      END
-------------------------------------------------------
```

Subroutine SPEICH

SPEICH transfers to the data pool Y for the parents of the next generation the data of a descendant representing a successful mutation (the object variables X and the strategy parameters S (and P, if used) together with the corresponding value of the objective function). A check is made that S (and P) fall within specified bounds.

J is the address in array Y from which point onwards the data are to be written and is provided by KORR.

ZZ represents ZSTERN or Z, XX represents XSTERN or X.

```
-------------------------------------------------------
      SUBROUTINE SPEICH
     1(J,BKORRL,EPSILO,N,NS,NP,NY,ZZ,XX,S,P,Y)
      LOGICAL BKORRL
      DIMENSION EPSILO(4),XX(N),S(NS),P(NP),Y(NY)
      COMMON/PIDATA/PIHALB,PIEINS,PIZWEI
      K=J
      DO 1 I=1,N
      K=K+1
1     Y(K)=XX(I)
      DO 2 I=1,NS
      K=K+1
2     Y(K)=AMAX1(S(I),EPSILO(1))
      IF(.NOT.BKORRL) GOTO 4
      DO 3 I=1,NP
      K=K+1
      PI=P(I)
      IF(ABS(PI).GT.PIEINS) PI=PI-SIGN(PIZWEI,PI)
3     Y(K)=PI
4     K=K+1
      Y(K)=ZZ
      RETURN
      END
-------------------------------------------------------
```

Subroutine MINMAX

MINMAX searches for the smallest or largest value in a series of values of the objective function held in an array. KORR calls this subroutine to determine the best or worst parent, in the first case in order to transfer its data to the location ZBEST (and perhaps also ZSTERN and XSTERN) and in the other case in order to give space for a better descendant. C=1.0 initiates a search for the best (smallest) value of the function, while C=−1.0 does the same for the worst (largest) value.

LL and NZ are auxiliary quantities used to transmit information on the position of the required values within array Y. ZM and LM contain the best (or worst) values of the objective function and the number of the corresponding parent minus one.

```
------------------------------------------------------------
      SUBROUTINE MINMAX
     1(C,LL,NZ,ZM,LM,IELTER,NY,Y)
      DIMENSION Y(NY)
      LM=LL
      K1=LL*NZ+NZ
      ZM=Y(K1)
      IF(IELTER.EQ.1) RETURN
      K1=K1+NZ
      K2=(LL+IELTER)*NZ
      KM=LL
      DO 1 K=K1,K2,NZ
      KM=KM+1
      ZZ=Y(K)
      IF((ZZ-ZM)*C.GT.0.) GOTO 1
      ZM=ZZ
      LM=KM
1     CONTINUE
      RETURN
      END
------------------------------------------------------------
```

Subroutine ABSCHA

ABSCHA tests the convergence criterion. If KONVKR = 1 has been selected, the difference between the objective function values representing the best and worst parents (ZBEST and ZSCHL) must be less than the limits set by EPSILO(3) (absolute) or EPSILO(4) (relative). Then the assignment BKONVG = .TRUE. is made.

Alternatively, the current difference ZSCHL-ZBEST is replaced by the change Z1−Z2 in the sum of all the parent objective function values occurring after KONVKR generations divided by IELTER.

The Boolean variable BKONVG transmits the result of the convergence test to KORR.

KONVZ is the generation counter if KONVKR > 1.

```
-------------------------------------------------------
      SUBROUTINE ABSCHA
     1(IELTER,KONVKR,IFALLK,EPSILO,ZBEST,ZSCHL,Z1,Z2,
     2KONVZ,BKONVG)
      LOGICAL BKONVG
      DIMENSION EPSILO(4)
      IF(KONVKR.EQ.1) GOTO 1
      KONVZ=KONVZ+1
      IF(KONVZ.LT.KONVKR) GOTO 3
      KONVZ=0
      DELTAF=Z1-Z2
      Z1=Z2
      GOTO 2
1     DELTAF=(ZSCHL-ZBEST)*IELTER
2     IF(DELTAF.GT.EPSILO(3)*IELTER) GOTO 3
      IF(DELTAF.GT.EPSILO(4)*ABS(Z2)) GOTO 3
      IFALLK=ISIGN(2,IFALLK)
      BKONVG=.TRUE.
      RETURN
3     BKONVG=.FALSE.
      RETURN
      END
-------------------------------------------------------
```

Function GAUSSN

GAUSSN converts a uniform random number distribution to a normal one. The function has been programmed for the trapezium algorithm (J. H. Ahrens and U. Dieter, Computer Methods for Sampling from the Exponential and Normal Distributions, Communications of the Association for Computing Machinery, vol. 15 (1972), pp. 873-882 and 1047). The Box-Muller rules require in many cases (machine-dependent) a longer run time even if both of the pair of numbers can be used.

SIGMA is the standard deviation, which is multiplied by the random number derived from a (0.0,1.0) normal distribution.

```
-------------------------------------------------------
      FUNCTION GAUSSN
     1(SIGMA,GLEICH)
1     U=GLEICH(D)
      U0=GLEICH(D)
      IF(U.GE..919544406) GOTO 2
      X=2.40375766*(U0+U*.825339283)-2.11402808
      GOTO 10
```

```
2     IF(U.LT..965487131) GOTO 4
3     U1=GLEICH(D)
      Y=SQRT(4.46911474-2.*ALOG(U1))
      U2=GLEICH(D)
      IF(Y*U2.GT.2.11402808) GOTO 3
      GOTO 9
4     IF(U.LT..949990709) GOTO 6
5     U1=GLEICH(D)
      Y=1.84039875+U1*.273629336
      U2=GLEICH(D)
      IF(.398942280*EXP(-.5*Y*Y)-.443299126+Y*.209694057
     1.LT.U2*.0427025816) GOTO 5
      GOTO 9
6     IF(U.LT..925852334) GOTO 8
7     U1=GLEICH(D)
      Y=.289729574+U1*1.55066917
      U2=GLEICH(D)
      IF(.398942280*EXP(-.5*Y*Y)-.443299126+Y*.209694057
     1.LT.U2*.0159745227) GOTO 7
      GOTO 9
8     U1=GLEICH(D)
      Y=U1*.289729574
      U2=GLEICH(D)
      IF(.398942280*EXP(-.5*Y*Y)-.382544556
     1.LT.U2*.0163977244) GOTO 8
9     X=Y
      IF(U0.GE..5) X=-Y
10    GAUSSN=SIGMA*X
      RETURN
      END
------------------------------------------------------------
```

Subroutine DREHNG

DREHNG is called from MUTATI only if BKORRL = .TRUE. and N > 1. DREHNG performs the coordinate transformation of the modification vector for the object variables. Although the components of this vector are initially mutually independent, they are linearly related on account of the rotation specified by the positional angles P and so are correlated. The transformation involves NP partial rotations, in each of which only two of the components of the modification vector are involved.

```
------------------------------------------------------------
      SUBROUTINE DREHNG
     1(NL,NM,N,NP,X,P)
      DIMENSION X(N),P(NP)
      NQ=NP
      DO 1 II=NL,NM
      N1=N-II
      N2=N
      DO 1 I=1,II
      X1=X(N1)
      X2=X(N2)
      SI=SIN(P(NQ))
      CO=COS(P(NQ))
      X(N2)=X1*SI+X2*CO
      X(N1)=X1*CO-X2*SI
      N2=N2-1
1     NQ=NQ-1
      RETURN
      END
------------------------------------------------------------
```

Logical function BLETAL

BLETAL tests the feasibility of an object variable vector immediately on production if constraints are imposed. The first constraint to be violated causes BLETAL to signal to KORR via the function name (declared as a Boolean variable) that the mutation was lethal.

```
------------------------------------------------------------
      LOGICAL FUNCTION BLETAL
     1(N,M,X,RESTRI)
      DIMENSION X(N)
      DO 1 J=1,M
      IF(RESTRI(J,N,X).LT.0.) GOTO 2
1     CONTINUE
      BLETAL=.FALSE.
      RETURN
2     BLETAL=.TRUE.
      RETURN
      END
------------------------------------------------------------
```

Subroutine MUTATI

MUTATI handles the random alteration of the strategy variables and the object variables. First, the step sizes are altered in accordance with the DELTAS and DELTAI

parameters by multiplication by two random factors with log-normal distributions. The resulting normal distribution is used in a random vector X that represents the changes in the object variables. If BKORRL = .TRUE. is set when KORR is called, i.e., linear correlation is required, the positional angle P is also mutated, with random numbers from a (0.0,DELTAP) normal distribution added to the original values. Also, DREHNG is called in that case to transform the vector of modifications to the object variable.

NL and NM are auxiliary quantities transmitted from KORR via MUTATI to DREHNG.

```
------------------------------------------------------------
      SUBROUTINE MUTATI
     1(NL,NM,BKORRL,DELTAS,DELTAI,DELTAP,N,NS,NP,X,S,P,
     2GAUSSN,GLEICH)
      LOGICAL BKORRL
      DIMENSION X(N),S(NS),P(NP)
      EXTERNAL GLEICH
      DS=GAUSSN(DELTAS,GLEICH)
      DO 1 I=1,NS
1     S(I)=S(I)*EXP(DS+GAUSSN(DELTAI,GLEICH))
      DO 2 I=1,N
2     X(I)=GAUSSN(S(MIN0(I,NS)),GLEICH)
      IF(.NOT.BKORRL) RETURN
      DO 3 I=1,NP
3     P(I)=P(I)+GAUSSN(DELTAP,GLEICH)
      CALL DREHNG
     1(NL,NM,N,NP,X,P)
      RETURN
      END
------------------------------------------------------------
```

Note

Without modifications the subroutines EVOL, GRUP, and KORR may be used to solve optimization problems with integer (or discrete) and mixed-integer variables. The search for an optimum then, however, will only lead into the vicinity of the exact solution.

The discreteness may be induced by the user when formulating the objective function, by merely rounding the correspondent variables to integers or by attributing discrete values to them.

The following two examples will give hints only to possible formulations. In order to get the results in the form wanted the variables will have to be transformed in the same manner at the end of the optimum search with EVOL, GRUP, or KORR, as is done within the objective function.

Example 1

Minimize

$$F(x) = \sum_{i=1}^{n} (x_i - i)$$

with $x_i \geq 0$, integer for all $i = 1(1)n$

```
      FUNCTION F(N,X)
      DIMENSION X(N)
      F=0.
      DO 1 I=1,N
      IX=IFIX(ABS(X(I)))
      XI=FLOAT(IX-I)
      F=F+XI*XI
 1    CONTINUE
      RETURN
      END
```

Example 2

Minimize

$$F(x) = (x_1 - 2)^2 + (x_1 - 2x_2)^2$$

with x_1 from $\{1.3, 1.5, 2.2, 2.8\}$ only

```
      FUNCTION F(N,X)
      DIMENSION X(N), Y(4)
      DATA Y /1.3,1.5,2.2,2.8/
      DO 1 I=1,4
      X1=Y(I)
      IF (X(1)-X1) 2,2,1
 1    CONTINUE
 2    F1=X1-2.
      F2=X1-X(2)-X(2)
      F =F1*F1+F2*F2
      RETURN
      END
```