4 Optimization

4.1 One-dimensional Minimization

The subroutine MINGUS (with second entry MINGAS) performs a one-dimensional minimization of a unimodular function $f(x)$ with a method combining the golden section and the parabolic approximation strategies. It can be used especially for a line search minimization during n-dimensional minimization by application to a function $f(x) \equiv F(\boldsymbol{p}_0 + x \cdot \Delta\boldsymbol{p})$, where $\boldsymbol{p}_0$ is an initial n-vector and $\Delta\boldsymbol{p}$ a step vector; minimization of the function $f(x)$ then corresponds to minimization of the function $F(\boldsymbol{p})$ along the line $\boldsymbol{p}_0 + x \cdot \Delta\boldsymbol{p}$ in n-dimensional space.

```
CALL MINGUS(FX,DX,RX)
```

Purpose: one-dimensional minimization of a function $f(x)$ in a loop (see example below)
FX : function value for the current value of the function argument x; at the end of the loop the final value of the minimization
DX : returned step factor for the change of the argument x
RX : returned flag (see below); for RX=0.0 another iteration is necessary

```
CALL MINGAS(XL,XR,TOL)
```

Purpose: optional definition of limits and tolerance parameter for MINGUS, to be called *before* the MINGUS-loop (see example below)
XL : specification of the left (lower) limit of the fuunction argument, relative to the start value and the step size
XR : specification of the right (upper) limit of the function argument
TOL : tolerance parameter MINGUS
MINGAS is a second entry of subroutine MINGUS. Default values, assumed if MINGAS is not called, are: XL = -5.0, XR = +5.0 and TOL= $1 \cdot 10^{-6}$.

Example: The code example below explains the use of MINGUS to find the minimum of the function $f(x) = 1 - \exp(-x)$, which is at $x = 1$. First the starting value of variable x is defined by X= 0.1; a fixed step size STEP is defined. The optional call of MINGAS is used to define lower and upper limits for the total steps; the limits are relative to the starting value and are measured in units of the value of STEP. The example below defines the limits -3.0 and $+5.5$, corresponding to a range of $0.1 - 3.0 \times 0.3 = -0.8$ to $0.1 + 5.5 \times 0.3 = 1.6$ for variable X. The last parameter (TOL) of the call to MINGAS defines the convergence tolerance: convergence is assumed if the approximate minimum is enclosed in an interval with function values at the interval boundaries less than TOL above the approximate minimum value. Note that the size of the x-interval depends on the function behaviour near the minimum.

```
      PARAMETER (TOL=1.0E-6)
      X   =0.1                    ! initial value of x
      STEP=0.3                    ! initial step value for changes of x
      CALL MINGAS(-3.0,+5.0,TOL)
   10 FUN=1.0-X*EXP(-X)           ! evaluation of function
      CALL MINGUS(FUN,DX,RX)      !
      X=X+DX*STEP                 ! modification of x
      IF(RX.EQ.0.0) GOTO 10       ! check for end of loop
```

The statements from statement number 10 to the IF-statement constitute the search loop. The function is evaluated at the current value of X and the function value FUN is transmitted to the subroutine MINGUS; inside this subroutine the function value is compared to previous function values. The returned value of DX multiplied by the (fixed) step STEP is added to the variable X. If the returned value of the argument RX is zero, the loop has to be executed another time.
For a *non-zero* value of RX the loop is left. If RX>0, then the search has converged and RX is the total size of the final search interval in X; in case of non-convergence (after 15 function evaluations) the variable RX is the total size of the last search interval in X with a minus-sign. In any case, the value

of variable `X` is the value at the minimal function value found during the loop, which is the value `FUN` returned at the last call.

4.2 The n-dimensional Minimization

A standard optimization method is the definition of an *objective Function* $F(\boldsymbol{p})$, depending on the parameter vector $\boldsymbol{p}$ or on n parameters p_1, p_2, $\ldots p_n$, and the determination of the minimum of the objective function with respect to the parameters. In the **Least Squares** method the objective function is the sum of weighted squares of deviations, often called the χ^2-function. In the **Maximum-Likelihood** Method a Likelihood function $L(\boldsymbol{p}$ is defined, usually as the product of values of density functions, depending on the parameter vector. The objective function $F(\boldsymbol{p}) = -(1/2) \times L(\boldsymbol{p})$ (factor 1/2 times the *negative* logarithm of the Likelihood function) has properties similar to the χ^2-function of Least Squares. In both cases the optimization is equivalent to the minimization of the objective function; information about the errors, or more general, the covariance matrix of the parameters, can be determined from the matrix of second derivatives of the objective function at the minimum.
Two subroutines are described in this section, which search for the minimum of the function $F(\boldsymbol{p})$, where the argument is a n-dimensional vector. Both minimization subroutines require a user function `FUNC(PAR,NULL)`, which returns for the value of the parameter vector $\boldsymbol{p}$ in array `PAR` the value of the objective function $F(\boldsymbol{p})$. The result of both minimization subroutines is the approximate value of the parameter vector at the minimum. The subroutine `MININ` in addition returns the covariance matrix of the parameter vector.
Subroutine `NELMED` is based on a simple algorithm of Nelder and Mead and does not makes use of any derivatives. It is robust and rather effective in a region far from the minimum, but slow in the task to define the minimum with high precision. No errors are provided by the method.
In subroutine `MININ` first and second derivatives of the objective function are calculated numerically. The derivatives are used in the determination of search directions and the final determination of the covariance matrix. Internally the subroutine `MINGUS` is used to search for the minimum along a search direction. The method should be effective especially in the final phase of the determination of the minimum with high precision. Internally subroutine `NELMED` is called in the initial phase of minimization.
The function `FUNC(PAR,NULL)` has to be declared as `EXTERNAL` in the program calling `MININ` or `NELMED`. The first argument is the array of current parameter values, and the function has to return the value of the objective function for these paramater values. It is recommended to use a `DOUBLE PRECISION` variable for forming sums within function `FUNC`.

```
FUNCTION FUNC(PAR,NULL)
REAL PAR(*)
DOUBLE PRECISION SUM
DO I = 1, ...
 ...
 SUM = SUM + ...
END DO
FUNC = SUM
END
```

The argument called `NULL` is an integer, which in calls by `MININ` and `NELMED` has always the value zero (`NULL = 0`). Within function `FUNC` the value of `NULL` may be used to perform initialization (for example for `NULL = 1`), and to print some result (for example for `NULL = 2`). The user then has to call the function outside the subroutines`MININ` and `NELMED`.
Both subroutines require initial values of the parameters. Minimization requires more time if the initial values are far from the minimum, and the methods may also fail for very *bad* initial values. In addition both subroutines require initial step values, which determine the size of the variation of the parameters in the initial phase of the minimization. Roughly the **step size** of a parameter should be of the order of the **standard deviation** of the parameter. After the initial phase the step size is automatically adapted to the behaviour of the function. The initial size of the step is more critical for the subroutine `MININ`, which calculates and uses *derivatives*. Sometimes the sensitive region of

parameters differ by order of magnitudes; therefore the initial step size is used in MININ as a *scaling* parameter. Bad values of step sizes may increase the time for the minimum determination; the method may even fail to find the minimum.

```
CALL MININ(NPAR,STEP,COV,NPAR,STOL,FUNC,NCMAX,NCYCLE,LUNP)
```

Purpose: Minimization of the function FUNC(PAR,0) for fitting applications with determination of the covariance matrix

PAR(.) : array of the parameter vector $\boldsymbol{p}$

STEP(.) : array of the parameter step vector $\Delta\boldsymbol{p}$

COV(.) : covariance matrix in symmetric storage mode

NPAR : number of elements of the parameter vector

STOL : tolerance parameter for convergence recognition; a value STOL $= \kappa^2$ corresponds roughly to the requirement that the minimum found has a distance of κ standard deviations from the true minimum. A choice STOL$= 0.01$ to $1.E-4$ should be safe in most cases. A value likeSTOL$= 1.0E-8$ may be too small to reach convergence, due to round-off errors.

FUNC : name of function subprogram, which has to be declared as EXTERNAL in the calling program

NCMAX : maximum allowed number of minimization cycles (the value 10 is used in case of NCMAX ¡ 10).

NCYCLE : number of cycles actually used (returned). The value is **negative** in case of **non-convergence**!

LUNP : unit number for printout; for LUNP = 0 the standard output unit is used, for LUNP < 0 or > 100 printout is suppressed.

The subroutine MININ searches for the minimum of a function FUNC, depending on NPAR parameters. The function value is assumed to be the χ^2 of data with respect to a fit expression, or -2 times a log-likelihood function. The arguments PAR and STEP are arrays of NPAR words of initial parameter values and steps; at return the array PAR contains the fitted parameter values. The covariance matrix (COV) for the parameters is determined from the matrix of second derivatives (Hessian).

The method used in subroutine MININ is the following. Within one cycle the gradient and the matrix of second derivatives is calculated from finite differences, and a line search along the Newton direction is performed; the Hessian is rotated to a diagonal matrix. Calculation of derivatives is done in the rotated system, where the correlations between the parameters are zero or small. In case of difficult behaviour of the function a minimization is in addition performed using the eigenvectors as search directions (column vectors of the rotation matrix). The method is rather safe and efficient close to the minimum, but could need larger number of function evaluations far from the minimum. Therefore the simpler method of Nelder and Mead (subroutine NELMED) is used initially for an improvement of the initial parameter values.

```
      REAL PA(3),ST(3),COV(6)
      SAVE
      EXTERNAL FUNC04
*     ...
      WRITE(*,*) 'Example of a typical fit, three parameter'
      PA(1)=100000.0
      PA(2)=0.005
      PA(3)=0.0006
      ST(1)=0.1*PA(1)
      ST(2)=0.1*PA(2)
      ST(3)=0.1*PA(3)
      CALL MININ (PA,ST,COV,3,0.001,FUNC04,30,NCYCLE,0)
      DUM=FUNC04(PA,2)
      END

      FUNCTION FUNC04(PA,NULL)
*     typical chi-square fit
```

```
      REAL PA(*)
      DOUBLE PRECISION SUM
      INTEGER XN(60)          ! array of measured counting rates
      DATA XN/029,099,154,214,230,225,238,240,261,264,258,247,257,
     +        250,240,251,240,201,211,215,188,186,175,194,182,147,
     +        162,136,144,165,147,166,121,134,120,111,127,114,109,
     +        104,102,075,100,087,102,069,063,063,064,076,073,079,
     +        068,074,069,057,060,064,047,053/
*     fit function = difference of two exponentials
      FUN(A1,A2,A3,T)=A1*A2*A3/(A3-A2)*(EXP(-A2*T)-EXP(-A3*T))
*     ...
      SUM=0.0D0
      IF(NULL.EQ.0) THEN
         DO I=1,60      ! chi square calculation
          FIT=FUN(PA(1),PA(2),PA(3),FLOAT(I)*10.0-5.0)
          SUM=SUM+(FIT-FLOAT(XN(I)))**2/FLOAT(XN(I)) ! chi square sum
         END DO
      ELSE
         WRITE(*,101)
         DO I=1,15      ! print out
          FIT1=FUN(PA(1),PA(2),PA(3),FLOAT(I   )*10.0-5.0)
          FIT2=FUN(PA(1),PA(2),PA(3),FLOAT(I+30)*10.0-5.0)
          WRITE(*,102) I,XN(I),FIT1, I+30,XN(I+30),FIT2
         END DO
      END IF
      FUNC04=SUM
 101  FORMAT(1X,2(11X,'i  measured   fitted'))
 102  FORMAT(1X,2(I12,I10,F9.1))
      END
```

Output of the program:

```
 Example of a typical fit, three parameter

 MININ start---------------------
 Cycle       Func         N(Func)
 start      5290.6              1
 smplx      78.674             55
     1      65.336             74
     2      64.628             93
     3      64.627            112
            64.627            132
 stol=     0.10000E-02

        Param         error       correlation coeff. (in %)
    1  0.10001E+06   1383.7
    2  0.23818E-01  0.14160E-02   40
    3  0.35874E-02  0.11301E-03  -62 -74
 MININ end-----------------------

           i  measured   fitted          i  measured   fitted
           1        29     39.9         31       147    141.1
           2        99    104.8         32       166    136.2
           3       154    153.3         33       121    131.5
           4       214    189.0         34       134    126.9
           5       230    214.8         35       120    122.4
```

6	225	232.8	36	111	118.1
7	238	244.7	37	127	114.0
8	240	252.0	38	114	110.0
9	261	255.6	39	109	106.1
10	264	256.5	40	104	102.4
11	258	255.2	41	102	98.8
12	247	252.3	42	75	95.3
13	257	248.2	43	100	91.9
14	250	243.3	44	87	88.7
15	240	237.7	45	102	85.6

```
CALL NELMED(PAR,STEP,NPAR,STOL,FUNC,NCMAX,NCYCLE)
```

Purpose: Search for the approximate minimum of a function of n parameters by the polyhedron or simplex method of Nelder and Mead

PAR(.) : array of the parameter vector $\boldsymbol{p}$
STEP(.) : array of the parameter step vector $\Delta\boldsymbol{p}$
NPAR : number of elements of the parameter vector
FUNC : name of function subprogram, which has to be declared as EXTERNAL in the calling program
STOL : tolerance parameter
NCMAX : maximum allowed number of cycles
NUMFUN : number of function evaluations (returned) actually used

The subroutine NELMED searches for the minimum of a function FUNC, depending on NPAR parameters, by the method of Nelder and Mead. The function FUNC has to be supplied and declared as EXTERNAL by the user.

The arguments of the subroutine, the function FUNC and the two steering parameters STOL and NCMAX, and the returned argument NCYCLE, have the same meaning as in subroutine MININ.

The arguments PAR and STEP are arrays of NPAR words of initial parameter values and steps; at return the array PAR contains the parameter values at the approximate minimum found by NELMED. The size of the steps (variations of the parameter value) should be meaningful; the number of function evaluations required to reach the approximate minimum depends on the size of the steps. Parameters corresponding to steps with a value of zero are *not* modified during the search. The argument STOL is the tolerance parameter for the convergence test: convergence is assumed if the standard deviation of all function values on the simplex is less than STOL. The argument NCMAX is the maximum number of cycles.

4.3 Parameter limits

Optionally lower and upper limits for parameters may be specified before the call of the minimization subroutines MININ and NELMED. This may be necessary to avoid function evaluations in certain ares of the parameter space. For example if the parameter PAR(2) has the meaning of a width of a function describing a peak, then zero or negative values of this parameter are not meaningful and cause arithmetic problems. In this case a small positive and a large positive value should be specified as lower and upper limit for PAR(2), for example by CALL PARLIM(2, 0.001,1000.0) before the call of NELMED or MININ. Limits can be specified for all or selected parameters. Within the minimization subroutines the initial parameter values are compared to the limits; all limits are *removed* if the initial parameter values violate the limits. At return from the minimization subroutines all limits are automatically removed.

The algorithms in MININ and in NELMED take the limits into account. In MININ line searches are limited by the given paremeter limits; in numerical differentiation for a parameter at a limit *one-sided* formulas are used. The algorithm in NELMED takes the limits into account, although parameter limits were not foreseen in the original Nelder and Mead algorithm. Eventually step sizes are reduced and the expansion of the simplex is limited.

The behaviour of the algorithm is not yet well-tuned for parameter values at the minimum exactly on one limit. Also the result of the error analysis in this case is still under test.

```
CALL PARLIM(I,PAR1,PAR2)
```

Purpose: Specification of lower (left) and upper (right) limit of the parameter with index `I`. The limits are active for the next call of `MININ` and `NELMED`. The special value `I = 0` has the effect to remove *all* limits. Calls with indices outside the range are ignored.

`I :` index of the parameter

`PAR1 :` lower (left) limit of the value of the parameter

`PAR2 :` upper (right) limit of the value of the parameter