Solve a general nonlinear programming problem using the successive quadratic programming algorithm and a finite difference gradient.

Usage

```
CALL NCONF (FCN, M, ME, N, XGUESS, IBTYPE, XLB, XUB,
            XSCALE, IPRINT, MAXITN, X, FVALUE)
```

Arguments

FCN — User-supplied SUBROUTINE to evaluate the functions at a given point. The usage is CALL FCN (M, ME, N, X, ACTIVE, F, G), where

M — Total number of constraints. (Input)

ME — Number of equality constraints. (Input)

N — Number of variables. (Input)

X — The point at which the functions are evaluated. (Input)
X should not be changed by FCN.

ACTIVE — Logical vector of length MMAX indicating the active constraints. (Input) MMAX = MAX(1, M)

F — The computed function value at the point X. (Output)

G — Vector of length MMAX containing the values of constraints at point X. (Output)

FCN must be declared EXTERNAL in the calling program.

M — Total number of constraints. (Input)

ME — Number of equality constraints. (Input)

N — Number of variables. (Input)

XGUESS — Vector of length N containing an initial guess of the computed solution. (Input)

IBTYPE — Scalar indicating the types of bounds on variables. (Input)

IBTYPE	Action
0	User will supply all the bounds.
1	All variables are nonnegative.
2	All variables are nonpositive.
3	User supplies only the bounds on 1st variable; all other variables will have the same bounds.

XLB — Vector of length N containing the lower bounds on variables. (Input, if IBTYPE = 0; output, if IBTYPE = 1 or 2; input/output, if IBTYPE = 3)
If there is no lower bound for a variable, then the corresponding XLB value should be set to −1.0E6.

XUB — Vector of length N containing the upper bounds on variables. (Input, if IBTYPE = 0; output, if IBTYPE = 1 or 2; input/output, if IBTYPE = 3)
If there is no upper bound for a variable, then the corresponding XLB value should be set to 1.0E6.

XSCALE — Vector of length N containing the diagonal scaling matrix for the variables. (Input)
All values of XSCALE must be greater than zero. In the absence of other information, set all entries to 1.0.

IPRINT — Parameter indicating the desired output level. (Input)

IPRINT	Action
0	No output printed.
1	Only a final convergence analysis is given.
2	One line of intermediate results are printed in each iteration.
3	Detailed information is printed in each iteration.

MAXITN — Maximum number of iterations allowed. (Input)

X — Vector of length N containing the computed solution. (Output)

FVALUE — Scalar containing the value of the objective function at the computed solution. (Output)

Comments

1. Automatic workspace usage is

NCONF N * (3 * N + 38 + MMAX) + 7 * MMAX + 6 * M + MAX(N, M) + 91 units, or

DNCONF 2 * N * (3 * N + 38 + MMAX) + 14 * MMAX + 12 * M + MAX(N, M) + 163 units.
MMAX = MAX(1, M)

Workspace may be explicitly provided, if desired, by use of N2ONF/DN2ONF. The reference is

```
CALL N2ONF (FCN, M, ME, N, XGUESS, IBTYPE, XLB, XUB,
            XSCALE, IPRINT, MAXITN, X, FVALUE, WK,
            LWK, IWK, LIWK, CONWK)
```

The additional arguments are as follows:

WK — Work vector of length N * (3 * N + 38 + MMAX) + 6 * MMAX + 6 * M + 72

LWK — Length of WK.

IWK — Work vector of length 19 + MAX(M, N).

LIWK — Length of LIWK.

CONWK — Work vector of length MMAX.

2. Informational errors

Type	Code	
4	1	Search direction uphill.
4	2	Line search took more than 5 function calls.
4	3	Maximum number of iterations exceeded.
4	4	Search direction is close to zero.
4	5	The constraints for the QP subproblem are inconsistent.

3. If reverse communication is desired, then N0ONF/DN0ONF can be used rather than using an external subroutine to evaluate the function. The reference is

```
CALL N0ONF (IDO, M, ME, N, IBTYPE, XLB, XUB, IPRINT,
            MAXITN, X, FVALUE, G, DF, DG, LDDG, U,
            C, LDC, D, ACC, SCBOU, MAXFUN, ACTIVE,
            MODE, WK, IWK, CONWK)
```

The additional arguments are as follows:

IDO — Reverse communication parameter indicating task to be done. (Input/Output)

On the initial call, IDO must be set to 0; and initial values must be passed into N0ONF/DN0ONF for X, FVALUE, G, DF, and DG. If the routine returns with IDO = 1, then the user has to compute FVALUE and G with respect to X. If the routine returns with IDO = 2, then the user has to compute DF and DG with respect to X. The user has to call the routine repeatedly until IDO does not equal to 1 or 2.

X — Vector of length N containing the initial guesses to the solution on input and the solution on output. (Input/Output)

FVALUE — Scalar containing the objective function value evaluated at the current X. (Input)

G — Vector of length MMAX containing constraint values at the current X. (Input)
MMAX is MAX(1, M).

DF — Vector of length N containing the gradient of the objective function evaluated at the current X. (Input)

DG — Array of dimension MMAX by N containing the gradient of the constraints evaluated at the current X. (Input)

LDDG — Leading dimension of DG exactly as specified in the dimension statement of the calling program. (Input)

U — Vector of length M + N + N + 2 containing the multipliers of the nonlinear constraints and of the bounds. (Output)
The first M locations contain the multipliers for the nonlinear constraints. The second N locations contain the multipliers for the lower bounds. The third N locations contain the multipliers for the upper bounds.

C — Array of dimension N + 1 by N + 1 containing the final approximation to the Hessian. (Output)

LDC — Leading dimension of C exactly as specified in the dimension statement of the calling program. (Input)

D — Vector of length N + 1 containing the diagonal elements of the Hessian. (Output)

ACC — Final accuracy. (Input)

SCBOU — Scalar containing the scaling variable for the problem function. (Input)
In the absence of further information, SCBOU may be set to 1.0E3.

MAXFUN — Scalar containing the maximum allowable function calls during the line search. (Input)

ACTIVE — Logical vector of length 2 * MMAX + 13. (Input/Output)
The first MMAX locations are used to determine which gradient constraints are active and must be set to .TRUE. on input. If ACTIVE(I) is .TRUE., then DG(I, K) is evaluated for K = 1, N. The last MMAX + 13 locations are used for workspace.

MODE — Desired solving version for the algorithm. (Input)
If MODE = 2; then reverse communication is used. If MODE = 3; then reverse communication is used and initial guesses for the multipliers and Hessian matrix of the Lagrange function are provided on input.

WK — Work vector of length 2 * N * (N + 16) + 4 * MMAX + 5 * M + 68.

IWK — Work vector of length 19 + MAX(M, N).

CONWK — Work vector of length M.

Algorithm

The routine NCONF is based on subroutine NLPQL, a FORTRAN code developed by Schittkowski (1986). It uses a successive quadratic programming method to solve the general nonlinear programming problem. The problem is stated as follows:

$$\min_{x \in R^n} f(x)$$

$$\text{subject to} \quad g_j(x) = 0, \text{ for} \quad j = 1, \ldots, m_e$$

$$g_j(x) \geq 0, \text{ for} \quad j = m_e + 1, \ldots, m$$

$$x_l \leq x \leq x_u$$

where all problem functions are assumed to be continuously differentiable. The method, based on the iterative formulation and solution of quadratic programming subproblems, obtains subproblems by using a quadratic approximation of the Lagrangian and by linearizing the constraints. That is,

$$\min_{d \in R^n} \frac{1}{2} d^T B_k d + \nabla f(x_k)^T d$$

$$\text{subject to} \quad \nabla g_j(x_k)^T d + g_j(x_k) = 0, \quad j = 1, \ldots, m_e$$

$$\nabla g_j(x_k)^T d + g_j(x_k) \geq 0, \quad j = m_e + 1, \ldots, m$$

$$x_l - x_k \leq d \leq x_u - x_k$$

where *B*k is a positive definite approximation of the Hessian and *x*k is the current iterate. Let *d*k be the solution of the subproblem. A line search is used to find a new point *x*k+1,

*x*k+1= *x*k + λ*d*k, $\lambda \in (0, 1]$

such that a "merit function" will have a lower function value at the new point. Here, the augmented Lagrange function (Schittkowski 1986) is used as the merit function.

When optimality is not achieved, *B*k is updated according to the modified BFGS formula (Powell 1978). Note that this algorithm may generate infeasible points during the solution process. Therefore, if feasibility must be maintained for intermediate points, then this routine may not be suitable. For more theoretical and practical details, see Stoer (1985), Schittkowski (1983, 1986) and Gill et al. (1985).

Since a finite-difference method is used to estimate the gradient for some single precision calculations, an inaccurate estimate of the gradient may cause the algorithm to terminate at a noncritical point. In such cases, high precision arithmetic is recommended. Also, whenever the exact gradient can be easily provided, routine NCONG should be used instead.

Within NCONF, there is a user-callable subroutine N0ONF that gives the user the option to use "reverse communication." This option allows the user to evaluate the functions and gradients in the main program. This option is useful when it is difficult to do the function evaluation in the fixed form required by NCONF.

Example 1

The problem

$$\min F(x) = (x_1 - 2)^2 + (x_2 - 1)^2$$

$$\text{subject to} \quad g_1(x) = x_1 - 2x_2 + 1 = 0$$

$$g_2(x) = -\left(x_1^2\right)/4 - x_2^2 + 1 \geq 0$$

is solved with an initial guess (2.0, 2.0).

```
      INTEGER    IBTYPE, IPRINT, M, MAXITN, ME, N
      PARAMETER  (IBTYPE=0, IPRINT=0, M=2, MAXITN=100, ME=1, N=2)
C
      REAL       FVALUE, X(N), XGUESS(N), XLB(N), XSCALE(N), XUB(N)
      EXTERNAL   FCN, NCONF, WRRRN
C
      DATA XGUESS/2.0E0, 2.0E0/, XSCALE/2*1.0E0/
      DATA XLB/-1.0E6, -1.0E6/, XUB/1.0E6, 1.0E6/
C
      CALL NCONF (FCN, M, ME, N, XGUESS, IBTYPE, XLB, XUB, XSCALE,
     &            IPRINT, MAXITN, X, FVALUE)
C
      CALL WRRRN ('The solution is', N, 1, X, N, 0)
      END
C
      SUBROUTINE FCN (M, ME, N, X, ACTIVE, F, G)
      INTEGER    M, ME, N
      REAL       X(*), F, G(*)
      LOGICAL    ACTIVE(*)
C                                Himmelblau problem 1
      F = (X(1)-2.0E0)**2 + (X(2)-1.0E0)**2
C
      IF (ACTIVE(1)) G(1) = X(1) - 2.0E0*X(2) + 1.0E0
      IF (ACTIVE(2)) G(2) = -(X(1)**2)/4.0E0 - X(2)**2 + 1.0E0
      RETURN
      END
```

Output

```
The solution is
 1   0.8229
 2   0.9114
```

Example 2

This example uses the reverse communication option to solve the same problem as Example 1.

```
      INTEGER    LDC, LDDG, LWK, M, ME, N
      PARAMETER  (M=2, ME=1, N=2, LDC=N+1, LDDG=M,
     &            LWK=2*N*(N+16)+9*M+68)
C
      INTEGER    IBTYPE, IDO, IPRINT, IWK(19+M), MAXFUN, MAXITN,
     &           MODE
      REAL       ACC, AMACH, C(LDC,N+1), CONWK(M), D(N+1), DF(N),
     &           DG(LDDG,N), FVALUE, G(M), SCBOU, SQRT,
     &           U(M+N+N+2), WK(LWK), X(N), XLB(N), XUB(N)
      LOGICAL    ACTIVE(2*M+13)
      INTRINSIC  SQRT
      EXTERNAL   AMACH, N0ONF
C
      DATA IBTYPE/3/, MAXITN/100/, MODE/2/, MAXFUN/10/, IPRINT/0/
      DATA X/2.0E0, 2.0E0/, XLB(1)/-1.0E6/, XUB(1)/1.0E6/,
SCBOU/1.0E3/
C                                Set final accuracy (ACC)
      ACC = SQRT(AMACH(4))
C
      ACTIVE(1) = .TRUE.
```

```
      ACTIVE(2) = .TRUE.
      IDO       = 0
   10 IF (IDO.EQ.0 .OR. IDO.EQ.1) THEN
C                                  Evaluate the function at X.
         FVALUE = (X(1)-2.0E0)**2 + (X(2)-1.0E0)**2
C                                  Evaluate the constraints at X.
         G(1) = X(1) - 2.0E0*X(2) + 1.0E0
         G(2) = -(X(1)**2)/4.0E0 - X(2)**2 + 1.0E0
      END IF
C
      IF (IDO.EQ.0 .OR. IDO.EQ.2) THEN
C                                  Evaluate the function gradient at
X.
         DF(1) = 2.0E0*(X(1)-2.0E0)
         DF(2) = 2.0E0*(X(2)-1.0E0)
C                                  If active evaluate the constraint
C                                  gradient at X.
         IF (ACTIVE(1)) THEN
            DG(1,1) = 1.0E0
            DG(1,2) = -2.0E0
         END IF
C
         IF (ACTIVE(2)) THEN
            DG(2,1) = -0.5E0*X(1)
            DG(2,2) = -2.0E0*X(2)
         END IF
      END IF
C                                 Call N0ONF for the next update.
C
      CALL N0ONF (IDO, M, ME, N, IBTYPE, XLB, XUB, IPRINT, MAXITN, X,
     &            FVALUE, G, DF, DG, LDDG, U, C, LDC, D, ACC, SCBOU,
     &            MAXFUN, ACTIVE, MODE, WK, IWK, CONWK)
C                                 If IDO does not equal 1 or 2, exit.
      IF (IDO.EQ.1 .OR. IDO.EQ.2) GO TO 10
C                                  Print the solution
      CALL WRRRN ('The solution is', N, 1, X, N, 0)
C
      END
```

Output

```
The solution is
 1   0.8229
 2   0.9114
```