DONLP2 USERS GUIDE

P. SPELLUCCI
Technical University at Darmstadt, Department of Mathematics
64289 Darmstadt, Germany
email: spellucci@mathematik.tu-darmstadt.de

1 Copyright

2 GENERAL DESCRIPTION

Purpose:
Minimization of an (in general nonlinear) differentiable real function f subject to (in general nonlinear) inequality and equality constraints g, h.

$$\begin{aligned} f(x) &= \min_{x \in \mathcal{S}} \\ \mathcal{S} &= \{x \in \mathbb{R}^n : h(x) = 0, g(x) \geq 0\} \ . \end{aligned}$$

Here g and h are vectorvalued functions of dimension `NG` and `NH`.

Bound constraints are integrated in the inequality constraints g. These might be identified by a special indicator (see `GUNIT` below) in order to simplify calculation of its gradients and also in order to allow a special treatment, known as the gradient projection technique. Also fixed variables might be introduced via h in the same manner.

2.1 Method employed

The method implemented is a sequential equality constrained quadratic programming method (with an active set technique) with an alternative usage of a fully regularized mixed constrained subproblem in case of nonregular constraints (i.e. linear dependent gradients in the "working set"). It uses a slightly modified version of the Pantoja-Mayne update for the Hessian of the Lagrangian, variable dual scaling and an improved Armijo-type stepsize algorithm. Bounds on the variables are treated in a gradient-projection like fashion. Details may be found in the following two papers :
P. Spellucci: *An SQP method for general nonlinear programs using only equality constrained subproblems.* Math. Prog. 82, (1998), 413–448
P. Spellucci: *A new technique for inconsistent problems in the SQP method.* Math. Meth. of Oper. Res. 47, (1998), 355–400. (published by Physica Verlag, Heidelberg, Germany).

2.2 System requirements

The software is written in ANSI-F77 with MILSTD 1753 extensions with the exception of two routines `O8CPU()` for giving the processes cpu-time and `O8TIDA()` for writing the run's date and starting time,

which in the present implementation use the unix-system-call gettimeofday and times. These usually can be found in the C-library (include gettimeofday.c in compiling if not automatic for your FORTRAN-call-script) The user should replace (or remove) these calls for other systems. The corresponding routines are to be found in the file sysdepen.f. The user may replace this by dummy.f from the distribution in the simplest case.

2.3 RESTRICTIONS

Since the algorithm makes no use of sparse matrix techniques, its proper use will be limited to small and medium sized problems with dimensions up to 300 (for the number of unknowns) say. The number of inequality constraints however may be much larger. (E.g. the code did solve the Ecker-Kupferschmid-Marin "robot-design" examples in SIJSC 15, 1994, with n=51 unknowns and m=3618 inequalities successfully within 400-900 sec's depending on the variation of the problem on an HP9000/735). The maximal allowed dimensions are set in the file O8PARA.INC. The distribution comes with NX=300, NRESM=300.

2.4 SYSTEM DEPENDENCIES

The code has been successfully tested on the HP9000/7xy , SGI , SUN and DEC ALPHA, INTEL PENTIUM (under LINUX) systems but should run on any UNIX-V system without any changes. It has been designed such that changes for other systems are quite simple.
In order to change the maximal possible dimensions, reedit the `PARAMETER (NX=.....)` line of `O8PARA.INC` and recompile the code.
In order to use the code on CRAY-like systems, reedit donlp2.f, and the include-files (files ending with *.INC) changing globally `DOUBLE PRECISION` to `REAL` and the double precision constants which are assembled in the `PARAMETER`-statement of `O8CONS.INC` to real format.

3 USAGE

`DONLP2` is written as a selfcontained system of subprograms. The user issues a

call donlp2

supplying necessary information in a series of userwritten subprograms and in some common areas described below. A rudimentary main-program donlp2main.f is contained in the distribution of the code.

3.1 PROBLEM DESCRIPTION

The user has to define the problem through function evaluation routines. This may be done in two ways: Method one gives any function and gradient evaluation code individually. In this case the parameter `BLOC` has to be set to `FALSE`. This is the default.

1 `EF(X,FX)` returns `FX`=$f(x)$ given x as input. Arguments `DOUBLE PRECISION X(*),FX`

2 `EGRADF(X,GRADF)` returns `GRADF`=$\nabla f(x)$ given x as input.
Arguments `DOUBLE PRECISION X(*),GRADF(*)`.

3 `EH(I,X,HXI)` returns the value of the i-th equality constraint as hxi given i and x as input. Arguments `INTEGER I; DOUBLE PRECISION X(*),HXI`.

4 `EGRADH(I,X,GRADHI)` returns $\nabla h_i(x)$ as `GRADHI` given i and x as input. Arguments `INTEGER I ; DOUBLE PRECISION X(*),GRADHI(*)`.

5 EG(I,X,GXI) returns $g_i(x)$ as GXI given i and x as input. Arguments INTEGER I; DOUBLE PRECISION X(*),GXI. Bound constraints have to be programmed here too. The evaluation of their gradients however may be much simplified. See below concerning the usage of GUNIT.

6 EGRADG(I,X,GRADGI) returns $\nabla g_i(x)$ as GRADGI given i and x as input. Arguments INTEGER I ; DOUBLE PRECISION X(*),GRADGI(*). For bound constraints, no evaluations must be coded here if GUNIT is initialized properly.

7 SETUP0 has to set some problem specific data in DONLP2's common areas as described below. SETUP0 is the first subprogram called by DONLP2 prior to any other computation.

8 SETUP may set user specific data (not transparent to DONLP2) as well as parameters of DONLP2 e.g. in order to supersede DONLP2's standard parameter setting.
!!! SETUP is called after SETUP0 and after DONLP2's standard initialization, which is done in O8ST.

9 SOLCHK may add additional final computations, graphical output using e.g. information from ACCINF (see below) and so on. This distribution contains a solchk.f with an empty body.

donlp2usrfc.f in this directory contains a rudimentary version of these routines which easily may be completed by the user. The bodies of SETUP, EG, EGRADG, EH, EGRADG and SOLCHK may be empty of course. In that case unconstrained minimization using the BFGS-method is done. The correctness of EGRADF, EGRADH and EGRADG can to some extent be checked by running testgrad.f (in this directory) with the set of user routines .

Alternatively the user might prefer or may be forced to evaluate the problem functions by a black-box external routine. In this case he has to set

```
BLOC=.TRUE.
```

within SETUP0 and to use a subroutine

```
EVAL_EXTERN(MODE)
```

This routine has to take XTR, which is a copy of X, from the O8FUEXTR common area as input and to compute , according to the setting of MODE and ANALYT either FU(0:NH+NG) or both FU(0:NH+NG) and FUGRAD(I,0:NH+NG), I=1,N and then return to the user. The meaning of MODE is as follows:

1 MODE=0 compute only those constraints which depend linearly on exactly one variable, i.e. fixed variables conditions and bound constraints. These constraints are identified by GUNIT(1,I)=1.

2 MODE=1 compute only FU, i.e. function values.

3 MODE=3 compute function and gradient values FU and FUGRAD.

The routines SETUP0, SETUP and SOLCHK must be present in this case too, SETUP and SOLCHK may have an empty body of course.

DONLP2 has a built-in feature for doing gradients numerically. This can be used for both methods of problem description. This is controlled by the variables ANALYT , EPSFCN, TAUBND and DIFFTYPE. If ANALYT=.TRUE. then DONLP2 uses the values from the EGRAD.. routines or from FUGRAD, according to the setting of BLOC. If ANALYT=.FALSE., then numerical differentiation is done internally using a method depending on DIFFTYPE.

1 DIFFTYPE=1 Use the ordinary forward difference quotient with discretization stepsize $0.1\texttt{EPSFCN}^{1/2}$ componentwise relative.

2 DIFFTYPE=2 Use the symmetric difference quotient with discretization stepsize $0.1\texttt{EPSFCN}^{1/3}$ componentwise relative.

3 DIFFTYPE=3 Use a sixth order approximation computing a Richardson extrapolation of three symmetric difference quotient values. This uses a discretization stepsize 0.01EPSFCN$^{1/7}$.

Here

EPSFCN

is the expected relative precision of the function evaluation. The precision obtained in the gradients is then of the order of EPSFCN$^{1/2}$, EPSFCN$^{2/3}$, EPSFCN$^{6/7}$ respectively. Since numerical differentiation can also occur if variables are on their bounds, the user must supply a further parameter

TAUBND

which gives a positive value by which any bound present may be violated. The user is warned that numerical differentiation uses n, $2n$, $6n$ additional function evaluations for a single gradient for DIFFTYPE=1,2,3 respectively.

The sequence of routines described above has to be present in any case. (If EVAL_EXTERN is used, then of course the bodies of EF, EGRADF, EH, EGRADH, EG, EGRADG are to be empty. This distribution contains such a file donlp2dummyfu.f .) Bound constraints are treated in a special way by DONLP2. The initial point is corrected automatically to fit the bounds. Thereafter any succeeding point is projected with respect to the bounds. The presence of bound constraints is indicated using the GUNIT-array. This is a descriptive array for f, $h_1, ..., h_e, g_1, ..., g_m$ (with e=NH and m=NG) (exactly in that order) with the following meaning:

1 GUNIT(1,j)=-1: j-th function is "general".

2 GUNIT(1,j)=1: j-th function depends linearly on X(GUNIT(2,j)) alone with GUNIT(3,j) as partial derivative, that is this function has the form gunit(3,j)*x(gunit(2,j)) + const.

E. g. let

$$3x_7 - 5.6 \ \geq \ 0$$

be the 20-th inequality constraint (the call EG(20,X,GXI) gives GXI=$3x_7 - 5.6$). This will be expressed as

```
GUNIT(1,NH+20)=1
GUNIT(2,NH+20)=7
GUNIT(3,NH+20)=3
```

The evaluation of these bound constraints however must be done in EG and EH in the usual way, e.g.

```
.............................
IF ( I .EQ. 20 ) THEN
 GXI=3.D0*X(7)-5.6D0
 RETURN
ENDIF
.............................
```

GUNIT must be given completely from (I,0) (corresponding to f), over (I,1,...NH), (corresponding h) until (I,NH+NG), I=1,2,3. If a user doesn't want to use this feature , she(he) simple writes

```
DO I=0,NRESM
  GUNIT(1,I)=-1
  GUNIT(2,I)=0
  GUNIT(3,I)=0
ENDDO
```

within SETUP0. This however may be inefficient, since now bound-constraints are treated like general nonlinear ones and may also become violated(!).

Within SETUP0, the following data must be given:

1 N number of variables (≤ 300 at present. check O8PARA.INC)

2 NH number of equality constraints (may be zero).

3 NG number of inequality constraints including bounds. (may be zero). NH+NG $\leq$ NRESM ≤ 1800 by default setting in O8PARA.INC at present.

4 NAME Problem identifier, 40 characters maximum. The leading 8 characters (if any) have to be alphanumeric, the first one alphabetic. Nonalphabetic characters are interpreted as 'X'.

5 X Initial guess (also holds the current solution).

6 TAU0 !!! This parameter should be carefully selected by the user: It gives a (universal) bound describing how much the unscaled penalty-term (the l1-norm of the constraint violation) may deviate from zero. DONLP2 assumes that within the region described by

$$\sum_{i=1}^{\mathrm{NH}} |h_i(x)| - \sum_{i=1}^{\mathrm{NG}} \min\{0, g_i(x)\} \ \leq \ \mathrm{TAU0}$$

all functions may be evaluated safely. The initial guess however may violate these requirements. In that case an initial feasibility improvement phase is run by DONLP2 until a point is found, that fits them. This is done by scaling f by SCF=0, such that the pure unscaled penalty term is minimized. TAU0 may be chosen as large as a user may want. A small TAU0 diminishes the efficiency of DONLP2, because the iterates then will follow the boundary of the feasible set closely. These remarks do not apply for bound constraints, since DONLP2 treats these by a projection technique. Bounds always remain satisfied. In the example HS114 TAU0=1 works by good luck only. Contrary, a large TAU0 may degrade the reliability of the code (outdoors the wolfes are howling.)

7 DEL0. In the initial phase of minimization a constraint is considered binding if

$$g_i(x) / \max\{1, ||\nabla g_i(x)||\} \ \leq \ \mathrm{DEL0}.$$

Good values are between 1.d0 and 1.d-2 . If equal to zero , DEL0 is set equal to TAU0. If DEL0 is chosen too small, then identification of the correct set of binding constraints may be delayed. Contrary, if DEL0 is too large, then the method will often escape to the full regularized SQP method, using individual slack variables for any active constraint, which is quite costly. For well scaled problems DEL0=1 is reasonable. However, sometimes it should be much smaller (compare the list of testresults in the accompanying paper).

8 NRESET. If there are more then NRESET steps using "small" stepsizes and therefore small corrections, a "restart" of the accumulated quasi-Newton-update is tried. NRESET is internally bounded by 4 from below. Good values are between N and 3*N. In the standard setting NRESET is bounded by N. Override this by using SETUP if desired.

9 ANALYT. ANALYT=.TRUE., if the gradients are given by analytical expressions. With ANALYT=.FALSE. donlp2 relaxes its termination criteria using the variable

10 EPSDIF. DONLP2 assumes relative precision of EPSDIF in the gradients if ANALYT is set to .FALSE.

11 EPSFCN. This is used only if ANALYT=.FALSE.. In this case it must give the relative precision of the function evaluation routine. Since precision of the numerical gradients is lower than this, using the method with very crude function values makes no sense.

12 DIFFTYPE Type of numerical differentiation to be used, if any. See above.

13 PROU. FORTRAN unit number for output protocol

14 MEU. FORTRAN unit number for messages concerning "special events" (PROU and MEU must be different!!!!)

15 SILENT. logical. If .TRUE., neither PROU nor MEU are written. DONLP2 returns silently with its results in the appropriate common-areas giving no form of output. See the description of this areas below. The user might extract his desired information from these.

16 TAUBND. Amount by which bounds may be violated if numerical differentiation is used.

17 COLD. If true, then any information is computed afresh . Otherwise, the penalty-weights and quasi-newton-udates as stored in the common-fields are used for the current run. This is useful if a parametric problem is to be solved.

18 GUNIT see above

Default settings are

```
TAUBND=1.D0
EPSFCN=1.D-16
EPSDIF=1.D-14
SILENT=.FALSE.
COLD=.TRUE.
```

As an example for coding the problem case 114 from Hock and Schittkowski (Lecture notes on economics and mathematical systems 187, alkylation problem from Bracken and McCormick) is given here.

The problem is to minimize

$$f(x) = 5.04x_1 + 0.035x_2 + 10x_3 + 3.36x_5 - 0.063x_4x_7$$

subject to

$$\begin{aligned}
h_1(x) &= 1.22x_4 - x_1 - x_5 \;=\; 0 \\
h_2(x) &= 98000x_3/(x_4x_9 + 1000x_3) - x_6 \;=\; 0 \\
h3(x) &= (x_2 + x_5)/x_1 - x_8 \;=\; 0 \\
g_1(x) &= 35.82 - 0.222x_10 - bx_9 \;\geq\; 0, \quad b = 0.9 \\
g_2(x) &= -133 + 3x_7 - ax_10 \;\geq\; 0, \quad a = 0.99 \\
g_3(x) &= -g_1(x) + x_9(1/b - b) \;\geq\; 0, \\
g_4(x) &= -g_2(x) + (1/a - a)x_{10} \;\geq\; 0, \\
g_5(x) &= 1.12x_1 + 0.13167x_1x_8 - 0.00667x_1x_8^2 - ax_4 \;\geq\; 0, \\
g_6(x) &= 57.425 + 1.098x_8 - 0.038x_8^2 + 0.325x_6 - ax_7 \;\geq\; 0, \\
g_7(x) &= -g_5(x) + (1/a - a)x_4 \;\geq\; 0, \\
g_8(x) &= -g_6(x) + (1/a - a)x_7 \;\geq\; 0
\end{aligned}$$

and the bounds

$$\begin{aligned}
0.00001 \;\leq\; & x_1 \;\leq\; 2000 \\
0.00001 \;\leq\; & x_2 \;\leq\; 16000 \\
0.00001 \;\leq\; & x_3 \;\leq\; 120 \\
0.00001 \;\leq\; & x_4 \;\leq\; 5000 \\
0.00001 \;\leq\; & x_5 \;\leq\; 2000 \\
85 \;\leq\; & x_6 \;\leq\; 93
\end{aligned}$$

$$
\begin{aligned}
90 &\leq x_7 \leq 95 \\
3 &\leq x_8 \leq 12 \\
1.2 &\leq x_9 \leq 4 \\
145 &\leq x_{10} \leq 162\ .
\end{aligned}
$$

This may be coded as follows (given as alkylati.f in the directory examples): (For easier handling, INCLUDE-files are used here, which are to be found in this distribution too.)

```
      SUBROUTINE SETUP
C   NO SPECIAL USER DATA
      RETURN
      END
      SUBROUTINE SETUP0
      INCLUDE 'O8COMM.INC'
      INTEGER I,J
      DOUBLE PRECISION XST0(10)
      DATA (XST0(I),I=1,10)
     F     /1745.D0,12.D3,11.D1,3048.D0,1974.D0,89.2D0,92.8D0,8.D0,
     1       3.6D0,145.D0/
C   NAME IS IDENT OF THE EXAMPLE/USER AND CAN BE SET AT USERS WILL
C   THE FIRST CHARACTER MUST BE ALPHABETIC.  40 CHARACTERS MAXIMUM
       NAME='ALKYLATION'
C   X IS INITIAL GUESS AND ALSO HOLDS THE CURRENT SOLUTION
C   PROBLEM DIMENSION N=DIM(X), NH=DIM(H), NG=DIM(G)
      N=10
      NH=3
      NG=28
      ANALYT=.TRUE.
      EPSDIF=1.D-16
      EPSFCN=1.D-16
      PROU=10
      MEU=20
C   DEL0 AND TAU0: SEE BELOW
      DEL0=1.D0
      TAU0=1.D0
      TAU=0.1D0
      DO I=1,N
      X(I)=XST0(I)
      ENDDO
C    GUNIT-ARRAY, SEE BELOW
      DO J=0,11
      GUNIT(1,J)=-1
      GUNIT(2,J)=0
      GUNIT(3,J)=0
      ENDDO
      DO J=12,31
      GUNIT(1,J)=1
      IF ( J .LE. 21 ) THEN
        GUNIT(2,J)=J-11
        GUNIT(3,J)=1
      ELSE
        GUNIT(2,J)=J-21
        GUNIT(3,J)=-1
      ENDIF
      ENDDO
```

```
      RETURN
      END

C     OBJECTIVE FUNCTION
      SUBROUTINE EF(X,FX)
      INCLUDE 'O8FUC0.INC'
      DOUBLE PRECISION FX,X(*)
      ICF=ICF+1
      FX=5.04D0*X(1)+.035D0*X(2)+10.D0*X(3)+3.36D0*X(5)-.063D0*X(4)*X(7)
      RETURN
      END
C     GRADIENT OF OBJECTIVE FUNCTION
      SUBROUTINE EGRADF(X,GRADF)
      INCLUDE 'O8FUC0.INC'
      INTEGER J
      DOUBLE PRECISION X(*),GRADF(*),A(10)
      SAVE A
      DATA A/5.04D0,0.035D0,10.D0,0.D0,3.36D0,5*0.D0/
      ICGF=ICGF+1
      DO      100      J=1,10
      GRADF(J)=A(J)
 100  CONTINUE
      GRADF(4)=-0.063D0*X(7)
      GRADF(7)=-0.063D0*X(4)
      RETURN
      END

C  COMPUTE THE I-TH EQUALITY CONSTAINT, VALUE IS HXI
      SUBROUTINE EH(I,X,HXI)
      INCLUDE 'O8FUC0.INC'
      DOUBLE PRECISION X(*),HXI
      INTEGER I
      CRES(I)=CRES(I)+1
      GOTO (100,200,300),I
  100 CONTINUE
      HXI=1.22D0*X(4)-X(1)-X(5)
      RETURN
  200 CONTINUE
      HXI=9.8D4*X(3)/(X(4)*X(9)+1.D3*X(3))-X(6)
      RETURN
  300 CONTINUE
      HXI=(X(2)+X(5))/X(1)-X(8)
      RETURN
      END
C  COMPUTE THE GRADIENT OF THE I-TH EQUALITY CONSTRAINT
      SUBROUTINE EGRADH(I,X,GRADHI)
      INCLUDE 'O8FUC0.INC'
      INTEGER I,J
      DOUBLE PRECISION X(*),GRADHI(*),T,T1
      CGRES(I)=CGRES(I)+1
      DO      100      J=1,10
      GRADHI(J)=0.D0
  100 CONTINUE
      GOTO(200,300,400),I
```

```
  200 CONTINUE
      GRADHI(1)=-1.D0
      GRADHI(4)=1.22D0
      GRADHI(5)=-1.D0
      RETURN
  300 CONTINUE
      T=9.8D4/(X(4)*X(9)+1.D3*X(3))
      T1=T/(X(4)*X(9)+1.D3*X(3))*X(3)
      GRADHI(3)=T-1.D3*T1
      GRADHI(4)=-X(9)*T1
      GRADHI(9)=-X(4)*T1
      GRADHI(6)=-1.D0
      RETURN
  400 CONTINUE
      GRADHI(1)=-(X(2)+X(5))/X(1)**2
      GRADHI(2)=1.D0/X(1)
      GRADHI(5)=GRADHI(2)
      GRADHI(8)=-1.D0
      RETURN
      END

C COMPUTE THE I-TH INEQUALITY CONSTAINT, BOUNDS INCLUDED

      SUBROUTINE EG(I,X,GXI)
      INCLUDE 'O8FUC0.INC'
      INTEGER I,K
      DOUBLE PRECISION GXI
      DOUBLE PRECISION X(NX),OG(10),UG(10),A,B,C,D,T
      SAVE A,B,C,D,UG,OG
      DATA A/.99D0/, B/.9D0/, C/2.01010101010101D-2/, D/
     1       2.11111111111111D-1/
      DATA OG/2.D3,16.D3,1.2D2,5.D3,2.D3,93.D0,95.D0,12.D0,4.D0,162.D0/
      DATA UG/5*1.D-5,85.D0,90.D0,3.D0,1.2D0,145.D0/
      IF ( GUNIT(1,I+NH) .EQ. -1 ) CRES(I+NH)=CRES(I+NH)+1
      IF(I .GT. 8)      GOTO 500
      K=(I+1)/2
      GOTO(100,200,300,400) K
  100 CONTINUE
      T=35.82D0-.222D0*X(10)-B*X(9)
      IF(K+K .EQ. I)    T=-T+X(9)*D
      GXI=T
      RETURN
  200 CONTINUE
      T=-133.D0+3.D0*X(7)-A*X(10)
      IF(K+K .EQ. I)    T=-T+C*X(10)
      GXI=T
      RETURN
  300 CONTINUE
      T=1.12D0*X(1)+.13167D0*X(1)*X(8)-.00667D0*X(1)*X(8)**2-A*X(4)
      IF(K+K .EQ. I)    T=-T+C*X( 4)
      GXI=T
      RETURN
  400 CONTINUE
      T=57.425D0+1.098D0*X(8)-.038D0*X(8)**2+.325D0*X(6)-A*X(7)
```

```
      IF(K+K .EQ. I)    T=-T+C*X( 7)
      GXI=T
      RETURN
  500 CONTINUE
      IF(I .GT. 18)   GXI=OG(I-18)-X(I-18)
      IF(I .LE. 18)   GXI=X(I-8)-UG(I-8)
      END
C COMPUTE THE GRADIENT OF THE I-TH INEQUALITY CONSTAINT
      SUBROUTINE EGRADG(I,X,GRADGI)
      INCLUDE 'O8FUCO.INC'
      INTEGER I,J,K
      DOUBLE PRECISION X(NX) ,GRADGI(NX),A,B,C,D
      SAVE A,B,C,D
      DATA A/.99D0/, B/.9D0/, C/1.01010101010101D0 /, D/
     1       1.11111111111111D0/
       DO     50       J=1,NX
       GRADGI(J)=0.D0
   50  CONTINUE
      IF(I .GT. 8)     GOTO 500
      K=(I+1)/2
      GOTO(100,200,300,400) K
  100  CONTINUE
      IF(K+K .EQ. I)      GOTO 150
      GRADGI(9)=-B
      GRADGI(10)=-.222D0
      RETURN
  150 CONTINUE
      GRADGI( 9)= D
      GRADGI(10)= .222D0
      RETURN
  200 CONTINUE
      IF(K+K .EQ. I)      GOTO 250
      GRADGI( 7)=3.D0
      GRADGI(10)=-A
      RETURN
  250 CONTINUE
      GRADGI( 7)=-3.D0
      GRADGI(10)=  C
      RETURN
  300 CONTINUE
      GRADGI( 1)=1.12D0+.13167D0*X(8)-.00667D0*X(8)**2
      GRADGI( 4)= -A
      GRADGI( 8)=.13167D0*X(1)-.01334D0*X(1)*X(8)
      IF(K+K .NE. I)      RETURN
      GRADGI( 1)=-GRADGI(1)
      GRADGI( 8)=-GRADGI(8)
      GRADGI( 4)=  C
      RETURN
  400 CONTINUE
      GRADGI( 6)=.325D0
      GRADGI( 7)= -A
      GRADGI( 8)=1.098D0-.076D0*X(8)
      IF(K+K .NE. I)      RETURN
      GRADGI( 6)=-.325D0
```

```
      GRADGI( 7)=C
      GRADGI( 8)=-GRADGI(8)
      RETURN
  500 CONTINUE
      IF(I .GT. 18)      GRADGI(I-18)=-1.D0
      IF(I .LE. 18)      GRADGI(I-8)=1.D0
      RETURN
      END
```

If the user wants valid statistics concerning the number of function evaluations etc. she(he) has to increase the counters in the named common block `O8CNT` as done in the example above.

1 `ICF` counts the number of function calls of the objective,

2 `CRES(1:NH)` those of the equality and

3 `CRES(NH+1:NG+NH` of the inequality constraints.

4 `ICGF`, and `CGRES(1:NH+NG)` are the counters of the gradient calls.

`DONLP2` does no evaluation statistics for its own.

3.2 ADDITIONAL FEATURES

There are additional possibilities to enhance the performance of the code:

1 Affine linear functions.
If some function is affine-linear in x (i.e. has constant gradient) then the user might set `GCONST(I)=.TRUE.` in `SETUP0`, with `I` in the range of 0 to `NH+NG`. In that case the gradient is evaluated once only and the corresponding function is excluded from the computation of the Maratos-correction. Also the evaluation of this gradient is counted once only. Feasibility for affine linear functions other than bounds and fixed variables is <u>not</u> maintained.

2 Parametric problems.
If the user has to solve a problem in a parametric fashion, she/he may do this using `COLD=.FALSE.` in a subsequent call of `SETUP0` , thereby avoiding the standard initialization of `DONLP2` (done in `O8ST`). In that case the old Quasi-Newton-update and the old penalty-weights are used as initial guesses. The files `USER1.INC`, parmain.f and parametric.f show how this may be done.

3 Parameter settings.
Any of the parameters of `DONLP2` (e.g. the parameters controlling amount of output `TE0, TE1, TE2, TE3`) may be changed at the users will within `SETUP` by redefining the values located in the corresponding common blocks described below. This <u>must</u> be done within `SETUP` whose call follows that of the standard initialization routine `O8ST`. In `SETUP` use `INCLUDE 'O8COMM.INC'` in order to access these parameters.

4 Error return from function evaluation.
The user has the possibility to prevent `DONLP2` from leaving the domain of definition of one of her(his) functions using the error-indicators of the COMMON-block `O8ERR`. The code proposes changes of the current solution using some formula

$$x_{new} = x_{current} + \sigma d + \sigma^2 dd$$

with correction vectors d and dd. These directions might well be "downhill" but too large, hence x_{new} leaving the domain of definition of some function involved. The user may check the actual parameter of the function evaluation and set `FFUERR=.TRUE.` (in the objective function evaluation) or `CFUERR(I)=.TRUE.` in one of the constraint function evaluations and return to the optimizer then. In this case `DONLP2` tries to avoid the problem by reducing σ. If such an error occurs with the current x, i.e. $\sigma = 0$, the run is terminated with an appropriate error message.

5 Scaling.
The user has the possibility to prescribe an internal scaling of x, setting the array `XSC` appropriately in `SETUP0` (must be done there, if any). The initial value given and the function and gradient evaluations however are always in the original unscaled variables. The external variables are then `XSC(I)*X(I)`, with `X(I)` as internal variables. The first internal variable is obtained by dividing the given initial values `X(I)` from `SETUP0` or `BLOCKDATA` by `XSC(I)`.

6 Changing termination criteria.
Termination criteria can be changed in various ways. Seven variables enter these criteria (see below), namely

`EPSX, DELMIN, SMALLW, EPSDIF, NRESET, NUMSM , EPSPHI`

The first five have been mentioned already. The remaining two may be changed from their default values `MAX(10,N)` and `1000*EPSMAC` in order to avoid many steps with only very marginally changing objective function values in the terminal phase of optimization. (Such a situation occurs e.g. for illconditioned cases). The user is warned that being too sloppy with `EPSPHI` might result in premature termination and solutions far from optimum, if the problem is nonconvex.

4 OUTPUT / RESULTS

`DONLP2` computes the names of the two files for output using the first 8 characters from `NAME` . If `NAME` is shorter than 8 characters `XXX..` is appended to `NAME` . These files are used for the following: The `NAME(1:8).PRO` file contains the results of the optimization run . Of course these results are also contained in the named common blocks described below. In case of a failure a short protocol of the complete run is appended, which may be evaluated by a knowledgeable person in order to locate the problem.

!!!! FORTRAN files `PROU` and `MEU` are used for output and must not be used otherwise by the user

Output can be of various stages of volume: Normally the final results are written to the PRO- and the "special events" to the MES-file. That means TE0=TE1=TE2=TE3=.FALSE., as given in the head of `DONLP2`. Then the XXXXXXXX.PRO -file holds the following:

1. copyright information
2. date and time of run
3. name of problem as defined by the user
4. parameter settings of `DONLP2` for this run, as left by `SETUP`.
5. starting value of primal variable
6. termination reason (text)
7. final scaling parameter of primal objective (output of other data is relative to scaling=1) If this is zero, then `DONLP2` failed in the infeasibility improvement phase, where it tries to reduce infeasibility below `TAU0`. In that case a larger `TAU0` may be tried, or otherwise a different initial guess.
8. $||\nabla f||$
9. error in Lagrangian condition, i.e. $||\nabla_x L||$. Compare with $||\nabla f||$ in order to assess accuracy. In an unconstrained case $||\nabla f|| = ||\nabla_x L||$ of course.
10. primal and dual feasibility errors. For primal the unscaled l1-penalty term and for dual the most negative multiplier, if any.

11. DONLP2's cpu-time.
12. optimal value of primal objective ($f(x^*)$).
13. optimal primal value x^*.
14. list of constraint values, max(1,gradient norms) (if evaluated at all) and multipliers. (The norm of a gradient of a constraint which never has been evaluated is set to one. Therefore the norms of the gradients of the active constraints are meaningful only).
15. statistics concerning the evaluation of constraints
16. condition number estimates for matrix of binding gradients and for quasi-Newton update for Hessian of the augmented Lagrangian.
17. run statistics.

For termination reason (variable OPTITE) see below, description of common-blocks.

If INTAKT is set to .TRUE. in SETUP, any output which appears in the PRO-file is written to std-out too.

if TE0 is set true, then a one-line-information is written to std-out for every step of iteration, giving sufficient information on the progress of solution. This is the following:

1. step number
2. FX = value of primal objective
3. UPSI = value of the unscaled l1-penalty
4. B2N = value of l2-norm of error in Lagrange condition (projected gradient for sytem transformed by the Cholesky-factor R^T of A), i.e. $||(R^T)^{-1}\nabla_x L||$.
5. UMI = smallest negative multiplier or zero
6. NR = number of binding constraints
7. SI -1 = constraints satisfy the regularity condition, 1 = constraints do not satisfy it. In that case a full regularized qp-problem is solved.

If TE1 is set true, the short protocol accumulated in the array ACCINF is printed at termination. TE1 is set to true automatically in case of unsuccessful termination. The ACCINF-array is a 32-column array, each line of which holds information on an iteration in condensed form, see below .

If TE2 is set true, intermediate results X,D, active constraint values and so on are written to the PRO-file. These are:

1. iteration step number
2. SCF = scaling factor of primal objective
3. PSIST = scaled penalty term at x_initial
4. PSI = scaled penalty term (current)
5. UPSI = unscaled l1-penalty term
6. FXST = f at x_initial (end of feasibility improvement phase)
7. FX = f current

8. X = current value of optimization variables (in the internal scaling).
9. permutation of variables applied to make qr-decomposition of matrix of gradients of binding constraints continuous (see Coleman and Sorensen, Math.Prog 29)
10. DEL }
11. B2NO } see below, description of common-blocks
12. B2N }
13. GFN = $||\nabla f(x)||$.
14. list of values of binding constraints and gradient norms (Euclidean norm)
15. DIAG(R)= diagonal of r-part of qr-decompostion of matrix of binding gradients
16. U = list of multipliers
17. eventually message concerning singularity of the problem and the final result from qp
18. eventually a list of constraints considered for inactivation and multipliers obtained thereafter
19. condition number estimator for matrix of binding constraints and the Hessian of the augmented Lagrangian used in the Pantoja and Mayne update
20. D = direction of descent for penalty function
21. information on new scaling:
    (a) SCF scaling for f,
    (b) CLOW : number of penalty decreases
    (c) ETA: penalty decrease takes place only , if for the new tentative weights
    (SCF*(FXST-FX)+PSIST-PSI) $\geq$ CLOW*ETA.
    ETA is adapted during the run.
    (d) SCALRES: the penalty weights
22. START UNIMIN: entry to unidimensional search (stepsize selection)
23. PHI = penalty function
24. DPHI = its directional derivative along D
25. BOUND UPSI = TAU0/2
26. PSI weighted penlty term
27. UPSI unweighted penalty term
28. DSCAL scaling of D. If stepsize 1 would change the current X too strong, as expressed by

$$||d|| > \beta(||x|| + 1)$$

    with the internal parameter BETA (=4 by default), then D is shortened accordingly.
29. FX $f(x)$
30. SCF current scaling of f in the penalty function.
31. SIG = stepsize tried (σ)
32. final accepted stepsize

33. list of constraints not binding at X but hit during search

34. kind of matrix updates applied

35. the parameters used in the update

If `TE3` is true, the values of the gradients and the approximated Hessians are printed too, provided $n \leq 50$ and `NRES=NH+NG`≤ 100. The *.MES-file is of value if the optimizer ends irregularly. It contains a message for every action which is "abnormal" in the sense of good progress for a regular and well conditioned problem. In most cases special advice is necessary to evaluate it. However, if the user observes many restarts reported in the *.MES file or similarly many calls to the full sqp-method, she(he) should check the problem against bad scaling and redundant constraints!

5 TERMINATION CRITERIA USED

`donlp2` has a lot of termination criteria built in, many of which can be controlled by the user. However, an unexperienced user is warned to use too sloppy criteria, since often for nonconvex problems progress in the iteration may be quite irregular. There may occur many steps with seemingly small progress followed by a sudden improvement. The following termination messages may occur, which are explained here in detail:

1. `'CONSTRAINT EVALUATION RETURNS ERROR WITH CURRENT POINT'`
   The user evaluation routines EH/EGRADH or EG/EGRADG did set an error indicator then asked for a new value and reduction of the trial stepsize down to its minimum did not remove the situation. Usually this means that an inadequate starting point has been provided. Also box constraints may have not been used adequately in order to prevent boundary points of the feasible set there some function is undefined.

2. `'OBJECTIVE EVALUATION RETURNS ERROR WITH CURRENT POINT'`
   Same reason as above with respect to the objective function.

3. `'QPSOLVER: EXTENDED QP-PROBLEM SEEMINGLY INFEASIBLE '`
   Theoretically, this is impossible, since the QP problem is constructed such that a feasible solution always exists. Hence this message means that the solution process is severely corrupted by roundoff, most probably a problem of bad scaling which was not overcome by the internal scaling techniques.

4. `'QPSOLVER: NO DESCENT FOR INFEAS FROM QP FOR TAU=TAU_MAX'`
   This means that a stationary point of the penalty function has been located which is not feasible. From the theory of the method it follows that the problem is not penalizable, at least in the region of the current point, since the Mangasarian Fromowitz condition, under which global convergence is proved, is the weakest possible for this purpose. Incresing TAUMAX may sometimes remove the situation. (TAUMAX represents a "very big" penalty).

5. `'QPSOLVER: ON EXIT CORRECTION SMALL, INFEASIBLE POINT'`
   Same reason as above

6. `'STEPSIZESELECTION: COMPUTED D FROM QP NOT A DIR. OF DESCENT'`
   Same as above or limiting accuracy reached for a singular problem, with termination criteria being too strong. (The method usually tries the outcome d of the QP-solver in spite of problems signaled from the solver as a direction of descent. The final check is done in the stepsize selection subprogram.)

7. `'MORE THAN MAXIT ITERATION STEPS'`
   Progress is too slow to meet the termination criteria within the given iteration limit (either MAXIT or the user defined ITERMA number of steps)

8. 'STEPSIZESELECTION: NO ACCEPTABLE STEPSIZE IN [SIGSM,SIGLA]'
   This may have a lot of different reasons. The most usual one is simply a programming error in the user supplied (analytical) gradients. It may also be due to termination criteria which are too stringent for the problem at hand, because of evaluation unpreciseness of functions and/or gradients or because of illconditioning of the (projected) Hessian matrix. Often the results are acceptable nevertheless, but the user should check that (e.g. by supplying an appropriate routine SOLCHK)

9. 'SMALL CORRECTION FROM QP, INFEASIBLE POINT'
   A stationary point of the penalty function which is not feasible for the original problem has been located. Surely the problem is incompatible locally at least. Try some other initial guess.

10. 'KT-CONDITIONS SATISFIED, NO FURTHER CORRECTION COMPUTED'
    means

$$\begin{aligned} ||h(x)||_1 + ||g(x)^-||_1 &\leq \texttt{DELMIN} \\ ||\lambda^-||_\infty &\leq \texttt{SMALLW} \\ ||\nabla L(x,\mu,\lambda)|| &\leq \texttt{EPSX}(1 + ||\nabla f(x)||) \\ |\lambda^T||g(x)| &\leq \texttt{DELMIN*SMALLW*(NH+NG)} \end{aligned}$$

11. 'COMPUTED CORRECTION SMALL, REGULAR CASE '
    means

$$\begin{aligned} ||d|| &\leq (||x|| + \texttt{EPSX})\texttt{EPSX} \\ ||\nabla L(x,\mu,\lambda)|| &\leq \texttt{EPSX}(1 + ||\nabla f(x)||) \\ ||h(x)||_1 + ||g(x)^-||_1 &\leq \texttt{DELMIN} \\ ||\lambda^-||_\infty &\leq \texttt{SMALLW} \end{aligned}$$

    where d is the computed correction for the current solution x.

12. 'STEPSIZESELECTION: X ALMOST FEASIBLE, DIR. DERIV. VERY SMALL'
    means that the directional derivative has become so small that progress cannot be expected. More precisely

$$D\Phi(x;d) \;\geq\; -100(|\Phi(x)| + 1) * \texttt{EPSMAC}$$

    where Φ is the current penalty function. This occurs as a termination reason for illconditioned problems usually.

13. 'KT-CONDITIONS (RELAXED) SATISFIED, SINGULAR POINT'
    means that we are at a point x where the matrix the constraint normals is illconditioned or singular and the following conditions are met, which describe a relaxed form of the KKT-conditions:

$$\begin{aligned} ||h(x)||_1 + ||g(x)^-||_1 &\leq \texttt{DELMIN(NH+NG)} \\ ||h(x)_{prev}||_1 + ||g(x_{prev})^-||_1 &\leq \texttt{DELMIN(NH+NG)} \\ ||\nabla L(x,\mu,\lambda)|| &\leq 100 \times \texttt{EPSX}(1 + ||\nabla f(x)||) \\ ||sl||_1 &\leq \texttt{DELMIN(NH+NG)} \end{aligned}$$

    where sl is the vector of articial slacks introduced to make the QP always consistent. Observe that dual feasibility and complementary slackness hold automatically for the current iterate, because we are solving a full QP subproblem.

14. 'VERY SLOW PRIMAL PROGRESS, SINGULAR OR ILLCONDITONED PROBLEM'
    means the occurence of the following conditon for at least N consecutive steps:

$$||h(x)||_1 + ||g(x)^-||_1 \quad \leq \quad \texttt{DELMIN(NH+NG)}$$

$$
\begin{aligned}
||h(x)_{prev}||_1 + ||g(x_{prev})^-||_1 &\leq \texttt{DELMIN(NH+NG)} \\
||\lambda^-||_\infty &\leq \texttt{SMALLW} \\
|\lambda^T||g(x)| &\leq \texttt{DELMIN*SMALLW*(NH+NG)} \\
||\nabla L(x,\mu,\lambda)|| &\leq \texttt{10*EPSX}(1 + ||\nabla f(x)||) \\
|f(x) - f(x_{prev})| &\leq (|f(x)| + 1)\texttt{EPS}
\end{aligned}
$$

where EPS is computed from the machine precision EPSMAC and the variables EPSX and EPSDIF as follows:

```
IF ( ANALYT ) THEN
  EPS=MIN(EPSX,SQRT(EPSMAC))
ELSE
  EPS=EPSDIF
  IF ( EPSX .LT. EPSDIF**2 ) EPSX=EPSDIF**2
ENDIF
EPS=MAX(EPSMAC*1000,MIN(1.D-3,EPS))
```

15. 'MORE THAN NRESET SMALL CORRECTIONS IN X '
means that for more than NRESET consecutive steps there were only componentwise small changes in the solution x as given by

$$|x_{prev,i} - x_i| \;\leq\; (|x_i| + 0.01) \times \texttt{EPSX}\,, \quad i = 1, \ldots, n$$

16. 'CORRECTION FROM QP VERY SMALL, ALMOST FEASIBLE, SINGULAR '
means that the gradients in the working set are linearly dependent or almost so, such that a full regularized QP is solved, with the outcome of a direction d satisfying

$$||d|| \;\leq\; 0.01(\min\{1, ||x||\} + \texttt{EPSX})\texttt{EPSX}\,.$$

This could effect changes in $||x||$ less than $0.01 \times$ EPSX (relative) at most, hence we terminate since this may be highly inefficient. It may occur in a problem there the second order sufficiency condition is not satisfied and the matrix of gradients of binding constraints is singular or very illconditioned.

17. 'NUMSM SMALL DIFFERENCES IN PENALTY FUNCTION,TERMINATE'
For NUMSM consecutive steps there were only small changes in the penalty function (without the other termination criteria satisfied) as given by

$$\Phi(x_{prev}) - \Phi(x) \;\leq\; \texttt{EPSPHI}(s|f(x)| + \Psi(x))$$

where Φ is the current Penaltyfunction, Ψ the current penalty term and s the current scaling of the objective function.

6 RESTRICTIONS

1) !!!!! names beginning with O8 are reserved by DONLP2 and must not be used otherwise within the users load module.

2) !!!!! the parameters of O8PARA.INC must not be changed by the user without having donlp2.f recompiled

files involved: O8BLO1.INC O8COMM.INC O8DER.INC O8PARA.INC O8QPDU.INC O8XBLO.INC O8BLOC.INC O8CONS.INC O8FUCO.INC O8QP.INC O8RDAT.INC O8XDAT.INC O8FINT.INC
donlp2.f, sysdepen.f (or dummy.f), donlp2main.f, solchk.f, user_eval.f, eval_extern.f (possibly empty body). (donlpusrfc.f, testgrad.f, example files)

7 THEORY

Mathematical background of the method may be found in the two papers cited above.

8 DESCRIPTION OF DONLP2's COMMON BLOCKS

The common blocks used by DONLP2 are listed here: (may be copied from O8COMM.INC and its further included *.INC files).

```
      IMPLICIT NONE
      INCLUDE 'O8PARA.INC'
c accinf contains information about intermediate results useful for a
c postmortem analysis .
C****
C   ACCINF A C C U M U L A T E D   I N F O R M A T I O N
C   ON ITERATION SEQUENCE
C   1: STEP-NR
C   2: F(X-K) CURRENT VALUE OF OBJECTIVE (ZERO IN FEASIBILITY IMPROVEMENT
C             PHASE (-1)  )
C   3: SCF    INTERNAL SCALING OF OBJECTIVE (ZERO IN PHASE -1)
C   4: PSI    THE WEIGHTED PENALTY-TERM
C   5: UPSI   THE UNWEIGHTED PENALTY-TERM (L1-NORM OF CONSTRAINT VECTOR)
C   6: DEL_K_1 BOUND FOR CURRENTLY ACTIVE CONSTRAINTS
C   7: B2NO   L2-NORM OF PROJECTED GRADIENT, BASED ON CONSTRAINTS IN LEVEL DELMIN
C             AND BELOW
C   8: B2N    L2-NORM OF PROJECTED GRADIENT BASED ON DEL_K_1
C   9: NR     NUMBER OF BINDING CONSTRAINTS
C  10: SING   IF 1, THE BINDING CONSTRAINTS DON'T SATISFY THE REGULARITY CONDITION
C  11: UMIN   INFINITY NORM OF NEGATIVE PART OF MULTIPLIER
C  12: -------------
C  13: COND_R CONDITION NUMBER OF DIAGONAL PART OF QR-DECOMP. OF NORMALIZED
C             GRADIENTS OF BINDING CONSTRAINTS
C  14: COND_H CONDITION NUMBER OF DIAGONAL OF CHOLESKY-FACTOR OF UPDATED FULL HESSIAN
C  15: SCFO   THE RELATIVE DAMPING OF TANGENTIAL COMPONENT IF UPSI>TAUO/2
C  16: XNORM  L2-NORM OF X
C  17: DNORM  L2-NORM OF D (CORRECTION FROM QP -SUBPROBLEM, UNSCALED)
C  18: PHASE  -1 : INFEASIBILITY IMPROVEMENT PHASE, 0: INITIAL OPTIMIZATION
C             1: BINDING CONSTRAINTS UNCHANGED , 2: D SMALL, MARATOS CORRECTION
C             IN USE
C  19: C_K    NUMBER OF DECREASES OF PENALTY WEIGHTS
C  20: WMAX   INFINITY NORM OF WEIGHTS
C  21: SIG_K  STEPSIZE FROM UNIDIMENSIONAL MINIMIZATION (BACKTRACKING)
C  22: CFINCR NUMBER OF OBJECTIVE EVALUATIONS FOR STEPSIZE-ALGORITHM
C  23: DIRDER DIRECTIONAL DERIVATIVE OF PENALTY-FUNCTION ALONG D (SCALED)
C  24: DSCAL  SCALING FACTOR FOR D
C  25: COSPHI COS OF ARC BETWEEN D AND D_PREVIOUS. IF LARGER THETA , SIG LARGER THAN
C             ONE (UP TO SIGLA) IS TRIED
C  26: VIOLIS(0) NUMBER OF CONSTRAINTS NOT BINDING AT X BUT HIT DURING LINE SEARCH
C  27:        TYPE OF UPDATE FOR H: 1 NORMAL P&M-BFGS-UPDATE, 0 UPDATE SUPPRESSED,
C             -1 RESTART WITH SCALED UNIT MATRIX , 2 STANDARD BFGS, 3 BFGS MODIFIED
C             BY POWELLS DEVICE
C  28: NY_K/TK   MODIFICATION FACTOR FOR DAMPING THE PROJECTOR IN BFGS/PANTOJA-
C             MAYNE-TERM RESPECTIVELY
```

```
C  29: 1-MY_K/XSIK MODIFICATION FACTOR FOR DAMPING THE QUASI-NEWTON-RELATION IN BFGS
C             FOR UNMODIFIED BFGS NY_K SHOULD BE LARGER THAN UPDMYO (NEAR ONE)
C             AND 1-MY_K EQUAL ONE./PANTOJA-MAYNE TERM RESPECTIVELYC****
C  30: QPTERM 0, IF SING=-1, TERMINATION INDICATOR OF QP-SOLVER OTHERWISE
C             1: SUCCESSFUL, -1: TAU BECOMES LARGER THAN TAUQP WITHOUT SLACK-
C             VARIABLES BECOMING SUFFICIENTLY SMALL .
C             -2: INFEASIBLE QP-PROBLEM (THEORETICALLY IMPOSSIBLE)
C  31: TAUQP: WEIGHT OF SLACK-VARIABLES IN QP-SOLVER
C  32: INFEAS L1-NORM OF SLACK-VARIABLES IN QP-SOLVER
C
c optite is the termination parameter.
c optite ranges from -10 to 7, a negative value indicating some irregular
c event. optite+11 corresponds one the following messages:
c
c
c     'CONSTRAINT EVALUATION RETURNS ERROR WITH CURRENT POINT',
c     'OBJECTIVE EVALUATION RETURNS ERROR WITH CURRENT POINT',
c     'QPSOLVER: WORKING SET SINGULAR IN DUAL EXTENDED QP ',
c     'QPSOLVER: EXTENDED QP-PROBLEM SEEMINGLY INFEASIBLE ',
c     'QPSOLVER: NO DESCENT FOR INFEAS FROM QP FOR TAU=TAU_MAX',
c     'QPSOLVER: ON EXIT CORRECTION SMALL, INFEASIBLE POINT',
c     'STEPSIZESELECTION: COMPUTED D FROM QP NOT A DIR. OF DESCENT',
c     'MORE THAN MAXIT ITERATION STEPS',
c     'STEPSIZESELECTION: NO ACCEPTABLE STEPSIZE IN [SIGSM,SIGLA]',
c     'SMALL CORRECTION FROM QP,  INFEASIBLE POINT',
c     'KT-CONDITIONS SATISFIED, NO FURTHER CORRECTION COMPUTED',
c     'COMPUTED CORRECTION SMALL, REGULAR CASE ',
c     'STEPSIZESELECTION: X ALMOST FEASIBLE, DIR. DERIV. VERY SMALL',
c     'KT-CONDITIONS (RELAXED) SATISFIED, SINGULAR POINT',
c     'VERY SLOW PRIMAL PROGRESS, SINGULAR OR ILLCONDITONED PROBLEM',
c     'MORE THAN NRESET SMALL CORRECTIONS IN X ',
c     'CORRECTION FROM QP VERY SMALL, ALMOST FEASIBLE, SINGULAR ',
c     'NUMSM SMALL DIFFERENCES IN PENALTY FUNCTION,TERMINATE'
c
c   a "good" termination is case 8,9,10.(value 0,1,2)
c   In the cases 12-15 the final
c   precision might be quite poor, since in the singular case  DONLP2
c   reduces its requirements automatically by multiplying epsx by 100.
c
c runtim : the processes cpu-time
c phase :    PHASE  -1 : INFEASIBILITY IMPROVEMENT PHASE, 0: INITIAL OPTIMIZATION
c            1: BINDING CONSTRAINTS UNCHANGED , 2: D SMALL, MARATOS CORRECTION
c            IN USE
c
      REAL RUNTIM
      DOUBLE PRECISION ACCINF
      REAL  OPTITE
      INTEGER ITSTEP,PHASE
      COMMON/O8ITIN/OPTITE,ITSTEP,PHASE,RUNTIM,
     F      ACCINF(0:MAXIT,32)
c  component "0" introduced to overcome errors of some optimizing compilers
c  o8xdat contains information pertaining the value of current x's
c  x = current point, x0= previous point, x1 = tentative new point
c  xst = starting point
```

```
c  same convention for names upsi , psi , fx , sig, -norm
c
c  fx = function value , psi =scaled penalty-term
c  upsi = unscaled penalty term , phi=scf*f+psi
c  xmin = during stepsize increase current best point
c  (with values sigmin, upsim, psimin, fmin, phimin, resmin )
c  sig : current step-size
c  dirder = directional derivative of phi along d
c  cosphi=cos(arc(d,d0))
c  dscal: scaling of d (if too long, resulting in a too large change of x)
c  d = direction , dd = second order correction
c  difx = x_new-x_old
c  b2n = the projected gradient (in a system transformed by the
c  cholesky-factor of H) , b2n0 = same based on a reduced minimal
c  working set ,
      DOUBLE PRECISION UPSI,UPSI0,UPSI1,UPSIST,PSI,PSI0,
     F    PSI1,PSIST,PSIMIN,
     F    PHI,PHI0,PHI1,PHIMIN,FX,FX0,FX1,
     F    FXST,FMIN,B2N,B2N0,XNORM,X0NORM,SIG0,DSCAL,DNORM,DONORM
      DOUBLE PRECISION SIG,SIGMIN,DIRDER,COSPHI,UPSIM
      DOUBLE PRECISION X,X0,X1,XMIN,D,D0,DD,DIFX,RESMIN
      COMMON/O8XDAT/X(NX),X0(NX),X1(NX),XMIN(NX),RESMIN(NRESM),
     F             D(NX),D0(NX),DD(NX),DIFX(NX),
     F             XNORM,X0NORM,DNORM,DONORM,SIG,SIG0,SIGMIN,DSCAL,
     F             UPSI,UPSI0,UPSI1,UPSIST,UPSIM,
     F             PSI,PSI0,PSI1,PSIST,PSIMIN,
     F             PHI,PHI0,PHI1,PHIMIN,
     F             FX,FX0,FX1,FXST,FMIN,
     F             B2N,B2N0,DIRDER,COSPHI
c  gradf= gradient(f), gfn= norm of gradient of f ,
c  qgf = internal transform of gradf
c  gphi1 and gphi0 : gradient of lagrangian at new and old point
c  gres : gradients of constraints,
c  gresn(i)=max(1,norm of gradient(constraint_i)))
      DOUBLE PRECISION GRADF,GFN,QGF,GPHI0,GPHI1,GRES,GRESN
      COMMON/O8GRD/GRADF(NX),GFN,QGF(NX),GRES(NX,NRESM),
     F        GRESN(NRESM),GPHI0(NX),GPHI1(NX)
C**
c information pertaining the qr-decomposition of the working set's
c gradients
c perm = current valid row permutation making it continuous
c perm1 = new permutation
c colno = column permutation using column pivoting
c rank = rank of working set
c qr : matrix containing information of the householder-decomposition
c diag : diagonal of the r-part
c betaq : factor for reflection normals
c cscal: the column scaling
c colle: column lenght for the qp_problem (extended columns)
      INTEGER PERM,PERM1,COLNO,RANK
      DOUBLE PRECISION QR,BETAQ,DIAG,CSCAL,COLLE
      COMMON/O8RDAT/QR(NX,NRESM),BETAQ(NRESM),DIAG(NRESM),CSCAL(NRESM),
     F            COLLE(NRESM),
     F            COLNO(2*NRESM),PERM(NX),
```

```
     F            PERM1(NX),RANK
C    COLNO ALSO USED IN  O8QPSO WITH DOUBLE LENGTH !
c    information pertaining the constraints
c    val(i)=true, if gradient of constraint i has been evaluated at
c    the new point
c    i=0 corresponds to f
c    gconst(i)=.true. if constraint i is linear.
c    such a constraint gradient is evaluated once only
c    gunit : see descrition above
c    llow and lup are computed by {\tt DONLP2} using gunit
c    llow(i)=.true. if there is a finite lower bound for x(i)
c    lup(i)=.true. if there is a finite upper bound for x(i)
c    sort: internal order of evaluation of constraints
       LOGICAL VAL,GCONST,LLOW,LUP
       INTEGER GUNIT
       COMMON/O8GRI/VAL(0:NRESM),GCONST(0:NRESM),GUNIT(3,0:NRESM),
     F             LLOW(NX),LUP(NX)
c intakt= .true. : give output to prou-channel and to console
c inx= presently not used
c std=.true. use standard initialization
c this may be superseded by statements in setup
c te0 = .true. : give one-line-output for every step on console
c te1 = .true. : give post-mortem-dump of accumulated information in accinf
c te2 = .true. : give detailed protocol of iteration
c te3 = .true. : also print the gradients and the approximated hessian
c singul = .true. : the working set is singular
c ident = .true. : we try a modified step with the same point
c eqres = .true. : the working set stays fixed
c silent= .true. : give no output at all (user must evaluate information
c in the common-blocks himself)
c analyt = .true. : assume full precision in the gradients
c otherwise {\tt DONLP2} assumes epsdif relative precision in the gradients and
c automatically reduces its precision requirements
c cold = .true. : initialize weights, hessian and so on, otherwise use them
c unchanged in the first step.
       LOGICAL INTAKT,INX,STD,TE0,TE1,TE2,TE3,SINGUL
       LOGICAL IDENT,EQRES,SILENT,ANALYT,COLD
       COMMON/O8STPA/INTAKT,INX,STD,TE0,TE1,TE2,TE3,SINGUL,
     F        IDENT,EQRES,SILENT,ANALYT,COLD
c information pertaining the quasi newton update:
c A :the quasi-newton-update in cholesky-factors: upper triangle :the new one
c lower triangle + diag0 : the previous one
c matsc : scaling for identity to be used in case of restart
c  (computed internally)
       DOUBLE PRECISION A,DIAG0,MATSC
       COMMON/O8QN/A(NX,NX),DIAG0(NX),
     F           ,MATSC
c information pertaining the working set:
c violis: list of constraints hit during current linesearch
c alist: indices of  the working set
c bind : constraints binding at x (violated or at zero) bind(i)=1
c        (=0 otherwise)
c bind0 : the same for x0
c sort : internal evaluaution order for constraints
```

```
      INTEGER VIOLIS,ALIST,BIND,BIND0,SORT
      COMMON/O8RESI/BIND(NRESM),BIND0(NRESM),VIOLIS(0:NSTEP*NRESM),
     F               ALIST(0:NRESM),SORT(NRESM)
c u = mulipliers , u0 = old multipliers , w = weights ,
c w1 = tentative new weights , res = constraint values , res0  = previous
c constraint values, res1 = same at tentative point , resst =
c constraint values at starting point , scf = current scaling of f
c scf0 = damping factor for tangential part of d if infeasibility > tau0/2
c yu internal varable, stores multipliers for working set ,
c slack =  slack values in qp-problem , infeas = one-norm of slack ,
c work = work array
      DOUBLE PRECISION U,U0,W,W1,RES,RES0,RES1,RESST,SCF,SCF0,
     F                 YU,SLACK,INFEAS,WORK
      COMMON/O8RESD/RES(NRESM),RES0(NRESM),RES1(NRESM),RESST(NRESM),
     F            U(NRESM),U0(NRESM),W(NRESM),W1(NRESM),WORK(0:NRESM),
     F            YU(NRESM),SLACK(NRESM),SCF,SCF0,INFEAS
c n=dim(x), nh=dim(h), ng=dim(g), nres=nh+ng, nr=dim(working set)
c (may be larger than n)
      INTEGER N,NR,NRES,NH,NG
      COMMON/O8DIM/N,NH,NG,NR,NRES
c epsmac = machine precision, tolmac= smallest positive machine number
c (roughly approximated) both computed automatically by  DONLP2
c  deldif is a suitable difference stepsize for the automatic
c  numerical differentiation routine supplied with this distribution
      DOUBLE PRECISION EPSMAC,TOLMAC,DELDIF
      COMMON/O8MPAR/EPSMAC,TOLMAC,DELDIF
c iterma= maximum number of steps allowed. (30*n a good value from
c experience)
c del = current delta, del0 = maximum of delta , del01=del0/10 ,
c delmin:  constraint are satisfied if |h_i(x)|<=delmin , g_j(x)>=-delmin
c respectively
c tau0 : upper bound for the unscaled penalty-term, i.e. the one-norm of the
c constraint violation
c tau gives a weight between descent for f and infeasibility.
c the smaller tau, the  more may f be scaled down.
c also used as additive increase for the penalty-weights
c ny increase factor for penalty-weights when computed from the multipliers
c smalld : use second order correction if ||d||<= smalld and the working
c set stays fixed
c smallw: multipliers absolutely smaller than smallw are considered as zero
c rho : working set considered singular if condition number of r-part
c of its qr-decomposition (after scaling the gradients to one) is larger than
c 1/rho.
c rho1: if condition number of quasi-newton-update of hessian becomes larger
c than 1/rho1, a restart with a scaled unit matrix is performed
c eta: level required for descent if weights are to be diminished,
c computed automatically
c epsx : termination parameter. successful termination is indicated if
c (roughly speaking) the kuhn-tucker-criteria are satisfied within an error of
c epsx
c epsphi: variable entering the termination criterion for small progress
c in the penalty function, see under termination criteria
c c1d: inactivation of an inequality contraint occurs if a multiplier*c1d
c is less than - a scaled projected gradient at the current point
```

```
c scfmax: the internal automatic scaling of f is bounded by 1/scfmax from below
c and scfmax from above. this scaling aims at  one multiplier at least being
c in the  order of magnitude of one.
c tauqp: the penalty-parameter in the qp-problem
c taufac: increase factor for tauqp
c taumax: maximum allowable tauqp
c updmy0 : presently not used
c
      INTEGER ITERMA,IFILL1
      DOUBLE PRECISION DEL,DEL0,DEL01,DELMIN,TAU0,TAU,NY
      DOUBLE PRECISION SMALLD,SMALLW,RHO,RHO1,ETA,EPSX,EPSPHI,C1D,
     F                 SCFMAX,UPDMY0,TAUQP,TAUFAC,TAUMAX
      COMMON/O8PAR/DEL0,DEL01,DEL,DELMIN,TAU0,TAU,
     F             SMALLD,SMALLW,RHO,RHO1,ETA,NY,EPSX,EPSPHI,
     F             C1D,SCFMAX,TAUQP,TAUFAC,TAUMAX,UPDMY0,
     F             ITERMA,IFILL1
c the parameters of the stepsize-algorithm:
c alpha=smallest reduction factor, beta=feasible increase factor
c for change of x-norm adding the correction d. theta: if theta<=cos(arc(d,d0))
c then stepsizes > 1 are tried for d, where d0=previous correction.
c sigsm= smallest stepsize allowed, sigla = largest stepsize allowed,
c delta = the armijo-parameter, stptrm = termination indicator
c for stepsize-algorithm, delta1= armijo-parameter for reduction
c of infeasibility, stmaxl= min(sigla, maximum stepsize for which the
c point on the ray x+sigma*d, projected to the bounds, changes)
      DOUBLE PRECISION ALPHA,BETA,THETA,SIGSM,SIGLA,DELTA,STPTRM
      DOUBLE PRECISION DELTA1,STMAXL
      COMMON/O8STEP/ALPHA,BETA,THETA,SIGSM,SIGLA,DELTA,STPTRM,
     F              DELTA1,STMAXL
c evaluation counters cf= number of calls to f, cgf= number of calls to grad(f)
c cfincr= number of function values used in the current stepsize-selection
c cres(i)= number of calls to individual constraints (bounds not counted)
c cgres = same for the gradients.
c i=1,..,nh corresponds  h and i=nh+1,...,nh+ng  g.
      INTEGER CF,CGF,CFINCR,CRES,CGRES
      COMMON/O8CNT/CF,CGF,CFINCR,CRES(NRESM),CGRES(NRESM)
c FFUERR error indicator for evaluating f
c CFUERR(i) error indicator for evaluating a constraining function
c  i=1,..,nh corresponds to  h and i=nh+1,...,nh+ng to  g.
c these are all initialized by .FALSE.
      LOGICAL FFUERR,CFUERR(NRESM)
      COMMON/O8ERR/FFUERR,CFUERR
c clow = number of times the weights have been dimished
c lastdw = last step number with weights diminished
c lastup = last step number with weights increased
c lastch = last step number with weights changed
      INTEGER CLOW,LASTDW,LASTUP,LASTCH
      COMMON/O8WEI/CLOW,LASTDW,LASTUP,LASTCH
c  problem identifier
      CHARACTER*40 NAME
      COMMON/O8ID/NAME
c  precision given by the u s e r,  computing gradients numerically.
c  not used if analyt=.true.
      DOUBLE PRECISION EPSDIF
```

```
      COMMON/O8DER/EPSDIF
c  channel numbers for i/o of intermediate and final results
      INTEGER PROU,MEU
      COMMON/O8IO/PROU,MEU
c  ug, og and delfac (lower and upper bounds for x(i) and a special
c  scaling weight for detection of binding bounds) are computed by
c  DONLP2 itself, using GUNIT and evaluation of the bounds using x=0
      DOUBLE PRECISION UG(NX),OG(NX),DELFAC(NRESM)
      COMMON/O8BD/UG,OG,DELFAC
c  reset the quasi-newton-update after "nreset" small or unsuccessful steps
c  numsm the number of allowed consecutive small changes in the penalty function
      INTEGER NRESET,NUMSM
      COMMON/O8RST/NRESET,NUMSM
c  here we store the starting point , for later printing only
      DOUBLE PRECISION XST(NX)
      COMMON/O8STV/XST
      INTRINSIC SQRT,EXP,LOG,MAX,MIN,ABS
```

The following common-area serves as an interface for transmitting and receiving information from a black-box evaluation routine which a user might wish to use:

```
C**** IF BLOC=.TRUE. THEN IT IS ASSUMED THAT FUNCTIONEVALUATION TAKES PLACE
C**** AT ONCE IN AN EXTERNAL ROUTINE AND
C**** THAT USER_EVAL HAS BEEN CALLED (WHICH IN TURN CALLS
C**** EVAL_EXTERN(MODE) BEFORE CALLING FOR EVALUATION OF FUNCTIONS
C**** THE LATTER THEN SIMPLY CONSISTS IN COPYING DATA FROM COMMON O8FUEXT
C**** TO DONLP2'S OWN DATA-AREAS.
C**** CORR IS SET TO TRUE BY DONLP2, IF THE INITIAL X DOES NOT SATISFY
C**** THE BOUND CONSTRAINTS. X IS MODIFIED IN THIS CASE
C**** DIFFTYPE=1,2,3 NUMERICAL DIFFERENTIATION BY THE ORDINARY FORWARD
C**** DIFFERENCES, BY CENTRAL DIFFERENCES OR BY RICHARDSON-EXTRAPOLATION
C**** OF ORDER 6, REQUIRING N, 2N , 6N ADDITIONAL FUNCTION EVALUATIONS
C**** RESPECTIVELY
C**** EPSFCN IS THE ASSUMED PRECISION OF THE FUNCTION EVALUATION, TO BE
C**** SET BY THE USER
C**** TAUBND: AMOUNT BY WHICH BOUND CONSTRAINTS MAY BE VIOLATED DURING
C**** FINITE DIFFERENCING, SET BY THE USER
      LOGICAL SCALE,BLOC,VALID,CORR
      INTEGER DIFFTYPE
C**** XTR : THE CURRENT DESIGN X
C**** FU : THE VECTOR OF FUNCTION VALUES IN THE ODER 0=OBJECTIVE FUNCTION,
C**** 1,...,NH EQUALITY CONSTRAINTS, NH+1,NH+NG INEQUALITY CONSTRAINTS.
C**** SIMILARLY FUGRAD(.,I) THE CORRESPONDING GRADIENTS
C**** WITH BLOC=.TRUE. AND ANALYT=.FALSE. ONLY FU MUST BE COMPUTED, USING XTR
C**** AS INPUT
      DOUBLE PRECISION XTR(NX),XSC(NX),FU(0:NRESM),FUGRAD(NX,0:NRESM),
     F                 FUD(0:NRESM,1:6),EPSFCN,TAUBND
      COMMON/O8FUEXT/XTR,XSC,FU,FUGRAD,FUD,EPSFCN,TAUBND
      COMMON/O8FUPAR/SCALE,BLOC,VALID,CORR,DIFFTYPE
```

9 RESULTS OBTAINED IN TESTING DONLP 2

These can be found in the accompanying papers describing the underlying theory and in the file donlp2results accompanying this distribution.

10 SOME ADVICE

How to proceed:

1) Adapt sysdepen.f to your system. If nothing works, use dummy.f instead of sysdepen.f . In this case you will miss information on date and time of the run and the cpu-time used. Nothing else.

2) Check whether the setting in `O8PARA.INC` is appropriate for your machine. The present setting needs about 8MB user memory (due to the setting of `NRESM` and `NX`), the maximum number of variables and constraints allowed) Reset these parameters appropriately. You need about

```
NX*(NX+NRESM+10)+NRESM*(50+NSTEP)+ (NX+NRESM)*((NX+NRESM)*2+8)+CONST
```

   double precision numbers to store. **Observe that there must hold `NRESM`≥`NX`.**

3) If you work on a UNIX-system, edit the Makefile (you possibly must modify the compilerflags).

4) Having written your problem-description, first compile and link it with testgrad.f in order to check your analytical gradients if you have supplied these. You should supply analytical expressions whenever this is possible. The numerical differentiation feature will normally degrade reliability and efficiency. With `DIFFTYPE=3` , which of course is quite costly, all testcases but one could be solved to almost the same precision as with analytical gradients. However the costs are very high ($6n$ function evaluations for a single gradient).

5) !!! In writing your example, remember you must completely specify your problem, such that the evaluation of any function is well defined in the box given by the bound constraints. E.g. if you have to use log, $\sqrt{}$, x^y with y real or double, /(....) when you must add the appropriate lower bounds $\geq \epsilon > 0$ (or $\leq -\epsilon < 0$ for denominators) for the expressions occuring there.

6) !!! Many machines nowadays use IEEE-arithmetic and have floating-errors disabled. Optimizing NaN's of course is useless. If possible, use linker- or compiler switches which enable error reporting of floating point errors in order to detect such behaviour.

7) Compile your application programs and link with donlp2.o , donlp2main.o and sysdepen.o . If you work on a UNIX-system, you may copy the suite of function evaluation routines to run.f and type "make exe"

8) !!! If `DONLP2` behaves "strange", proceed as follows: use

```
      SUBROUTINE SETUP
      'INCLUDE O8COMM.INC'
      TE2=.TRUE.
C**** the following for small N, NG only
      TE3=.TRUE.
      .......
      RETURN
      END
```

   after runnning the testcase, grep for 'NaN' in the *.PRO-file. If these are present, you can be sure that there is an error in your coding of the example of the kind indicated above.