IMM
INFORMATICS AND MATHEMATICAL MODELLING

Technical University of Denmark
DK-2800 Kongens Lyngby – Denmark

J. No. RS
31.1.2002
HBN/ms

ROBUST SUBROUTINES FOR NON-LINEAR OPTIMIZATION

Kaj Madsen

Hans Bruun Nielsen

Jacob Søndergaard

TECHNICAL REPORT

IMM-REP-2002-02

IMM

Contents

1. Introduction

This report presents a package of robust and easy-to-use Fortran subroutines for solving unconstrained and constrained non-linear optimization problems. The intention is that the routines should use the currently best algorithms available. All routines have standardized calls, and the user does not have to worry about special parameters controlling the iteration. For convenience we include an option for numerical checking of the user's implementation of the gradient.

The present report is a new and updated version of a previous report NI-90-06 with the same title. The software changes are listed in Section 1.5.

1.1. Problem Formulation

We consider minimization of functions of vector arguments, $F : \mathbb{R}^n \mapsto \mathbb{R}$. The function may be a norm of vector valued function $\mathbf{f} : \mathbb{R}^n \mapsto \mathbb{R}^m$. For the scalar case the user must provide a subroutine, which – for a given $\mathbf{x}$ – returns both the function value $F(\mathbf{x})$ and the *gradient* $\mathbf{g}(\mathbf{x}) \in \mathbb{R}^n$, defined by

$$\mathbf{g} = \begin{bmatrix} \dfrac{\partial F}{\partial x_1} \\ \vdots \\ \dfrac{\partial F}{\partial x_n} \end{bmatrix} . \tag{1.1}$$

In case of a vector function the user's subroutine must return the vector $\mathbf{f}(\mathbf{x})$ and the *Jacobian matrix* $\mathbf{J}(\mathbf{x}) \in \mathbb{R}^{m\times n}$, defined by

$$\mathbf{J} = \begin{bmatrix} \dfrac{\partial f_1}{\partial x_1} & \cdots & \dfrac{\partial f_1}{\partial x_n} \\ \vdots & & \vdots \\ \dfrac{\partial f_m}{\partial x_1} & \cdots & \dfrac{\partial f_m}{\partial x_n} \end{bmatrix} , \tag{1.2}$$

i.e., the ith row in $\mathbf{J}$ is the gradient of f_i, the ith component of $\mathbf{f}$.

For an efficient performance of the optimization algorithm the function and the gradients must be implemented without errors. It is not possible to check the correctness of the implementation of F (or $\mathbf{f}$), but we provide the possibility of checking the corresponding gradient (or Jacobian).

1.2. Checking the Gradients

This is done by *difference approximations.* First, consider a scalar function $F(\mathbf{x})$: For given $\mathbf{x}$ and steplength h we compute

$$\left.\begin{array}{l} D_j^F = (F(\mathbf{x}+h\mathbf{e}_j) - F(\mathbf{x}))/h \\ D_j^B = (F(\mathbf{x}) - F(\mathbf{x}-\frac{1}{2}h\mathbf{e}_j))/(\frac{1}{2}h) \\ D_j^E = (D_j^F + 2D_j^B))/3 \end{array}\right\}, \quad j=1,\ldots,n , \tag{1.3}$$

where $\mathbf{e}_j$ is the jth unit vector (the jth column of $\mathbf{I}$), and the superscripts stand for Forward, Backward and Extrapolated difference approximation, respectively.

We assume that F is three times continuously differentiable with respect to each of its arguments. Then a Taylor expansion from $\mathbf{x}$ shows that

$$\begin{aligned} F(\mathbf{x}) + \eta\mathbf{e}_j) &= F(\mathbf{x}) + \eta\frac{\partial F}{\partial x_j}(\mathbf{x}) + \tfrac{1}{2}\eta^2\frac{\partial^2 F}{\partial x_j^2}(\mathbf{x}) + \mathcal{O}(\eta^3) \\ &= F(\mathbf{x}) + \eta g_j(\mathbf{x}) + \eta^2 S_j(\mathbf{x}) + \mathcal{O}(\eta^3) \quad . \end{aligned} \tag{1.4}$$

Inserting this in (1.3) we see that

$$\begin{aligned} D_j^F &= g_j + hS_j + \mathcal{O}(h^2) \\ D_j^B &= g_j - \tfrac{1}{2}hS_j + \mathcal{O}(h^2) \quad \text{with} \quad S_j = \tfrac{1}{2}\frac{\partial^2 F}{\partial x_j^2}(\mathbf{x}) \ . \\ D_j^E &= g_j + \mathcal{O}(h^2) \end{aligned} \tag{1.5}$$

Now, let G_j denote the jth component of the gradient as returned from the user's subroutine, and let

$$G_j = g_j - \psi_j \ , \tag{1.6}$$

where $\psi_j = \psi_j(\mathbf{x})$ is zero if the implementation is correct. Inserting this in (1.5) we get

$$\begin{aligned} \delta_j^F &\equiv D_j^F - G_j = \psi_j + hS_j + \mathcal{O}(h^2) \ , \\ \delta_j^B &\equiv D_j^B - G_j = \psi_j - \tfrac{1}{2}hS_j + \mathcal{O}(h^2) \ , \\ \delta_j^E &\equiv D_j^E - G_j = \psi_j + \mathcal{O}(h^2) \ \ . \end{aligned} \tag{1.7}$$

If $\psi_j = 0$, $S_j \neq 0$ and h is so small that the last term in each right-hand side of (1.7) can be neglected, then we can expect $\delta_j^B \simeq -\frac{1}{2}\delta_j^F$ and δ_j^E to be of the order of magnitude $(\delta_j^F)^2$. Also, if the approximation is recomputed with h replaced by θh, where $0 < \theta < 1$, then both δ_j^F and δ_j^B are reduced by a factor θ, while δ_j^E is reduced by a factor θ^2.

If $\psi_j \neq 0$ and h is sufficiently small, then the error will be recognized by $\delta_j^F \simeq \delta_j^B \simeq \delta_j^E \simeq \psi_j$.

The computed values are affected by rounding errors. Especially, instead of $F(\mathbf{z})$ we get $\mathsf{fl}(F(\mathbf{z})) = F(\mathbf{z}) + \varepsilon$. The best that we can hope for is that $|\varepsilon| \leq u \cdot |F(\mathbf{x})|$, where u is the "unit round-off". (The subroutines use `REAL*8` corresponding to $u = 2^{-53} \simeq 10^{-16}$ on most computers). This has the consequence that for the computed difference approximations (1.7) should be replaced by

$$\begin{aligned} |\delta_j^F| &\leq |\psi_j + hS_j| + A_j h^{-1} + \mathcal{O}(u) + \mathcal{O}(h^2) \ , \\ |\delta_j^B| &\leq |\psi_j - \tfrac{1}{2}hS_j| + A_j h^{-1} + \mathcal{O}(u) + \mathcal{O}(h^2) \ , \\ |\delta_j^E| &\leq |\psi_j| + B_j h^{-1} + \mathcal{O}(u) + \mathcal{O}(h^2)) \ \ , \end{aligned} \tag{1.8}$$

where A_j and B_j are positive values, that depend on F and $\mathbf{x}$, but not on h. In the case of correct implementation of the gradient, (1.8) shows that for large h the errors are dominated by truncation error, while effects of rounding errors dominate if h is too small. Assuming that $|S_j|$ and A_j are of the same order of magnitude, the smallest error with the forward and backward difference approximations is obtained with $h \simeq \sqrt{u}\|\mathbf{x}\|$. Similarly, we can expect that $|\delta_j^E|$ is minimal for $h \simeq \sqrt[3]{u}\|\mathbf{x}\|$.

In order to enhance accuracy the one-sided difference approximations in (1.3) should be computed by the formulae

$$D_j^F = \frac{F(\mathbf{x}+h\mathbf{e}_j) - F(\mathbf{x})}{\widehat{h}_j}, \quad D_j^B = \frac{F(\mathbf{x}) - F(\mathbf{x}-\frac{1}{2}h\mathbf{e}_j)}{\widetilde{h}_j} \ , \tag{1.9a}$$

where $\widehat{h}_j$ and $\widetilde{h}_j$ are the **actual** steps,

$$\widehat{h}_j = \mathsf{fl}(\mathsf{fl}(x_j+h) - x_j), \quad \widetilde{h}_j = \mathsf{fl}(x_j - \mathsf{fl}(x_j - \tfrac{1}{2}h)) \ . \tag{1.9b}$$

Note that if $|h|$ is too small, then we get $\widehat{h}_j = 0$ and/or $\widetilde{h}_j = 0$. In that case the gradient checker gives an error return.

The subroutines MINF and MINCF deal with scalar functions of vector variables. If they are called with the option of checking the gradient, then they return $\{\delta_j^A, j^A\}$ defined by

$$j^A = \underset{j=1,\ldots,n}{\text{argmax}} \{|\delta_j^A|\}, \qquad \delta^A = \delta_{j^A}^A \tag{1.10}$$

for $A = F, B, E$, i.e. δ^A is the extreme value and j^A is its position.

The other subroutines deal with problems where $F(\mathbf{x})$ is some norm of a vector function $\mathbf{f}(\mathbf{x})$. In this case it is relevant to check the implementation of the Jacobian $\mathbf{J}(\mathbf{x})$, (1.2). The ith row in $\mathbf{J}$ is the gradient of f_i, the ith component of $\mathbf{f}$, and a straightforward generalization of (1.3) is

$$\left.\begin{aligned} D_{ij}^F &= (f_i(\mathbf{x}+h\mathbf{e}_j) - f_i(\mathbf{x}))/h \\ D_{ij}^B &= (f_i(\mathbf{x}) - f_i(\mathbf{x}-\tfrac{1}{2}h\mathbf{e}_j))/(\tfrac{1}{2}h) \\ D_{ij}^E &= (D_{ij}^F + 2D_{ij}^B)/3 \end{aligned}\right\}, \quad \begin{cases} i=1,\ldots,m \\ j=1,\ldots,n \end{cases}, \tag{1.11}$$

leading to

$$\begin{aligned} \delta_{ij}^F &\equiv D_{ij}^F - \mathtt{J}_{ij} = \psi_{ij} + hS_{ij} + \mathcal{O}(h^2) + \mathcal{O}(uh^{-1}) \ , \\ \delta_{ij}^B &\equiv D_{ij}^B - \mathtt{J}_{ij} = \psi_{ij} - \tfrac{1}{2}hS_{ij} + \mathcal{O}(h^2) + \mathcal{O}(uh^{-1}) \ , \\ \delta_{ij}^E &\equiv D_{ij}^B - \mathtt{J}_{ij} = \psi_{ij} + \mathcal{O}(h^2) + \mathcal{O}(uh^{-1}) \ , \end{aligned} \tag{1.12}$$

where $\mathtt{J}_{ij}$ is the (i,j)th element in the implemented Jacobian, ψ_{ij} is its error and $S_{ij} = \frac{1}{2}\partial^2 f_i/\partial x_j^2$. If the subroutines are called with the option of checking the Jacobian, they return δ^A, i^A, j^A for $A = F, B, E$ defined as in (1.10).

1.3. Examples

First, consider the scalar problem ($n = 2$)

$$F(\mathbf{x}) = \cos x_1 + e^{2x_2}, \qquad \mathbf{g}(\mathbf{x}) = \begin{bmatrix} -\sin x_1 \\ 2e^{2x_2} \end{bmatrix} \ , \tag{1.13}$$

implemented by the subroutine (note the sign error in g_1)

```
      SUBROUTINE FDF(N,X,DF,F)
c   Scalar function with gradient error
      INTEGER           N
      DOUBLE PRECISION  X(N),DF(N),F,E
      INTRINSIC         COS,EXP,SIN
        E = EXP(2D0 * X(2))
        F = COS(X(1)) + E
        DF(1) = SIN(X(1))
        DF(2) = 2D0 * E
      RETURN
      END
```

If we call e.g. MINF with the checking option and $h = 10^{-3}$, we get the results

$$\begin{aligned} \texttt{Max|DF| = 1.4778E+01}, \quad \delta^F &= \texttt{-1.6832E+00}, \quad \mathtt{j}^F = 1 \ , \\ \delta^B &= \texttt{-1.6828E+00}, \quad \mathtt{j}^B = 1 \ , \\ \delta^E &= \texttt{-1.6829E+00}, \quad \mathtt{j}^F = 1 \ , \end{aligned}$$

indicating an error in the first element of the computed gradient. After correcting the error we get

$$\delta^F = \texttt{1.4788E-02}, \quad \delta^B = \texttt{-7.3866E-03}, \quad \delta^E = \texttt{4.9273E-06}$$

h	δ^F	j^F	δ^B	j^B	δ^E	j^E
1	3.24E+01	2	-5.44E+00	2	7.19E+00	2
1.0E-01	1.58E+00	2	-7.15E-01	2	5.06E-02	2
1.0E-02	1.49E-01	2	-7.36E-02	2	4.94E-04	2
1.0E-03	1.48E-02	2	-7.39E-03	2	4.93E-06	2
1.0E-04	1.48E-03	2	-7.39E-04	2	4.93E-08	2
1.0E-05	1.48E-04	2	-7.39E-05	2	6.38E-10	2
1.0E-06	1.48E-05	2	-7.39E-06	2	-1.38E-09	2
1.0E-07	1.49E-06	2	-7.34E-07	2	-5.86E-09	1
1.0E-08	4.71E-08	2	-1.31E-07	2	-7.13E-08	2
1.0E-09	7.57E-07	2	-1.68E-06	1	-1.09E-06	1
1.0E-10	-4.58E-06	2	-7.01E-06	1	-4.05E-06	1
1.0E-11	2.21E-05	2	-1.67E-04	1	-1.08E-04	1
1.0E-12	1.18E-03	2	1.18E-03	2	1.18E-03	2

Table 1.1. *Gradient check with varying h*

and $j^F = j^B = j^E = 2$. This agrees with expectation: $\delta^B \simeq -\frac{1}{2}\delta^F$ and δ^E is orders of magnitude smaller.

To illustrate the behaviour for varying steplength we give results in Table 1.1 for the extreme values of of the differences for $h = 1, 10^{-1}, \cdots, 10^{-12}$.

For large values of h (the first two rows) the results are dominated by truncation error. Then follows a series of results where the δ^A behave as described above **and** $\delta^A(0.1h) \simeq 0.1\delta^A(h)$ for the forward and backward approximation, while $\delta^E(0.1h) \simeq 0.01\delta^E(h)$. Finally, for the smallest h-values rounding errors dominate. For the one-sided approximations this happens for $h \simeq 10^{-7} \simeq 10\sqrt{u}$ and for the extrapolated approximation the turning point is $h \simeq 10^{-5} \simeq 2\sqrt[3]{u}$, where $u = 2^{-53} \simeq 10^{-16}$ is the unit round-off used for the computations. This agrees with the discussion after (1.8).

1.4. Test Functions

Many of the examples in this report use the following set of functions, $\mathbf{f} : \mathbb{R}^2 \mapsto \mathbb{R}^3$, originally given by Beale [1],

$$\begin{aligned} f_1(\mathbf{x}) &= 1.5 - x_1(1 - x_2) \\ f_2(\mathbf{x}) &= 2.25 - x_1(1 - x_2^2) \\ f_3(\mathbf{x}) &= 2.625 - x_1(1 - x_2^3) \end{aligned} \tag{1.14}$$

1.5. Modifications

The following modifications were made compared with the version of the package described in [12],

1° `MINF` is completely new. It provides an option for warm start and requires a smaller workspace.

2° `MINL2` is completely new and requires a smaller workspace.

3° The gradient checker has been changed in all subroutines so that the difference approximations are more accurate (cf. (1.9)) and the extreme differences are returned with their sign.

4° The LP solver in `MINCIN` has been replaced by a corrected version, dated June 2001.

5° The subroutines dealing with vector functions return both $F(\mathbf{x})$ and $\mathbf{f}(\mathbf{x})$. The old versions returned $\mathbf{f}(\mathbf{x})$ only.

6° The subroutines `MINCL1` and `MINCIN` return the values of the constraints together with $F(\mathbf{x})$ and $\mathbf{f}(\mathbf{x})$.

2. Unconstrained Optimization

2.1. MINF. Minimization of a Scalar Function

Purpose. Find $\mathbf{x}^*$ that minimizes $F(\mathbf{x})$, where $\mathbf{x} = [x_1, \ldots, x_n]^\top \in \mathbb{R}^n$ is the vector of unknown parameters and the scalar objective function F is twice continuously differentiable. The user must supply a subroutine that evaluates $F(\mathbf{x})$ and the gradient $\mathbf{g}(\mathbf{x})$. There are options for checking the implementation of $\mathbf{g}$ and for warm start of the algorithm.

Method. The algorithm is a quasi-Newton method with BFGS updating of the inverse Hessian[1)] and soft line search,, see e.g. [3, Chapters 9 (and 6)] or [15, Chapters 3, 4 and 8]. This is combined with a trust region type monitoring of the input to the line search algorithm, see [14].

Remark. The user has to give an initial value for Δ, the length of the step $\mathbf{h}$ between two consecutive iterates. Ideally, a step of this length is accepted by the line search, and during iteration this "trust region radius" is adjusted by the output from the line search algorithm, see [14, Section 2.3].

The algorithm is not very sensitive to Δ_0, the initial value of this parameter. If the function F is almost linear, then we recommend to use a value for Δ_0, which is an estimate of the distance between $\mathbf{x}_0$ and the solution $\mathbf{x}^*$. Otherwise, we recommend $\Delta_0 = 0.1\|\mathbf{x}_0\|$.

Origin. `MINF` is a modified version of the subroutine `UCMINF`, [14]. The modification was made to make it consistent with the other routines in the present package.

Use of other Subprograms. The subroutine calls the following BLAS (see [4]) subroutines and functions

Level 0: `LSAME` `XERBLA`
Level 1: `DAXPY` `DCOPY` `DDOT` `DNRM2` `DSCAL` `IDAMAX`
Level 2: `DSPMV` `DSPR` `DSPR2`

Copies of these were obtained from
http://www.netlib.org/blas/blas.tgz
and are included in the file minf.f. At lines 65, 261 and 370 you can find instructions about how to modify the file if BLAS is available on your computer.

Use. The subroutine call is

```
CALL MINF(FDF,N,X,DX,EPS,MAXFUN,W,IW,ICONTR)
```

The parameters are

`FDF` `SUBROUTINE` written by the user with the following declaration

```
SUBROUTINE FDF(N,X,DF,F)
REAL*8 X(N),DF(N),F
```

It must calculate the value of the objective function and its gradient at the point $\mathbf{x} = [\mathtt{X(1)}, \ldots, \mathtt{X(N)}]^\top$ and store these numbers as follows,

$\mathtt{F} = F(\mathbf{x}), \qquad \mathtt{DF(J)} = \dfrac{\partial F}{\partial x_{\mathtt{J}}}(\mathbf{x}), \quad \mathtt{J} = 1, \ldots \mathtt{N}$

The name of this subroutine (which can be chosen freely by the user) must appear in an `EXTERNAL` statement in the calling program.

`N` `INTEGER`. Number of unknowns, n. Must be positive. Is not changed.

`X` `REAL*8 ARRAY` with `N` elements. The use depends on the entry value of `ICONTR`.

[1)] The Hessian $\mathbf{H}(\mathbf{x})$ is the matrix of second derivatives, $H_{ij} = \dfrac{\partial^2 F}{\partial x_i \partial x_j}$.

ICONTR > 0 : *On entry*: Initial approximation to $\mathbf{x}^*$.
On exit: Computed solution.
ICONTR ≤ 0 : Point at which the Jacobian should be checked. Not changed.

DX REAL*8. The use depends on the entry value of ICONTR.
ICONTR > 0 : "Trust region radius", see **Remark** above.
On entry: DX $= \Delta_0$. Must be positive.
On exit: Final trust region radius.
ICONTR ≤ 0 : Gradient check with DX used for h in (1.3). Must be significantly nonzero.
Is not changed.

EPS REAL*8. Used only if the entry value of ICONTR is positive. Desired accuracy.
The algorithm stops when it suggests to change the iterate from $\mathbf{x}_k$ to $\mathbf{x}_k+\mathbf{h}_k$ with $\|\mathbf{h}_k\| <$ EPS$\cdot(\|\mathbf{x}_k\| +$ EPS$)$. Must be positive. Is not changed.

MAXFUN INTEGER. Used only if the entry value of ICONTR is positive.
On entry: Upper bound on the number of calls of FDF. Must be positive.
On exit: Number of calls of FDF.

W REAL*8 ARRAY with IW elements. Work space.
On entry: If ICONTR>2, then W(4N+1,..,4N+$\frac{1}{2}$N(N+1)) should hold (an approximation to) the lower triangle of the inverse of $\mathbf{H}$(X), stored columnwise. This matrix must be positive definite.
If ICONTR≤ 2, then entry values of W are not used.

On exit with $\texttt{ICONTR}_{\text{entry}} > 0$:
W(1) $= F$(X), the computed minimum,
W(2) $= \max\ |g_i(\texttt{X})|$,
W(3) = length of the last step,
W(4,..,N+3) $= \mathbf{g}$(X), the gradient at X,
W(4N+1,..,4N+$\frac{1}{2}$N(N+1)): lower triangle of the final approximation to the inverse Hessian, stored columnwise.

On exit with $\texttt{ICONTR}_{\text{entry}} \leq 0$: Results of the gradient check are returned in the first 7 elements of W as follows, cf. (1.10)

W(1)	Maximum element in |DF|.
W(2), W(5)	δ^F and j^F.
W(3), W(6)	δ^F and j^F.
W(4), W(7)	δ^F and j^F.

In case of an error the indices W(5..7) point out the erroneous gradient component.

IW INTEGER. Length of work space W. Must be at least

$$\begin{cases} \max\{\frac{1}{2}n(n{+}11),\ 7\}. & \text{if } \texttt{ICONTR}_{\text{entry}} \leq 2 \\ n\cdot\max\{n{+}1,\ \frac{1}{2}(n{+}11)\} & \text{otherwise.} \end{cases}$$

Is not changed.

ICONTR INTEGER. *On entry*: Controls the computation,
ICONTR ≤ 0 : Check gradient. No iteration.
ICONTR > 0 : Start minimization with the inverse Hessian $\mathbf{D}$ initialized to the unit matrix if ICONTR≤ 2. Otherwise, $\mathbf{D}_0$ is given in W(4N+1,..,4N+$\frac{1}{2}$N(N+1)).
If ICONTR $= 2$ or ICONTR > 3, then information is printed during the iteration.

On exit: Information about performance,
ICONTR $= 0$: Successful call.
ICONTR $= 2$: Iteration stopped because too many iterations were needed, see MAXFUN.
ICONTR $= 3$: Iteration stopped by zero step from the line search.

ICONTR < 0 : Computation did not start for the following reason,
ICONTR $= -2$: N ≤ 0
ICONTR $= -4$: |DX| is too small
ICONTR $= -5$: EPS ≤ 0
ICONTR $= -6$: MAXFUN ≤ 0
ICONTR $= -7$: Given $\mathbf{D}$ is not positive definite
ICONTR $= -8$: IW is too small

Example. Minimize

$$F(\mathbf{x}) = \sin(x_1 x_2) + 2e^{x_1+x_2} + e^{-x_1-x_2} \ .$$

```
      PROGRAM  TINF
*********************************************************************
* Test MINF.  27.1.2002
*********************************************************************
      IMPLICIT          NONE
      INTEGER           ICONTR,IW,MAXFUN,N
      DOUBLE PRECISION  DX,EPS,X(2),W(18)
      EXTERNAL          FDF
C       ... Set parameters
        DATA  N, IW, MAXFUN, EPS,     X
     &       /2, 18,   25,  1D-10, 1D0,2D0/
        ICONTR = 0
C        ICONTR = 1
        IF  (ICONTR .LE. 0)  THEN
C         ... Check gradients
          DX = 1D-3
          CALL MINF(FDF,N,X,DX,EPS,MAXFUN,W,IW,ICONTR)
          IF  (ICONTR .LT. 0)  THEN
            WRITE(6,10) -ICONTR
          ELSE
            WRITE(6,20) W(1), W(2),INT(W(5))
            WRITE(6,30) 'Backward', W(3),INT(W(6))
            WRITE(6,30) ' Extrap.', W(4),INT(W(7))
          ENDIF
        ELSE
C         ... Optimize
          DX = 1D0
          CALL MINF(FDF,N,X,DX,EPS,MAXFUN,W,IW,ICONTR)
          IF  (ICONTR .LT. 0)  THEN
            WRITE(6,10) -ICONTR
          ELSE
            WRITE(6,40) ICONTR,MAXFUN, X(1),X(2), W(1),W(2)
          ENDIF
        ENDIF
  10    FORMAT('Parameter number',I3,'  is outside its range')
  20    FORMAT('Test of gradient.  Max|DF| = ',1P1D11.4/
     &    'Max difference,    Forward :',1P1D12.4,'  at  j =',I3)
  30    FORMAT(18X,A8,' :',1P1D12.4,'  at  j =',I3)
  40    FORMAT('Optimization.  ICONTR =',I2,'.',I4,' calls of FDF'
     &     /'   x =',1P2D17.9/'F(x) =',1P1D17.9,4X,
     &     '||g(X)|| =',1P1D10.2)
        STOP
       END

      SUBROUTINE FDF(N,X,DF,F)
      INTEGER           N
      DOUBLE PRECISION  X(N),DF(N),F, C,E
      INTRINSIC         COS,SIN,EXP
        C = COS(X(1) * X(2))
```

```
      E = EXP(X(1) + X(2))
      F = SIN(X(1) * X(2)) + 2D0*E + 1D0/E
      DF(1) = X(2)*C + 2D0*E - 1D0/E
      DF(2) = X(1)*C + 2D0*E - 1D0/E
      RETURN
    END
```

We get the results

```
Test of gradient.  Max|DF| =  3.9705E+01
Max difference,    Forward :  1.9663E-02  at  j =  2
                  Backward : -9.8262E-03  at  j =  2
                   Extrap. :  3.6214E-06  at  j =  1
```

These results indicate that there is no error in the gradient, and changing the initial value of `ICONTR` from 0 to 1 we get

```
Optimization.  ICONTR = 0.  14 calls of FDF
   x = -1.438523785E+00  1.091950195E+00
F(x) =  1.828427125E+00    ||g(X)|| =  1.69E-12
```

2.2. MINL2. Minimization of the ℓ_2-Norm of a Vector Function (Least Squares)

Purpose. Find $\mathbf{x}^*$ that minimizes $F(\mathbf{x})$, where

$$F(\mathbf{x}) = \tfrac{1}{2}\sum_{i=1}^{m} \left(f_i(\mathbf{x})\right)^2 \quad . \tag{2.1}$$

Here $\mathbf{x} = [x_1, \ldots, x_n]^\top \in \mathbb{R}^n$ is the vector of unknown parameters and $f_i,\ i = 1, \ldots, m$ is a set of functions that are twice continuously differentiable. The user must supply a subroutine that evaluates $\mathbf{f}(\mathbf{x})$ and the Jacobian $\mathbf{J}(\mathbf{x})$. There is an option for checking the implementation of $\mathbf{J}$.

Method. `MINL2` uses the Levenberg-Marquardt algorithm, see e.g. [5, Section 5.2]: At the current iterate $\mathbf{x}$ a step $\mathbf{h}$ is computed as the solution to

$$(\mathbf{J}^\top\mathbf{J} + \mu\mathbf{I})\mathbf{h} = -\mathbf{J}^\top\mathbf{f} \ ,$$

where $\mathbf{f}$ and $\mathbf{J}$ are the vector function and its Jacobian evaluated at $\mathbf{x}$. The damping parameter μ is updated during iteration [13], and its initial value is given by

$$\mu_0 = \tau \cdot \max\{\mathrm{diag}(\mathbf{J}_0^\top\mathbf{J}_0)\} \ , \tag{2.2}$$

where τ is given by the user (parameter `DX`) and $\mathbf{J}_0 = \mathbf{J}(\mathbf{x}_0)$. We recommend to use $\tau = 10^{-4}$ if $\mathbf{x}_0$ is believed to be close to $\mathbf{x}^*$, otherwise $\tau = 1$.

Iteration stops when

$$\|\mathbf{h}\|_2 < \varepsilon(\|\mathbf{x}\|_2 + \varepsilon) \ , \tag{2.3}$$

where ε is given by the user (parameter `EPS`).

Origin. The subroutine was written specially for this package. It is based on the MATLAB function `marquardt`, [13].

Use of other Subprograms. The subroutine calls the following BLAS (see [4]) subroutines and functions

Level 0: `LSAME` `XERBLA`
Level 1: `DAXPY` `DCOPY` `DDOT` `DNRM2` `DSCAL` `IDAMAX`
Level 2: `DGEMV` `DTRSV`
Level 3: `DGEMM` `DSYR`

Copies of these were obtained from
http://www.netlib.org/blas/blas.tgz
and are included in the file minl2.f. At lines 63, 233 and 275 you can find instructions about how to modify the file if BLAS is available on your computer.

Use. The subroutine call is

```
CALL MINL2(FDF,N,M,X,DX,EPS,MAXFUN,W,IW,ICONTR)
```

The parameters are

`FDF` `SUBROUTINE` written by the user with the following declaration

```
SUBROUTINE FDF(N,M,X,DF,F)
REAL*8 X(N),DF(M,N),F(N)
```

It must calculate the values of the functions and their gradients at the point $\mathbf{x} = [\texttt{X(1)}, \ldots, \texttt{X(N)}]^\top$ and store these numbers as follows,

$\texttt{F(I)} = f_{\texttt{I}}(\mathbf{x}),\ \texttt{I} = 1, \ldots, \texttt{M},$

$\texttt{DF(I,J)} = \dfrac{\partial f_{\texttt{I}}}{\partial x_{\texttt{J}}}(\mathbf{x}),\ \begin{cases} \texttt{I} = & 1, \ldots, \texttt{M} \\ \texttt{J} = & 1, \ldots, \texttt{N} \end{cases}$

The name of this subroutine (which can be chosen freely by the user) must appear in an EXTERNAL statement in the calling program.

N INTEGER. Number of unknowns, n. Must be positive. Is not changed.

M INTEGER. Number of functions, m. Must be positive. Is not changed.

X REAL*8 ARRAY with N elements. The use depends on the entry value of ICONTR.
ICONTR > 0 : *On entry*: Initial approximation to $\mathbf{x}^*$.
On exit: Computed solution.
ICONTR ≤ 0 : Point at which the Jacobian should be checked. Not changed.

DX REAL*8. The use depends on the entry value of ICONTR.
ICONTR > 0 : *On entry*: The initial damping parameter is given by (2.2) with $\tau =$ DX. Must be positive.
On exit: The τ-value given by (2.2) with μ_0 replaced by the current μ-value and $\mathbf{J}_0$ replaced by $\mathbf{J}$(X). Can be used for an ensuing warm start.
ICONTR ≤ 0 : Check of Jacobian matrix with DX used for h in (1.11). Must be significantly nonzero. Is not changed.

EPS REAL*8. Used only when $\texttt{ICONTR}_{\text{entry}} > 0$.
On entry: Desired accuracy: used for ε in (2.3). Must be positive.
On exit: If EPS was chosen too small, then the iteration stops when there is indication that rounding errors dominate, and EPS $= 0.0$, ICONTR $= 2$ are returned. Otherwise not changed.

MAXFUN INTEGER. Used only if the entry value of ICONTR is positive.
On entry: Upper bound on the number of calls of FDF. Must be positive.
On exit: Number of calls of FDF.

W REAL*8 ARRAY with IW elements. Work space. Entry values are not used.
Exit values depend on the entry value of ICONTR,
$\texttt{ICONTR}_{\text{entry}} > 0$:
W(1) $= F$(X), defined by (2.1),
W(2) $= \max\ |g_i(\texttt{X})|$,
W(3) = length of the last step,
W(4,..,M+3) : $f_{\texttt{I}}$(X), I $= 1, \ldots,$ M,
W(M+4,..,M+M*N+3) : Jacobian $\mathbf{J}$(X), stored columnswise,
W(M+M*N+4,..,M+N+M*N+3) : gradient $\mathbf{g}$(X).

$\texttt{ICONTR}_{\text{entry}} \leq 0$: Results of the gradient check are returned in the first 10 elements of W as follows, cf. (1.10)

W(1)	Maximum element in |DF|.
W(2), W(5), W(6)	$\delta^F,\ i^F,\ j^F$.
W(3), W(7), W(8)	$\delta^B,\ i^B,\ j^B$.
W(4), W(9), W(10)	$\delta^E,\ i^E,\ j^E$.

In case of an error the indices point out the erroneous element of the Jacobian matrix.

IW INTEGER. Length of work space W. Must be at least
If ICONTR > 0 then $2m(n{+}1) + n(n{+}3)$
Otherwise, $2m(n{+}2) + n + 10$
Is not changed.

ICONTR INTEGER. *On entry*: Controls the computation,
ICONTR $= 1$: Start minimization.
ICONTR $= 2$: Start minimization and print information during the iteration.
ICONTR ≤ 0 : Check gradient. No iteration.

On exit: Information about performance,
ICONTR $= 0$: Successful call.
ICONTR $= 2$: Iteration stopped because too many iterations were needed, see MAXFUN, or because rounding errors dominate, see EPS.
ICONTR < 0 : Computation did not start for the following reason,
ICONTR $= -2$: N ≤ 0
ICONTR $= -3$: M ≤ 0
ICONTR $= -5$: |DX| is too small in case of gradient check, or
DX ≤ 0 in case of optimization
ICONTR $= -6$: EPS ≤ 0
ICONTR $= -7$: MAXFUN ≤ 0
ICONTR $= -9$: IW is too small

Example. Minimize

$$F(\mathbf{x}) = \tfrac{1}{2}\sum_{i=1}^{3} f_i^2(\mathbf{x}) \quad ,$$

where the f_i are given by (1.14), page 6.

```
      PROGRAM  TINL2
*********************************************************************
* Test MINL2.  30.1.2002
*********************************************************************
      IMPLICIT          NONE
      INTEGER           I,ICONTR,IW,M,MAXFUN,N
      DOUBLE PRECISION  DX,EPS,X(2),W(36)
      EXTERNAL          FDF
C       ... Set parameters
        DATA   N, M, IW, MAXFUN, EPS,   X
     &        /2, 3, 36,   25,  1D-10, 2*1D0 /
        ICONTR = 0
C        ICONTR = 1
        IF  (ICONTR .LE. 0)  THEN
C         ... Check  Jacobian
          DX = 1D-3
          CALL MINL2(FDF,N,M,X,DX,EPS,MAXFUN,W,IW,ICONTR)
          IF  (ICONTR .LT. 0)  THEN
            WRITE(6,10) -ICONTR
          ELSE
            WRITE(6,20) W(1), W(2),INT(W(5)), INT(W(6))
            WRITE(6,30) 'Backward', W(3),INT(W(7)),INT(W(8))
            WRITE(6,30) ' Extrap.', W(4),INT(W(9)),INT(W(10))
          ENDIF
        ELSE
C         ... Optimize
          DX = 1D0
          CALL MINL2(FDF,N,M,X,DX,EPS,MAXFUN,W,IW,ICONTR)
          IF  (ICONTR .LT. 0)  THEN
            WRITE(6,10) -ICONTR
          ELSE
            WRITE(6,40) ICONTR,MAXFUN,X(1),X(2)
            WRITE(6,50) W(1),W(2), (W(I), I=4,M+3)
          ENDIF
```

```
      ENDIF
 10   FORMAT('Parameter number',I3,'  is outside its range')
 20   FORMAT('Test of Jacobian.  Max|DF| =',1P1D12.4/'Max difference',
     &    5X,'Forward :',1P1D12.4,'  at  i,j =',I2,',',I2)
 30   FORMAT(18X,A8,' :',1P1D12.4,'  at  i,j =',I2,',',I2)
 40   FORMAT('Optimization.  ICONTR =',I2,'.',I4,' calls of FDF'
     &     /'   x =',1P2D17.9)
 50   FORMAT('F(x) =',1P1D17.9,',    norm(g(x)) =',1P1D9.2
     &    /'f(x) =',1P3D17.9)
      STOP
     END

     SUBROUTINE FDF(N,M,X,DF,F)
     INTEGER           N,M
     DOUBLE PRECISION  X(N),DF(M,N),F(M)
       F(1) = 1.5D0   - X(1)*(1D0 - X(2))
       F(2) = 2.25D0  - X(1)*(1D0 - X(2)**2)
       F(3) = 2.625D0 - X(1)*(1D0 - X(2)**3)
       DF(1,1) = X(2)-1D0
       DF(1,2) = X(1)
       DF(2,1) = X(2)**2 - 1D0
       DF(2,2) = 2D0*X(1)*X(2)
       DF(3,1) = X(2)**3 - 1D0
       DF(3,2) = 3D0*X(1)*X(2)**2
       RETURN
     END
```

We get the results

```
Test of Jacobian.  Max|DF| =  3.0000E+00
Max difference     Forward :  3.0010E-03  at  i,j = 3, 2
                  Backward : -1.4998E-03  at  i,j = 3, 2
                   Extrap. :  5.0000E-07  at  i,j = 3, 2
```

These results indicate that there is no error in the Jacobian, and changing the initial value of `ICONTR` from 0 to 1 we get

```
Optimization.  ICONTR = 0.  13 calls of FDF
   x =  3.000000000E+00  4.999999999E-01
F(x) =  3.540144388E-21,    norm(g(x)) = 3.17E-11
f(x) = -5.231232114E-11  2.579603198E-13  6.590632207E-11
```

2.3. MINL1. Minimization of the ℓ_1-Norm of a Vector Function

Purpose. Find $\mathbf{x}^*$ that minimizes $F(\mathbf{x})$, where

$$F(\mathbf{x}) = \sum_{i=1}^{m} | f_i(\mathbf{x}) | \quad . \tag{2.4}$$

Here $\mathbf{x} = [x_1, \ldots, x_n]^\top \in \mathbb{R}^n$ is the vector of unknown parameters and $f_i,\ i = 1, \ldots, m$ is a set of functions that are twice continuously differentiable. The user must supply a subroutine that evaluates $\mathbf{f}(\mathbf{x})$ and the Jacobian $\mathbf{J}(\mathbf{x})$. There is an option for checking the implementation of $\mathbf{J}$.

Method. The algorithm is iterative. It is based on successive linearizations of the nonlinear functions f_i, combining a first order trust region method with a local method which uses approximate second order information, see [9].

Origin. Subroutine `L1NLS` from [7].

Remark. The trust region around the the current $\mathbf{x}$ is the ball centered at $\mathbf{x}$ with radius Δ defined so that the linearizations of the nonlinear functions f_i are reasonably accurate for all points inside the ball. During iteration this bound is adjusted according to how well the linear approximations centered at the previous iterate predict the gain in F.

The user has to give an initial value for Δ. If the functions are almost linear, then we recommend to use an estimate of the distance between $\mathbf{x}_0$ and the solution $\mathbf{x}^*$. Otherwise, we recommend $\Delta_0 = 0.1\|\mathbf{x}_0\|$.

Use. The subroutine call is

```
CALL MINL1(FDF,N,M,X,DX,EPS,MAXFUN,W,IW,ICONTR)
```

The parameters are

`FDF` `SUBROUTINE` written by the user with the following declaration

```
SUBROUTINE FDF(N,M,X,DF,F)
REAL*8 X(N),DF(M,N),F(N)
```

It must calculate the values of the functions and their gradients at the point $\mathbf{x} = [\texttt{X(1)}, \ldots, \texttt{X(N)}]^\top$ and store these numbers as follows,

$\texttt{F(I)} = f_{\texttt{I}}(\mathbf{x}),\ \texttt{I} = 1, \ldots, \texttt{M},$

$$\texttt{DF(I,J)} = \frac{\partial f_{\texttt{I}}}{\partial x_{\texttt{J}}}(\mathbf{x}),\ \begin{cases} \texttt{I} = 1, \ldots, \texttt{M} \\ \texttt{J} = 1, \ldots, \texttt{N} \end{cases}$$

The name of this subroutine (which can be chosen freely by the user) must appear in an `EXTERNAL` statement in the calling program.

`N` `INTEGER`. Number of unknowns, n. Must be positive. Is not changed.

`M` `INTEGER`. Number of functions, m. Must be positive. Is not changed.

`X` `REAL*8 ARRAY` with `N` elements. The use depends on the entry value of `ICONTR`,

`ICONTR` > 0 : *On entry*: Initial approximation to $\mathbf{x}^*$.
On exit: Computed solution.

`ICONTR` ≤ 0 : Point at which the Jacobian should be checked. Not changed.

`DX` `REAL*8`. The use depends on the entry value of `ICONTR`,

`ICONTR` > 0 : Radius of trust region, see **Remark** above.
On entry: `DX` $= \Delta_0$. Must be positive.
On exit: Final trust region radius.

ICONTR ≤ 0 : h in (1.11) for check of Jacobian matrix. Must be significantly nonzero. Is not changed.

EPS REAL*8. Used only when ICONTR > 0. Must be positive.
On entry: Desired accuracy: The algorithm stops when it suggests to change the iterate from $\mathbf{x}_k$ to $\mathbf{x}_k+\mathbf{h}_k$ with $\|\mathbf{h}_k\| <$ EPS$\cdot\|\mathbf{x}_k\|$.
On exit: If EPS was chosen too small, then the iteration stops when there is indication that rounding errors dominate, and EPS $= 0.0$, ICONTR $= 2$ are returned.

MAXFUN INTEGER. Used only if the entry value of ICONTR is positive.
On entry: Upper bound on the number of calls of FDF. Must be positive.
On exit: Number of calls of FDF.

W REAL*8 ARRAY with IW elements. Work space. Entry values are not used.
Exit values depend on the entry value of ICONTR,
ICONTR$_{\text{entry}} > 0$:
W(1) $= F$(X), defined by (2.4),
W(2,..,M+1) : f_{I}(X), I $= 1, \ldots,$ M,
W(2M+1,..,2M+M*N) : Jacobian $\mathbf{J}$(X), stored columnswise..

ICONTR$_{\text{entry}} \le 0$: Results of the gradient check are returned in the first 10 elements of W as follows, cf. (1.10)

W(1)	Maximum element in |DF|.
W(2), W(5), W(6)	$\delta^F,\ i^F,\ j^F$.
W(3), W(7), W(8)	$\delta^B,\ i^B,\ j^B$.
W(4), W(9), W(10)	$\delta^E,\ i^E,\ j^E$.

In case of an error the indices point out the erroneous element of the Jacobian matrix.

IW INTEGER. Length of work space W. Must be at least $2mn+5(n^2+m+1)+11n$. Is not changed.

ICONTR INTEGER. *On entry*: Controls the computation,
ICONTR > 0 : Start minimization.
ICONTR ≤ 0 : Check Jacobian. No iteration.

On exit: Information about performance,
ICONTR $= 0, 1$: Successful call.
ICONTR $= 2$: Iteration stopped because too many iterations were needed, see MAXFUN, or rounding errors dominate, see EPS.
ICONTR < 0 : Computation did not start for the following reason,
ICONTR $= -2$: N ≤ 0
ICONTR $= -3$: M ≤ 0
ICONTR $= -5$: \|DX\| is too small in case of gradient check, or
DX ≤ 0 in case of optimization
ICONTR $= -6$: EPS ≤ 0
ICONTR $= -7$: MAXFUN ≤ 0
ICONTR $= -9$: IW $< 2mn + 5(n^2 + m + 1) + 11n$

Example. Minimize

$$F(\mathbf{x}) = \sum_{i=1}^{3} |f_i(\mathbf{x})| \ ,$$

where the f_i are given by (1.14), page 6.

```
      PROGRAM  TINL1
*************************************************************************
* Test MINL1.  31.1.2002
*************************************************************************
      IMPLICIT          NONE
      INTEGER           I,ICONTR,IW,M,MAXFUN,N
      DOUBLE PRECISION  DX,EPS,X(2),W(74)
      EXTERNAL          FDF
C       ... Set parameters
        DATA   N, M, IW, MAXFUN, EPS,   X
     &        /2, 3, 74,   25,  1D-10, 2*1D0 /
        ICONTR = 0
C        ICONTR = 1
        IF  (ICONTR .LE. 0)  THEN
C         ... Check  Jacobian
          DX = 1D-3
          CALL MINL1(FDF,N,M,X,DX,EPS,MAXFUN,W,IW,ICONTR)
          IF  (ICONTR .LT. 0)  THEN
            WRITE(6,10) -ICONTR
          ELSE
            WRITE(6,20) W(1), W(2),INT(W(5)), INT(W(6))
            WRITE(6,30) 'Backward', W(3),INT(W(7)),INT(W(8))
            WRITE(6,30) ' Extrap.', W(4),INT(W(9)),INT(W(10))
          ENDIF
        ELSE
C         ... Optimize
          DX = .1D0
          CALL MINL1(FDF,N,M,X,DX,EPS,MAXFUN,W,IW,ICONTR)
          IF  (ICONTR .LT. 0)  THEN
            WRITE(6,10) -ICONTR
          ELSE
            WRITE(6,40) ICONTR,MAXFUN
            WRITE(6,50) X(1),X(2), W(1), (W(I), I=2,M+1)
          ENDIF
        ENDIF
  10    FORMAT('Parameter number',I3,'  is outside its range')
  20    FORMAT('Test of Jacobian.  Max|DF| =',1P1D12.4/'Max difference',
     &    5X,'Forward :',1P1D12.4,'  at  i,j =',I2,',',I2)
  30    FORMAT(18X,A8,' :',1P1D12.4,'  at  i,j =',I2,',',I2)
  40    FORMAT('Optimization.  ICONTR =',I2,'.',I4,' calls of FDF')
  50    FORMAT('   x =',1P2D17.9/'F(x) =',1P1D17.9/'f(x) =',1P3D17.9)
        STOP
      END

      SUBROUTINE FDF(N,M,X,DF,F)
      INTEGER           N,M
      DOUBLE PRECISION  X(N),DF(M,N),F(M)
        F(1) = 1.5D0   - X(1)*(1D0 - X(2))
        F(2) = 2.25D0  - X(1)*(1D0 - X(2)**2)
        F(3) = 2.625D0 - X(1)*(1D0 - X(2)**3)
        DF(1,1) = X(2)-1D0
        DF(1,2) = X(1)
        DF(2,1) = X(2)**2 - 1D0
        DF(2,2) = 2D0*X(1)*X(2)
        DF(3,1) = X(2)**3 - 1D0
        DF(3,2) = 3D0*X(1)*X(2)**2
        RETURN
      END
```

We get the results

```
Test of Jacobian.  Max|DF| =  3.0000E+00
Max difference     Forward :  3.0010E-03  at  i,j = 3, 2
                  Backward : -1.4998E-03  at  i,j = 3, 2
                   Extrap. :  5.0000E-07  at  i,j = 3, 2
```

These results indicate that there is no error in the Jacobian, and changing the initial value of `ICONTR` from 0 to 1 we get

```
Optimization.  ICONTR = 0.  10 calls of FDF
   x =  3.000000000E+00  5.000000000E-01
F(x) =  0.000000000E+00
f(x) =  0.000000000E+00  0.000000000E+00  0.000000000E+00
```

2.4. MININF. Minimization of the ℓ_∞-Norm of a Vector Function

Purpose. Find $\mathbf{x}^*$ that minimizes $F(\mathbf{x})$, where

$$F(\mathbf{x}) = \max_i |f_i(\mathbf{x})| \quad . \tag{2.5}$$

Here $\mathbf{x} = [x_1, \ldots, x_n]^\top \in \mathbb{R}^n$ is the vector of unknown parameters and $f_i,\ i = 1, \ldots, m$ is a set of functions that are twice continuously differentiable. The user must supply a subroutine that evaluates $\mathbf{f}(\mathbf{x})$ and the Jacobian $\mathbf{J}(\mathbf{x})$. There is an option for checking the implementation of $\mathbf{J}$.

Method. The algorithm is iterative. It is based on successive linearizations of the nonlinear functions f_i and uses constraints on the step vector. The linearized problems are solved by a linear programming technique, see [11].

Origin. The main part of the subroutine was written by K. Madsen and was published as `VE01AD` in the the Harwell Subroutine Library [10]. We use K. Madsen's original subroutine `SUB1W` which is consistent with the other subroutines in the present package

Remark. The user has to give an initial value for Δ, which appears in the constraint $\|\mathbf{h}\| \le \Delta$, where $\mathbf{h}$ is the step between two consecutive iterates. During iteration this bound (trust region radius) is adjusted according to how well the current linear approximations predict the actual gain in F.

If the functions f_i are almost linear, then we recommend to use a value for Δ_0, which is an estimate of the distance between $\mathbf{x}_0$ and the solution $\mathbf{x}^*$. Otherwise, we recommend $\Delta_0 = 0.1\|\mathbf{x}_0\|$.

Use. The subroutine call is

```
CALL MININF(FDF,N,M,X,DX,EPS,MAXFUN,W,IW,ICONTR)
```

The parameters are

`FDF` `SUBROUTINE` written by the user with the following declaration

```
SUBROUTINE FDF(N,M,X,DF,F)
REAL*8 X(N),DF(M,N),F(N)
```

It must calculate the values of the functions and their gradients at the point $\mathbf{x} = [\mathtt{X(1)}, \ldots, \mathtt{X(N)}]^\top$ and store these numbers as follows,

$\mathtt{F(I)} = f_{\mathtt{I}}(\mathbf{x}),\ \mathtt{I} = 1, \ldots, \mathtt{M},$

$$\mathtt{DF(I,J)} = \frac{\partial f_{\mathtt{I}}}{\partial x_{\mathtt{J}}}(\mathbf{x}),\ \begin{cases} \mathtt{I} = & 1, \ldots, \mathtt{M} \\ \mathtt{J} = & 1, \ldots, \mathtt{N} \end{cases}$$

The name of this subroutine (which can be chosen freely by the user) must appear in an `EXTERNAL` statement in the calling program.

`N` `INTEGER`. Number of unknowns, n. Must be positive. Is not changed.

`M` `INTEGER`. Number of functions, m. Must be positive. Is not changed.

`X` `REAL*8 ARRAY` with `N` elements. The use depends on the entry value of `ICONTR`.
`ICONTR` > 0 : *On entry*: Initial approximation to $\mathbf{x}^*$.
On exit: Computed solution.
`ICONTR` ≤ 0 : Point at which the Jacobian should be checked. Not changed.

`DX` `REAL*8`. The use depends on the entry value of `ICONTR`,
`ICONTR` > 0 : Radius of trust region, see **Remark** above.
On entry: `DX` $= \Delta_0$. Must be positive.
On exit: Final trust region radius.
`ICONTR` ≤ 0 : h in (1.11) for check of Jacobian matrix. Must be significantly nonzero. Is not changed.

EPS REAL*8. Used only when $\texttt{ICONTR}_{\text{entry}} > 0$. Must be positive.
On entry: Desired accuracy: The algorithm stops when it suggests to change the iterate from $\mathbf{x}_k$ to $\mathbf{x}_k+\mathbf{h}_k$ with $\|\mathbf{h}_k\| < \texttt{EPS}\cdot\|\mathbf{x}_k\|$.
On exit: If EPS was chosen too small, then the iteration stops when there is indication that rounding errors dominate, and EPS = 0.0, ICONTR = 2 are returned.

MAXFUN INTEGER. Used only if the entry value of ICONTR is positive.
On entry: Upper bound on the number of calls of FDF. Must be positive.
On exit: Number of calls of FDF.

W REAL*8 ARRAY with IW elements. Work space. Entry values are not used.
Exit values depend on the entry value of ICONTR,
$\texttt{ICONTR}_{\text{entry}} > 0$:
W(1) $= F($X$)$, defined by (2.5),
W(2,..,M+1) : $f_{\texttt{I}}($X$)$, I $= 1, \ldots,$ M,
W(2M+1,..,2M+M*N) : Jacobian $\mathbf{J}($X$)$, stored columnswise..

$\texttt{ICONTR}_{\text{entry}} \le 0$: Results of the gradient check are returned in the first 10 elements of W as follows, cf. (1.10)

W(1)	Maximum element in |DF|.
W(2), W(5), W(6)	$\delta^F,\ i^F,\ j^F$.
W(3), W(7), W(8)	$\delta^B,\ i^B,\ j^B$.
W(4), W(9), W(10)	$\delta^E,\ i^E,\ j^E$.

In case of an error the indices point out the erroneous element of the Jacobian matrix.

IW INTEGER. Length of work space W. Must be at least $2mn+n^2+14n+4m+11$. Is not changed.

ICONTR INTEGER. *On entry*: Controls the computation,
ICONTR > 0 : Start minimization.
ICONTR ≤ 0 : Check gradient. No iteration.

On exit: Information about performance,
ICONTR $= 0, 1$: No problems encountered.
ICONTR $= 2$: Iteration stopped because too many iterations were needed, see MAXFUN, or rounding errors dominate, see EPS.
ICONTR < 0 : Computation did not start for the following reason,
ICONTR $= -2$: N ≤ 0
ICONTR $= -3$: M ≤ 0
ICONTR $= -5$: \|DX\| is too small in case of gradient check, or DX ≤ 0 in case of optimization
ICONTR $= -6$: EPS ≤ 0
ICONTR $= -7$: MAXFUN ≤ 0
ICONTR $= -9$: IW $< 2mn + n^2 + 14n + 4m + 11$

Example. Minimize

$$F(\mathbf{x}) = \max_i |\, f_i(\mathbf{x}) \,| \quad ,$$

where the f_i are given by (1.14), page 6.

```
      PROGRAM  TININF
*********************************************************************
* Test MININF.  31.1.2002
*********************************************************************
      IMPLICIT          NONE
      INTEGER           I,ICONTR,IW,M,MAXFUN,N
      DOUBLE PRECISION  DX,EPS,X(2),W(67)
      EXTERNAL          FDF
```

```
C       ... Set parameters
        DATA   N, M, IW, MAXFUN, EPS,   X
     &        /2, 3, 67,   25,  1D-10, 2*1D0 /
        ICONTR = 0
C        ICONTR = 1
        IF  (ICONTR .LE. 0)  THEN
C         ... Check  Jacobian
          DX = 1D-3
          CALL MININF(FDF,N,M,X,DX,EPS,MAXFUN,W,IW,ICONTR)
          IF  (ICONTR .LT. 0)  THEN
            WRITE(6,10) -ICONTR
          ELSE
            WRITE(6,20) W(1), W(2),INT(W(5)), INT(W(6))
            WRITE(6,30) 'Backward', W(3),INT(W(7)),INT(W(8))
            WRITE(6,30) ' Extrap.', W(4),INT(W(9)),INT(W(10))
          ENDIF
        ELSE
C         ... Optimize
          DX = .1D0
          CALL MININF(FDF,N,M,X,DX,EPS,MAXFUN,W,IW,ICONTR)
          IF  (ICONTR .LT. 0)  THEN
            WRITE(6,10) -ICONTR
          ELSE
            WRITE(6,40) ICONTR,MAXFUN
            WRITE(6,50) X(1),X(2), W(1), (W(I), I=2,M+1)
          ENDIF
        ENDIF
  10    FORMAT('Parameter number',I3,'  is outside its range')
  20    FORMAT('Test of Jacobian.  Max|DF| =',1P1D12.4/'Max difference',
     &    5X,'Forward :',1P1D12.4,'  at  i,j =',I2,',',I2)
  30    FORMAT(18X,A8,' :',1P1D12.4,'  at  i,j =',I2,',',I2)
  40    FORMAT('Optimization.  ICONTR =',I2,'.',I4,' calls of FDF')
  50    FORMAT('   x =',1P2D17.9/'F(x) =',1P1D17.9/'f(x) =',1P3D17.9)
        STOP
      END

      SUBROUTINE FDF(N,M,X,DF,F)
      INTEGER           N,M
      DOUBLE PRECISION  X(N),DF(M,N),F(M)
        F(1) = 1.5D0   - X(1)*(1D0 - X(2))
        F(2) = 2.25D0  - X(1)*(1D0 - X(2)**2)
        F(3) = 2.625D0 - X(1)*(1D0 - X(2)**3)
        DF(1,1) = X(2)-1D0
        DF(1,2) = X(1)
        DF(2,1) = X(2)**2 - 1D0
        DF(2,2) = 2D0*X(1)*X(2)
        DF(3,1) = X(2)**3 - 1D0
        DF(3,2) = 3D0*X(1)*X(2)**2
        RETURN
      END
```

We get the results

```
Test of Jacobian.  Max|DF| =  3.0000E+00
Max difference     Forward :  3.0010E-03  at  i,j = 3, 2
                  Backward : -1.4998E-03  at  i,j = 3, 2
                   Extrap. :  5.0000E-07  at  i,j = 3, 2
```

These results indicate that there is no error in the Jacobian, and changing the initial value of `ICONTR` from 0 to 1 we get

```
Optimization.  ICONTR = 0.  11 calls of FDF
   x =  3.000000000E+00  5.000000000E-01
F(x) =  0.000000000E+00
f(x) =  0.000000000E+00  0.000000000E+00  0.000000000E+00
```

3. Constrained Optimization

3.1. MINCF. Constrained Minimization of a Scalar Function

Purpose. Find $\mathbf{x}^*$ that minimizes $F(\mathbf{x})$, where the vector of unknown parameters $\mathbf{x} = [x_1, \ldots, x_n]^\top \in \mathbb{R}^n$ must satisfy the following non-linear equality and inequality constraints,

$$c_i(\mathbf{x}) = 0 \ , \quad i = 1, 2, \ldots, L_{\text{eq}} \ ,$$
$$c_i(\mathbf{x}) \geq 0 \ , \quad i = L_{\text{eq}}{+}1, \ldots, L \ .$$

The objective function F and the constraint functions $\{c_i\}$ must be twice continuously differentiable. The user must supply a subroutine that evaluates $F(\mathbf{x})$, $\{c_i(\mathbf{x})\}$ and the gradients of F and $\{c_i\}$. There is an option for checking the implementation of these gradients.

Method. The algorithm is iterative. It is based on successively approximating the non-linear problem with quadratic problems, i.e. at the current iterate the objective function is approximated by a quadratic function and the constraints are approximated by linear functions. The algorithm uses the so-called *"Watch-dog technique"* as described in [2] and [16]. The quadratic programming algorithm is described in [17]

Origin. Harwell subroutine `VF13AD` [10].

Use. The subroutine call is

```
CALL MINCF(FDFCDC,N,L,LEQ,X,DX,EPS,MAXFUN,W,IW,ICONTR)
```

The parameters are

`FDFCDC` `SUBROUTINE` written by the user with the following declaration

```
SUBROUTINE FDFCDC(N,N1,L,X,F,DF,C,DC)
REAL*8 X(N),F,DF(N),C(L),DC(L,N1)
```

where `N1` = `N`+1. It must calculate the value of the objective function and its gradient at the point $\mathbf{x} = [\texttt{X(1)}, \ldots, \texttt{X(N)}]^\top$ and store these numbers as follows,

$\texttt{F} = F(\mathbf{x})$,

$$\texttt{DF(J)} = \frac{\partial F}{\partial x_{\texttt{J}}}(\mathbf{x}) \ , \qquad \texttt{J} = 1, \ldots \texttt{N}$$
$$\texttt{C(I)} = c_{\texttt{I}} \ , \qquad \texttt{I} = 1, \ldots \texttt{L}$$
$$\texttt{DC(I,J)} = \frac{\partial c_{\texttt{I}}}{\partial x_{\texttt{J}}}(\mathbf{x}) \ , \quad \texttt{I} = 1, \ldots \texttt{L} \text{ and} \texttt{J} = 1, \ldots \texttt{N}$$

The name of this subroutine (which can be chosen freely by the user) must appear in an `EXTERNAL` statement in the calling program.

For internal reasons in the minimization routine the number of columns in `ARRAY DC` must be exactly one more than actually addressed. This number, `N1 = N+1`, is transported to `SUBROUTINE FDFCDC` as its second parameter.

It is essential that the equality constraints (if any) are numbered first.

`N` `INTEGER`. Number of unknowns, n. Must be positive. Is not changed.

`L` `INTEGER`. Number of constraints, L. Must be positive. Is not changed.

`LEQ` `INTEGER`. Number of equality constraints, L_{eq}. Must satisfy $0 \leq \texttt{LEQ} \leq \min\{\texttt{L}, \texttt{N}\}$. Is not changed.

`X` `REAL*8 ARRAY` with `N` elements. The use depends on the entry value of `ICONTR`.

ICONTR > 0 : *On entry*: Initial approximation to $\mathbf{x}^*$.
On exit: Computed solution.
ICONTR ≤ 0 : Point at which the gradients should be checked. Not changed.

DX REAL*8. Used only when ICONTR ≤ 0 on entry, in which case DX is used for h in (1.3) for checking the gradients of the objective function and the constraints.
Must be significantly nonzero. Is not changed.

EPS REAL*8. Used only when ICONTR > 0 on entry. Desired accuracy of the results: Iteration stops when the Kuhn-Tucker conditions are satisfied within a tolerance of EPS.
Must be positive. Is not changed.

MAXFUN INTEGER. Used only if the entry value of ICONTR is positive.
On entry: Upper bound on the number of calls of FDF. Must be positive.
On exit: Number of calls of FDF.

W REAL*8 ARRAY with IW elements. Work space. Entry values are not used.
Exit values depend on the entry value of ICONTR,

ICONTR$_{\text{entry}} > 0$:
W(1) $= F$(X), the computed minimum.
W(I+1) $= c_{\text{I}}$(X), I $= 1, \ldots,$ L

ICONTR$_{\text{entry}} \leq 0$: Results of the gradient check are returned in the first 17 elements of W as follows, cf. (1.10)

Objective function:	
W(1)	Maximum element in |DF|.
W(2), W(9)	δ^F and j^F.
W(3), W(10)	δ^B and j^B.
W(4), W(11)	δ^E and j^E.
Constraints:	
W(5)	Maximum element in |DC|.
W(6),W(12),W(13)	δ^F and j^F.
W(7),W(14),W(15)	δ^B and j^B.
W(8),W(16),W(17)	δ^E and j^E.

In case of an error the indices W(9..17) point out the erroneous gradient component.

IW INTEGER. Length of work space W. Must be at least $\frac{5}{2}n(n{+}9) + (n{+}8)L + 15$. Is not changed.

ICONTR INTEGER. *On entry*: Controls the computation,
ICONTR $= 1$: Start minimization.
ICONTR $= 2$: Start minimization and print information during the iteration.
ICONTR ≤ 0 : Check gradient. No iteration

On exit: Information about performance,
ICONTR $= 1$: Successful call.
ICONTR $= 2$: Iteration stopped because too many iterations were needed, see MAXFUN.
ICONTR $= 3$: Iteration stopped because more than 5 calls of FDFCDC was needed in one line search. Check your gradients.
ICONTR $= 4$: Iteration stopped because an uphill search direction was suggested. Check your gradients.
ICONTR $= 5$: Iteration failed because it was not possible to find a starting point satisfying all constraints.
ICONTR < 0 : Computation did not start for the following reason,
ICONTR $= -2$: N ≤ 0
ICONTR $= -3$: L ≤ 0
ICONTR $= -4$: LEQ < 0 or LEQ $> \min\{$L, N$\}$
ICONTR $= -6$: \|DX\| is too small

$$\texttt{ICONTR} = -7: \ \texttt{EPS} \leq 0$$
$$\texttt{ICONTR} = -8: \ \texttt{MAXFUN} \leq 0$$
$$\texttt{ICONTR} = -9: \ \texttt{IW} < \tfrac{5}{2}n(n{+}9) + (n{+}8)L + 15$$

Example. Minimize

$$F(\mathbf{x}) = \sin(x_1 x_2) + 2e^{x_1+x_2} + e^{-x_1-x_2}$$

subject to the constraints

$$c_1(\mathbf{x}) \equiv 1 - x_1^2 - x_2^2 \geq 0$$
$$c_2(\mathbf{x}) \equiv x_2 - x_1^3 \geq 0$$
$$c_3(\mathbf{x}) \equiv x_1 + 2x_2 \geq 0$$

```
      PROGRAM  TINCF
************************************************************************
* Test MINCF.  9.10.2000
************************************************************************
      IMPLICIT          NONE
      INTEGER           ICONTR,IW,L,MAXFUN,N,I
      DOUBLE PRECISION  DX,EPS,X(2),W(100)
      EXTERNAL          FDFCDC
C       ... Set parameters
        DATA  N, L, IW, MAXFUN, EPS,   DX,     X
     &       /2, 3,100,   25,  1D-10, 1D-3, 1D0,2D0/
        ICONTR = 0
C        ICONTR = 1
        IF  (ICONTR .LE. 0)  THEN
C         ... Check  Jacobian
          DX = 1D-3
          CALL MINCF(FDFCDC,N,L,0,X,DX,EPS,MAXFUN,W,IW,ICONTR)
          IF  (ICONTR .LT. 0)  THEN
            WRITE(6,10) -ICONTR
          ELSE
            WRITE(6,20) ' function ','DF',W(1)
            WRITE(6,30) W(2),INT(W(9))
            WRITE(6,40) 'Backward', W(3),INT(W(10))
            WRITE(6,40) 'Extrap.', W(4),INT(W(11))
            WRITE(6,20) 'constraint','DC',W(5)
            WRITE(6,50) W(6),INT(W(12)),INT(W(13))
            WRITE(6,60) 'Backward', W(7),INT(W(14)),INT(W(15))
            WRITE(6,60) 'Extrap.', W(8),INT(W(16)),INT(W(17))
          ENDIF
        ELSE
C         ... Optimize
          CALL MINCF(FDFCDC,N,L,0,X,DX,EPS,MAXFUN,W,IW,ICONTR)
          IF  (ICONTR .LT. 0)  THEN
            WRITE(6,10) -ICONTR
          ELSE
            WRITE(6,70) ICONTR,MAXFUN
            WRITE(6,80) X(1),X(2), W(1), (W(I+1),I=1,L)
          ENDIF
        ENDIF
  10    FORMAT('Parameter number',I3,'  is outside its range')
  20    FORMAT('Test of ',A10,' gradient.  Max|',A2,'| =',1P1D12.4)
  30    FORMAT(11X,'Max difference,    Forward :',1P1D12.4,
     &    '  at  j =',I2)
  40    FORMAT(29X,A8,' :',1P1D12.4,'  at  j =',I2)
  50    FORMAT(11X,'Max difference,    Forward :',1P1D12.4,
     &    '  at  i,j =',I2,',',I2)
```

```
  60    FORMAT(29X,A8,' :',1P1D12.4,'  at  i,j =',I2,',',I2)
  70    FORMAT('Optimization.  ICONTR =',I2,'.',I4,' calls of FDF')
  80    FORMAT('  x  =',1P2D17.9/'F(x) =',1P1D17.9/
     &    'c(x) =',1P3D17.9)
        STOP
      END

      SUBROUTINE FDFCDC(N,N1,L,X,F,DF,C,DC)
      INTEGER           N,N1,L
      DOUBLE PRECISION  X(N),F,DF(N),C(L),DC(L,N1), CC,EE
      INTRINSIC         COS,SIN,EXP
        CC = COS(X(1) * X(2))
        EE = EXP(X(1) + X(2))
        F = SIN(X(1) * X(2)) + 2D0*EE + 1D0/EE
        DF(1) = X(2)*CC + 2D0*EE - 1D0/EE
        DF(2) = X(1)*CC + 2D0*EE - 1D0/EE
C       ... Constraints
        C(1) = 1D0 -X(1)**2 - X(2)**2
        C(2) = X(2) - X(1)**3
        C(3) = X(1) + 2D0*X(2)
        DC(1,1) = -2D0 * X(1)
        DC(1,2) = -2D0 * X(2)
        DC(2,1) = -3D0 * X(1)**2
        DC(2,2) = 1D0
        DC(3,1) = 1D0
        DC(3,2) = 2D0
        RETURN
      END
```

We get the results

```
Test of  function  gradient.  Max|DF| =  3.9705E+01
           Max difference,    Forward :  1.9663E-02  at  j = 2
                             Backward : -9.8262E-03  at  j = 2
                              Extrap. :  3.6214E-06  at  j = 1
Test of constraint gradient.  Max|DC| =  4.0000E+00
           Max difference,    Forward : -3.0010E-03  at  i,j = 2, 1
                             Backward :  1.4998E-03  at  i,j = 2, 1
                              Extrap. : -5.0000E-07  at  i,j = 2, 1
```

These results indicate that the gradients of both the objective function and the constraints are implemented correctly, and changing the initial value of `ICONTR` from 0 to 1 we get

```
Optimization.  ICONTR = 1.  11 calls of FDF
  x  = -8.264473889E-01  5.630139549E-01
F(x) =  2.389516015E+00
c(x) = -3.051107544E-15  1.127490155E+00  2.995805209E-01
```

3.2. MINCL1. Linearly Constrained Minimization of the ℓ_1-Norm of a Vector Function

Purpose. Find $\mathbf{x}^*$ that minimizes $F(\mathbf{x})$, where

$$F(\mathbf{x}) = \sum_{i=1}^{m} | f_i(\mathbf{x}) | \quad , \tag{3.1a}$$

and where the vector of unknown parameters $\mathbf{x} = [x_1, \ldots, x_n]^\top \in \mathbb{R}^n$ must satisfy the following linear equality and inequality constraints,

$$\begin{aligned} c_i(\mathbf{x}) &\equiv \mathbf{a}_i^\top \mathbf{x} + b_i = 0 \ , \quad i = 1, 2, \ldots, L_{\text{eq}} \ , \\ c_i(\mathbf{x}) &\equiv \mathbf{a}_i^\top \mathbf{x} + b_i \geq 0 \ , \quad i = L_{\text{eq}}{+}1, \ldots, L \end{aligned} \tag{3.1b}$$

for given vectors $\{\mathbf{a}_i\}$ and scalars $\{b_i\}$. The f_i, $i = 1, \ldots, m$ is a set of functions that are twice continuously differentiable. The user must supply a subroutine that evaluates $\mathbf{f}(\mathbf{x})$ and the Jacobian $\mathbf{J}(\mathbf{x})$. There is an option for checking the implementation of $\mathbf{J}$.

Method. The algorithm is iterative. It is based on successive linearizations of the on-linear functions f_i, combining a first order trust region method with a local method that uses approximate second order information, see [9].

Origin. Subroutine `L1NLS` by Jørgen Hald [7].

Remarks. The trust region around the the current $\mathbf{x}$ is the ball centered at $\mathbf{x}$ with radius Δ defined so that the linearizations of the nonlinear functions f_i are reasonably accurate for all points inside the ball. During iteration this bound is adjusted according to how well the linear approximations centered at the previous iterate predict the gain in F.

The user has to give an initial value for Δ. If the functions are almost linear, then we recommend to use an estimate of the distance between $\mathbf{x}_0$ and the solution $\mathbf{x}^*$. Otherwise, we recommend $\Delta_0 = 0.1\|\mathbf{x}_0\|$.

A solution is said to be *"regular"* when it is a strict local minimum, i.e. there exists a positive number K such that

$$F(\mathbf{x}) - F(\mathbf{x}^*) \geq K\|\mathbf{x} - \mathbf{x}^*\|$$

for any feasible $\mathbf{x}$ near $\mathbf{x}^*$. Otherwise, the solution is said to be *"singular"*.

Use. The subroutine call is

```
CALL MINCL1(FDF,N,M,L,LEQ,B,A,X,DX,EPS,MAXFUN,W,IW,ICONTR)
```

The parameters are

`FDF` `SUBROUTINE` written by the user with the following declaration

```
SUBROUTINE FDF(N,M,X,DF,F)
REAL*8 X(N),DF(M,N),F(N)
```

It must calculate the values of the functions and their gradients at the point $\mathbf{x} = [\texttt{X(1)}, \ldots, \texttt{X(N)}]^\top$ and store these numbers as follows,

`F(I)` $= f_{\texttt{I}}(\mathbf{x})$, `I` $= 1, \ldots,$ `M`,

`DF(I,J)` $= \dfrac{\partial f_{\texttt{I}}}{\partial x_{\texttt{J}}}(\mathbf{x})$, $\begin{cases} \texttt{I} = 1, \ldots, \texttt{M} \\ \texttt{J} = 1, \ldots, \texttt{N} \end{cases}$

The name of this subroutine (which can be chosen freely by the user) must appear in an `EXTERNAL` statement in the calling program.

`N` `INTEGER`. Number of unknowns, n. Must be positive. Is not changed.

M INTEGER. Number of functions, m. Must be positive. Is not changed.

L INTEGER. Number of constraints, L. Must be positive. Is not changed.

LEQ INTEGER. Number of equality constraints, L_{eq}. Must be positive, less than N and at most L. Is not changed.

B REAL*8 ARRAY with L elements. Vector with the constant terms in the constraints (3.1b),
B(I) $= b_{\text{I}}$, I $= 1, \ldots,$ L .
Is not changed.

A REAL*8 2-dimensional ARRAY with L rows and N columns. Matrix with with the coefficients of the constraints (3.1b) arranged rowwise,
A(I,J) $= a_{\text{J}}^{(\text{I})}$, I $= 1, \ldots,$ L, J $= 1, \ldots,$ N .
Is not changed.

X REAL*8 ARRAY with N elements. The use depends on the entry value of ICONTR.
ICONTR > 0 : *On entry*: Initial approximation to $\mathbf{x}^*$.
On exit: Computed solution.
ICONTR ≤ 0 : Point at which the gradients should be checked. Not changed.

DX REAL*8. The use depends on the entry value of ICONTR,
ICONTR > 0 : Radius of trust region, see **Remarks** above.
On entry: DX $= \Delta_0$. Must be positive.
On exit: Final trust region radius.
ICONTR ≤ 0 : Check of Jacobian matrix with DX used for h in (1.11). Must be significantly nonzero. Is not changed.

EPS REAL*8. Used only when ICONTR > 0. Must be positive.
On entry: Desired accuracy: The algorithm stops when it suggests to change the iterate from $\mathbf{x}_k$ to $\mathbf{x}_k+\mathbf{h}_k$ with $\|\mathbf{h}_k\| <$ EPS$\cdot\|\mathbf{x}_k\|$.
If EPS was chosen too small, then the iteration stops when there is indication that rounding errors dominate, and ICONTR $= 2$ is returned.
On exit: Length of the last step in the iteration.

MAXFUN INTEGER. Used only if the entry value of ICONTR is positive.
On entry: Upper bound on the number of calls of FDF. Must be positive.
On exit: Number of calls of FDF.

W REAL*8 ARRAY with IW elements. Work space.
Entry values are not used, and exit values depend on the entry value of ICONTR,
ICONTR$_{\text{entry}} > 0$:
W(1) $= F$(X) defined by (3.1a),
W(2,..,M+1) : f_{I}(X), I $= 1, \ldots,$ M,
W(M+2,..,M+M*N+1) : Jacobian $\mathbf{J}$(X), stored columnswise..
W(M+M*N+2,..,M+M*N+L+1) : c_{I}(X), I $= 1, \ldots,$ L.

ICONTR$_{\text{entry}} \leq 0$: Results of the gradient check are returned in the first 10 elements of W as follows, cf. (1.10)

W(1)	Maximum element in |DF|.
W(2), W(5), W(6)	δ^F, i^F, j^F.
W(3), W(7), W(8)	δ^B, i^B, j^B.
W(4), W(9), W(10)	δ^E, i^E, j^E.

In case of an error the indices point out the erroneous element of the Jacobian matrix.

IW INTEGER. Length of work space W. Must be at least
$2mn + 5n^2 + 5m + 10n + 4L$. Is not changed.

ICONTR INTEGER. *On entry*: Controls the computation,

ICONTR > 0 : Start minimization.
ICONTR ≤ 0 : Check gradient. No iteration.

On exit: Information about performance,
ICONTR $= 0$: Successful call. Regular Solution.
ICONTR $= 1$: Successful call. Singular Solution.
ICONTR $= 2$: Iteration stopped because too many iterations were needed, see MAXFUN, or rounding errors dominate, see EPS.
ICONTR $= 3$: The subroutine failed to find a point $\mathbf{x}$ satisfying all constraints. The feasible region is presumably empty.
ICONTR < 0 : Computation did not start for the following reason,
ICONTR $= -2$: N ≤ 0
ICONTR $= -3$: M ≤ 0
ICONTR $= -4$: L < 0
ICONTR $= -5$: LEQ < 0 or LEQ $> L$ or LEQ $\geq$ N
ICONTR $= -9$: |DX| is too small in case of gradient check, or DX ≤ 0 in case of optimization
ICONTR $= -10$: EPS ≤ 0
ICONTR $= -11$: MAXFUN ≤ 0
ICONTR $= -13$: IW$<2mn+5n^2+5m+10n+4L$

Example. Minimize

$$F(\mathbf{x}) = \sum_{i=1}^{3} |f_i(\mathbf{x})| \ ,$$

subject to the constraint

$$c(\mathbf{x}) \equiv -x_1 + x_2 + 2 \geq 0 \ .$$

The f_i are given by (1.14), page 6.

```
      PROGRAM  TINCL1
*************************************************************************
* Test MINCL1.  30.1.2002
*************************************************************************
      IMPLICIT          NONE
      INTEGER           I,IC,ICONTR,IW,L,LEQ,M,MAXFUN,N
      DOUBLE PRECISION  A(1,2),B(1),DX,EPS,X(2),W(71)
      EXTERNAL          FDF
C       ... Set parameters
        DATA  N, M, L, LEQ, IW, MAXFUN,   A,     B,   EPS,    X
     &       /2, 3, 1,  0,  71,   25, -1D0,1D0, 2D0, 1D-10,  2*1D0/
        ICONTR = 0
        ICONTR = 1
        IF  (ICONTR .LE. 0)  THEN
C         ... Check  Jacobian
          DX = 1D-3
          CALL MINCL1(FDF,N,M,L,LEQ,B,A,X,DX,EPS,MAXFUN,W,IW,ICONTR)
          IF  (ICONTR .LT. 0)  THEN
            WRITE(6,10) -ICONTR
          ELSE
            WRITE(6,20) W(1), W(2),INT(W(5)), INT(W(6))
            WRITE(6,30) 'Backward', W(3),INT(W(7)),INT(W(8))
            WRITE(6,30) ' Extrap.', W(4),INT(W(9)),INT(W(10))
          ENDIF
        ELSE
C         ... Optimize
```

```
        DX = .1D0
        CALL MINCL1(FDF,N,M,L,LEQ,B,A,X,DX,EPS,MAXFUN,W,IW,ICONTR)
        IF  (ICONTR .LT. 0)  THEN
          WRITE(6,10) -ICONTR
        ELSE
          WRITE(6,40) ICONTR,MAXFUN
          WRITE(6,50) X(1),X(2), W(1)
          IC = M+1 + M*N
          WRITE(6,60) (W(I),I=2,M+1), (W(I),I=IC+1,IC+L)
        ENDIF
      ENDIF
 10   FORMAT('Parameter number',I3,'  is outside its range')
 20   FORMAT('Test of Jacobian.  Max|DF| =',1P1D12.4/'Max difference',
     &    5X,'Forward :',1P1D12.4,'  at  i,j =',I2,',',I2)
 30   FORMAT(18X,A8,' :',1P1D12.4,'  at  i,j =',I2,',',I2)
 40   FORMAT('Optimization.  ICONTR =',I2,'.',I4,' calls of FDF')
 50   FORMAT('   x =',1P2D17.9/'F(x) =',1P2D17.9)
 60   FORMAT('f(x) =',1P3D17.9/'c(x) =',1P3D17.9)
      STOP
    END

    SUBROUTINE FDF(N,M,X,DF,F)
    INTEGER           N,M
    DOUBLE PRECISION  X(N),DF(M,N),F(M)
      F(1) = 1.5D0   - X(1)*(1D0 - X(2))
      F(2) = 2.25D0  - X(1)*(1D0 - X(2)**2)
      F(3) = 2.625D0 - X(1)*(1D0 - X(2)**3)
      DF(1,1) = X(2)-1D0
      DF(1,2) = X(1)
      DF(2,1) = X(2)**2 - 1D0
      DF(2,2) = 2D0*X(1)*X(2)
      DF(3,1) = X(2)**3 - 1D0
      DF(3,2) = 3D0*X(1)*X(2)**2
      RETURN
    END
```

We get the results

```
Test of Jacobian.  Max|DF| =  3.0000E+00
Max difference     Forward :  3.0010E-03  at  i,j = 3, 2
                  Backward : -1.4998E-03  at  i,j = 3, 2
                   Extrap. :  5.0000E-07  at  i,j = 3, 2
```

These results indicate that the gradients of the $\{f_i\}$ are implemented correctly, and changing the initial value of `ICONTR` from 0 to 1 we get

```
Optimization.  ICONTR = 0.   9 calls of FDF
   x =  2.366025404E+00  3.660254038E-01
F(x) =  5.759618943E-01
f(x) = -9.638557313E-17  2.009618943E-01  3.750000000E-01
c(x) =  0.000000000E+00
```

3.3. MINCIN. Linearly Constrained Minimax Optimization of a Vector Function

Purpose. Find $\mathbf{x}^*$ that minimizes $F(\mathbf{x})$, where

$$F(\mathbf{x}) = \max_i \{ f_i(\mathbf{x}) \} \quad , \tag{3.2a}$$

and where the vector of unknown parameters $\mathbf{x} = [x_1, \ldots, x_n]^\top \in \mathbb{R}^n$ must satisfy the following linear equality and inequality constraints,

$$\begin{aligned} c_i(\mathbf{x}) &\equiv \mathbf{a}_i^\top \mathbf{x} + b_i = 0 \ , \quad i = 1, 2, \ldots, L_{\text{eq}} \ , \\ c_i(\mathbf{x}) &\equiv \mathbf{a}_i^\top \mathbf{x} + b_i \geq 0 \ , \quad i = L_{\text{eq}}{+}1, \ldots, L \ , \end{aligned} \tag{3.2b}$$

for given vectors $\{\mathbf{a}_i\}$ and scalars $\{b_i\}$. The f_i, $i = 1, \ldots, m$ is a set of functions that are twice continuously differentiable. The user must supply a subroutine that evaluates $\mathbf{f}(\mathbf{x})$ and the Jacobian $\mathbf{J}(\mathbf{x})$. There is an option for checking the implementation of $\mathbf{J}$.

Method. The algorithm is iterative. It is based on successive linearizations of the on-linear functions f_i, combining a first order trust region method with a local method that uses approximate second order information, see [8].

Origin. Subroutine `MLA1QS` by Jørgen Hald [7].

Remarks. The trust region around the the current $\mathbf{x}$ is the ball centered at $\mathbf{x}$ with radius Δ defined so that the linearizations of the nonlinear functions f_i are reasonably accurate for all points inside the ball. During iteration this bound is adjusted according to how well the linear approximations centered at the previous iterate predict the gain in F.

The user has to give an initial value for Δ. If the functions are almost linear, then we recommend to use an estimate of the distance between $\mathbf{x}_0$ and the solution $\mathbf{x}^*$. Otherwise, we recommend $\Delta_0 = 0.1\|\mathbf{x}_0\|$.

A solution is said to be *"regular"* when it is a strict local minimum, i.e. there exists a positive number K such that

$$F(\mathbf{x}) - F(\mathbf{x}^*) \geq K\|\mathbf{x} - \mathbf{x}^*\|$$

for any feasible $\mathbf{x}$ near $\mathbf{x}^*$. Otherwise, the solution is said to be *"singular"*.

`MINCIN` can also be used to compute a linearly constrained minimizer of the ℓ_∞-norm of $\mathbf{f}$,

$$F(\mathbf{x}) = \max_i | f_i(\mathbf{x}) | \quad . \tag{3.3a}$$

For that purpose we introduce the extended vector function $\widehat{\mathbf{f}} : \mathbb{R}^n \mapsto \mathbb{R}^{2m}$ defined by

$$\widehat{f}_i(\mathbf{x}) = \begin{cases} f_i(\mathbf{x}) & \text{for } i = 1, 2, \ldots, m \\ -f_{i-m}(\mathbf{x}) & \text{for } i = m{+}1, \ldots, 2m \end{cases} \quad . \tag{3.3b}$$

It is easily seen that $\max\limits_{i=1,\ldots,2m} \{\widehat{f}_i(\mathbf{x})\} = \max\limits_{i=1,\ldots,m} \{|f_i(\mathbf{x})|\}$.

Use. The subroutine call is

```
CALL MINCIN(FDF,N,M,L,LEQ,B,A,X,DX,EPS,MAXFUN,W,IW,ICONTR)
```

The parameters are

`FDF` `SUBROUTINE` written by the user with the following declaration

```
SUBROUTINE FDF(N,M,X,DF,F)
REAL*8 X(N),DF(M,N),F(N)
```

It must calculate the values of the functions and their gradients at the point $\mathbf{x} = [\texttt{X(1)}, \ldots, \texttt{X(N)}]^\top$ and store these numbers as follows,

$\texttt{F(I)} = f_{\texttt{I}}(\mathbf{x}),\ \texttt{I} = 1, \ldots, \texttt{M},$

$$\texttt{DF(I,J)} = \frac{\partial f_{\texttt{I}}}{\partial x_{\texttt{J}}}(\mathbf{x}),\ \begin{cases} \texttt{I} = 1, \ldots, \texttt{M} \\ \texttt{J} = 1, \ldots, \texttt{N} \end{cases}$$

The name of this subroutine (which can be chosen freely by the user) must appear in an `EXTERNAL` statement in the calling program.

`N` `INTEGER`. Number of unknowns, n. Must be positive. Is not changed.

`M` `INTEGER`. Number of functions, m. Must be positive. Is not changed.

`L` `INTEGER`. Number of constraints, L. Must be positive. Is not changed.

`LEQ` `INTEGER`. Number of equality constraints, L_{eq}. Must be positive, less than `N` and at most `L`. Is not changed.

`B` `REAL*8 ARRAY` with `L` elements. Vector with the constant terms in the constraints (3.2b),
$\texttt{B(I)} = b_{\texttt{I}},\ \ \texttt{I} = 1, \ldots, \texttt{L}$.
Is not changed.

`A` `REAL*8` 2-dimensional `ARRAY` with `L` rows and `N` columns. Matrix with with the coefficients of the constraints (3.2b) arranged rowwise,
$\texttt{A(I,J)} = a_{\texttt{J}}^{(\texttt{I})},\ \ \texttt{I} = 1, \ldots, \texttt{L},\ \ \texttt{J} = 1, \ldots, \texttt{N}$.
Is not changed.

`X` `REAL*8 ARRAY` with `N` elements. The use depends on the entry value of `ICONTR`.
$\texttt{ICONTR} > 0$: *On entry*: Initial approximation to $\mathbf{x}^*$.
On exit: Computed solution.
$\texttt{ICONTR} \le 0$: Point at which the gradients should be checked. Not changed.

`DX` `REAL*8`. The use depends on the entry value of `ICONTR`,
$\texttt{ICONTR} > 0$: Radius of trust region, see **Remarks** above.
On entry: $\texttt{DX} = \Delta_0$. Must be positive.
On exit: Final trust region radius.
$\texttt{ICONTR} \le 0$: Check of Jacobian matrix with `DX` used for h in (1.11). Must be significantly nonzero. Is not changed.

`EPS` `REAL*8`. Used only when $\texttt{ICONTR} > 0$. Must be positive.
On entry: Desired accuracy: The algorithm stops when it suggests to change the iterate from $\mathbf{x}_k$ to $\mathbf{x}_k + \mathbf{h}_k$ with $\|\mathbf{h}_k\| < \texttt{EPS} \cdot \|\mathbf{x}_k\|$.
If `EPS` was chosen too small, then the iteration stops when there is indication that rounding errors dominate, and $\texttt{ICONTR} = 2$ is returned.
On exit: Length of the last step in the iteration.

`MAXFUN` `INTEGER`. Used only if the entry value of `ICONTR` is positive.
On entry: Upper bound on the number of calls of `FDF`. Must be positive.
On exit: Number of calls of `FDF`.

`W` `REAL*8 ARRAY` with `IW` elements. Work space.
Entry values are not used, and exit values depend on the entry value of `ICONTR`,
$\texttt{ICONTR}_{\text{entry}} > 0$:
$\texttt{W(1)} = F(\texttt{X})$ defined by (3.2a),
$\texttt{W(2,..,M+1)}$: $f_{\texttt{I}}(\texttt{X}),\ \texttt{I} = 1, \ldots, \texttt{M}$,
`W(M+2,..,M+M*N+1)` : Jacobian $\mathbf{J}(\texttt{X})$, stored columnswise..
$\texttt{W(M+M*N+2,..,M+M*N+L+1)}$: $c_{\texttt{I}}(\texttt{X}),\ \texttt{I} = 1, \ldots, \texttt{L}$.

ICONTR$_{\text{entry}} \le 0$: Results of the gradient check are returned in the first 10 elements of W as follows, cf. (1.10)

W(1)	Maximum element in |DF|.
W(2), W(5), W(6)	$\delta^F,\ i^F,\ j^F$.
W(3), W(7), W(8)	$\delta^B,\ i^B,\ j^B$.
W(4), W(9), W(10)	$\delta^E,\ i^E,\ j^E$.

In case of an error the indices point out the erroneous element of the Jacobian matrix.

IW INTEGER. Length of work space W. Must be at least $4mn+5n^2+8m+8n+4L+3$. Is not changed.

ICONTR INTEGER. *On entry*: Controls the computation,
ICONTR > 0 : Start minimization.
ICONTR ≤ 0 : Check gradient. No iteration.

On exit: Information about performance,
ICONTR $= 0$: Successful call. Regular Solution.
ICONTR $= 1$: Successful call. Singular Solution.
ICONTR $= 2$: Iteration stopped because too many iterations were needed, see MAXFUN, or rounding errors dominate, see EPS.
ICONTR $= 3$: The subroutine failed to find a point $\mathbf{x}$ satisfying all constraints. The feasible region is presumably empty.
ICONTR < 0 : Computation did not start for the following reason,
ICONTR $= -2$: N ≤ 0
ICONTR $= -3$: M ≤ 0
ICONTR $= -4$: L < 0
ICONTR $= -5$: LEQ<0 or LEQ$>L$ or LEQ$\ge$N
ICONTR $= -9$: \|DX\| is too small in case of gradient check, or DX ≤ 0 in case of optimization
ICONTR $= -10$: EPS ≤ 0
ICONTR $= -11$: MAXFUN ≤ 0
ICONTR $= -13$: IW$<4mn+5n^2+8m+8n+4L+3$

Example. Minimize

$$F(\mathbf{x}) = \max_i |\, f_i(\mathbf{x}) \,| \quad ,$$

subject to the constraint

$$c(\mathbf{x}) \equiv -x_1 + x_2 + 2 \ge 0 \ .$$

The f_i are given by (1.14), page 6. This is a problem of computing a linearly constrained minimizer of the ℓ_∞-norm of $\mathbf{f}$, and we extend the vector $\mathbf{f}$ to $\widehat{\mathbf{f}}$ as defined in (3.3b).

```
      PROGRAM  TINCIN
*********************************************************************
* Test MINCIN.  30.01.2002
*********************************************************************
      IMPLICIT          NONE
      INTEGER           I,IC,ICONTR,IW,L,LEQ,M,MAXFUN,N
      DOUBLE PRECISION  A(1,2),B(1),DX,EPS,X(2),W(91)
      EXTERNAL          FDF
C       ... Set parameters
        DATA  N, M, L, LEQ, IW, MAXFUN,   A,     B,   EPS,    X
     &       /2, 6, 1,  0,  91,   25, -1D0,1D0, 2D0, 1D-10,  2*1D0/
        ICONTR = 0
        ICONTR = 1
        IF  (ICONTR .LE. 0)  THEN
```

```
C         ... Check  Jacobian
          DX = 1D-3
          CALL MINCIN(FDF,N,M,L,LEQ,B,A,X,DX,EPS,MAXFUN,W,IW,ICONTR)
          IF  (ICONTR .LT. 0)  THEN
            WRITE(6,10) -ICONTR
          ELSE
            WRITE(6,20) W(1), W(2),INT(W(5)), INT(W(6))
            WRITE(6,30) 'Backward', W(3),INT(W(7)),INT(W(8))
            WRITE(6,30) ' Extrap.', W(4),INT(W(9)),INT(W(10))
            WRITE(7,20) W(1), W(2),INT(W(5)), INT(W(6))
            WRITE(7,30) 'Backward', W(3),INT(W(7)),INT(W(8))
            WRITE(7,30) ' Extrap.', W(4),INT(W(9)),INT(W(10))
          ENDIF
        ELSE
C         ... Optimize
          DX = .1D0
          CALL MINCIN(FDF,N,M,L,LEQ,B,A,X,DX,EPS,MAXFUN,W,IW,ICONTR)
          IF  (ICONTR .LT. 0)  THEN
            WRITE(6,10) -ICONTR
          ELSE
            WRITE(6,40) ICONTR,MAXFUN
            WRITE(6,50) X(1),X(2), W(1)
            IC = M+1 + M*N
            WRITE(6,60) (W(I),I=2,M+1), (W(I),I=IC+1,IC+L)
          ENDIF
        ENDIF
  10    FORMAT('Parameter number',I3,'  is outside its range')
  20    FORMAT('Test of Jacobian.  Max|DF| =',1P1D12.4/'Max difference',
     &    5X,'Forward :',1P1D12.4,'  at  i,j =',I2,',',I2)
  30    FORMAT(18X,A8,' :',1P1D12.4,'  at  i,j =',I2,',',I2)
  40    FORMAT('Optimization.  ICONTR =',I2,'.',I4,' calls of FDF')
  50    FORMAT('   x =',1P2D17.9/'F(x) =',1P2D17.9)
  60    FORMAT('f(x) =',1P3D17.9/6X,1P3D17.9/'c(x) =',1P3D17.9)
        STOP
      END

      SUBROUTINE FDF(N,M,X,DF,F)
      INTEGER           N,M,I,J
      DOUBLE PRECISION  X(N),DF(M,N),F(M)
        F(1) = 1.5D0   - X(1)*(1D0 - X(2))
        F(2) = 2.25D0  - X(1)*(1D0 - X(2)**2)
        F(3) = 2.625D0 - X(1)*(1D0 - X(2)**3)
        DF(1,1) = X(2)-1D0
        DF(1,2) = X(1)
        DF(2,1) = X(2)**2 - 1D0
        DF(2,2) = 2D0*X(1)*X(2)
        DF(3,1) = X(2)**3 - 1D0
        DF(3,2) = 3D0*X(1)*X(2)**2
C       ... Supply with -f
        DO  20  I = 1, 3
          DO  10  J = 1, 2
  10        DF(I+3,J) = -DF(I,J)
  20      F(I+3) = -F(I)
        RETURN
      END
```

We get the results

```
Test of Jacobian.  Max|DF| =  3.0000E+00
Max difference     Forward :  3.0010E-03  at  i,j = 3, 2
                  Backward : -1.4998E-03  at  i,j = 3, 2
                   Extrap. :  5.0000E-07  at  i,j = 3, 2
```

These results indicate that the gradients of the $\{f_i\}$ are implemented correctly, and changing the initial value of `ICONTR` from 0 to 1 we get

```
Optimization.  ICONTR = 1.  15 calls of FDF
   x =  2.366025404E+00  3.660254038E-01
F(x) =  3.750000000E-01
f(x) = -9.638557313E-17  2.009618943E-01  3.750000000E-01
        9.638557313E-17 -2.009618943E-01 -3.750000000E-01
c(x) =  0.000000000E+00
```

References

[1] E.M.L. Beale (1958): *On an Iterative Method of Finding a Local Minimum of a Function of More than one Variable.* Princeton Univ. Stat. Techn. Res. Group, Techn. Rep. 25.

[2] R.M. Chamberlain, C. Lemarechal, H.C. Pedersen and M.J.D. Powell (1982): *The Watchdog Technique for Forcing Convergence in Algorithms for Constrained Optimization.* MATHEMATICAL PROGRAMMING STUDY **16**, 1 – 17.

[3] J.E. Dennis and R.B. Schnabel (1983): *Numerical Methods for Unconstrained Optimization and Nonlinear Equations.* Prentice Hall Series in Computational Mathematics.

[4] J. Dongarra, C.B. Moler, J.R. Bunch and G.W. Stewart. (1988): *An Extended Set of Fortran Basic Linear Algebra Subprograms,* ACM Trans. Math. Soft. **14**, 1–17.

[5] R. Fletcher (1987): *Practical Methods of Optimization,* 2nd edition. Wiley.

[6] J. Hald (1981a): *MMLA1Q, a Fortran Subroutine for Linearly Constrained Minimax Optimization.* Report NI-81-01, Institute for Numerical Analysis (now part of IMM), Technical University of Denmark.

[7] J. Hald (1981b): *A 2-Stage Algorithm for Nonlinear ℓ_1 Optimization.* Report NI-81-03, Institute for Numerical Analysis (now part of IMM), Technical University of Denmark.

[8] J. Hald and K. Madsen (1981): *Combined LP and Quasi-Newton Methods for Minimax Optimization.* MATHEMATICAL PROGRAMMING **20**, 49 – 62.

[9] J. Hald and K. Madsen (1985): *Combined LP and Quasi-Newton Methods for Nonlinear ℓ_1 Optimization.* SIAM J. NUMER. ANAL. **20**, 68 – 80.

[10] *Harwell Subroutine Library.* (1984). Report R9185, Computer Science and Systems Division, Harwell Laboratory, Oxfordshire, OX11 ORA, England.

[11] K. Madsen (1975): *An Algorithm for Minimax Solution of Overdetermined Systems of Nonlinear Equations.* J. IMA **16**, 321 – 328.

[12] K. Madsen, O. Tingleff, P.C. Hansen and W. Owczarz (1990): *Robust Subroutines for Non-Linear Optimization.* Report NI-90-06, Institute for Numerical Analysis (now part of IMM), Technical University of Denmark.

[13] H.B. Nielsen (1999): *Damping Parameter in Marquardt's Method.* Report IMM-REP-1999-05, IMM, DTU. Available at
http://www.imm.dtu.dk/∼hbn/publ/TR9905.ps

[14] H.B. Nielsen (2000): *UCMINF – an Algorithm for Unconstrained Nonlinear Optimization.* Report IMM-REP-2000-19, IMM, DTU. Available at
http://www.imm.dtu.dk/∼hbn/publ/TR0019.ps

[15] J. Nocedal and S.J. Wright (1999): *Numerical Optimization.* Springer, New York.

[16] M.J.D. Powell (1982): *Extension to Subroutine VF02AD.* In R.F. Drenik and F. Kozin (eds.), "System Modeling and Optimization", LECTURE NOTES IN CONTROL AND INFORMATIONS SCIENCES **38**, Springer-Verlag, 529 – 538.

[17] M.J.D. Powell (1985): *On the Quadratic Programming Algorithm of Goldfarb and Idnani.* MATHEMATICAL PROGRAMMING STUDY **25**, 46 – 61.