1

STENMIN: User's Manual

Ali Bouaricha[†]

Argonne National Laboratory

1. Interfaces and Usage

Two interfaces have been provided with the package. If the user wishes to use all the defaults options provided by the package, then he (or she) should call STUMSD (STUMSS if single–precision is used). Only the number of variables N, the function FCN that computes $f(x)$, an initial guess X0, the number of nonzeros NZ stored in the lower or upper triangular part of the Hessian matrix, and the row and column indices IRN and ICN of these nonzeros given in any order, need to be supplied. The other interface, STUMCD (STUMCS if single–precision is used), requires the user to supply all parameters. The user may specify selected parameters only by first invoking the subroutine STDFLT, which sets all parameters to their default values, and then overriding only the desired values. The two calling sequences are as follows:

```
C STUMSD interface: the default options provided by STENMIN are used.

      CALL STUMSD(N, X0, NZ, IRN, LIRN, ICN, LICN, FCN, TYPX, MSG, XPLS,
     *            FPLS, GPLS, HESS, WRK, LWRK, IWRK, LIWRK, TERMCD)


C STUMCD interface: the user first invokes the subroutine STDFLT to obtain the default
C options provided by STENMIN, then overrides the desired values.

      CALL STDFLT(N, TYPX, FSCALE, GRADTL, STEPTL, ILIM, STEPMX,
     *            IPR, METHOD, GRDFLG, HSNFLG, NDIGIT, INFORM, MSG)

C USER OVERRIDES SPECIFIC DEFAULT VALUES PARAMETERS, E.G.

              GRADTL   = 1.0D-6
              ILIM     = 1000
              GRDFLG   = 1
              HSNFLG   = 1
```

*Part of this work was performed while the author was research associate at CERFACS (Centre Européen de Recherche et de Formation Avancée en Calcul Scientifique, Toulouse, France).

†Author's address: Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Illinois, 60439. bouarich@@mcs.anl.gov. This work was supported in part by the Office of Scientific Computing, U.S. Department of Energy, under Contract W-31-109-Eng-38.

```
 CALL STUMCD(N, XO, NZ, IRN, LIRN, ICN, LICN, FCN, UGR, USH,
*            TYPX, FSCALE, GRADTL, STEPTL, ILIM, STEPMX, IPR,
*            METHOD, GRDFLG, HSNFLG, NDIGIT, MSG, XPLS, FPLS,
*            GPLS, HESS, WRK, LWRK, IWRK, LIWRK, TERMCD, HTV,
*            INFORM)
```

2. Parameters and Default Values

The parameters used in the calling sequences of §1 are fully described here. STUMSD uses only those parameters that are preceded by an asterisk. When it is noted that module STDFLT returns a given value, this is the default employed by interface STUMSD. The user may override the default value by utilizing STUMCD.

Following each variable name in the list below appears a one– or a two–headed arrow symbol of forms $\rightarrow$, $\leftarrow$, and $\longleftrightarrow$. These symbols signify that the variable is for input, output, and input-output, respectively.

*N$\rightarrow$: A positive integer variable specifying the number of variables in the problem. **Restriction:** N ≥ 1.

*XO$\rightarrow$: An array of length N that contains an initial estimate of the minimizer x_*.

*NZ$\rightarrow$: An integer variable that must be set by the user to the number of nonzeros stored in the lower or upper triangular part of the Hessian matrix. It is not altered by the program. **Restriction:** NZ ≥ 1.

*IRN$\rightarrow$: An integer array of length LIRN. On entry, it must hold the row index of each nonzero stored in the lower or upper triangular part of the Hessian matrix.

*LIRN$\rightarrow$: An integer variable that must be set by the user to the length of array IRN. LIRN need not be as large as LICN; normally it need not be very much greater than NZ. It is not altered by the program. **Restriction:** LIRN $\geq$ NZ.

*ICN$\rightarrow$: An integer array of length LICN. On entry, it must hold the column index of the nonzeros stored in lower or upper triangular part of the Hessian matrix.

*LICN$\rightarrow$: An integer variable that must be set by the user to the length of the Hessian array HESS and ICN. LICN should ordinarily be 2 to 4 times as large as NZ. It is not altered by the program. **Restriction:** LICN $\geq$ NZ.

*FCN$\rightarrow$: The name of a user supplied subroutine that evaluates the function f at an arbitrary vector x. The subroutine must be declared EXTERNAL in the user's calling program and must conform to

```
CALL FCN(N, X, F),
```

where X is a vector of length N. The subroutine must not alter the values of X.

UGR$\rightarrow$: The name of a user supplied subroutine that returns in G, the value of the gradient $\nabla f(x)$ at the current point X. UGR must be declared EXTERNAL in the user's calling program and must conform to the usage

```
CALL UGR(N, X, G),
```

where N is the dimension of the problem, X is a vector of length N, and G is the gradient at X. UGR must not alter the values of N and X. When using the interface STUMCD, if no analytic gradient is supplied (GRDFLG = 0), the user must use the dummy name STDUGR.

USH$\rightarrow$: The name of a user supplied subroutine that returns in HESS, the value of the Hessian $\nabla^2 f(x)$ at the current point X. USH must be declared EXTERNAL in the user's calling program and must conform to the usage

```
CALL USH(N, X, NZ, LICN, HESS, IRN, ICN)
```

where N is the dimension of the problem, X is a vector of length N, HESS is the Hessian matrix at X, LICN is the length of HESS, NZ is the number of nonzeros in the lower or upper triangular part of HESS, and IRN and ICN are the row and column indices of the nonzeros in HESS. USH must not alter the values of N, X, and LICN. Only the lower or upper triangular part of HESS should be given. When using the interface STUMCD, if no analytic Hessian is supplied (HSNFLG = 0), the user must use the dummy name STDUSH.

*TYPX$\rightarrow$: An array of length N in which the typical size of the components of X are specified. The typical component sizes should be positive real scalars. If a negative value is specified, its absolute value will be used. When 0.0 is specified, 1.0 will be used. The program will not abort. This vector is used by the the package to determine the scaling matrix D_x. Although the package may work reasonably well in a large number of instances without scaling, it may fail when the components of x_* are of radically different magnitude and scaling is not invoked. If the sizes of the parameters are known to differ by many orders of magnitude, then the scale vector TYPX should definitely be used. Module STDFLT returns TYPX = (1.0,..., 1.0). For example, if it is anticipated that the range of values for the iterates x_k is

$$\begin{array}{lll} x_1 & \in & [-10^{10} \ , \ 10^{10}] \\ x_2 & \in & [-10^{2} \ , \ 10^{4}] \\ x_3 & \in & [-6 \times 10^{-6}, \ 9 \times 10^{-6}] \end{array}$$

then an appropriate choice will be TYPX = (1.0E+10, 1.0E+3, 7.0E-6).

FSCALE$\rightarrow$: A positive real number estimating the magnitude of $f(x)$ near the minimizer x_*. It is used in the gradient stopping condition given below. If $f(x_0)$ is much greater than $f(x_*)$, FSCALE should be approximately $f(x_*)$. If a negative value is specified for FSCALE, its absolute value is used. When 0.0 is specified, 1.0 will be used. The program will not abort.

GRADTL→: Positive scalar giving the tolerance at which the scaled gradient of $f(x)$ is considered close enough to zero to terminate the algorithm. The scaled gradient is a measure of the relative change in f in each direction x_i divided by the relative change in x_i. More precisely, the test used by the program is

$$\max_i \left\{ \frac{\mid \nabla f(x) \mid_i \ \max\{\mid x_i \mid, \mathtt{TYPX}_i\}}{\max\{\mid f \mid, \mathtt{FSCALE}\}} \right\} \leq \mathtt{GRADTL}.$$

The module STDFLT returns the value $\epsilon^{1/3}$. If the user specifies a negative value, the default value is used instead.

STEPTL→: A positive scalar providing the minimum allowable relative step length. STEPTL should be at least as small as 10^{-d}, where d is the number of accurate digits the user desires in the solution x_*. The actual test used is

$$\max_i \left\{ \frac{|{x_i}^k - {x_i}^{k-1}|}{\max\{|{x_i}^k, \mathtt{TYPX}_i|\}} \right\} \leq \mathtt{STEPTL},$$

where x^k and x^{k-1} are the new and old iterates, respectively. The program may terminate prematurely if STEPTL is too large. Module STDFLT returns the value $\epsilon^{2/3}$. If the user specifies a negative value, then the default value is used instead.

ILIM→: Positive integer specifying the maximum iterations to be performed before the program is terminated. Module STDFLT returns ILIM = 500. If the user specifies ILIM ≤ 0, the default value is used instead.

STEPMX→: A positive scalar providing the maximum allowable scaled step length $||D_x(x_+ - x_c)||_2$, where $D_x = diag(\mathtt{1/TYPX}_1, \ldots, \mathtt{1/TYPX}_n)$. STEPMX is used to prevent steps that would cause the optimization problem to overflow, to prevent the algorithm from leaving the area of interest in parameter space, or to detect divergence in the algorithm. STEPMX should be chosen small enough to prevent these occurrences but should be larger than any anticipated "reasonable" step. The algorithm will halt and provide a diagnostic if it attempts to exceed STEPMX on five successive iterations. If a nonpositive value is specified for STEPMX, the default is used. Module STDFLT returns the value STEPMX $= \max\{||x_0||_2 \cdot 10^3, 10^3\}$, where x_0 is the initial approximation provided by the user.

IPR→: The unit on which the routine outputs information. STDFLT returns the value 6, which is the standard FORTRAN unit for the printer.

METHOD→: An integer flag designating which method to use.

- METHOD $= 0$: Use Newton's method.
- METHOD $= 1$: Use the tensor method.

Module STDFLT returns value 1. If the user specifies an illegal value, module OPTCHK will set METHOD to 1; the program will not abort.

GRDFLG→: Integer flag designating whether or not analytic Hessian has been supplied by the user.

- GRDFLG $= 0$: No analytic gradient supplied.
- GRDFLG $= 1$: Analytic gradient supplied (will be checked against finite difference gradient.)
- GRDFLG $= 2$: Analytic gradient supplied (will not be checked against finite difference gradient.)

When GRDFLG $= 0$, the gradient is obtained by forward finite differences. When GRDFLG $= 1$ or 2, the name of the user supplied routine that evaluates $\nabla f(x)$ must be supplied in UGR. When GRDFLG $= 1$, the program compares the value of the user's analytic gradient routine at x_0 with a finite difference estimate and aborts if the relative difference between any two components is greater than 0.01. The module STDFLT returns GRDFLG $= 0$. If the user specifies an illegal value, the module OPTCHK supplies the value 0.

HSNFLG$\rightarrow$: Integer flag designating whether or not analytic Hessian has been supplied by the user.

- HSNFLG $= 0$: No analytic Hessian supplied.
- HSNFLG $= 1$: Analytic Hessian supplied (will be checked against finite difference Hessian.)
- HSNFLG $= 2$: Analytic Hessian supplied (will not be checked against finite difference Hessian.)

When HSNFLG $= 0$, the Hessian values are computed by forward finite differences based on gradient values. When HSNFLG $= 1$ or 2, the name of the user–supplied routine that evaluates $\nabla^2 f(x)$ must be supplied in USH. When HSNFLG $= 1$, the program compares the value of the user's analytic Hessian routine at x_0 with a finite difference estimate and aborts if the relative difference between any two components is greater than 0.01. The module STDFLT returns HSNFLG $= 0$. If the user specifies an illegal value, the module OPTCHK supplies the value 0.

NDIGIT$\rightarrow$: Integer estimating the number of accurate digits on the objective function $f(x)$. STDFLT returns the value -LOG$_{10}(\epsilon)$, where ϵ is machine precision. If NDIGIT ≤ 0 then the default value is used instead.

*MSG$\longleftrightarrow$: An integer variable that the user may set on input to inhibit certain automatic checks or override certain default characteristics of the package. Currently, three "message" features can be used individually or in combination.

- MSG $= 0$: No output will be produced.
- MSG $= 1$: Print the input state, the final results, and the stopping conditions.
- MSG $= 2$: Print the intermediate results, that is, the input state, the values of the objective function and the scaled gradient at each iteration, and the final results including the stopping conditions and the number of function, gradient, and Hessian evaluations.

The module STDFLT returns a value of 1. On output, if the program has terminated because of erroneous input, MSG contains an error code indicating the reason:

- MSG $=$ -1 : Illegal dimension N; N ≤ 0. The program aborts.
- MSG $=$ -2 : Illegal length of LIRN or LICN; LIRN ≤ 0 or LICN ≤ 0. The program aborts.
- MSG $=$ -3 : Illegal length of LIWRK or LWRK; LIWRK $<$ 2*LIRN+12*N+2 or LWRK $<$ 7*N. The program aborts.
- MSG $=$ -4 : Illegal number of nonzeros NZ; NZ ≤ 0. The program aborts.

- MSG = -5 : The K-th element of IRN or the K-th element of ICN is not an integer between 1 and N; (IRN(K) < 1 or IRN(K) > N) or (ICN(K) < 1 or ICN(K) > N). The program aborts.
- MSG = -6 : The K-th diagonal element is not in the sparsity pattern. This is checked only if HSNFLG = 0 because the finite difference Hessian approximation require that diagonal elements be in the sparsity pattern. The program aborts.
- MSG = -7 : Redundant entries in sparsity pattern was encountered. When HSNFLG = 1 or HSNFLG = 2, the program aborts. When HSNFLG = 0, the program eliminates the redundant entries and continue the execution (no error message is reported in this case).
- MSG = -8 : Probable coding error in the user's analytic gradient routine UGR. Analytic and finite difference gradient do not agree within a tolerance of 0.01. The program aborts. (This check can be overridden by setting GRDFLG = 2.)
- MSG = -9 : Probable coding error in the user's analytic Hessian routine USH. Analytic and finite difference Hessian do not agree within a tolerance of 0.01. The program aborts. (This check can be overridden by setting HSNFLG = 2.)

*XPLS←: An array of length N containing the best approximation to the minimizer x_* upon return. (If the algorithm has not converged, the last iterate is returned.)

*FPLS←: A scalar variable that contains the function value at the final iterate XPLS.

*GPLS←: An array of length N containing the gradient value at XPLS.

HESS⟷: An array that is used to store the Hessian matrix at each iteration. It needs to be at least of dimension LICN. Only the nonzeros in the lower or upper triangular part of the Hessian matrix is stored in HESS. On entry, these nonzeros may be given in any order. On exit, HESS contains the Hessian matrix at the minimizer x_* with the nonzeros sorted by columns if HSNFLG was set to 0.

*WRK→: An array of length LWRK. This is used as workspace by the package. Its length must be at least 8*N if the STUMSD interface is used and at least 7*N if the STUMCD interface is used.

*LWRK→: An integer variable. It must be set by the user to the length of array WRK and is not altered by the package.

*IWRK→: An integer array of length LIWRK. This is used as workspace by the package. Its length must be at least 2*LIRN+12*N+2.

*LIWRK→: An integer variable. It must be set by the user to the length of array IWRK and is not altered by the package.

*TERMCD←: An integer that specifies the reason why the algorithm has terminated.

- TERMCD = 1 : The scaled gradient at the final iterate was less than GRADTL.
- TERMCD = 2 : The length of the last step was less than STEPTL.

- TERMCD = 3 : Last global step failed to locate a point lower than XPLS. It is likely that either XPLS is an approximate solution of the function or STEPTL is too large.
- TERMCD = 4 : The iteration limit has been exceeded.
- TERMCD = 5 : Five consecutive steps of length STEPMX have been taken.

HTV$\leftrightarrow$: An array of length N. It need not be set by the user on entry. If INFORM is set to 1, a re-entry must be made with HTV set to HESS times HTV (see INFORM.)

INFORM$\longleftrightarrow$: An integer variable. If it is set to 1, the user must obtain HESS times HTV and re-enter STUMCD (STUMCS if single–precision is used) with INFORM unchanged. The result of HESS times HTV must be stored in HTV. The default value of INFORM is 0, meaning that HESS times HTV is computed by the package.

3. Summary of Default Values

The following parameters are returned by the module STDFLT:

ILIM = 500
GRDFLG = 0
HSNFLG = 0
IPR = 6
GRADTL = $\epsilon^{1/3}$ (ϵ is machine precision)
STEPTL = $\epsilon^{2/3}$
METHOD = 1
NDIGIT = $-\mathrm{LOG}_{10}(\epsilon)$
STEPMX = 0.0
TYPX = (1.0,...,1.0)
FSCALE = 1.0
MSG = 1
INFORM = 0

4. Implementation Details

This software package has been coded in Fortran 77. The user has the choice between single– and double–precision versions. The user must then preprocess the package at compile time using either the **tosngl** or **todble** tools from CUTE [2], for the single– and double–precision versions, respectively. The **tosngl** program picks up the appropriate version by selecting any statement that begins with CS in the first column, where the S character means that this is a single–precision version. On the other hand, the **todble** program picks up the appropriate version by selecting any statement that begins with CD in the first column, with D meaning that this is a double–precision version. Note that a statement that begins by neither CS nor CD will be picked by both tools.

The following software are included in the package:

1. The Coleman and Moré graph coloring software [6, 5, 4], which is used for estimating a finite-difference approximation of a sparse Hessian matrix.

2. The subroutine DSYPRC [7, 8], which is used for modifying the negative eigencomponents obtained when factorizing an indefinite Hessian matrix using the Harwell code MA27.

3. The function DPMEPS [3], which is used for dynamically determining the machine precision.

The program was developed and tested on a Sun SPARC 10 Model 40 computer.

The machine precision is calculated by the package and used in several places including finite differences stepsizes and stopping criteria. On some computers, the returned value may be incorrect because of compiler optimizations. The user may wish to check the computer value of the machine epsilon and, if it is incorrect, replace the code in the function DPMEPS with the following statement

```
DPMEPS = correct value of machine epsilon
```

5. Example of Use

In the example code shown in Figure 5.1, we first call the routine STDFLT, which returns the default values. We then override the values of GRADTL, GRDFLG and HSNFLG. Next we call either the interface STUMSD or STUMCD for the single– and double–precision version, respectively, to solve the sparse unconstrained optimization problem coded in FCN and whose gradient and Hessian are given by UGRAD and UHESS, respectively.

```
C
C STENMIN MINIMIZES AN UNCONSTRAINED NONLINEAR FUNCTION IN N
C UNKNOWNS WHERE THE HESSIAN IS LARGE AND SPARSE, USING TENSOR
C METHODS.
C
C EXAMPLE OF USE FOR STENMIN.  THE TEST PROBLEM IS THE
C THE BROYDEN TRIDIAGONAL [10].
C
C  ALI BOUARICHA, OCTOBER 1994.
C  MCS DIVISION, ARGONNE NATIONAL LAB.
C
      INTEGER          NMAX, N, NZ, LIRN, LICN, ILIM, IPR, METHOD
      INTEGER          GRDFLG, HSNFLG, NDIGIT, MSG, LWRK, LIWRK
      INTEGER          TERMCD, INFORM, I
CD    DOUBLE PRECISION FSCALE, GRADTL, STEPTL, FPLS, STEPMX, ONE
CS    REAL             FSCALE, GRADTL, STEPTL, FPLS, STEPMX, ONE
      PARAMETER      ( NMAX = 10000, LIRN = 50000, LICN = 500000 )
      PARAMETER      ( LIWRK = 2 * LIRN + 12 * NMAX + 2 )
      PARAMETER      ( LWRK = 7 * NMAX )
      INTEGER          IRN ( LIRN  ), ICN ( LICN )
      INTEGER          IWRK( LIWRK )
```

```
CD    DOUBLE PRECISION X    ( NMAX ),  TYPX( NMAX ), XPLS( NMAX )
CD    DOUBLE PRECISION GPLS( NMAX ),  HESS( LICN ), WRK ( LWRK )
CD    DOUBLE PRECISION HTV  ( NMAX )
CS    REAL             X    ( NMAX ),  TYPX( NMAX ), XPLS( NMAX )
CS    REAL             GPLS( NMAX ),  HESS( LICN ), WRK ( LWRK )
CS    REAL             HTV  ( NMAX )
      EXTERNAL         FCN, UGRAD, UHESS
CD    DATA ONE / 1.0D0 /
CS    DATA ONE / 1.0E0 /

C READ DATA

      READ(5,*) N

C  COMPUTE THE STANDARD STARTING POINT.

      DO 10 I = 1, N
         X(I) = -ONE
 10   CONTINUE

      CALL STDFLT(N,TYPX,FSCALE,GRADTL,STEPTL,ILIM,STEPMX,
     *            IPR,METHOD,GRDFLG,HSNFLG,NDIGIT,INFORM,MSG)

CD    GRADTL = 1.0D-5
CS    GRADTL = 1.0E-3
      GRDFLG = 2
      HSNFLG = 2

C  CALL THE SPARSE OPTIMIZER

CD    CALL STUMCD(N,X,NZ,IRN,LIRN,ICN,LICN,FCN,UGRAD,
CS    CALL STUMCS(N,X,NZ,IRN,LIRN,ICN,LICN,FCN,UGRAD,UHESS,TYPX,
     *            FSCALE,GRADTL,STEPTL,ILIM,STEPMX,IPR,METHOD,
     *            GRDFLG,HSNFLG,NDIGIT,MSG,XPLS,FPLS,GPLS,HESS,
     *            WRK,LWRK,IWRK,LIWRK,TERMCD,HTV,INFORM)

      STOP
      END

C THE FOLLOWING IS A SUBROUTINE FOR THE BROYDEN TRIDIAGONAL
C PROBLEM

        SUBROUTINE FCN(N, X, F)
        INTEGER N
```

```
CD      DOUBLE PRECISION X(N), F
CS      REAL X(N), F

C  LOCAL VARIABLES

        INTEGER I
CD      DOUBLE PRECISION ONE, TWO, THREE
CS      REAL ONE, TWO, THREE
CD      DATA ONE, TWO, THREE / 1.0D0, 2.0D0, 3.0D0 /
CS      DATA ONE, TWO, THREE / 1.0E0, 2.0E0, 3.0E0 /

        F = ((THREE - TWO * X(1)) * X(1) - TWO * X(2) + ONE) ** 2
        DO 10 I = 2, N-1
           F = F + ((THREE - TWO * X(I)) * X(I) - X(I-1) -
     *         TWO * X(I+1) + ONE) ** 2
 10     CONTINUE
        F = F + ((THREE - TWO * X(N)) * X(N) - X(N-1) + ONE) ** 2
        RETURN
        END

C THE FOLLOWING IS A SUBROUTINE FOR THE GRADIENT OF THE BROYDEN
C TRIDIAGONAL PROBLEM

        SUBROUTINE UGRAD(N, X, G)
        INTEGER N
CD      DOUBLE PRECISION X(N), G(N)
CS      REAL X(N), G(N)

C LOCAL VARIABLES

        INTEGER I
CD      DOUBLE PRECISION RL, RM, RR, ONE, TWO, THREE, FOUR
CS      REAL RL, RM, RR, ONE, TWO, THREE, FOUR
CD      DATA ONE, TWO, THREE, FOUR/1.0D0, 2.0D0, 3.0D0, 4.0D0/
CS      DATA ONE, TWO, THREE, FOUR/ 1.0E0, 2.0E0, 3.0E0, 4.0E0/

        RL = (THREE - TWO * X(1)) * X(1) - TWO * X(2) + ONE
        RR = (THREE - TWO * X(2)) * X(2) - X(1) - TWO * X(3) + ONE
        G(1) = TWO * (RL * (THREE - FOUR * X(1)) - RR)
        DO 10 I = 2, N-1
           IF(I .NE. 2) THEN
              RL = (THREE - TWO * X(I-1)) * X(I-1) - X(I-2) -
     *             TWO * X(I) + ONE
           ENDIF
```

```
          RM = (THREE - TWO * X(I)) * X(I) - X(I-1) -
     *             TWO * X(I+1) + ONE
          IF(I .EQ. N-1) THEN
             RR = (THREE - TWO * X(N)) * X(N) - X(N-1) + ONE
          ELSE
             RR = (THREE - TWO * X(I+1)) * X(I+1) - X(I) -
     *               TWO * X(I+2) + ONE
          ENDIF
       G(I) = -TWO * (TWO * RL - RM * (THREE - FOUR * X(I)) + RR)
 10    CONTINUE
       G(N) = -TWO * (TWO * RM - RR * (THREE - FOUR * X(N)))
       RETURN
       END

C THE FOLLOWING IS A SUBROUTINE FOR THE HESSIAN OF THE BROYDEN
C TRIDIAGONAL PROBLEM

       SUBROUTINE UHESS(N,X,NZ,LICN,HESS,IRN,ICN)
       INTEGER N, NZ, LICN
       INTEGER IRN(NZ), ICN(LICN)
CD     DOUBLE PRECISION X(N), HESS(LICN)
CS     REAL X(N), HESS(LICN)

C LOCAL VARIABLES

       INTEGER I
CD     DOUBLE PRECISION RL,RM,RR,DRLIM1,DRMI
CD     DOUBLE PRECISION ONE,TWO,THREE,FOUR
CS     REAL RL,RM,RR,DRLIM1,DRMI
CS     REAL ONE,TWO,THREE,FOUR
CD     DATA ONE, TWO, THREE, FOUR/1.0D0, 2.0D0, 3.0D0, 4.0D0/
CS     DATA ONE, TWO, THREE, FOUR/1.0E0, 2.0E0, 3.0E0, 4.0E0/

       NZ = 1
       RL = (THREE - TWO * X(1)) * X(1) - TWO * X(2) + ONE
       HESS(NZ) = TWO * ((THREE - FOUR * X(1))**2 -
     *             FOUR * RL + ONE)
       IRN(NZ) = 1
       ICN(NZ) = 1
       DO 10 I = 2, N-1
          DRLIM1 = THREE - FOUR * X(I-1)
          DRMI = THREE - FOUR * X(I)
          IF(I .NE. 2) THEN
             NZ = NZ + 1
```

```
            HESS(NZ) = FOUR
            IRN(NZ) = I
            ICN(NZ) = I-2
         ENDIF
         NZ = NZ + 1
         HESS(NZ) = -TWO * (TWO * (THREE - FOUR * X(I-1)) +
   *                ONE * (THREE - FOUR * X(I)))
         IRN(NZ) = I
         ICN(NZ) = I-1
         RM = (THREE - TWO * X(I)) * X(I) - X(I-1) -
   *          TWO * X(I+1) + ONE
         NZ = NZ + 1
         HESS(NZ) = -TWO * (-FOUR - (THREE - FOUR * X(I))**2 +
   *                FOUR * RM  - ONE)
         IRN(NZ) = I
         ICN(NZ) = I
10    CONTINUE
      RR = (THREE - TWO * X(N)) * X(N) - X(N-1) + ONE
      NZ = NZ + 1
      HESS(NZ) = FOUR
      IRN(NZ) = N
      ICN(NZ) = N-2
      NZ = NZ + 1
      HESS(NZ) = -TWO * (TWO * (THREE - FOUR * X(N-1)) +
   *             THREE - FOUR * X(N))
      IRN(NZ) = N
      ICN(NZ) = N-1
      NZ = NZ + 1
      HESS(NZ) = TWO * (FOUR + (THREE - FOUR * X(N))**2 - FOUR * RR)
      IRN(NZ) = N
      ICN(NZ) = N
      RETURN
      END
```

Figure 5.1: Code to solve the Broyden tridiagonal problem

If we use the double–precision version of the package to solve the Broyden tridiagonal problem given by FCN, for N = 10000, we obtain the following output:

```
STDRUO      GRADIENT FLAG      = 2
STDRUO      HESSIAN FLAG       = 2
STDRUO      METHOD             = 1
STDRUO      ITERATION LIMIT    =  500
STDRUO      MACHINE EPSILON    =  0.2220446049250E-15
```

```
  STDRUO      STEP TOLERANCE     =  0.3666852862501E-10
  STDRUO      GRADIENT TOLERANCE =  0.1000000000000E-04
  STDRUO      MAXIMUM STEP SIZE  =  0.1000000000000E+06

--------------------------------------------
  STRSLT    ITERATION K   =    0
  STRSLT    FUNCTION AT X(K)
  STRSLT        0.1001100000000E+05
  STRSLT    SCALED GRADIENT AT X(K)
  STRSLT        0.3800000000000E+02
--------------------------------------------

  STCHKS    RELATIVE GRADIENT CLOSE TO ZERO
  STCHKS    CURRENT ITERATE IS PROBABLY SOLUTION

--------------------------------------------
  STRSLT    ITERATION K   =    4
  STRSLT    FUNCTION AT X(K)
  STRSLT        0.1884575867777E-13
  STRSLT    SCALED GRADIENT AT X(K)
  STRSLT        0.1113397081739E-05
--------------------------------------------

  STRSLT    NUMBER OF FUNCTION EVALUATIONS    5
  STRSLT    NUMBER OF GRADIENT EVALUATIONS    5
  STRSLT    NUMBER OF HESSIAN EVALUATIONS    4
```

In the Appendix, we give another example of use–the optimal design with composite materials problem–from the `MINPACK-2` collection [1].

A. Appendix: Another Example of Use: The Optimal Design Problem

In the example given in Figure A.1, we first call the routine STDFLT, which returns the default values. We then override the value of GRADTL and GRDFLG. Next we call either the interface STUMCS or STUMCD for the single– and double–precision version, respectively, to solve the optimal design with composite materials problem (ODC) from the MINPACK-2 collection [1]. Since in the MINPACK-2 collection both the function and the gradient of the ODC problem are coded in the same subroutine DODCFG, we split DODCFG in two subroutines: DODCF and DODCG for the function and gradient evaluations, respectively.

```
C
C STENMIN MINIMIZES AN UNCONSTRAINED NONLINEAR FUNCTION IN N
C UNKNOWNS WHERE THE HESSIAN IS LARGE AND SPARSE, USING TENSOR
C METHODS.
C
C EXAMPLE OF USE FOR STENMIN.  THE TEST PROBLEM IS THE
C OPTIMAL DESIGN WITH COMPOSITE MATERIALS PROBLEM FROM
C THE MINPACK-2 TEST PROBLEM COLLECTION.
C
C  ALI BOUARICHA, OCTOBER 1994.
C  MCS DIVISION, ARGONNE NATIONAL LAB.
C
      INTEGER          NMAX, N, NZ, LIRN, LICN, ILIM, IPR, METHOD
      INTEGER          GRDFLG, HSNFLG, NDIGIT, MSG, LWRK, LIWRK
      INTEGER          TERMCD, INFORM, I, J, K, NX, NY
CD    DOUBLE PRECISION FSCALE, GRADTL, STEPTL, FPLS, STEPMX
CD    DOUBLE PRECISION LAMBDA, HX, HY, TEMP, ONE
CS    REAL             FSCALE, GRADTL, STEPTL, FPLS, STEPMX
CS    REAL             LAMBDA, HX, HY, TEMP, ONE
      PARAMETER      ( NMAX = 10000, LIRN = 50000, LICN = 500000 )
      PARAMETER      ( LIWRK = 2 * LIRN + 12 * NMAX + 2 )
      PARAMETER      ( LWRK = 7 * NMAX )
      INTEGER          IRN ( LIRN  ), ICN ( LICN )
      INTEGER          IWRK( LIWRK )
CD    DOUBLE PRECISION X   ( NMAX ),  TYPX( NMAX ), XPLS( NMAX )
CD    DOUBLE PRECISION GPLS( NMAX ),  HESS( LICN ), WRK ( LWRK )
CD    DOUBLE PRECISION HTV ( NMAX )
CS    REAL             X   ( NMAX ),  TYPX( NMAX ), XPLS( NMAX )
CS    REAL             GPLS( NMAX ),  HESS( LICN ), WRK ( LWRK )
CS    REAL             HTV ( NMAX )
      COMMON / PARAM / NX, NY
      COMMON / OTHER / LAMBDA
      EXTERNAL         DODCF, DODCG, STDUSH
CD    INTRINSIC        DBLE, MIN
```

```
CS    INTRINSIC        FLOAT, MIN
CD    DATA ONE / 1.0D0 /
CS    DATA ONE / 1.0E0 /

C READ DATA

      READ(5,*) NX, NY, LAMBDA
      N = NX * NY

C  COMPUTE THE STANDARD STARTING POINT.

CD    HX = ONE/DBLE(NX+1)
CD    HY = ONE/DBLE(NY+1)
CS    HX = ONE/FLOAT(NX+1)
CS    HY = ONE/FLOAT(NY+1)
      DO 20 J = 1, NY
CD       TEMP = DBLE(MIN(J,NY-J+1))*HY
CS       TEMP = FLOAT(MIN(J,NY-J+1))*HY
         DO 10 I = 1, NX
            K = NX*(J-1) + I
CD          X(K) = -(MIN(DBLE(MIN(I,NX-I+1))*HX,TEMP))**2
CS          X(K) = -(MIN(FLOAT(MIN(I,NX-I+1))*HX,TEMP))**2
   10    CONTINUE
   20 CONTINUE

C  DEFINE THE SPARSITY STRUCTURE OF THE HESSIAN.

      CALL DODCSP(NX,NY,NZ,IRN,ICN)

C SET THE DEFAULT VALUES OF THE PACKAGE.

      CALL STDFLT(N,TYPX,FSCALE,GRADTL,STEPTL,ILIM,STEPMX,
     *            IPR,METHOD,GRDFLG,HSNFLG,NDIGIT,INFORM,MSG)

CD    GRADTL = 1.0D-5
CS    GRADTL = 1.0E-3
      GRDFLG = 2

C  CALL THE SPARSE OPTIMIZER.

CD    CALL STUMCD(N,X,NZ,IRN,LIRN,ICN,LICN,DODCF,DODCG,
CS    CALL STUMCS(N,X,NZ,IRN,LIRN,ICN,LICN,DODCF,DODCG,STDUSH,
     *            TYPX,FSCALE,GRADTL,STEPTL,ILIM,STEPMX,IPR,
     *            METHOD,GRDFLG,HSNFLG,NDIGIT,MSG,XPLS,FPLS,GPLS,
```

```
*             HESS,WRK,LWRK,IWRK,LIWRK,TERMCD,HTV,INFORM)

      STOP
      END
```

Figure A.1: Code to solve the optimal design with composite materials problem

If we use the double–precision version of the package to solve the ODC problem for the following input:

```
NX, NY, LAMBDA :   100 100 0.008,
```

we obtain the following output:

```
  STDRUO      GRADIENT FLAG      = 2
  STDRUO      HESSIAN FLAG       = 0
  STDRUO      METHOD             = 1
  STDRUO      ITERATION LIMIT    =  500
  STDRUO      MACHINE EPSILON    =  0.2220446049250E-15
  STDRUO      STEP TOLERANCE     =  0.3666852862501E-10
  STDRUO      GRADIENT TOLERANCE =  0.1000000000000E-04
  STDRUO      MAXIMUM STEP SIZE  =  0.6521118878154E+04

---------------------------------------------
  STRSLT    ITERATION K   =    0
  STRSLT    FUNCTION AT X(K)
  STRSLT        0.4823420295546E-01
  STRSLT    SCALED GRADIENT AT X(K)
  STRSLT        0.1931183217332E-01
---------------------------------------------

  STCHKS    RELATIVE GRADIENT CLOSE TO ZERO
  STCHKS    CURRENT ITERATE IS PROBABLY SOLUTION

---------------------------------------------
  STRSLT    ITERATION K   =   20
  STRSLT    FUNCTION AT X(K)
  STRSLT       -0.1137724408643E-01
  STRSLT    SCALED GRADIENT AT X(K)
  STRSLT        0.3938142592477E-05
---------------------------------------------

  STRSLT    NUMBER OF FUNCTION EVALUATIONS   67
  STRSLT    NUMBER OF GRADIENT EVALUATIONS   21
  STRSLT    NUMBER OF HESSIAN EVALUATIONS   20
```

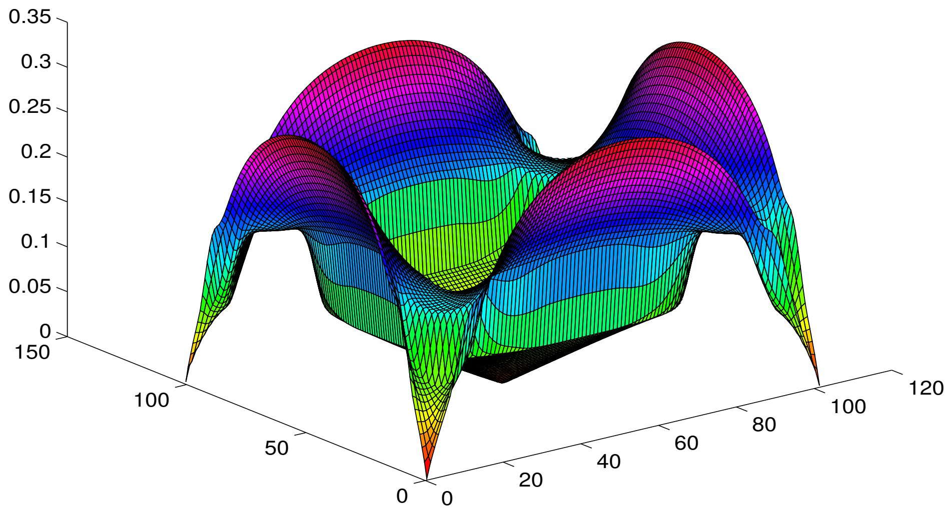

Figure A.2: Norm of $||\nabla v||$ for the stress field v in a design with composite materials

A plot of the norm $||\nabla v||$ of the gradient of the stress field v in the bounded domain $D = (0,1) \times (0,1)$ where `LAMBDA` $= 0.008$ is given in Figure A.2. Figure A.3 shows the contour plot for this surface.

References

[1] B. M. Averick, R. G. Carter, J. J. Moré, and G. L. Xue. The MINPACK-2 test problem collection. Technical Report ANL/MCS-P153-0692, Argonne National Laboratory, 1992.

[2] I. Bongartz, A. R. Conn, N. I. M. Gould, and Ph. L. Toint. CUTE: Constrained and Unconstrained Testing Environment. *ACM Trans. Math. Software*, 21(1):123–160, 1995.

[3] W. J. Cody. MACHAR: A subroutine to dynamically determine machine parameters. *ACM Trans. Math. Softw.*, 14:303–311, 1988.

[4] T. F. Coleman, B. S. Garbow, and J. J. Moré. Fortran subroutines for estimating sparse Hessian matrices. *ACM Trans. Math. Software*, 11:378, 1985.

[5] T. F. Coleman, B. S. Garbow, and J. J. Moré. Software for estimating sparse Hessian matrices. *ACM Trans. Math. Software*, 11:363–377, 1985.

[6] T. F. Coleman and J. J. Moré. Estimation of sparse Hessian matrices and graph coloring problems. *Math. Programming*, 28:243–270, 1984.

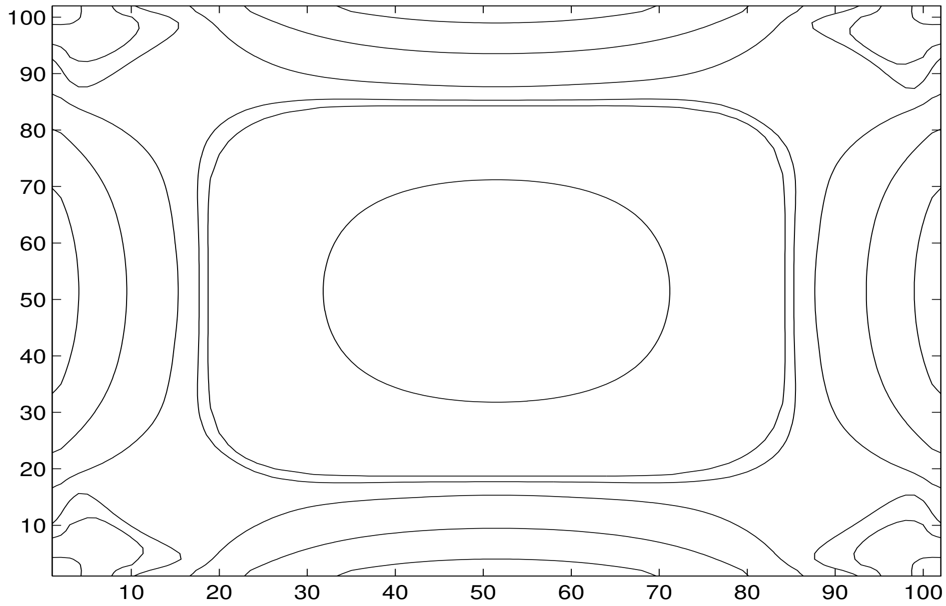

Figure A.3: Contours of $||\nabla v||$ for the stress field v in a design with composite materials

[7] A. R. Conn, N. I. M. Gould, and Ph. L. Toint. An introduction to the structure of large scale nonlinear optimization problems and the LANCELOT project. Report 89-19, Namur University, Namur, Belgium, 1989.

[8] A. R. Conn, N. I. M. Gould, and Ph. L. Toint. LANCELOT. Springer Series in Computational Mathematics. Springer-Verlag, 1992.

[9] I. S. Duff and J. K. Reid. MA27: A set of Fortran subroutines for solving sparse symmetric sets of linear equations. Technical Report R-10533, AERE Harwell Laboratory, Harwell, UK, 1983.

[10] J. J. Moré, B. S. Garbow, and K. E. Hillstrom. Testing unconstrained optimization software. *ACM Trans. Math. Software*, 7(1):17–41, 1981.