AN UNCONSTRAINED NONLINEAR OPTIMIZATION SOLVER: A USER'S GUIDE

1. Introduction

This modular software package has been designed to solve the unconstrained nonlinear optimization problem. This problem entails finding the local minimum of a twice continuously differentiable real-valued function f of n variables from a given start point x^0. Required input to the routine includes the dimension of the problem n, a subroutine to evaluate the function $f(x)$, and an estimate x^0 of the minimum x^*. In addition, the user may control the means for evaluating first and second derivatives of the optimization function and specify various tolerances. Upon completion, the program returns with an approximation $\hat{x}$ to the local minimum x^*, the value of the function $f(\hat{x})$, the value of the gradient $g(\hat{x})$, and a flag specifying under which stopping condition the algorithm was terminated. These algorithms are not intended for problems of dimension $n=1$. The program is inefficient for this case; the user is advised to find more suitable software.

This implementation provides three separate global strategies a user may choose for locating a next iterate x^k. These consist of a line search and two trust region methods known as dogleg and hookstep (see section 3). The user may provide analytic first and second derivatives of the optimization function, or elect to have them computed by finite differences or approximated by secant methods. A wide range of combinations of analytic and/or computed derivatives is allowed (see section 2). The non-linear problem is a difficult one; no software package can guarantee a correct solution for every case, but the alternatives provided here will hopefully demonstrate utility in a greater number of circumstances.

These algorithms correspond closely, but not always exactly, to those in *Numerical Methods for Unconstrained Optimization and Nonlinear Equations* by J. E. Dennis, Jr. and R. B. Schnabel.

2. Computing the Gradient and Hessian

The program provides considerable flexibility in the computation of the gradient and Hessian of the optimization function. The user may supply analytic routines for the gradient and/or the Hessian. If derivatives are not supplied, the package contains an assortment of routines for the computation of these quantities. These routines are invoked automatically in accordance with the user's choice of the parameters IAGFLG, IAHFLG and IEXP which specify whether an analytic gradient has been provided, whether an analytic Hessian has been provided and whether the optimization function is expensive to evaluate respectively (see section 6). Table A summarizes the use of these modules.

Secant methods do not require evaluation of the optimization function when obtaining the Hessian, whereas finite difference techniques do. When the user's optimization function routine FCN is expensive to evaluate, it is usually beneficial to set IEXP=1 in order to minimize the number of function evaluations. Occasionally, it may be less expensive to use secant methods than analytic or finite difference Hessians, even when the function evaluation is inexpensive. The secant update used is the Broyden, Fletcher, Goldfarb and Shanno (BFGS) update.

The default values are:

IAGFLG = 0
IAHFLG = 0
IEXP = 1

TABLE A
MODULES USED TO COMPUTE GRADIENT AND HESSIAN
DEPENDENT ON AVAILABILITY OF ANALYTIC ROUTINES

Availability of			Module used for	
analytic g?	analytic H?	expensive f?	gradient	Hessian
Yes	Yes	---	user g	user H
Yes	No	No	user g	FSTOFD
Yes	No	Yes	user g	SECFAC* SECUNF
No	No	No	FSTOFD	SNDOFD
No	No	Yes	FSTOFD	SECFAC* SECUNF
No	Yes	---	FSTOFD	user H

NOTE: FSTOFD = first order finite difference
SNDOFD = second order finite difference
SECFAC = secant update, factored
SECUNF = secant update, unfactored

* The factored secant update is used with line search and dogleg strategies; the unfactored secant update is used with the hookstep strategy.

3. Choice of Method

Three global strategies have been implemented in this software package. There is a line search and two trust region algorithms, known as the dogleg and hookstep (also known as locally constrained optimal step, or Levenberg-Marquardt step). Unless the user specifies differently, the package will employ the line search technique to locate a new iterate. The relative performance of the alternative methods will vary from problem to problem. Unfortunately, there are no *a priori* means of knowing which method will perform better in a particular situation. Consequently, if the user is solving a class of problems, it may pay to sample each method in turn and to choose the one that works best. The default choice is the line search.

4. Control of Output

The standard (default) output from this package consists of printing the input parameters and the final results. The reported input is as it will be taken by the algorithm and hence includes any corrections made by the program module OPTCHK, which examines the input specifications for illegal entries and consistency (see section 6). The program will provide an error message if it terminates as a result of input errors. The printed results include a termination indicator, an approximation $\hat{x}$ to the minimum, the optimization function f evaluated at $\hat{x}$, and the gradient vector g evaluated at $\hat{x}$.

The package provides two additional means for the control of output. One means is the variable MSG described in section 6. In addition, WRITE (IPR,.) statements have judiciously been scattered throughout the code of the global method subroutines, but concealed on special comment cards that have the characters C@ in columns 1-2. A user with access to an editor which performs string replacement can remove the characters C@ with the concomitant result of compiling these WRITE statements. This will permit the user to obtain a detailed history of the calculations within each iteration.

5. Interfaces

To accommodate both casual and sophisticated users, two interfaces have been provided with the system. OPTIFO requires the user to provide only the dimension n of the problem, a subroutine to evaluate the optimization function f and a starting vector x^0. Certain storage arrays must also be declared (see A, WRK in section 6). OPTIF9 requires the user to supply all parameters. The user may specify selected parameters only by first invoking the subroutine DFAULT which sets all parameters to their default values, and then overriding the desired values. Two examples of calling sequences are given below.

```
i)   CALL OPTIFO (NR,N,X,FCN,XPLS,FPLS,GPLS,ITRMCD,A,WRK)
ii)  CALL DFAULT (N,X,TYPX,TYPF,METHOD,IEXP,MSG,NDIGIT,ITNLIM,
         IAGFLG,IAHFLG,IPR,DLT,GRADTL,STEPMX,STEPTL)
```

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

      GRADTL = 1.0 E-9
      STEPMX = 1.0 E-9
      METHOD = 2
C
   CALL OPTIF9 (NR,N,X,FCN,D1FN,D2FN,TYPX,TYPF,METHOD,IEXP,MSG,
        NDIGIT,ITNLIM,IAGFLG,IAHFLG,IPR,DLT,GRADTL,STEPMX,
        STEPTL,XPLS,FPLS,GPLS,ITRMCD,A,WRK)
```

6. Parameters

The parameters employed with the calling sequences of section 5 are fully described here. OPTIFO uses only those parameters which are preceded by an asterisk. When it is noted that module DFAULT returns a given value, this is the default employed by interface OPTIFO. The user may override the default value by utilizing interface OPTIF9.

Following each variable name in the list below appears a one- or two-headed arrow symbol of forms -->, <--, and <-->. These symbols signify that the variable is for input, output, and input-output respectively.

* NR -->	A positive integer specifying the row dimension of the matrices A and WRK in the user's calling program. NR must satisfy the relation NR $\geq$ N. (see N). The provision of this variable allows the user the flexibility of solving several problems of different order N one after the other.
* N -->	A positive integer specifying the order or dimension of the problem. The program will abort if N $\leq$ 0. The program is inefficient for the one-dimensional case (N=1); the user is advised to find more efficient software. The package will abort for N=1 unless the parameter MSG is appropriately set (see MSG); in this instance the interface OPTIF9 must be utilized.
* X(N)<-->	An N-dimensional array which contains an initial estimate of the minimum x^*. On return X will contain x^{k-1}, the next-to-last iterate.
* FCN ---	The name of a user supplied subroutine that evaluates the optimization function at an arbitrary vector X. The subroutine must be declared EXTERNAL in the user's calling program and must conform to the usage: CALL FCN (N,X,F) where X is a vector of length N. The subroutine must not alter the values of X or N. On return F is the value of the optimization function at X.

D1FN -- The name of a user supplied subroutine that evaluates the first derivative (gradient) of the optimization function. The subroutine must be declared EXTERNAL in the user's program and must conform to the usage

```
CALL D1FN (N,X,G)
```

where X and G are vectors of length N. The subroutine must not alter the values of X or N. On return G is the value of the gradient at X. When using the interface OPTIF9, if no analytic gradient routine is supplied (IAGFLG=0), the user must use the dummy name D1FCN.

The program will automatically check the analytic derivative against a numerical derivative by comparing the user's gradient with a finite difference estimate to within a relative tolerance computed as follows:

relative noise = $\max\{10^{-\mathrm{NDIGIT}}, \text{machine } \varepsilon\}$

relative tolerance = $\max\{10^{-2}, \sqrt{\text{relative noise}}\}$

where NDIGIT is the number of good digits in the optimization function and machine ε is the smallest $\varepsilon>0$ such that $1+\varepsilon>1$ on the machine. To override this feature the parameter MSG must be appropriately set (see MSG).

D2FN -- The name of a user supplied subroutine that evaluates the second derivative (Hessian) of the optimization function. The subroutine must be declared EXTERNAL in the user's calling program and must conform to the usage

```
CALL D2FN (NR,N,X,H)
```

where X is a vector of length N and H is an N x N matrix. The subroutine must not alter the values of X or N. On return H is the value of the Hessian at X. When using the interface OPTIF9, if no analytic Hessian routine is supplied (IAHFLG=0), the user must use the dummy name D2FCN.

The program will automatically check the analytic Hessian against a numerical Hessian by comparing the user's Hessian with a finite difference estimate to within a relative tolerance computed as follows:

relative noise = $\max\{10^{-\mathrm{NDIGIT}}, \text{machine } \varepsilon\}$

relative tolerance = $\max\{10^{-2}, \sqrt{\text{relative noise}}\}$

where NDIGIT is the number of good digits in the optimization function. To take advantage of this feature the user Hessian routine should fill only the lower triangular elements of the matrix H. To override this feature the parameter MSG must be appropriately set (see MSG).

TYPX(N) <--> An n-dimensional array 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. is specified, 1.0 will be used. The program will not abort.

This vector is used by the package to determine a scaling matrix. 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 magnitudes 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 DFAULT returns TYPX = (1.,...,1.). For example, if it is anticipated that the range of values for the iterates x^k would be

$x_1 \ \varepsilon \ [-10^{10}, 10^{10}]$

$x_2 \ \varepsilon \ [-10^2, 10^4]$

$x_3 \ \varepsilon \ [-6 \times 10^{-6}, 9 \times 10^{-6}]$

then an appropriate choice would be TYPX = (1.E10, 1.E3, 7.E-6).

TYPF <--> A positive scalar estimating the magnitude of the optimization function near the minimum x^*. If too large a value is provided for TYPF, the program may terminate prematurely. In particular, if $f(x^0)$ is $>>$ $f(x^*)$, TYPF should be approximately $f(x^*)$, not $f(x^0)$.

If a negative value is specified its absolute value will be used. When 0. is specified, 1.0 will be used. The program will not abort. The module DFAULT returns the value 1.0.

For example, if $f(x^*) \sim 10^4$, then an appropriate choice would be TYPF = 1.E4.

METHOD<--> An integer flag designating which global strategy to use as follows:

= 1 Line search algorithm
= 2 Dogleg trust region algorithm
= 3 Hookstep trust region algorithm

Module DFAULT returns value of 1. If the user specifies an illegal value, module OPTCHK will set METHOD = 1; program will not abort. For further information, see section 3.

IEXP <--> An integer flag specifying whether or not the optimization function subroutine is expensive to evaluate as follows:

= 0 function evaluation is not expensive
= 1 function evaluation is expensive

When IEXP = 1 and IAHFLG = 0, secant methods will be employed to obtain the second derivative (Hessian) of the optimization function. When IEXP = 0 and IAHFLG = 0, finite difference methods will be used to obtain the Hessian. When IAHFLG = 1, IEXP is ignored.

Module DFAULT returns value of 1. If the user specifies an illegal value, module OPTCHK will set IEXP = 1. The program will not abort. For more information, see section 2.

MSG <--> An integer variable which the user may set on input to inhibit certain automatic checks or override certain default characteristics of the package. There are currently five "message" features which can be used individually or in combination.

= 0 No message.

= 1 Do not abort package for N=1.

= 2 Do not check user analytic gradient routine D1FN against
its finite difference estimate. This may be necessary if the user knows his gradient function is properly coded, but the program aborts because the comparative tolerance is too tight. It is also efficient if the gradient has previously been checked. Do not use MSG=2 if the analytic gradient is *not* supplied.

= 4 Do not check user analytic Hessian routine D2FN against its finite difference estimate. This may be necessary if the user knows his Hessian function is properly coded, but the program aborts because the comparative tolerance is too tight. It is also efficient if the Hessian has previously been checked. Do not use MSG=4 if the analytic Hessian is *not* supplied.

= 8 Suppress printing of the input state, the final results, and the stopping condition.

= 16 Print intermediate results.

The user may specify a combination of features by setting MSG to the sum of the individual components. As an example, suppose the user wishes to override automatic comparison of his analytic Hessian routine and to suppress the automatic outputting of all results. The user would set MSG=4+8=12.

The module DFAULT returns a value of 0. If the user specifies an illegal value, its value MOD 32 will be used.

On output, if the program has terminated because of erroneous input (ITRMCD=0), MSG contains an error code indicating the reason.

= 0 No error. (See ITRMCD for termination code)

= -1 Illegal dimension, $N \le 0$.

= -2 Attempt to run program for N=1. (See N, MSG above)

= -3 Illegal tolerance on gradient, GRADTL<0.

= -4 Iteration limit ITNLIM≤0.

= -5 No good digits in optimization function, NDIGIT=0.

= -6 Program asked to override check of analytic gradient against finite difference estimate, but routine D1FN not supplied. (Incompatible input: MSG=0 mod 2 and IAGFLG=0)

= -7 Program asked to override check of analytic Hessian against finite difference estimate, but routine D2FN not supplied. (Incompatible input: MSG=0 mod 4 and IAGFLG=0)

= -21 Probable coding error in the user's analytic gradient routine D1FN. Analytic and finite difference gradients do not agree within the assigned tolerance. (See computation of tolerance under D1FN)

= -22 Probable coding error in the user's analytic Hessian routine D2FN. Analytic and finite difference Hessians do not agree within the assigned tolerance. (See computation of tolerance under D2FN).

NDIGIT <-> The integer number of reliable digits returned by the optimization function FCN. This parameter is used in calculating finite difference stepsizes and checking analytic derivatives. If the function routine FCN provides the full number of significant digits obtainable on the host computer, enter -1 (or any negative integer).

For example, if the optimization function FCN is the result of an iterative procedure (e.g., quadrature, partial differential equations) which is expected to provide 5 good digits in the answer, the user should set NDIGIT=5. If FCN does not invoke an iterative procedure, NDIGIT=-1 is usually appropriate.

The module DFAULT returns the value -1. If the user sets NDIGIT=0, that is the minimization function has no good digits, the program aborts.

Specification of this parameter provides the software with an estimate of the relative error (noise) in the evaluation of the function, computed as $\eta = \max\{10^{-\mathrm{NDIGIT}}, \text{machine } \varepsilon\}$.

ITNLIM <-> Positive integer specifying the maximum iterations to be performed before the program is terminated. Module DFAULT returns the value 150. If the user specifies ITNLIM≤0, the program will abort.

IAGFLG <-> Integer flag designating whether or not an analytic gradient function D1FN has been supplied by the user.

= 0 No user analytic gradient supplied

= 1 User analytic gradient supplied

When IAGFLG=0, the gradient vector is obtained by a first order finite difference method.

The module DFAULT returns the value 0. If the user specifies an illegal value, the module OPTCHK will supply the value 0. If the user sets IAGFLG=1 but fails to provide the subroutine D1FN the program will abort in either the loading or execution phase, depending on the operating system.

IAHFLG <-> Integer flag designating whether or not an analytic Hessian function D2FN has been supplied by the user.

= 0 No user analytic Hessian supplied

= 1 User analytic Hessian supplied

When IAHFLG=0, the Hessian matrix is obtained by a finite difference method if IEXP=0 or by a secant update method if IEXP=1.

The module DFAULT returns the value 0. If the user specifies an illegal value, the module OPTCHK will supply the value 0. If the user sets IAHFLG=1 but fails to provide the subroutine D2FN, the program will abort either in the loading or execution phase, depending on the operating system.

IPR --> The unit on which the routine outputs information. DFAULT returns the value 6 which is the standard FORTRAN unit for the printer.

DLT <--> Positive scalar giving the initial trust region radius. When using the line search global strategy this parameter is ignored. For trust region algorithms, if it is supplied, its value should reflect what the user considers a maximum reasonable scaled step length at the first iteration. If not supplied (DLT = -1.), the routine uses the length of the scaled gradient instead.

The module DFAULT returns the value -1. If the user specifies a negative value, module OPTCHK sets DLT = -1.

GRADTL --> Positive scalar giving the tolerance at which the scaled gradient 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

$$\max_i \left\{ \frac{|\nabla f(x)_i| \max\{|x_i|, typx_i\}}{\max\{|f|, typf\}} \right\} \le gradtl \ . \qquad (1)$$

The module DFAULT returns the value 1.0E-5. If the user specifies a negative value the program aborts.

STEPMX --> A positive scalar providing the maximum allowable scaled step length. STEPMX is used to prevent steps which would cause the optimization function 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 the first two of these occurences 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.

Module DFAULT returns the value

STEPMX = $\max\{||x||_2 * 10^3, 10^3\}$

where X is the user provided initial guess of the minimum.

STEPTL --> A positive scalar providing the minimum allowable relative step length. STEPTL should be set to 10^{-p} where p is the number of digits of agreement between successive iterates of X which the user considers as a satisfactory test of convergence. The actual test used is

$$\max_i \left\{ \frac{|x_i^k - x_i^{k-1}|}{\max\{|x_i^k|, typx_i\}} \right\} \le steptl \ . \qquad (2)$$

Values of p between 3 and 7 are common when using machines such as double precision arithmetic (or single precision large word Control Data or CRAY). Values between 3 and 5 are usual when using single precision.

Module DFAULT returns the value 1.0E-5.

* XPLS(N) <-- An N-dimensional array containing the best approximation to the local minimum upon return (if the algorithm has converged).

* FPLS <-- A real scalar containing the function value at the approximate minimum (if the algorithm has converged).

* GPLS(N)<-- An N-dimensional array containing the gradient of the optimization function at the approximate minimum (if the algorithm has converged).

* ITRMCD <-- An integer which specifies the reason the algorithm was terminated.

= 0 Erroneous input detected. (see MSG)

= 1 Relative gradient is close to zero (see equation 1).
Current iterate is probably solution.

= 2 Successive iterates within tolerance (see equation 2).
Current iterate is probably solution.

= 3 Last global step failed to locate a point lower than
XPLS. Either XPLS is an approximate local minimum of the function, the function is too non-linear for this algorithm, or STEPTL is too large.

= 4 Iteration limit exceeded.

= 5 Maximum step size STEPMX exceeded five consecutive
times. Either the function is unbounded below, becomes asymtotic to a finite value from above in some direction, or STEPMX is too small.

* A(NR,NC) --> A real matrix array which is used to store the Hessian and its Cholesky decomposition. The user must declare this array of dimensions NRxNC in the calling program where both NR ≥ N and NC ≥ N. Only the upper left NxN sub-matrix will be used by this package. (Since stopping criteria are checked before the Hessian or its estimate is updated, the Hessian at XPLS is not available upon return.)

* WRK(NR,i) --> Workspace required by the program:

i = 9 when using interface OPTIF0
i = 8 when using interface OPTIF9

The user must declare this array in the calling routine. The workspace is broken into i NR-vectors by the interface modules, where NR ≥ N. Note that i may be greater than the integer specified here. The flexibility in the dimensions here allows the user to utilize this space to the user’s advantage in the user’s driving program.

7. Summary of Default Values

The following list summarizes the default parameter values returned by module DFAULT. When the interface routine OPTIF0 is employed, these default values are used by the program

Parameter	
TYPX	(1.,...,1.)
TYPF	1.0
METHOD	1
IEXP	0
MSG	0
NDIGIT	-1
ITNLIM	150
IAGFLG	0
IAHFLG	0
IPR	6
DLT	
GRADTL	1.0E-5
STEPMX	max$\|\|x^0\|\|_2 * 10^3, 10^3$ where x^0 is user's estimate of x^*
STEPTL	1.0E-5

8. Implementation Details

This program package has been coded in strict accordance to the 1966 ANSI FORTRAN standards; hence, it should port with no great difficulty to any machine of sufficient core capacity. The program was developed and tested on a Control Data 7600 at the National Center for Atmospheric Research in Boulder, Colorado.

The program package is self-contained except for inline function calls to SQRT and ALOG10 which, since they are described by the FORTRAN standards, should be resident on each host system.

There are a few machine dependencies. The printer is assumed to be FORTRAN unit 6. This value is set in module DFAULT and may easily be changed if the FORTRAN output unit does not conform to standard practice. WRITE statements for real quantities use an E20.13 format specification that is quite suitable for CDC 7600 single precision real values which contain 14 or 15 significant decimal digits. It is rather unappropriate for 32 or 36 bit single precision words. An installation might find it convenient to modify the code to render it a double precision implementation.

Finally, an installation may find it convenient to provide additional interfaces to its user community. These may be easily constructed in three steps, modeled on module OPTIF0: (1)

initiate a call to the module DFAULT; (2) override appropriate default parameter values; (3) initiate a call to module OPTDRV. For instance, if an installation finds that it is common practice for its users to supply an analytic gradient routine, to control scaling, and to use the dogleg strategy, an interface may be provided with the following framework:

```
      SUBROUTINE OPTIF1 (NR,N,X,FCN,D1FN,TYPX,XPLS,
            FPLS,GPLS,ITRMCD,A,WRK)
  C
  C   NOTE:  THE DUMMY ARRAY WRK(1,1) IS SUPPLIED
  C   IN PLACE OF 'TYPX' SO THAT USER'S SUPPLIED
  C   VALUE OF 'TYPX' IS NOT OVERWRITTEN
  C
      CALL DFAULT (N,X,WRK(1,1),...)
      IAGFLG=1
      METHOD=2
  C
      CALL OPTDRV.(...)
```

The provision of alternative interfaces may be an expedient labor saving device.

9. References:

More, J.J., B.S. Garbow, and K.E. Hillstrom (1981), "Testing unconstrained optimization software," *A.C.M. Transactions on Mathematical Software* 7, pp. 17-41.

Dennis, J.E. Jr., and R.B. Schnabel (1983), *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice-Hall, Englewood Cliffs, N.J.