8/26/83

Dr. [redacted]

1709

I-10918

SANDIA REPORT SAND83-0936 • Unlimited Release

Printed July 1983

SAND--83-0936

DE83 016773

A New Algorithm for Constrained Nonlinear Least-Squares Problems

Part I

Richard J. Hanson, Fred T. Krogh

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550
for the United States Department of Energy
under Contract DE-AC04-76DP00789

MASTER

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITE

SF 2900-Q(6-82)

SAND 83-0936
Unlimited Release
Printed July, 1983

A NEW ALGORITHM FOR CONSTRAINED NONLINEAR LEAST-SQUARES PROBLEMS PART I.

R. J. Hanson
Sandia National Laboratories

and

F. T. Krogh†
Jet Propulsion Lab.
4800 Oak Grove Dr.
Pasadena, CA 91109

ABSTRACT

A new Gauss-Newton algorithm is presented for solving nonlinear least squares problems. The problem statement may include simple bounds or more general constraints on the unknowns. The algorithm uses a trust region that allows the objective function to increase with logic for retreating to best values. The computations for the linear problem are done using a least squares system solver that allows for simple bounds and linear constraints. The trust region limits are defined by a box around the current point. In its current form the algorithm is effective only for problems with small residuals, linear constraints and dense Jacobian matrices. Results on a set of test problems are encouraging.

† The research of this author was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.

CONTENTS

TABLES

Introduction

The constrained nonlinear least squares problem which we consider is stated as follows.

Problem CNLLS

(1) Let $\underset{\sim}{f}(\underset{\sim}{x})$ be a differentiable mapping of a rectangular domain $T = \{\underset{\sim}{x}: \alpha_j \le x_j \le \beta_j,\ j = 1,\ldots,N\}$ into R^M, and $\underset{\sim}{g}(\underset{\sim}{x})$ be a differentiable mapping of T into R^L. Find an $\hat{\underset{\sim}{x}} \in T$, $\alpha_i' \le g_i(\hat{\underset{\sim}{x}}) \le \beta_i'$, $i=1,\ldots,L$, such that $\|\underset{\sim}{f}(\underset{\sim}{x})\|^2 = \sum_{i=1}^{M} f_i^2(\underset{\sim}{x})$ is minimized at $\underset{\sim}{x} = \hat{\underset{\sim}{x}}$.

In the statement of problem CNLLS we emphasize that the functions $\underset{\sim}{f}(\underset{\sim}{x})$ and $\underset{\sim}{g}(\underset{\sim}{x})$ are defined and differentiable for every $\underset{\sim}{x} \in T$. Problems for which $\underset{\sim}{f}(\underset{\sim}{x})$ is defined only when $\underset{\sim}{x} \in T$ <u>and</u> $\underset{\sim}{\alpha}' \le \underset{\sim}{g}(\underset{\sim}{x}) \le \underset{\sim}{\beta}'$ are not included. The solution $\underset{\sim}{x}$ will satisfy the bounds on $\underset{\sim}{g}(\underset{\sim}{x})$, if possible, but $\underset{\sim}{g}(\underset{\sim}{x})$ may be evaluated at an arbitrary point $\underset{\sim}{x} \in T$.

The following notation is used,

	$\underset{\sim}{x}_p$	$\underset{\sim}{x}$ on the p-th iteration
	f_p	$\underset{\sim}{f}(\underset{\sim}{x})$ on the p-th iteration
	g_p	$\underset{\sim}{g}(\underset{\sim}{x})$ on the p-th iteration
(2)	G_p	$\partial\underset{\sim}{g}/\partial\underset{\sim}{x}$ on the p-th iteration
	J_p	$\partial\underset{\sim}{f}/\partial\underset{\sim}{x}$ on the p-th iteration
	Δx_p	$\underset{\sim}{x}_p - \underset{\sim}{x}_{p+1}$

The basic iteration, known as the (constrained) Gauss-Newton method, is given by the sequence of constrained linear least squares problems

$$
(3) \quad
\begin{array}{c}
\underset{\sim}{x}_0 \in T \text{ is given} \\
J_p \Delta x_p \cong f_p \\
\text{subject to the linear constraints} \\
g_p - \underset{\sim}{\beta}' \leq G_p \Delta x_p \leq g_p - \underset{\sim}{\alpha}' \\
\text{and the simple bounds} \\
\underset{\sim}{x}_p \in T, \\
p = 0, 1, 2, \ldots
\end{array}
$$

Probably the most challenging aspect of developing an algorithm, based on (3), is in making the method converge with good efficiency. (Our notion of efficiency is based on the usual one, the number of function and derivative evaluations, f_p, J_p, g_p, and G_p. The cost of the numerical linear algebra to solve the constrained linear least squares problem of (3) is assumed to be less important than the cost of evaluating these functions and derivatives.) It is frequently true that the iteration (3) will fail to converge if $\|\Delta x_p\|$ is not restricted in some way. Too tight a restriction, however, will decrease the efficiency.

We describe an algorithm for bounding $\|\Delta x_p\|$ which (based on performance) gives better efficiency, on the average, than the MINPACK algorithm, [1]. Our work has not yet resulted in a finished software product. What we are reporting here is the fact that our basic methods look promising.

It must be emphasized that even when $\underset{\sim}{x}_p$ is close to a solution, the iteration (3) may be unacceptably slow or may not converge. Cases A-D below must be separately considered. They can, of course, appear in any combination. We feel that our algorithm currently performs reasonably in Case A.

A. The matrix J_p is full rank and $\|f_p\| \to c \doteq 0$. Both $J_p^T f_p$ and Δx_p converge quadratically to zero. (A simple model is $f(x) = \sin x$, near $x=0$.)

B. The matrix J_p is of full rank and $\|f_p\| \to c >> 0$. Iteration (3) gives only linear convergence to the solution. (A simple model is $\underset{\sim}{f}^T(x) = (1+\sin x,\ 2-e^x)$, near $x=0$.)

C. At the solution J_p is not of full rank, but this does not limit the amount $\|\underset{\sim}{f}(\underset{\sim}{x})\|$ can be reduced. Iteration (3) converges linearly to the solution. (A simple model is $f(x) = x \sin x$, near $x=0$.)

D. At the solution J_p is not of full rank and this limits the amount $\|\underset{\sim}{f}(\underset{\sim}{x})\|$ can be reduced. Iteration (3) does not converge. (A simple model is $f(x) = 1 + \sin^4 x$, near $x=0$.)

We anticipate that in Part II of this report (not yet written) modifications of the iteration (3) will be presented. These modifications will involve using (3) where it is effective and higher order terms where it is not. We believe that our modified algorithm will be efficient for all the cases except for case C when case D is also present. We believe this combination is a rare event in the computations that will apply our methods.

Our current algorithm is effective only on problems where G_p is a constant matrix. We expect to modify the algorithm for nonlinear constraints in Part II of this report.

Section 2 gives a broad outline of the algorithm and contrasts it to the Levenberg-Marquardt method used, for example in MINPACK [1] and the package NL2SOL [2]. Section 3 gives the details of our algorithm and the rationale for various aspects of its design. Section 4 summarizes our results on the test problems and contains some discussion of the test set itself.

2. Outline of the Algorithm

In this section we summarize our approach for solving the nonlinear least squares problem of (1). The key to efficiency is in accepting steps Δx_p for which $\|f_{p+1}\| > \|f_p\|$. Large moves that increase $\|\underset{\sim}{f}(\underset{\sim}{x})\|$ are frequently good moves in the sense that they get closer to the solution. In turn, this requires logic so that the algorithm can retreat to its previous best values if it senses that making these moves got it into trouble. We provide logic for sensing trouble and for making the retreat.

The algorithm begins with a user-given starting point, $\underset{\sim}{x}_0$. An N-dimensional box, B, is constructed that contains $\underset{\sim}{x}_p$. The lengths of each dimension of B are chosen so that B remains in T and $\|\Delta x_p\|$ remains "reasonable." We will elaborate on this below and in the next section.

There are five cases for altering these dimensions that we will discuss in detail in Section 3. These cases are

1. Selecting the dimensions of B initially.

2. Selecting the dimensions after successive best values of $\|\underset{\sim}{f}(\underset{\sim}{x})\|$ are obtained.

3. Selecting the dimensions after finding a best value of $\|\underset{\sim}{f}(\underset{\sim}{x})\|$ after a value that is not a best.

4. Selecting the dimensions after giving up and retreating to a best value for $\|f(x)\|$.

5. Selecting the dimensions after obtaining a value for $\|f(x)\|$ larger than the best obtained previously.

The Levenberg-Marquardt method for solving nonlinear least squares problems (with no constraints) can be regarded as placing an ellipsoidal trust region about each point x_p. The MINPACK implementation [1] of this method chooses the relative sizes of the axes of this ellipsoid based on the column norms for J_p. This type of scaling is appropriate when the error in each variable contributes equally to the residual, a condition that is frequently not satisified.

Our algorithm uses adaptive scaling that depends on changes in the individual components of x. Namely, the relative weights used to compute the future moves is approximated by the size of moves made in the past. We tried column scaling using reciprocals of column norms at successive best x values. Based on overall performance, we prefer the adaptive scaling outlined above. A Levenberg-Marquardt algorithm that uses our approach for column scaling could be developed.

Briefly stated, our decision to retreat to a best value for $\|f(x)\|$ is made in one of two ways:

1. If $\|J_p\Delta x_p-f_p\|\geq\frac{1}{2}[\|f(x_{BEST})\| + \|J_{BEST}\Delta x_{BEST}-f_{BEST}\|]$, $\Delta x_p \varepsilon B$, we retreat. If this condition is not satisfied, the linear approximation is telling us that we still have reasonable expectation for improvement, despite the fact that $\|f(x)\|$ is large.

2. If $\|f(x)\|$ gets too large after a move from a best x value, the initial move was probably too large. We retreat if $\|f_p\|>\bar{f}$, where $\bar{f}=2\|f_p\|$ at the first step with $\|f_p\|\geq\|f_{p-1}\|$. This allows an arbitrarily large increase on the first step; after that $\|f(x)\|$ is bounded. The final version of our algorithm is not expected to require this condition.

The characteristics that we consider vital here are: An arbitrarily large increase in $\|f(x)\|$ is tolerated; there is no ad hoc limit on how many iterations will be taken before retreating; and retreating immediately if the residual to the current linear problem does not hold the prospect of an eventual new best.

3. The nonlinear least squares algorithm using simple bounds to define the trust region.

Our trust region amounts to a choice for the dimensions of the box B that we place around $\underset{\sim}{x}_p$. The dimensions of the box are varied, roughly speaking, so that when the function decreases the dimensions increase. Otherwise the dimensions tend to decrease.

We introduce the following list of variables that are used to describe the details of the algorithm.

$\underset{\sim}{x}$ the vector of independent variables at Step p, i.e., $\underset{\sim}{x}_p$.

x_i component i of x, i=1,...,N

δ_i change on the current iteration, i.e., component i of Δx_p.

f function whose Euclidean length (least squares norm) is being minimized.

$\underset{\sim}{x}_B$ value of $\underset{\sim}{x}$ from among all the previous $\underset{\sim}{x}_p$ that gave the minimum of $\|\underset{\sim}{f}(\underset{\sim}{x})\|$; i.e., the "best" $\underset{\sim}{x}$.

f_B smallest value of $\|\underset{\sim}{f}(\underset{\sim}{x})\|$ so far, $f_B=\|f(\underset{\sim}{x}_B)\|$

f_L value of $\|\underset{\sim}{f}(\underset{\sim}{x}_{p-1})\|$, last value for $\|\underset{\sim}{f}(\underset{\sim}{x})\|$

f_C value of $\|\underset{\sim}{f}(\underset{\sim}{x}_p)\|$, current value for $\|\underset{\sim}{f}(\underset{\sim}{x})\|$

b_i bound used in computing δ_i.

B_i bound used in computing δ_i after a new best

k count of iterations since a new best. (Set to 0 to flag initialization.)

KASE if KASE>0 got a new best after KASE iterations. (Set to 0 for initialization; set to -1 when we retreat to best $\underset{\sim}{x}$.)

c_i length of column i of the Jacobian matrix J_p.

α a scalar used in computing b_i

P predicted value of f_C based on solving the constrained linear problem

boost, a boost factor for increasing the bounds B_i on successive best $\underset{\sim}{x}$'s.

Using this notation, we now describe our algorithm. The language we use here is close to SFTRAN3, [3]. We first describe the algorithm and follow this by some details about why we chose the specific factors to vary the trust region. The reader is warned that some details of the algorithm given here are obsolete even at this writing. The results shown in Tables 1-4 do correspond to the following algorithm, however.

<u>H-K-1 Algorithm for Nonlinear Least Squares</u>

```
        δi = 0, i=1,...,N
        k  = 0
        fB = ∞
        P  = 0
        retreat   = .false.
        terminate = .false.

        DO FOREVER
           IF (retreat) THEN

C       We must retreat back to best x.

             k      = 0
             KASE   = -1
             fL   = fB
             x      = xB

           Else
             KASE = k
             xi = xi - δi, i=1,...,N

             If (terminate) Exit from Algorithm

           End if
           Compute J, f, G, and g at x.

           fC = ||f(x)||
           Test for convergence.

           If (terminate) Exit from Algorithm

           If (fC < fB) k = 0

           If (k = 0) then

C       We want to position at best x values.

             fB = fC
```

```
      DO CASE (2-KASE, 3)

      Case 1

C     We immediately got a new best x~.

C     The following formula for α has the following effect.
C     If f_C ≤ P, then α = 1 + 1/ALFAC.

C     If f_C^2 = P * f_L, then α = 1.

C     If f_C^2 >> P * f_L, then α = 0.5.
           α = max(f_C^2 - P^2, 0)

                 α + P * (ALFAC + 1) * (f_L - P)
           α = -----------------------------------
                 2α + P * ALFAC * (f_L - P)

           ALFAC = 0.125 * (1/α + ALFAC)

           boost = min((α+1)*boost, 10^10)

      Case 2

C     We are at the initial x~.

           ALFAC = 1/128

           Do For i = 1,...,N
                B_i = -x_i
                If (B_i = 0) then
                     If (c_i ≠ 0) then
                          B_i = -f_C/c_i
                     Else
                          B_i = -1
                     End If
                End If
                b_i = B_i
           End For
           x~_B = x~
           α = 1
           boost = 0.5
           Exit from the "If (k = 0)" block.
           Case 3

C     We are retreating to best x~.
```

```
                    α = 0.125
                    boost = 0.25

                Case Other

C           Not immediately a best x.
                    α = 0.25
                    boost = 1

                End Case

                Do For i = 1,...,N

                    δi = (xB)i - xi

                    If (δi = 0) then
                        bi = α*Bi
                    Else
                        bi = α*SIGN(δi+Bi,δi) + boost*δi
                        (xB)i = xi
                    End If
                    Bi = bi
                End For
                Else

C           The nonlinear residual norm is not a new best
                    ALFAC = 0.25

                    If(k = 1) then
                        f = 2*fC
                        P̂ = (fB+P)/2
                    Else If (fC>f) then

C           The function norm is too big.

                        retreat = .true.

                        cycle on "Do Forever" loop

                    End if

                End if

                α = (0.25*fC + 0.5*fL)/(fC + fL)

                Do For i = 1,...,N
                    If (δi = 0) then
                        bi = α*bi
```

```
               Else

                     b_i = α*(SIGN(0.25*δ_i + b_i, δ_i) + 0.75*δ_i)
               End If

          End For

     End If

     k = k+1

C    Compute the box, with each dimension defined by
C    (ℓ_i,u_i) that will be the trust region for the linear
     problem.

     Do For i = 1,...,N

          If (b_i<0) then
               ℓ_i = b_i
               u_i = -ℓ_i*(8/9)

          Else

               u_i = b_i
               ℓ_i = u_i*(8/9)

          End If

C    Restrict the trust region bounds further if the user has
C    given bounds on the variables.

          ℓ_i = max (ℓ_i, x_i - β_i)
          u_i = min (u_i, x_i - α_i)

     End For

C    If the user has given general constraints, compute the

C    bounds (ℓ'_j, u'_j) for the constraint equations.

     Do For j = 1,...,L

          ℓ'_j= g_j(x) - β'_j

          u'_j = g_j(x) - α'_j
     End For

C    Solve linear least squares problem [4] with bounds and
C    generals constraints to get δ_j, j = 1,...,N, and the
C    linear residual, vector length, P.
```

```
If (P ≥ P̂) then
   retreat = .true.
End if
If (.not. retreat) then
    If (‖Δx_p‖ is small and certain other
       conditions are satisfied) then
       terminate = .true.
    End if
End if
f_L = f_C

C         Reduce bounds if the size of the move was well below what
C         was allowed

σ = 0
Do For j = 1,...,N
     t = δ_j/b_j
     If (t > 0) then

          σ = max (σ,t)

     Else

          σ = max (σ,-(9/8)*t)

     End If

End For

If (σ ≤ 0.5) then

     σ = 2*σ

     Do For j = 1,...,N

          b_j = σ *b_j

          If (k = 1) B_j = b_j

     End For

     boost = 1

End If

End Forever
```

We close this section with some discussion of the reasons for our choice of the dimensions of the box, B. These are cases 1-5 given in Section 2. The algorithm tends to select slightly smaller bounds for those components which are changing direction. The direction is set at the initial point as the sign of the value. For example, a variable with the value 1 can move to points in the interval [1/9, 2].

1. **Selecting the Dimensions of B Initially**

If a variable is nonzero, we assume that its value is in the approximate scale of a solution for this parameter. The user should avoid nonzero initial guesses if this is not the case. The dimension of the box, in this coordinate, is 17/9 the magnitude of the variable. If the variable is zero and $c_i \neq 0$, we use an approximation to a single variable Gauss-Newton step. This approximation assumes that the residual could be reduced to zero with that component. This yields $(17/9)\|f_C\|/c_i$ as the dimension of B. If $c_i = 0$, we use the value of (17/9) for the dimension in this coordinate. These bounds tend to be large, especially if one were to insist on a reduction in $\|\tilde{f}(\tilde{x})\|$.

We have not done much work on this choice of initial bounds. Perhaps the choice should be symmetric in a relative sense for example. More testing of this could be done, and it is likely that tuning would yield some improvements in the algorithm's performance. The choice we made was guided by retaining simplicity of implementation. If a choice of a nonzero initial value is off scale by several orders of magnitude, our algorithm may be inefficient. We plan to make changes to the algorithm to deal with this out-of-scale problem at a later time.

2. **Selecting the Dimensions After Successive Best Values of $\|\tilde{f}(\tilde{x})\|$ are obtained**

Our principle for choosing the scale on the size of B is based on past behavior. Specifically, the bounds are closely related to the distance between adjacent values of $\tilde{x}$. There are three further effects that we use to choose the dimensions. If the step direction changes sign, we prefer smaller moves. Secondly, if the function is exhibiting linear behavior on this move or thirdly, follows a sequence of best $\tilde{x}$ values, a boost in the size of the step is desired.

Linear behavior is measured using the formulas

$$\alpha = \max(f_C^2 - P^2, 0)$$

$\alpha = (\alpha + P*(ALFAC + 1)*(f_L - P))/(2\alpha + P*ALFAC*(f_L - P))$
$ALFAC = 0.125 * (1/\alpha + ALFAC)$

This formula has the following effect. If $f_C \le P$, then $\alpha = 1+1/ALFAC$. If $f_C^2 = P * f_L$, then $\alpha = 1$. If $f_C^2 >> P*f_L$, then $\alpha = 0.5$ (One can think of increases in α as measuring more linear behavior). The factor for boosting the bounds is updated using the formula boost $=$ $\min((\alpha+1)*\text{boost}, 10^{10})$. The formula for updating the bounds is

(4) $B_i = \alpha*SIGN(\delta_i+B_i, \delta_i) + \text{boost}*\delta_i$

The sign of B_i is used to store the direction of the last step. Note that this formula includes the three effects mentioned above.

3. <u>Selecting the Dimensions After Finding a Best Value of $\|\underset{\sim}{f}(\underset{\sim}{x})\|$ After a Value that is not a Best</u>

The size of the move is based on differences between best $\underset{\sim}{x}$ values using (4) with $\alpha = 1/4$ and boost $= 1$. This choice was based on performance. A conservative measure is used because a new best was not immediately found.

It is important to use the difference between best $\underset{\sim}{x}$ values rather than the size of the individual moves. If adjacent best $\underset{\sim}{x}$ values are smaller than moves taken, then there is no benefit to taking large moves. On the other hand, if adjacent best $\underset{\sim}{x}$ values are significantly farther apart than the size of moves taken, this is an indication that the large move can be allowed.

4. <u>Selecting the Dimensions After Giving Up and Retreating to a Best Value for $\|\underset{\sim}{f}(\underset{\sim}{x})\|$</u>

The logic for this case is the simplest of all. The dimensions of the box, after retreating, are reduced using (4) with $\alpha = 1/8$ and boost $= 1/4$. Some reduction is required given that a retreat took place. The values of 1/8 and 1/4 were chosen based on performance with the test set. The move δ_i is the last move that resulted in a new best value for $\|\underset{\sim}{f}(\underset{\sim}{x})\|$, or the value used after adjusting from a previous retreat.

5. Selection the Dimension After Obtaining a Value for $\|\underset{\sim}{f}(\underset{\sim}{x})\|$ Larger than the Best Obtained Previously

In this case it is important that the size of the box should tend to decrease. If the size decreases too much, however, our first rule for retreating would take effect and waste the effort. The formulas for the size of the bound in this case are given by

$$\alpha = ((1/4)*f_C + (1/2)*f_L)/(f_C + f_L)$$

(5) $$B_i = \alpha*B_i, \text{ if } \delta_i = 0$$

$$= \alpha*[SIGN((1/4)*\delta_i + B_i,\delta_i) + (3/4)*\delta_i], \ \delta_i \neq 0$$

This formula has the following properties in each separate coordinate of the box. If the move gave a good decrease in f_C and the direction of the move is unchanged, then the dimension of the box stays the same. If f_C is a good decrease in the opposite direction, the box shrinks by 3/4. If f_C significantly increases and the move is in the same direction, the box shrinks by 1/2. Finally, if f_C significantly increases and the move is in the opposite direction, the box is reduced by 3/8.

We emphasize here that it is not necessarily a bad move if the value of f_C increases but the linear residual P is still small.

4. Some Test Results for the New Algorithm

We developed our algorithm using the test set [5] for unconstrained nonlinear least squares problems. Having this set of testing software was most appreciated by the authors. In this section, we give the results of our algorithm compared to the MINPACK [1] codes for nonlinear least squares problems.

There are characteristics of some test problems that make comparisons hard to draw. For example certain problems, (8, 9, 10, 15 (10 by 10), 16 (N by N), N=10, 30, 40) have more than one local minimum. The authors of [5] claimed that a package they tested, NLSQ2, failed on problem 10, second starting point. This problem has N = 3 variables and M = 16 data values. In fact, the result corresponds to a local minimum near $x_2 = x_3 = \infty$. The residual reported for this problem is the residual variance obtained using the average of the data values. For the class of algorithms being considered, we would not characterize any convergence to a local minimum as a failure.

For these reasons, we present results in Tables 1 and 3 for the H-K-1 and MINPACK codes for those problems that are directly comparable. In particular, the points that the two codes converge to are the same, and the convergence criteria are (effectively) the same. The output flags INFO/H-K-1 and INFO/ MINPACK are defined below. Their values indicate the reasons why each algorithm converged on each problem.

Tables 2 and 4 show the results of the H-K-1 and MINPACK codes for those problems that have the same convergence criteria but with different solution points obtained. Note that the distance of the solution from the initial point is listed for both solutions.

Reliability of software for nonlinear least squares has been discussed in the evaluation paper [6]. A reader may get the impression from the paper that efficiency is not an issue in evaluating code packages, only reliability. Obviously, efficiency of the algorithm is important. One of us (Krogh) has been concerned with applications involving multipoint boundary value problems with modest sized systems of ordinary differential equations. Each evaluation of the function and Jacobian matrix requires the integration of the system over a long trajectory, and the computation is expensive. In fact, this part of the process dominates the linear algebra involved in the Levenberg-Marquardt algorithm used to implement the Gauss-Newton method [7]. For problems where the functions are cheap to evaluate, the issue of efficiency in the linear algebra can become important. Of course, reliability is important. The software should not say that it has found a solution when it has not; it should not fail because an intermediate point has a singular Jacobian matrix; it should satisfy all of its claimed tests for convergence, etc. It is necessary that a careful study of the output from a package be made so that fair conclusions are obtained when one makes comparisons. For example, all of the packages evaluated in [6] converge to local minima; none of them guarantee that a global minimum has been found. Thus, one cannot fault any algorithm tested for finding one local mininum versus another. (One might prefer the algorithm that moves to the nearest min.) Our view is that performance should be compared only for those problems that result in convergence to the same local min.

We used constraints only for problem 10. (Results are also given without constraints.) Constraints used were: (a) nonnegativity on the variables and (b) restricting the argument of the exponential functions so that the value is within machine limits. These constraints seem natural for this problem. From the point of view of a user of nonlinear least squares software, one would probably solve the problem without constraints at the outset. Then after seeing the unsatisfactory results, constraints would be applied. After this the user would have more confidence that the desired minimum value had been obtained.

We now give a list of meanings for the output flag INFO. The machine precision was $\eta \doteq 7.1 \times 10^{-15}$ on a CDC 6600.

INFO/H-K-1	Meaning
2	Obtained a small value for $\Vert \underset{\sim}{f}(\underset{\sim}{x}) \Vert$. (Tolerance is $\eta^{1/2}$).
3	Obtained a minimum for $\Vert \underset{\sim}{f}(\underset{\sim}{x}) \Vert$. Tests satisfied are: (a) $\max(0, f_L^2 - P^2)^{1/2} \le 10^{-2} * P$ and (b) $\Vert f_B - f_C \Vert \le 10^{-2} * f_B$ and (c) $\Vert f_C - f_L \Vert \le f_B * \eta^{1/2}$
4	No projection of residual into column space of Jacobian matrix, i.e., $P \ge f_C$. (Usually means residual is pure noise.)
5	$\Vert \Delta x_p \Vert \le \eta$ and retreat = .false.
6	$\Vert \Delta x_p \Vert \le (10\eta) * \Vert x_p \Vert$ and retreat = .false.

INFO/MINPACK	Meaning
1	Obtained a minimum for $\Vert \underset{\sim}{f}(\underset{\sim}{x}) \Vert$. Test is: $f_C \le (1 + \eta^{1/2}) * P$
2	$\Vert \Delta x_p \Vert \le (10\eta) * \Vert \underset{\sim}{x}_B \Vert$
3	INFO/MINPACK = 1 and 2 are both satisfied
4	No projection of residual into column space of Jacobian matrix.
5	Obtained a small value for $\Vert \underset{\sim}{f}(\underset{\sim}{x}) \Vert$. (Tolerance is $\eta^{1/2}$).

The results listed in Tables 3 and 4 are only for the reader's general information. No conclusions should be drawn from the numbers. We plan to present algorithms that will solve the problems 6, 7, 13, 14 and 15 in Part II of this report (not yet written). We expect the performance of our algorithm to improve most markedly for the nonsmall residual problems.

Interpretation of the results of Tables 1-2, with regard to efficiency, can vary. For example, problems where the Jacobian matrices are analytically available frequently have the property that evaluating $\tilde{f}(\tilde{x})$ and $\partial\tilde{f}/\partial\tilde{x}$ involve little more work than evaluating $\tilde{f}(\tilde{x})$ alone. In many applications these quantities can be efficiently computed at the same time, and users often will do so. In this case, the amount of work for each problem would be proportional to the numbers NFEV in Tables 1-4. Under this measure for efficiency, our new algorithm is (almost) uniformly doing a superior job since the function and Jacobian are required at the same point. At the other extreme, in applications where the Jacobians are approximated by one-sided divided differences, the amount of work for $\partial\tilde{f}/\partial\tilde{x}$ will be about N times the work for evaluating $\tilde{f}(\tilde{x})$. For this comparison we compute WORK = NFEV + N*NJEV, summarized in Table 1 and 3. Using this measure for work penalizes our performance because of the fact that at each iteration we evaluate both $\tilde{f}(\tilde{x})$ and $\partial\tilde{f}/\partial\tilde{x}$. Even under these conditions we are generally performing in a superior manner with regard to efficiency.

The problem listed under NPROB in the tables has either a blank, a ' or a " by its number. These denote the standard starting point, [5], 10 times it, or 100 times it, respectively.

RJH:sdc:7289A:08/07/84X

References

[1] Garbow, B. S., K. E. Hillstrom, J. J. Moré, Implementation Guide for MINPACK-1, Argonne National Laboratories, Rept. ANL-80-68, Argonne, IL, 60439 (July, 1980).

[2] Dennis, J. E., Jr., D. M. Gay, R. E. Welsch, Algorithm 573--An Adaptive Nonlinear Least Squares Algorithm, ACM Trans. Math. Software, Vol. 7, No. 3, p. 369 (Sept., 1981).

[3] Lawson, C. L., SFTRAN3--Programmer's Reference Manual, JPL Document No. 1846-98, Rev. A, California Institute of Technology, Pasadena, CA (April, 1981).

[4] Hanson, R. J., Linear Least Squares with Bounds and Linear Constraints, (submitted to SIAM JSSC, April 1983). Also see Sandia Laboratories Rept. SAND82-1517, Albuquerque, NM (August, 1982).

[5] Moré, J. J., B. S. Garbow, K. E. Hillstrom, Algorithm 566 FORTRAN Subroutines for Testing Unconstrained Optimization Software, ACM Trans. Math. Software, Vol. 7, No. 1, p. 136 (March, 1981).

[6] Hiebert, K. L., An Evaluation of Mathematical Software that Solves Nonlinear Least Squares Problems, ACM Trans. Math. Soft., Vol. 7, No. 1, p. 1 (March, 1981).

[7] JPL Fortran V Subprogram Directory, Ed. 5, 1846-23, Rev. A, Ch. 8.1 (Feb. 1, 1975).

TABLE 1

Comparable Results for Small Residual Problems

			H-K-1 Algorithm					MINPACK				
NPROB	N	M	NFEV	NJAC**	WORK	INFO	Final L_2 Norm	NFEV	NJAC	WORK	INFO	Final L_2 Norm
4	2	2	10	10	30	2	-0-	22	15	52	5	-0-
4'	2	2	5	5	15	2	5.7×10^{-13}	8	4	16	5	1.28×10^{-12}
4"	2	2	6	6	18	2	3.6×10^{-12}	6	3	12	5	1.60×10^{-12}
5	3	3	9	9	36	2	9.6×10^{-9}	11	7	32	5	5.00×10^{-8}
5'	3	3	11	11	44	2	3.5×10^{-9}	20	14	62	5	1.46×10^{-9}
5"	3	3	10	10	40	2	3.4×10^{-15}	19	15	64	5	2.36×10^{-14}
8	3	15	6	6	24	3	9.0635961×10^{-2}	7	5	22	1	9.0635960×10^{-2}
9	4	11	10	10	50	3	1.7536343×10^{-2}	17	14	73	1	1.7535838×10^{-2}
10	3	16	14	14	56	3	9.3779451	127	116	475	1	9.3779451
10'	3	16	45	45	180	3	9.3779453	1830	1479	6267	1	9.3779451
* 10	3	16	16	16	64	3	9.3779452					
* 10'	3	16	26	26	104	3	9.3779453					
* 10"	3	16	26	26	104	3	9.3779470					
11	6	31	7	7	49	3	4.7831216×10^{-2}	8	6	44	1	4.7829594×10^{-2}
11'	6	31	12	12	84	3	4.7829624×10^{-2}	14	12	86	1	4.7829594×10^{-2}
11"	6	31	15	15	105	3	4.7829939×10^{-2}	16	14	100	1	4.7829594×10^{-2}
11	9	31	7	7	70	3	1.1831146×10^{-3}	8	6	62	1	1.1831146×10^{-3}
11'	9	31	12	12	120	3	1.1831173×10^{-3}	21	16	165	1	1.1831146×10^{-3}
11"	9	31	15	15	150	3	1.1831146×10^{-3}	19	15	154	1	1.1831146×10^{-3}
11	12	31	7	7	91	3	2.1731041×10^{-5}	10	8	106	1	2.1731040×10^{-5}
11'	12	31	13	13	169	3	2.1731041×10^{-5}	14	12	158	1	2.1731040×10^{-5}
11"	12	31	16	16	208	3	2.1731041×10^{-5}	35	28	371	1	2.1731041×10^{-5}
12	3	10	7	7	28	2	8.1×10^{-9}	7	5	22	5	3.37×10^{-10}
16'	10	10	20	20	220	2	5.7×10^{-12}	12	6	72	5	6.82×10^{-8}
17	5	33	6	6	36	3	7.3925101×10^{-3}	19	15	94	1	7.3924926×10^{-3}
18	11	65	11	11	132	3	2.0034984×10^{-1}	17	12	149	1	2.0034404×10^{-1}

*Using Constraints

**One Jacobian evaluation could be subtracted from entries where INFO=2 in H-K-1 algorithm. This was not done.

TABLE 2

Noncomparable Results for Small Residual Problems

			H-K-1 Algorithm					MINPACK Algorithm				
NPROB	N	M	NFEV	NJEV	INFO	Final L_2 Norm	Distance from Initial Point	NFEV	NJAC	INFO	Final L_2 Norm	Distance from Initial Point
8'	3	15	7	7	3	9.0635960×10^{-2}	1.535×10	32	30	1	4.1747690	2.857×10^{7}
8"	3	15	6	6	3	4.1747688	1.231×10^{12}	9	7	1	4.1747690	3.318×10^{7}
9'	4	11	18	18	3	1.7536343×10^{-2}	7.034	73	64	1	3.2052195×10^{-2}	4.860×10^{6}
9"	4	11	23	23	3	3.2052210×10^{-2}	4.691×10^{6}	490	379	1	1.7535838×10^{-2}	7.310×10
10"	3	16	40	40	3	3.7652312×10^{4}	$> 10^{8}$	298	257	3	9.3779451	3.946×10^{5}
16	10	10	2	2		-0-	1.581	14	11	5	2.18×10^{-9}	1.602
16"	10	10	5	5	4	1.0	1.550×10^{2}	43	39	5	6.52×10^{-10}	1.550×10^{2}
16	30	30	2	2	2	-0-	2.739	6	2	1	1.0	2.806×10
16	40	40	2	2	2	9.71×10^{-12}	3.162	6	2	1	1.0	3.295×10

TABLE 3

Comparable Results for Non-Small Residual Problems

			H-K-1 Algorithm					MINPACK				
NPROB	N	M	NFEV	NJAC	WORK	INFO	Final L_2 Norm	NFEV	NJAC	WORK	INFO	Final L_2 Norm
1	5	10	3	3	18	6	2.236080	4	2	14	1	2.2360680
1'	5	50	3	3	18	6	6.7082039	4	2	14	1	6.7082039
6	4	4	15	15	75	2	4.73×10^{-8}	16	14	72	5	4.73×10^{-8}
6'	4	4	18	18	90	2	7.39×10^{-8}	19	17	87	5	7.39×10^{-8}
6"	4	4	22	22	110	2	2.86×10^{-8}	23	21	107	5	2.86×10^{-8}
7	2	2	24	24	72	3	6.9988835	15	8	31	1	6.9988752
7'	2	2	27	27	81	3	6.9988759	20	12	44	1	6.9988752
7"	2	2	34	34	102	3	6.9988873	25	17	59	1	6.9988752
13	2	10	25	25	75	3	1.1151794×10	21	11	43	1	1.1151779×10
14	4	20	30	30	150	3	2.9295444×10^2	240	222	1120	1	2.9295440×10^2
14'	4	20	37	37	185	3	2.9295439×10^2	50	38	202	1	2.9295428×10^2
14"	4	20	37	37	185	3	2.9295446×10^2	214	198	1006	1	2.9295443×10^2
15	1	8	1	1	2	5	1.8862380	2	1	3	4	1.8862380
15'	1	8	26	26	52	3	1.8842581	29	27	56	1	1.8842482
15"	1	8	44	44	88	3	1.8842575	47	45	92	1	1.8842482

TABLE 4

Noncomparable Results for Non-Small Residual Problems

			H-K-1 Algorithm					MINPACK Algorithm				
NPROB	N	M	NFEV	NJEV	INFO	Final L_2 Norm	Distance from Initial Point	NFEV	NJAC	INFO	Final L_2 Norm	Distance from Initial Point
2	5	10	3	3	5	1.4638501	2.141	4	2	1	1.4638501	2.012×10^{2}
2'	5	10	3	3	6	3.4826302	2.154	4	2	1	3.4826302	2.341×10^{2}
3	5	10	3	3	4	1.9097274	1.667	4	2	1	1.9097274	1.589×10^{2}
3'	5	50	3	3	5	3.6917294	1.691	4	2	1	3.6917294	1.794×10^{2}
15	8	8	35	35	3	5.9303359×10^{-2}	2.055×10^{-1}	38	19	1	5.9303236×10^{-2}	1.615×10^{-1}
15	9	9	14	14	2	1.48×10^{-12}	2.151×10^{-1}	12	8	5	9.26×10^{-11}	1.226×10^{-1}
15	10	10	24	24	3	6.9085678×10^{-2}	1.233×10^{-1}	23	11	1	8.0647101×10^{-2}	1.155×10^{-1}

Distribution

T. J. Aird
IMSL, Inc.
7500 Bellaire
Houston, TX 77036

Richard Allen
Dept. of Math, and Statistics
University of New Mexico
Albuquerque, NM 87131

Richard Bartels
University of Waterloo
Waterloo, Ontario N2L 3G1
CANADA

Billy L. Buzbee
Los Alamos National Laboratories
P. O. Box 808
Los Alamos, NM 87545

Roy Danchick
The Rand Corp.
1700 Main St.
Santa Monica, CA 90406

John E. Dennis, Jr.
Dept. of Math. Sciences
Rice University
Houston, TX 77001

Albert M. Erisman
Boeing Comp. Services
565 Andover Park W. 9C-01
Tukwila, WA 98188

Kirby Fong
Lawrence Livermore Laboratory
P. O. Box 808
Livermore, CA 94550

Brian Ford
NAG, Ltd.
7 Banbury Rd.
Oxford OX2 6NN
ENGLAND

Fred N. Fritsch
Lawrence Livermore Laboratory
P. O. Box 808
Livermore, CA 94550

Pat Gaffney
Oak Ridge National Laboratories
P. O. Box Y
Oak Ridge, TN 37830

David Gay
Bell Laboratories
600 Mountain Rd.
Murray Hill, NJ 07974

Phillip Gill
Stanford University
Dept. of Operations Research
Stanford, CA 94305

Fred T. Krogh (5)
Jet Propulsion Laboratory
4800 Oak Grove Drive
Pasadena, CA 91103

Charles L. Lawson
Jet Propulsion Laboratory
4800 Oak Grove Drive
Pasadena, CA 91103

Per Lindström
Univ. of Umeå
Inst. of Information Processing
S-901 87 Umeå, SWEDEN

Tom A. Manteuffel, C-3
Los Alamos National Laboratories
P. O. Box 808
Los Alamos, NM 87545

Roy E. Marsten
Dept. of Management Info. Systems
University of Arizona
Tucson, AZ 85718

Mike Minkoff
Argonne National Laboratories
9700 S. Cass Ave.
Argonne, IL 60439

Cleve Moler
Dept. of Computer Science
University of New Mexico
Albuquerque, NM 87131

Jorge J. Moré
Argonne National Laboratories
9700 S. Cass Ave.
Argonne, IL 60439

Alfred Morris
Naval Surface Weapons Center
DK-74
Dahlgren, VA 22448

Walter Murray
Dept. of Operations Research
Stanford University
Stanford, CA 94305

M. J. D. Powell
Dept. of Appl. Math. &
Theor. Physics
Univ. of Cambridge
Silver St.
Cambridge CB39EW, ENGLAND

John Reid
AERE Harwell
Comp. Sci. & Systems Div.
Oxford, OX11 ORA
ENGLAND

Report Section
Dept. of Computer Science
Royal Inst. of Technology
Stockholm
SWEDEN

Mike Saunders
Dept. of Operations Research
Stanford University
Stanford, CA 94305

Don Scheck
Dept. of Indust. Engineering
Ohio University
Athens, OH 45701

Danny Sorensen
Argonne National Laboratories
9700 S. Cass Ave.
Argonne, IL 60439

Richard A. Tapia
Dept. of Math. Sciences
Rice University
Houston, TX 77001

Homer Walker
Dept. of Math. & Stat.
University of New Mexico
Albuquerque, NM 87131

Robert Ward
Oak Ridge National Laboratories
P. O. Box Y
Oak Ridge, TN 37830

Margaret Wright
Dept. of Operations Research
Stanford University
Stanford, CA 94305

1636 P. G. Kaestner
1640 G. J. Simmons
1641 R. J. Thompson
1642 L. F. Shampine
1642 D. E. Amos
1642 B. L. Hulme
2113 J. A. Wisniewski
2401 C. F. Diegert
2614 A. R. Iacoletti
2640 J. L. Tischhauser
2644 D. M. Darsey
2644 D. C. Ghiglia
2646 M. [illegible]. Scott
2646 E. A. Aronson
2646 R. J. Hanson (30)
2646 L. A. Romero
2646 H. A. Watts
2646 W. H. Vandevender
7112 F. Biggs
7223 K. V. Diegert
7223 R. R. Prairie
7241 K. L. Hiebert
8214 M. A. Pound
8328 J. F. Lathrop
833[illegible] R. J. Kee
8335 R. E. Huddleston
9336 P. J. McDaniel
3141 L. J. Erickson (5)
3151 W. L. Garner (3)
3154-3 C. H. Dalin (25)
(For DOE/TIC)