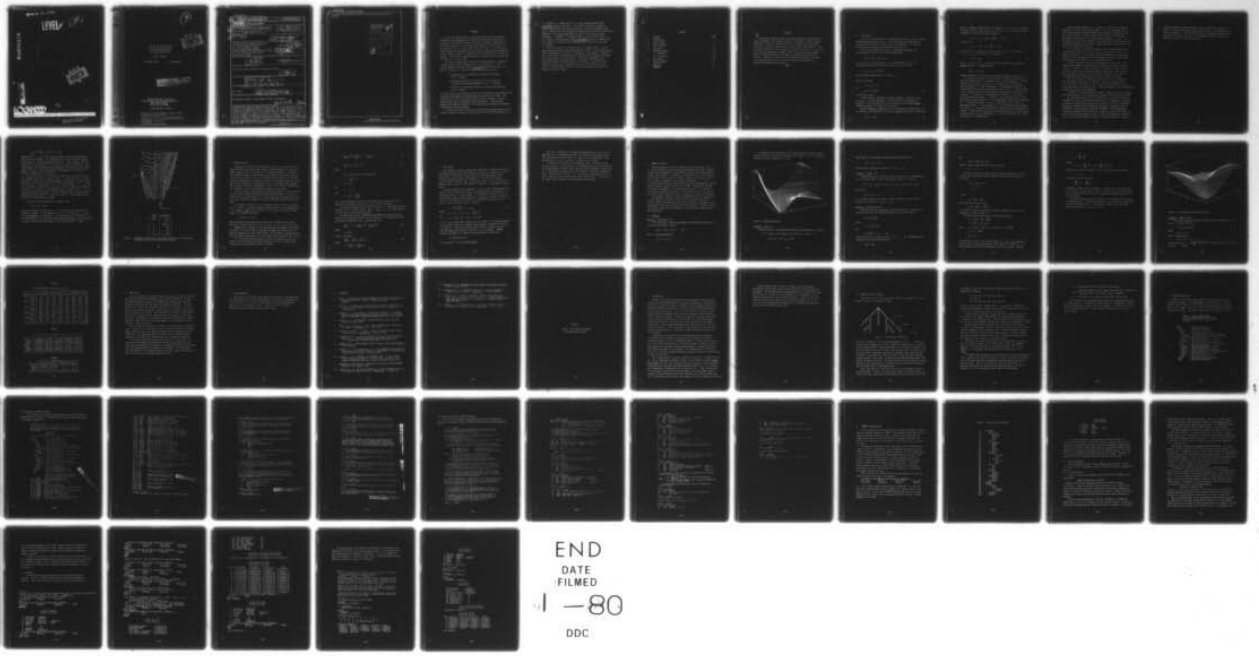
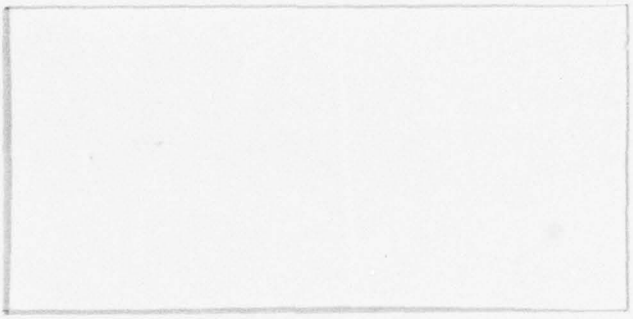
AD-A078 220 LOCKHEED MISSILES AND SPACE CO INC PALO ALTO CA PALO --ETC F/G 12/1
DESIGN AND IMPLEMENTATION OF A NEW DESCENT ALGORITHM FOR LOCAL --ETC(U)
NOV 79 P S JENSEN F49620-76-C-0003
UNCLASSIFIED LMSC/D683306 AFOSR-TR-79-1226 NL

| OF |
ADA
078220



END
DATE
FILMED
1 -80
DDC

LEVEL II



LOCKHEED
MISSILES & SPACE COMPANY, INC • SUNNYVALE, CALIFORNIA

(9)

DESIGN AND IMPLEMENTATION OF A NEW DESCENT ALGORITHM FOR LOCAL OPTIMIZATION

Paul S. Jensen

15 November 1979 LMSC-D683306

Applied Mechanics Laboratory
Lockheed Palo Alto Research Laboratory
3251 Hanover Street
Palo Alto, CA 94304

(415) 493-4411, X 45133

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: AFOSR-TR-79-1226	2. GOVT ACCESSION NO. / 3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle): DESIGN AND IMPLEMENTATION OF A NEW DESCENT ALGORITHM FOR LOCAL OPTIMIZATION	5. TYPE OF REPORT & PERIOD COVERED: Final report for period 7/15/76 through 9/15/79
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): Paul S. Jensen	8. CONTRACT OR GRANT NUMBER(s): F49620-76-C-0003
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Lockheed Palo Alto Research Laboratory, 3251 Hanover Street (52-33/205), Palo Alto, California 94304	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 61102F 2304/A3
11. CONTROLLING OFFICE NAME AND ADDRESS: Air Force Office of Scientific Research (NM), Bldg. 410, Bolling Air Force Base, Washington, DC 20332	12. REPORT DATE: 15 Nov 79 / 13. NUMBER OF PAGES: 61
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report): Unclassified / 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Final rept. 15 Jul 76-15 Sep 79

18. SUPPLEMENTARY NOTES

Tech. Other

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Optimization, descent, planar search, CRATER, minimization.

210 118

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

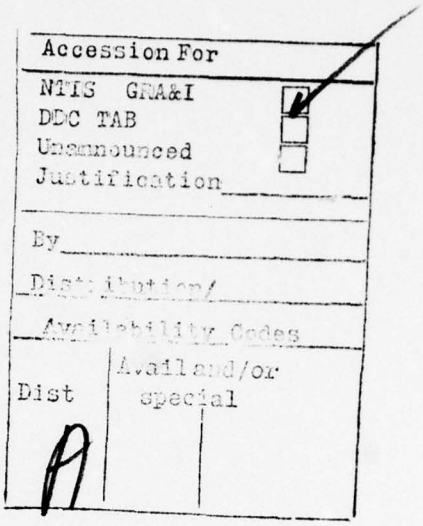
A new descent algorithm based on direction searches in the osculating plane as opposed to the negative gradient is described. The major practical difference between the implementation of this approach and quasi-Newton algorithms is the use of the Hessian in place of the inverse Hessian. Thus, the descent algorithm is most suitable for problems for which it is practical to recalculate the Hessian a number of times during the solution process. A number of comparative results with popular quasi-Newton algorithms are provided. Extensive discussion of the implementation details is included along with documentation of a

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. convenient computer program CRATER for unconstrained optimization analysis



FOREWORD

This is the final report for a three year research effort entitled "Development and Implementation of Efficient Sparse Matrix Algorithms". The scope of this effort has been very broad, covering the development of efficient sparse matrix factorization and processing algorithms and the development of local and global optimization algorithms. The topics form a logical sequence in the sense that the results of each study are supportive of the successor studies in a natural way for the treatment of large sparse systems. Yet, they are autonomous efforts, each having a broad spectrum of applicability in its own right.

The study on sparse matrix factorization and processing is reported in "Sparse Symmetric Matrix Processing" by P. S. Jensen and J. K. Reid, Lockheed Research Laboratory Report LMSC-D626184 dated 26 May 1978. This report is written as a volume of three related papers entitled:

"A Comparison of Two Sparse Matrix Processing Techniques," by P. S. Jensen,

"A Package of Subroutines for Solution of Very Large Sets of Linear Finite-Element Equations", by J. K. Reid and

"A Fortran Virtual Storage Simulator for Non-Virtual Computers", by P. S. Jensen.

The second paper, also available as Atomic Energy Authority report AERE-M 2947, documents the computer program developed as a result of this study. The utilization of auxilliary storage by this program is based on a virtual memory concept that is documented in the third paper. These computer programs have been distributed in the United States and England and requests from Canada are being processed.

A preliminary report of the work on local and global optimization is provided in "Numerical Techniques for Optimization and Nonlinear Equations" by

P. S. Jensen, E. R. Hansen and H. M. C. Yee, Lockheed Research Report LMSC-D630055 dated 29 June 1978. It included a preliminary version of "Global Optimization Using Interval Analysis — The Multi-Dimensional Case" by E. R. Hansen, which was subsequently accepted for publication in Numerische Mathematik. That paper established the basis of the computer program documented in "GLOBALMIN — A Computer Program for Global Optimization" by E. R. Hansen, Lockheed Research Report LMSC-D683307, dated 15 November 1979. This program has just been completed as of this writing and has not been distributed.

The remaining work on local optimization is reported here. The main text covers the technical development and test results. The computer program CRATER, developed for the study, is documented in the appendix. CRATER is a very flexible program implementing both descent and quasi-Newton algorithms for unconstrained, local optimization. The present implementation is oriented toward small problems having full Hessians. An effort to link CRATER to a sparse matrix processing system and a non-linear structural analysis computer program (STAGS) for large scale engineering analysis is planned for the near future.

CONTENTS

ABSTRACT

A new descent algorithm based on direction searches in the osculating plane as opposed to the negative gradient is described. The major practical difference between the implementation of this approach and quasi-Newton algorithms is the use of the Hessian in place of the inverse Hessian. Thus, the descent algorithm is most suitable for problems for which it is practical to recalculate the Hessian a number of times during the solution process. A number of comparative results with popular quasi-Newton algorithms are provided. Extensive discussion of the implementation details is included along with documentation of a convenient computer program CRATER for unconstrained optimization analysis.

1. INTRODUCTION

Let $f:R^n \to R$ be continuously differentiable and let $\underset{\sim}{g}(\underset{\sim}{x})$ and $H(\underset{\sim}{x})$ denote the gradient and Hessian of f at $\underset{\sim}{x}$. We are interested in determining a vector $\underset{\sim}{x}^*$ that minimizes f locally, i.e., such that $\underset{\sim}{g}(\underset{\sim}{x}^*) = \underset{\sim}{0}$. For this discussion, we consider only unconstrained minimization.

The methods considered here are all linearization methods [13], characterized by the use of an affine approximation

$$\underset{\sim}{g}(\underset{\sim}{x}) \tilde{=} A_k(\underset{\sim}{x} - \underset{\sim}{x}_k) + \underset{\sim}{g}(\underset{\sim}{x}_k) \tag{1}$$

of $\underset{\sim}{g}$, which is to be valid for $\underset{\sim}{x}$ in a neighborhood of $\underset{\sim}{x}_k$. The practical application of this approximation is to use its zero

$$\underset{\sim}{x}_{k+1} = \underset{\sim}{x}_k - A_k^{-1}\, \underset{\sim}{g}(\underset{\sim}{x}_k)$$

as an improved approximation to a zero of $\underset{\sim}{g}$.

We call the change

$$\underset{\sim}{s}_k = \underset{\sim}{x}_{k+1} - \underset{\sim}{x}_k$$

or

$$\underset{\sim}{s}_k = -A_k^{-1}\, \underset{\sim}{g}(\underset{\sim}{x}_k) \tag{2}$$

the k^{th} step vector.

The most famous linearization method is Newton's method which uses $A_k = H(\underset{\sim}{x}_k)$. For many problems, H is very difficult or expensive to construct. This fact has given rise to the widely studied quasi-Newton methods, surveyed in [3 and 5] for example.

Another approach in determining A_k is motivated by considering $\underset{\sim}{x}:R \to R^n$ a continuous, differentiable function of a pseudo-time variable t defined by

$$\dot{\underset{\sim}{x}}(t) = -\underset{\sim}{g}(\underset{\sim}{x}), \tag{3}$$

where $\overset{\circ}{\underset{\sim}{x}}$ denotes the derivative with respect to t. If $\underset{\sim}{x}(0)$ is in a region of attraction of $\underset{\sim}{x}^*$, then clearly $\underset{\sim}{x}(t) \to \underset{\sim}{x}^*$ as $t \to \infty$. For a small time increment h, we may use a Taylor expansion

$$\underset{\sim}{x}(t + h) = \underset{\sim}{x}(t) + h\overset{\circ}{\underset{\sim}{x}}(t) + \frac{h^2}{2}\ddot{\underset{\sim}{x}}(t) + O(h^3)$$

or, from (3),

$$\underset{\sim}{x}(t + h) = \underset{\sim}{x}(t) - h\underset{\sim}{g} + \frac{h^2}{2} H\underset{\sim}{g} + O(h^3)$$

to approximate $\underset{\sim}{x}$ by neglecting the high order terms in h. Defining $\underset{\sim}{s} = \underset{\sim}{x}(t + h) - \underset{\sim}{x}(t)$ similar to (2), we obtain

$$\underset{\sim}{s} \doteq \alpha\underset{\sim}{g} + \beta H\underset{\sim}{g}, \tag{4}$$

where $\alpha = -h$ and $\beta = h^2/2$. We shall refer to methods of this nature as descent methods. In descent methods

$$A_k^{-1} = \alpha I + \beta H(\underset{\sim}{x}_k).$$

Steepest descents is the most well known descent method, for which $\beta = 0$.

An important practical difference between quasi-Newton and descent is that quasi-Newton uses H^{-1} and descent uses H to approximate $\underset{\sim}{s}$. If the problem being treated is such that H is difficult to calculate, the noted difference is of minor consequence. The appropriate algorithms for either model usually construct approximations to H^{-1} or H using low rank (1 or 2) updates and the two approximations are about equally difficult to construct. If, however, it is reasonable to calculate H occasionally during the solution process, (4) presents a clear advantage since it does not require a factorization of H. This advantage is particularly noticeable if H is large and sparse. In fact, we note in (4) that an explicit representation of H is not required, i.e., only the product $H\underset{\sim}{g}$ is needed for various $\underset{\sim}{g}$. Thus, we can conveniently introduce low rank updates to the current H with minimal impact on an algorithm based on (4) by applying the updates directly to $\underset{\sim}{g}$ rather than to H prior to forming $H\underset{\sim}{g}$ (i.e., retain the updated H in product form).

There have been studies, e.g., [11] and [16], applying quasi-Newton algorithms to fairly large, sparse systems. In [16], the updates for the inverse Hessian were initially formed in the conventional manner and then zeros were inserted to conform to the sparsity pattern of the problem. It has apparently been shown to be superlinearly convergent [5 p 60]. In [1], the updates are not directly applied to the approximate inverse Hessian but are held and applied in product form. After every ten or so iterations, the Hessian is reformed and refactored.

In this paper, we consider the application of a descent algorithm to sparse problems under the assumption that it is reasonable to reform the Hessian periodically. As indicated in the above discussion, we should not expect as rapid convergence as with quasi-Newton methods but, should expect the average cost per iteration to be substantially less.

In the next section, we discuss the descent algorithm used. It is a planar descent method in the sense that a direction in the [$\underset{\sim}{g}$, $H\underset{\sim}{g}$] plane (see (4)) is chosen in place of the traditional steepest descent direction. In Section 3 we discuss the matrix updating schemes used. This topic has been extensively discussed in the literature and here we simply use the two approaches that have enjoyed the greatest success in the past.

Once the direction is determined, we use a line search to establish the step length as discussed in Section 4. Thus, the function is not truly minimized in the plane [$\underset{\sim}{g}$, $H\underset{\sim}{g}$], but only along a direction in that plane determined on the basis of (4). A study of complete planar minimization at each step remains to be conducted.

In Section 5 we present some test results. Several "classical" problems that have appeared in the literature are presented along with some new ones.

Our overriding objective for this effort was to implement an effective algorithm for large, sparse problems for which it is reasonable to construct the Hessian occasionally during a solution process. Although the research was directed toward descent methods, we considered it imperative that no commitment to descent methods should be reflected in the implementation. Furthermore, because of many variations in updating, step direction and line search that must be considered in such a study, we considered it important to use a very flexible (modular) design in the implementation and include a

convenient problem oriented language in order to facilitate a systematic study. The resulting computer program, called CRATER, is discussed in some detail in Appendix. We feel that CRATER meets its design objectives very well and should prove very useful for production development as well as additional studies in local optimization.

2. PLANAR SEARCH

Instead of defining coefficients α and β as in (4), we consider predicting α and β to minimize $f(\underset{\sim}{x} + \underset{\sim}{s}(\alpha,\beta))$. Using a Taylor expansion about $\underset{\sim}{x}$ we have

$$f(\underset{\sim}{x} + \underset{\sim}{s}) \simeq f(\underset{\sim}{x}) + \underset{\sim}{s}^T \underset{\sim}{g}(\underset{\sim}{x}) + \frac{1}{2}\underset{\sim}{s}^T H(\underset{\sim}{x})\underset{\sim}{s},$$

which may be combined with (4) to yield

$$f(\underset{\sim}{x} + \underset{\sim}{s}) \simeq f + \alpha\sigma_0 + \beta\sigma_1 + \frac{1}{2}(\alpha^2\sigma_1 + 2\alpha\beta\sigma_2 + \beta^2\sigma_3), \tag{5}$$

where we define

$$\sigma_i \equiv \underset{\sim}{g}^T H^i \underset{\sim}{g}, \quad i = 0, 1, 2, 3.$$

The extrema occur at the zeros of the partial derivations of (5) wrt α and β, which are formally given by

$$\alpha' = \frac{\sigma_1\sigma_2 - \sigma_0\sigma_3}{\sigma_1\sigma_3 - \sigma_2^2} \tag{6}$$

and

$$\beta' = \frac{\sigma_0\sigma_2 - \sigma_1^2}{\sigma_1\sigma_3 - \sigma_2^2}$$

if $\sigma_1\sigma_3 \neq \sigma_2^2$. Otherwise, we choose $\alpha' = -\sigma_0/\sigma_1$ and $\beta' = 0$.

Because the approximation (5) is valid only for small $\underset{\sim}{s}$, we have no guarantee that the step

$$\underset{\sim}{s}' = \alpha'\underset{\sim}{g} + \beta' H\underset{\sim}{g} \tag{7}$$

minimizes $f(\underset{\sim}{x} + \underset{\sim}{s})$ over α and β, or even that $f(\underset{\sim}{x} + \underset{\sim}{s}') \leq f(\underset{\sim}{x})$. The following theorem at least shows that $\underset{\sim}{s}'$ is in a descent direction.

Theorem If $f:R^n \to R$ is twice continuously differentiable with non-negative definite Hessian H, and at point $\underset{\sim}{x} \in R^n$ the gradient $\underset{\sim}{g}(\underset{\sim}{x}) \neq \underset{\sim}{0}$, then the step $\underset{\sim}{s}'$ defined by (7) is in a descent direction.

Proof

We must show that $\underset{\sim}{s}'^T\underset{\sim}{g} < 0$. If $\sigma_1\sigma_3 = \sigma_2^2$ then $\underset{\sim}{s}'$ is parallel to the negative gradient, which satisfies our theorem. Otherwise $\sigma_1\sigma_3 - \sigma_2^2 > 0$ by the Cauchy-Schwarz theorem and so from (7) we evidently need to show

$$\theta \equiv (\sigma_1\sigma_3 - \sigma_2^2)(\alpha'\sigma_0 + \beta'\sigma_1) < 0.$$

Now from (6) and the definition of σ_i, we have

$$\theta = -(\sigma_0^2\sigma_3 - 2\sigma_0\sigma_1\sigma_2 + \sigma_1^3).$$

Defining

$$\underset{\sim}{v} = \sigma_0 H\underset{\sim}{g} - \sigma_1\underset{\sim}{g},$$

we observe that

$$\theta = -\underset{\sim}{v}^T H\underset{\sim}{v} < 0. \tag{8}$$

We have strict inequality in (8) because $\underset{\sim}{v}$ is not in the null space of H. If it were, then the denominators in the expressions for α' and β' in (6) would be zero (as would the numerators). That follows immediately from the facts that $H\underset{\sim}{v} = \underset{\sim}{0}$ implies

$$\sigma_0\sigma_2 = \sigma_1^2$$

and

$$\sigma_0\sigma_3 = \sigma_1\sigma_2.$$

-QED-

If we simply implement the planar search algorithm as outlined, and test it on the standard Rosenbrock function

$$f_R(\underset{\sim}{x}) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

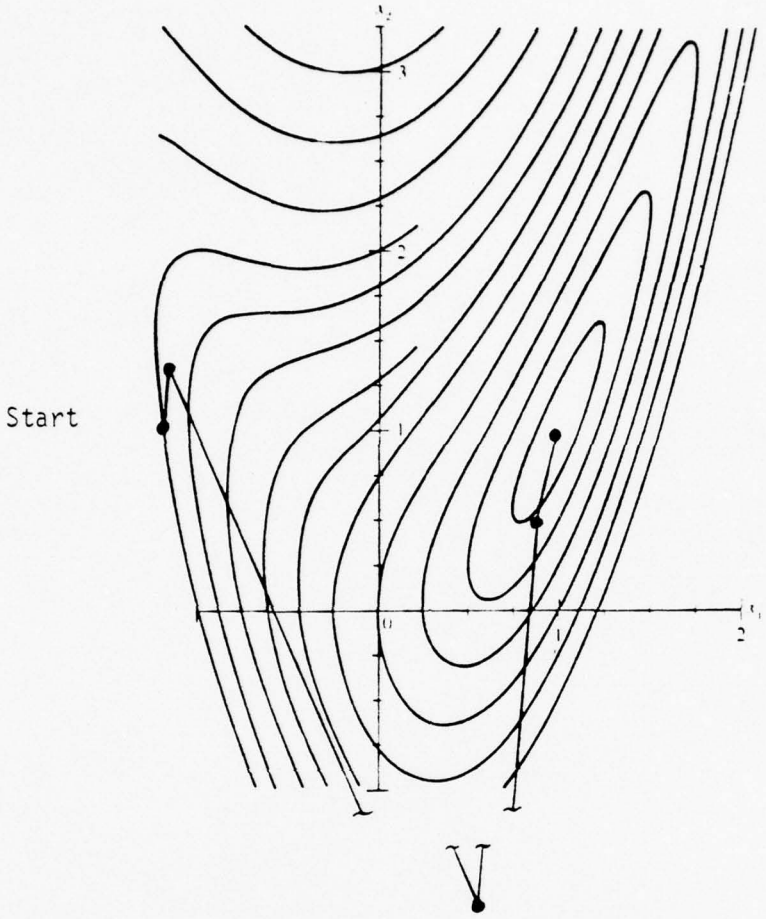
starting at $\underset{\sim}{x}^T = (-1.2, 1)$, we find the pleasing result that convergence is achieved in only 5 steps. This compares with 30 steps for Powell's method and 18 steps for the Fletcher-Powell method [6]. Note, however, that this comparison is not completely fair since here we use the true Hessian. Also, unfortunately, the convergence here progresses in the rather erratic fashion shown in Figure 1. It would be somewhat more appealing if the function decreased monotonically, or nearly so.

We can arbitrarily limit the factor by which the function value at any point exceeds the value at a previous point. For example, making the factor less than (or equal to) unity forces monotonic convergence. In such a case, a multiple $\gamma s'$ of the step given in (7) must be used, where $0 < \gamma < 1$. Note that a suitable γ always exists since $\underset{\sim}{s}'$ is a descent direction as shown in the theorem. The value of γ must be determined by a search along $\underset{\sim}{s}'$. This matter is discussed in Section 4. We shall see later that imposing monotonic convergence on our planar search algorithm leads to a mildly slower convergence rate for the Rosenbrock problem.

If we apply the basic method to a quadratic form

$$f(\underset{\sim}{x}) = \underset{\sim}{x}^T\underset{\sim}{b} - \frac{1}{2}\underset{\sim}{x}^T H \underset{\sim}{x}$$

we observe instant convergence for $n = 2$ (as expected) and monotonic convergence for larger n. For example, if $b_i = n + 1 - i$ and $H_{ij} = i + i$, we obtain convergence in 5 steps for $n = 3$ and convergence in 15 steps for $n = 10$. The Newton method, of course always gives instant convergence for this problem and quasi-Newton methods theoretically require no more than n steps.



STEP	FUNCTIONAL
START	24.20
1	4.73
2	1403.74
3	.06
4	.33
5	.00

Figure 1. Convergence of the Basic Planar Search Algorithm for the Standard Rosenbrock Function Using the Analytic Hessian.

3. MATRIX UPDATING

Techniques for updating the approximate Hessian or inverse Hessian have been very widely studied for over 25 years. Dennis and Moré [5] provide a recent survey of much of the work in this area and Broyden [3] provides a very readable earlier survey of general techniques for optimization and nonlinear equation systems. The work by Broyden was presented at the NSF-CBMS Regional Conference on "The Numerical Solution of Nonlinear Algebraic Systems" held at the University of Pittsburgh in 1972. A highlight of that meeting was a series of lectures by Rheinboldt [13] that also provides excellent survey information as well as considerable theoretical background.

For this study, we used two popular update procedures, viz: DFP (Davidon-Fletcher-Powell) and BFGS (Broyden-Fletcher-Goldfarb-Shanno). In order to present these formulas and the form in which they were implemented, we introduce some notation that appears to be widely accepted among articles on this topic.

As discussed in Section 1, we are seeking a point $\underset{\sim}{x}^* \in R^n$ that minimizes $f: R^n \to R$ locally. We use an iterative procedure which, given a current point $\underset{\sim}{x} \in R^n$, produces a step $\underset{\sim}{s} \in R^n$ such that $\underset{\sim}{x} + \underset{\sim}{s}$ is closer to $\underset{\sim}{x}^*$ than $\underset{\sim}{x}$ in some sense. We introduce the vector

$$\underset{\sim}{y} = \underset{\sim}{g}(\underset{\sim}{x} + \underset{\sim}{s}) - \underset{\sim}{g}(\underset{\sim}{x})$$

which represents the change in the gradient $\underset{\sim}{g}$ of f resulting from step $\underset{\sim}{s}$. Obviously, we would like $||\underset{\sim}{g}(\underset{\sim}{x} + \underset{\sim}{s})|| \leq ||\underset{\sim}{g}(\underset{\sim}{x})||$ or $f(\underset{\sim}{x} + \underset{\sim}{s}) \leq f(\underset{\sim}{x})$ or both.

Probably out of respect for the extensive works of Broyden in this area, the letter B is usually used to represent the approximate Hessian and H is used for something else (the inverse of B). In this paper, we shall also use B for the approximate Hessian, but H will denote the true Hessian (as in Sec. 1).

If B is the approximate Hessian at point $\underset{\sim}{x}$, we construct $\underset{\sim}{s}$, e.g., by means of the planar search algorithm discussed in Section 2, and form the updated Hessian using either of the two formulas

$$B_{BFGS} = B + \frac{1}{(y,s)} y\, y^T - \frac{1}{(v,s)} v\, v^T \tag{9}$$

or

$$B_{DFP} = B + u\, z^T + z\, u^T \tag{10}$$

where

(y,s) denotes the inner product $y^T s$,

$$z = \frac{1}{(y,s)}\, s,$$

$$v = Bs$$

and

$$u = \beta y - v$$

with

$$\beta = \frac{1}{2}\left(1 + \frac{(v,s)}{(y,s)}\right).$$

It is a simple algebraic exercise to show that (9) and (10) are formally equivalent to the forms given on pp 72, 74 of Dennis and Moré [5].

The important features for our application are: 1. The updates preserve symmetry (which is obvious) and 2. $B_{BFGS}\, s = B_{DFP}\, s = y$. The latter feature is basic to all useful update formulas.

For comparison, we also implemented quasi-Newton algorithms using the BFGS and DFP counterpart formulas for the inverse of B. Letting B^{-1} be the approximate inverse Hessian at x we have from [6].

$$B_{DFP}^{-1} = B^{-1} + \frac{1}{(y,s)}\, ss^T - \frac{1}{(y,w)}\, w\, w^T \tag{11}$$

where

$$w = B^{-1}y.$$

From [6], we have

$$B_{BFGS}^{-1} = B_{DFP}^{-1} + (y,w)\, rr^T \tag{12}$$

where

$$r = \frac{1}{(y,s)}\, s - \frac{1}{(y,w)}\, w.$$

4. LINE SEARCH

Two fundamental problems that must be addressed in iterative optimization algorithms are selecting a step direction, as discussed in Section 2, and a step length. Intuitively, a length that minimizes the function in the direction of the step seems most appropriate. This choice of length is called "perfect iteration" [3]. However, that length is fairly costly to determine and, as we saw in the example of Section 2, is not always the best choice.

A popular algorithm due to Davidon [4] uses a cubic interpolation of the function and its partial derivative in the step direction corresponding to lengths of 0 and 1. If we use length h instead of 1 and let f_0, f_0', f_1, f_1' denote the values of the function and its derivatives (along $\underset{\sim}{s}$) at points 0 and h on the step vector $\underset{\sim}{s}$, then cubic interpolation suggests that the length ℓ that minimizes f along $\underset{\sim}{s}$ is given by

$$\ell = (\sqrt{\eta_1^2 - 3f_0'\eta_2} - \eta_1)/3\eta_2 \qquad (13)$$

where

$$\eta_1 = (3(f_1 - f_0) - h(f_1' + 2f_0'))/h^2$$

and

$$\eta_2 = -(2(f_1 - f_0) - h(f_1' + f_0'))/h^3.$$

If $f(\underset{\sim}{x} + \ell\underset{\sim}{s})$ is less than $f(\underset{\sim}{x})$ and $f(\underset{\sim}{x} + h\underset{\sim}{s})$, then ℓ is the accepted length without further searching. Otherwise the process is repeated with smaller h, e.g., $\frac{1}{2}h$ or ℓ. Since the algorithms usually used to generate the step $\underset{\sim}{s}$ take its length into account, an initial $h = 1$ is suitable. However, Fletcher and Powell [6] suggest using conservative extrapolation

$$h = \min(1, 2(f - f^*)/f')$$

if the value $f^* = f(\underset{\sim}{x}^*)$ is known a-priori.

If $f(\underset{\sim}{x})$ is such that it is practical computationally to use $||g||^2$ in the place of f in the above algorithm, some benefit accrues from the fact that the converged value is zero, viz: there is less cancellation in the calculation of η_1 and η_2, and the f* is always known a-priori.

A quadratic line search is also frequently used. Since four values f_o, f_o', f_1 and f_1' are available for the interpolation, one can be discarded or a least squares fit can be used. We have chosen a quadratic interpolation of f' values that continues until a zero (within a prescribed tolerance) is found. Note that this is cubic in f. The processes are started with f_o', f_1', and f_{ℓ}', where f_{ℓ}' comes from the Davidon cubic interpolation. The amount of work done in the line search is controlled by the tolerance.

5. NUMERICAL RESULTS

There have been a variety of comparative results published (see [1, 2, 7, 15] for examples) on the unconstrained optimization problem. Several problems appear repeatedly in such studies and seem to have become defacto standard test functions. Occasionally the author of such a study makes a definite conclusion such as "The Fletcher algorithm was clearly superior to all the others, followed by the Davidon-Fletcher-Powell . . ." in [7]. This author appreciates the value of the test results presented but cautions against drawing a sweeping conclusion from them. The problems involved relatively few unknowns (15, 10 and the rest less than 6) but certainly represented a substantial complexity. Questions relating to sparsity and the use of analytic Hessians (always or occasionally) were not considered.

Unfortunately, the test results presented here are also inadequate to draw any sweeping conclusions. They are only intended to provide a minimal indication of the functioning of the algorithms discussed. A more comprehensive study on the behavior of these algorithms for the optimization of structural panels will be forthcoming. The problems treated here are described below.

5.1 PROBLEMS

Problem 1. Rosenbrock [14]

A two dimensional problem that presents a considerable challenge for numerical optimization is given

$$r_\alpha(\underset{\sim}{x}) = \alpha(x_1^2 - x_2)^2 + (1 - x_1)^2$$

with $\alpha = 100$ and starting at

$$\underset{\sim}{x}_0^T = (-1.2, 1).$$

In appearance, this function is like a deep canyon with a curved, gently sloping bottom as illustrated in Figures 1 and 2 (using a $\alpha = 10$). Rosenbrocks function converges to zero at $x^{*T} = (1,1)$.

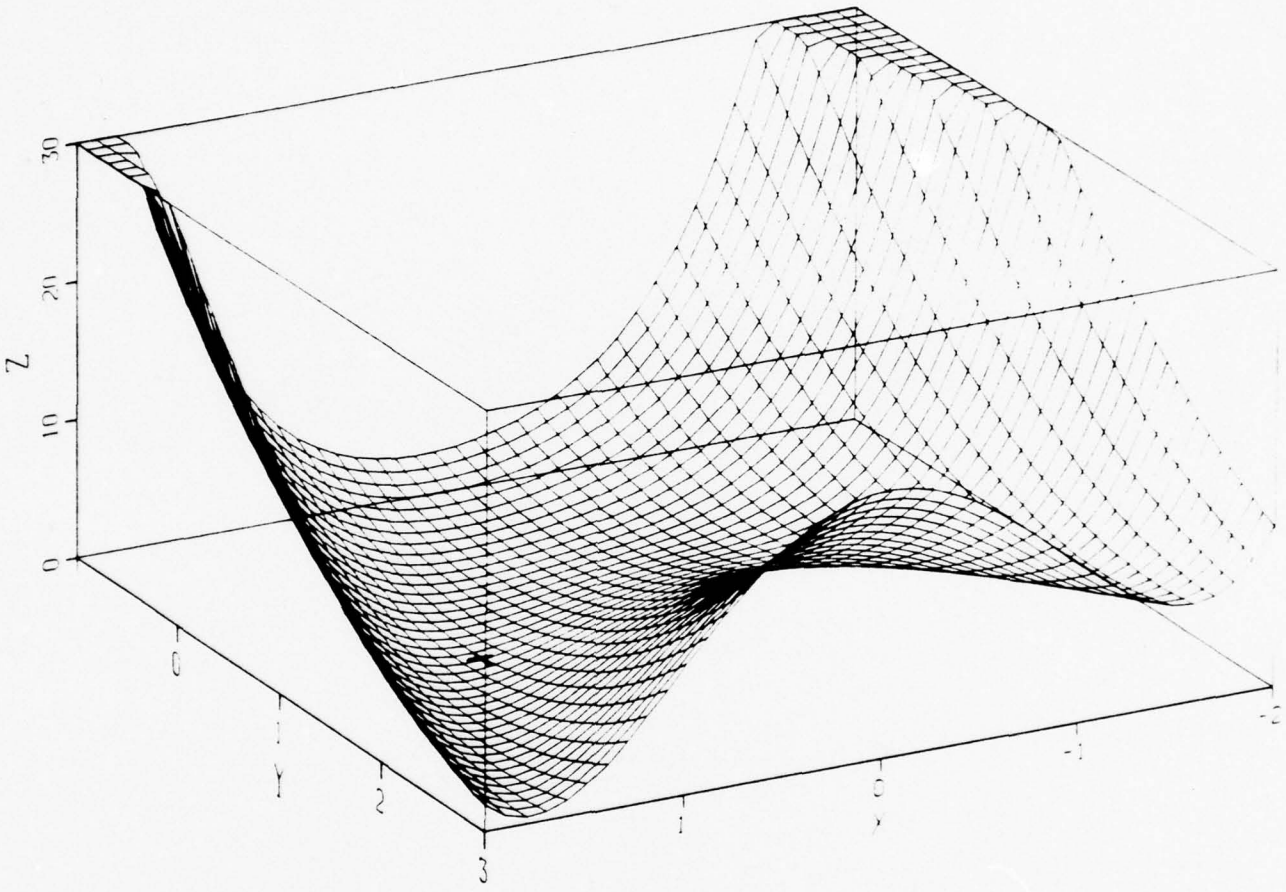


Figure 2. Rosenbrock Function

Problem 2. Wood [15]

A generalization of the Rosenbrock problem to four dimensions is given by

$$f(x) = r_{100}(x_1, x_2) + r_{90}(x_3, x_4) + 19.8\,(x_2 - 1)(x_4 - 1)$$

$$+ 10.1\,((x_2 - 1)^2 + (x_4 - 1)^2)$$

where $r_\alpha(\underset{\sim}{x})$ is the Rosenbrock function and the starting point is

$$\underset{\sim}{x}_o^T = -(3, 1, 3, 1).$$

Woods function converges to zero at $\underset{\sim}{x}^{*T} = (1, 1, 1, 1)$.

Problem 3. Powell [24]

A problem that has a singular Hessian at the point $\underset{\sim}{x}^*$ of convergence and has an extremely gentle slope in one direction near $\underset{\sim}{x}^*$ is given by

$$f(\underset{\sim}{x}) = (x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4$$

starting at

$$\underset{\sim}{x}_o^T = (3, -1, 0, 1).$$

It converges to zero at the origin. Within a distance ε of the origin, the slope is $O(\varepsilon^3)$ along $x_1 = -10x_2$, $x_3 = x_4$.

Problem 4. Rayleigh Quotient

A problem that does not appear to have been used much in the past for studies of this sort is a Rayleigh quotient, such as that given by

$$f(\underset{\sim}{x}) = \frac{1}{2}\, \eta \underset{\sim}{x}^T N \underset{\sim}{x}$$

where

$$\eta \equiv \eta(\underset{\sim}{x}) = 2/\underset{\sim}{x}^T\underset{\sim}{x}$$

and

$$N = \text{diag}\ (1, 2, \ldots, n).$$

The solution $f(\underset{\sim}{x}^*) = 1$ occurs at $\underset{\sim}{x}^{*T} = (1, 0, \ldots, 0)$. The gradient and Hessian of this function are given by

$$\underset{\sim}{g}(\underset{\sim}{x}) = \eta \bar{N} \underset{\sim}{x}$$

and

$$H(\underset{\sim}{x}) = \eta(\overline{N} - \underset{\sim}{g}\underset{\sim}{x}^T - \underset{\sim}{x}\underset{\sim}{g}^T)$$

where $\overline{N}$ is the shifted coefficient matrix given by

$$\overline{N} \equiv N - fI.$$

For application of the planar search minimization algorithm to this problem, we are concerned with the nature of step vectors of the form (4). Note that

$$\begin{aligned}\underset{\sim}{x}^T\underset{\sim}{g} &= \eta(\underset{\sim}{x}^T N\underset{\sim}{x} - f\underset{\sim}{x}^T\underset{\sim}{x}) \\ &= 2f - 2f \\ &= 0\end{aligned}$$

so that

$$H\underset{\sim}{g} = \eta(\overline{N}\underset{\sim}{g} - \underset{\sim}{g}^T\underset{\sim}{g}\underset{\sim}{x}).$$

Since these results imply that

$$\underset{\sim}{g}^T H\underset{\sim}{g} = \eta\underset{\sim}{g}^T\overline{N}\underset{\sim}{g} = \eta^3\underset{\sim}{x}^T\overline{N}^3\underset{\sim}{x}$$

we see that the vectors $\underset{\sim}{g}$ and $H\underset{\sim}{g}$ are linearly independent unless $\underset{\sim}{x}$ is an eigenvector, in which case $\underset{\sim}{g} = \underset{\sim}{0}$.

Combining (4) with the above, we obtain the expression

$$\begin{aligned}\underset{\sim}{s} &= \eta(\beta\eta\overline{N}^2 + \alpha\overline{N} - \beta\underset{\sim}{g}^T\underset{\sim}{g}I)\underset{\sim}{x} \\ &= (\gamma_2 N^2 + \gamma_1 N + \gamma_0 I)\underset{\sim}{x},\end{aligned}$$

where $\gamma_0 = \eta(\beta(\eta f^2 - \underset{\sim}{g}^T\underset{\sim}{g}) - \alpha f) = \eta f(3\beta f - \alpha) - \beta\eta^2\underset{\sim}{x}^T N^2\underset{\sim}{x}$

$$\gamma_1 = \eta(\alpha - 2\beta\eta f)$$

$$\gamma_2 = \eta^2\beta.$$

For arbitrary i and j ($i \neq j$) we may write $\underset{\sim}{x} = \hat{\underset{\sim}{x}} + x_i\underset{\sim}{e}_i + x_j\underset{\sim}{e}_j$, where $\underset{\sim}{e}_i$ is the ith column of the identity matrix. Note that this implies $\hat{x}_i = \hat{x}_j = 0$. Note also that the ith and jth components of $N\hat{x}$ are also zero. Letting

$$\hat{N} = \sum_{0}^{2} \gamma_k N^k$$

we have

$$\underset{\sim}{s} = \hat{N}\hat{x} + (\sum_{0}^{2} \gamma_k i^k)\, x_i \underset{\sim}{e}_i + (\sum_{0}^{2} \gamma_k j^k)\, x_j \underset{\sim}{e}_j$$

Thus, we can eliminate components i and j from the next iterate vector

$$\underset{\sim}{x}' = \underset{\sim}{x} + \underset{\sim}{s}$$

by choosing α and β such that

$$\sum_{0}^{2} \gamma_k i^k = \sum_{0}^{2} \gamma_k j^k = -1.$$

Consequently, there exists a sequence of α's and β's such that the problem can be solved in no more than n/2 steps. Unfortunately, the current planar search algorithm does not produce this sequence and somewhat more than n/2 steps are taken.

The three dimensional Rayleigh quotient may be graphically illustrated by parameterizing it in terms of coordinates in a plane passing through the three unit eigenvectors of N. The result in Figure 3, shows the three stationery points at (0,0), (0,1) and (1,0).

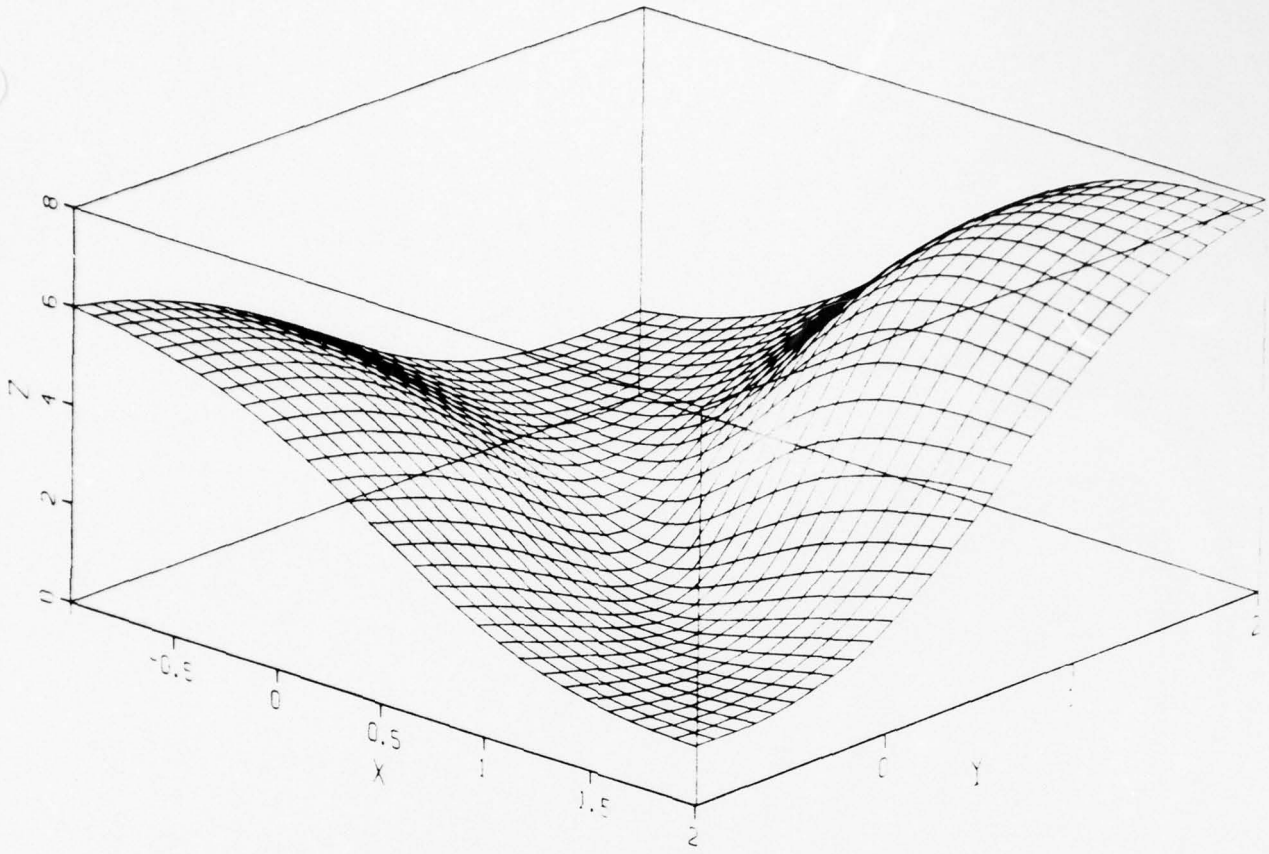


Figure 3. Illustration of a Rayleigh Quotient.

Problem 5. Quadratic form

A very simple quadratic form test problem was also included in the test series. The form of this was

$$f(\underset{\sim}{x}) = \underset{\sim}{x}^T(\underset{\sim}{b} + \frac{1}{2} H\underset{\sim}{x})$$

where

$$b_i = n + 1 - i$$

and H is diagonal with

$$H_{ii} = 2i, \quad i = 1, \ldots, n.$$

The solution $f(\underset{\sim}{x}^*) = -\frac{1}{4}\sum_{1}^{n} \frac{(n + 1 - i)^2}{i}$ occurs at $x^{*T} = (n/2, (n-1)/4, (n-2)/6, \ldots, 1/2n)$.

5.2 TEST RESULTS

Each of the 5 problems described was solved using each of the 3 methods (planar search, DFP and BFGS), utilizing each of the 3 line search techniques (cubic, extrapolated cubic and quadratic). For the planar search studies each of the 3 Hessian updates (DFP, BFGS and analytic) were used, making a total of 75 tests. The results of these tests are summarized in Tables 1 and 2.

Not all of the solutions converged within the maximum number of steps allowed. These cases are noted by an * in Tables 1 and 2. The values of the functions and their gradient norms at the end of the iteration are given in Tables 3 and 4. Zero entries in these tables correspond to converged results.

Table 1

STATISTICS FOR COMPLETE PLANAR SEARCH SOLUTIONS

PROBLEM	HESSIAN UPDATE	SEARCH	STEPS	FUNCTION EVALS.	GRADIENT EVALS.	HESSIAN -VECTOR MULTS.
QUAD.	DFP	CUBIC	26	46	46	78
		EX-CU	26	46	46	78
		QUAD	23	24	24	69
	BFGS	CUBIC	20	74	74	60
		EX-CU	20	74	74	60
		QUAD	23	24	24	69
	ANALYTIC	CUBIC	18	67	67	36
		EX-CU	18	67	67	36
		QUAD	24	25	25	48
RAYL	DFP	CUBIC	23	35	34	69
		EX-CU	153	164	163	459
		QUAD	12	13	68	36
	BFGS	CUBIC	35	46	45	105
		EX-CU	153	164	163	459
		QUAD	11	12	63	33
	ANALYTIC	CUBIC	17	24	24	34
		EX-CU	17	20	20	34
		QUAD	42	43	167	84
ROSN.	DFP	CUBIC	42	51	51	126
		EX-CU	42	53	53	126
		QUAD	28	29	129	84
	BFGS	CUBIC	36	44	44	108
		EX-CU	39	51	51	117
		QUAD	33	34	138	99
	ANALYTIC	CUBIC	21	32	32	42
		EX-CU	21	32	32	42
		QUAD	14	15	87	28
WOOD	* DFP	CUBIC	303	304	304	909
	*	EX-CU	303	304	304	909
	*	QUAD	303	304	796	909
	* BFGS	CUBIC	303	308	308	909
	*	EX-CU	303	308	308	909
	*	QUAD	303	304	681	909
	*ANALYTIC	CUBIC	303	307	306	606
	*	EX-CU	303	307	306	606
		QUAD	243	244	502	486
POWELL	* **DFP**	**CUBIC**	**303**	**304**	**304**	**909**
	*	**EX-CU**	**303**	**304**	**304**	**909**
	*	QUAD	303	304	426	909
	*ANALYTIC	CUBIC	303	304	304	606
	*	EX-CU	303	304	304	606
	*	QUAD	303	304	379	606
	* BFGS	CUBIC	303	304	304	909
	*	EX-CU	303	304	304	909
	*	QUAD	303	304	383	909

Table 2

STATISTICS FOR COMPLETE QUASI-NEWTON SOLUTIONS

		DFP				BFGS			
PROBLEM	SEARCH	STEPS	FUNCT. EVALS	GRAD. EVALS.	HESSIAN -VECTOR MULTS.	STEPS	FUNCT. EVALS.	GRAD. EVALS.	HESSIAN -VECTOR MULTS.
QUAD	CUBIC	11	22	22	22	11	22	22	22
	EX-CU	17	19	19	34	16	23	23	32
	QUAD.	10	11	21	20	11	12	22	22
RAYL.	CUBIC	21	23	23	42	20	22	22	40
	EX-CU	153	154	154	306	153	154	154	306
	QUAD.	153	154	182	306	79	80	116	158
ROSN.	CUBIC	54	69	69	108	39	57	57	78
	EX-CU	46	57	57	92	41	55	55	82
	QUAD.	28	29	135	56	35	36	160	70
WOOD	CUBIC	303	322	322	606	31	51	51	62
	EX-CU	303	354	352	606	74	92	92	148
	QUAD.	88	89	356	176	73	74	300	146
POWELL	CUBIC	94	110	110	188	38	51	51	766
	EX-CU	106	116	116	212	31	33	33	622
	QUAD.	74	75	136	148	27	28	81	544

Table 3

	PLANE SEARCH					
	DFP		BFGS		ANAL	
POWELL 303	FUNCTION	GRADIENT	FUNCTION	GRADIENT	FUNCTION	
CUBIC	8.20 E-08	4.01 E-05	3.02 E-06	4.40 E-04	1.53 E-05	
EXCUBIC	8.20 E-08	4.01 E-05	3.02 E-06	4.40 E-04	1.53 E-05	
QUAD.	3.20 E-06	3.46 E-04	6.58 E-07	1.59 E-04	6.14 E-06	
WOOD 303	FUNCTION	GRADIENT	FUNCTION	GRADIENT	FUNCTION	
CUBIC	7.87 E 00	2.85 E-01	1.31 E-02	3.87 E-01	7.81 E 00	
EXCUBIC	7.87 E 00	2.85 E-01	3.59 E 00	3.82 E 00	7.83 E 00	
QUAD	6.60 E-02	1.09 E 00	6.78 E-09	1.03 E-03	0	

Table 4

	QUASI-NEWTON			
	DFP		BFGS	
WOOD - 303	FUNCTION	GRADIENT	FUNCTION	GRADIENT
CUBIC	4.08 E-03	2.21 E 00	0	0
EXCUBIC	2.75 E-11	1.15 E-04	0	0
QUAD	0	0	0	0

6. CONCLUSIONS

We have presented a new descent method that selects its search directions from the osculating planes instead of the traditional gradients. We have described and tested a variety of implementation considerations and made extensive comparisons with two popular quasi-Newton algorithms. The major difference in implementation between the descent and the Newton (or quasi-Newton) algorithms is the use of a representation of a Hessian in the former and its inverse in the latter. When the nature of a problem is such that it is practical to work with the inverse Hessian, there appears to be little doubt that a good Newton or quasi-Newton algorithm will outperform a descent algorithm. When this is not the case; however, this new descent algorithm provides a viable alternative.

For both types of algorithm, we find that the expenditure of a moderate amount of effort in the line search, such as through the use of our quadratic search algorithm, has a very favorable effect on both convergence rate and robustness. We recommend somewhat more effort in this respect than the simple one or two step cubic interpolation that is frequently suggested.

Two studies relating to this work should be made in the future. The first should investigate the possibility of doing a plane search for the step length and direction simultaneously as opposed to obtaining the direction from the plane and doing a line search for the length. The second should test the behavior of the descent algorithm on large, sparse problems and compare it with that of a good quasi-Newton algorithm.

7. ACKNOWLEDGEMENTS

The ideas for the planar search algorithms discussed in this paper were inspired by earlier work of Dr. Eldon Hansen. He originally studied the use of (4) with Dr. H. M. C. Yee in its derived form [9] and suggested the possibility of using general α and β. Without his ideas and helpful discussions, this study would not have been pursued.

8. REFERENCES

1. Bard, Y., "Comparison of Gradient Methods for the Solution of Nonlinear Parametric Estimation Problems", SIAM J. Num. Anal., 7 (1970) 159-186.

2. Box, M. J., "A Comparison of Several Current Optimization Methods, and the Use of Transformations in Constrained Problems", Comput. J., 9 (1966) 67-77.

3. Broyden, C. G., "Quasi-Newton, or Modification Methods", in Numerical Solution of Systems of Nonlinear Algebraic Equations, (G. D. Byrne and C. A. Hall, eds.) Academic Press, New York, N.Y. (1973) 241-280.

4. Davidon, W. C., "Variable Metric Method for Minimization", AEC R&D Report. ANL-5990, (1959).

5. Dennis, J.E., Jr., and J. J. Moré, "Quasi-Newton Methods, Motivation and Theory", SIAM Review, 19, 1 (1977) 46-89.

6. Fletcher, R. and M. J. D. Powell, "A Rapidly Converged Descent Method for Minimization", Computer Jnl. 6 (1963) 163-170.

7. Himmelblau, D. M., "A Uniform Evaluation of Unconstrained Optimization Techniques", in Numerical Methods for Non-Linear Optimization (F. A. Lootsma, ed.), Academic Press, New York (1972).

8. Jensen, Paul S., "An Engineering Analysis System," Proc. 1978 ACM Conf. 1 (1978) 490-495.

9. Jensen, P. S., E. R. Hansen and H. M. C. Yee, "Numerical Techniques for Optimization and Nonlinear Equations", LMSC-D630055, Lockheed Missiles & Space Corp., Palo Alto, CA (29 June 1978).

10. Lawson, C. L., R. J. Hanson, D. R. Kincaid, and F. T. Krogh, "Basic Linear Algebra Subprograms for FORTRAN Usage", CNA-124, TR-72, Center for Numerical Analysis, University of Texas, Austin (1977).

11. Matthies, H. and G. Strang, "The Solution of Nonlinear Finite Element Equations", MIT Report, (1978).

12. Powell, M. J. D., "An Iterative Method for Finding Stationary Values of a Function of Several Variables", Computer Jnl., 5 (1962) 147.

13. Rheinboldt, W. C., Methods for Solving Systems of Nonlinear Equations, SIAM, Philadelphia (1974).

14. Rosenbrock, H. H., "An Automatic Method for Finding the Greatest or Least Value of a Function", Computer Jnl., 3 (1960) 175-184.

15. Sargent, R. W. H. and D. J. Sebastian, "Numerical Experience with Algorithms for Unconstrained Minimization", in Numerical Methods for Non-linear Optimization (F. A. Lootsma, ed.), Academic Press, New York (1972).

16. Schubert, L. K., "Modification of a Quasi-Newton Method for Nonlinear Equations with a Sparse Jacobian", Math Comp, 24 (1970) 27-30.

APPENDIX

CRATER — AN INTERACTIVE PROGRAM FOR OPTIMIZATION STUDIES

1. INTRODUCTION

CRATER is an interactive computer program designed for the convenient solution of optimization problems using various algorithms. Convenience of operation is achieved by means of a problem oriented language that provides prompting whenever a user needs it but does not burden the experienced user with unnecessary questions. Provision is also made for inspecting intermediate results and changing various control parameters such as the print control during an execution. The user can even stop a run and re-initialize it at a different starting point if so desired. If a user should become confused at some point in an execution, he can type HELP for assistance or STOP to quit. One or several commands may be given on one line. Finally, CRATER is forgiving. A user needs only to get the first four letters of each command spelled correctly. If he fails at that, CRATER will politely ask him to repeat.

Convenience of problem setup is achieved through program modularity. Each routine used by CRATER is designed to serve a specific and rather isolated prupose, and is well documented internally. Three specific routines with which a user is particularly concerned are: USRFNL, USRGRD and USRHES, which define the functional, gradient and (optionally) Hessian on which the optimization procedure is to be applied. The starting point can be set at zero or at a variety of random points using internal options or it can be keyed or read in (free field).

For transportability, the code is essentially written in standard FORTRAN 66. We say "essentially" because, in fact, it is written in a special master source code (MSC) form that includes directives for special, machine dependent, characteristics along with the FORTRAN code and resides in a library (called EASY) developed at Lockheed Missiles and Space Co. [8]. EASY is maintained by a reasonably sophisticated librarian program that, among other things, is capable of interpreting the special MSC directives and producing source code that is immediately operational on one of several specific computing environments.

Another feature of MSC is that it includes a structured program documentation system that imposes a significant amount of discipline upon developers. The librarian checks the documentation supplied with MSC codes and complains if it feels that the documentation is inadequate. It also extracts and tabulates a copy of the documentation in order to facilitate library searches and the construction of program documents such as this. Thus, much of the documentation appearing in the subsequent Sections came directly from the CRATER program itself.

2. PROBLEM ORIENTED LANGUAGE

The commands that a user imposes upon CRATER are organized in a simple tree structure as illustrated in Figure 1.

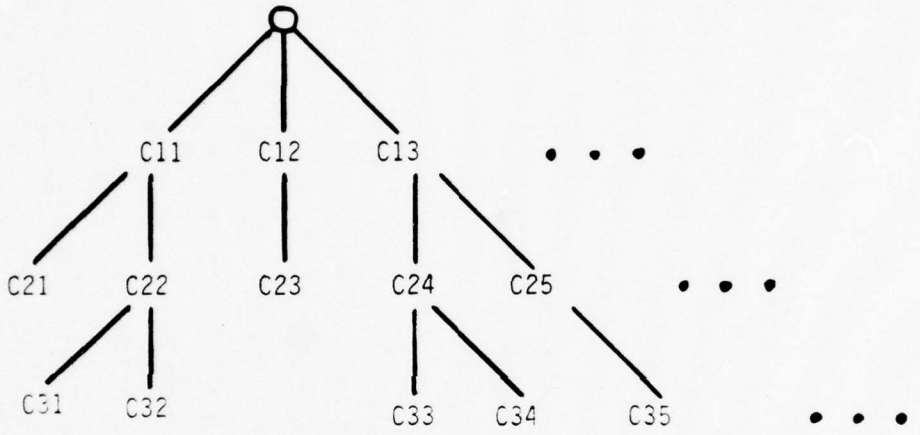


Figure 1. Command Tree Illustration

At level 1, CRATER will list the command options C11, C12, . . . from which the user should select one, C13 say. CRATER then lists the level 2 command options under the level 1 command selected, e.g., C24 and C25, from which the user should select one, and so forth. When the user makes a selection at the end of a branch in the tree, the effect is recorded in the state variables of CRATER. Then CRATER moves back to the next higher level and repeats the command options at that level. The user may then select another command and follow its branch to the end. If the user does not wish to invoke any command options at the current position in the tree, he may enter BACKUP to move to the next higher level in the command tree.

After some practice, a user will tire of all the prompting (listing of command options). To avoid it, he may simply enter a string of commands on one or more lines using the line continuation flag $ at the end of each line.

For example, a user could invoke every command illustrated in Figure 1 by entering the following

C11, C21, C22, C31, C32, BACK, BACK, $
C12, C23, $
C13, C24, C33, C34, BACK, C25, C35, STOP

with only a single level 1 prompting of C11, C12, and C13. Note that a command may be repeated at various times as long as the position in the tree, at the time the command is issued, is appropriate.

There are three "universal" commands that may be invoked at any point in the tree, viz: BACKUP, STOP and HELP. We have discussed the BACKUP (or BACK for short) command above. The STOP command simply terminates a run and HELP will provide helpful instructions relating to the operation of CRATER.

We define two classes of command, viz: transitional and terminal. Transitional commands have "offspring" commands at the next level of the tree structure where-as terminal commands do not. Thus, the terminal commands reside at the ends of branches of the tree structure. A terminal command may require numerical data to follow it but a transitional command can only be followed by other commands in the command stream.

The actual words used as commands are simply established in a data list and can be readily changed to suit a users taste. See arrays COMMND and CMNDPT in block data deck LOP91 of the code for CRATER. The i^{th} word, CMNDPT(i), in the pointer list points to the i^{th} command word in list COMMND.

The command tree structure is also specified by data (see array TREE) in LOP91. However, the organization of TREE is closely related to the organization of the code itself and, consequently, more care must be exercised in modifying TREE. Users who are quite familiar with the code can, in a straight forward fashion, readily add or delete branches on the tree. A triple (O, S, W) is associated with each node of the tree where

O is the "oldest offspring" node at the next level

S is { the next younger "sibling" node at the same level if positive or
     { the parent node at the previous level if negative

W is the index of the command word associated with the node.

Up to this point, we have described the syntax of a general POL organized in a tree structure. The semantics of the specific POL implemented in CRATER remain to be discussed. Before doing this, however, it is helpful to discuss the organization of CRATER in order to establish the setting in which the POL is couched.

3. PROGRAM ORGANIZATION

CRATER has a relatively small executive program that oversees the optimization process. It is supported by 20 subroutines that carry out specific tasks. These, in turn, utilize 4 utility subroutines from the Basic Linear Algebra Library [10] and 10 other utilities all of which are provided with CRATER. The program organization is illustrated in Table 1.

Table 1. Program Organization.
Calling routines appear directly above and to the left of the list of called routines.

```
CRATER          -  EXECUTIVE PROCESSOR
   LOP05        -  USER COMMAND INTERPRETER
   LOP10        -  INITIALIZATION MANAGER
      LOP05     -  USER COMMAND INTERPRETER
      LOP11     -  INITIALIZE POSITION VECTOR
      LOP12     -  INITIALIZE HESSIAN (JACOBIAN) MATRIX
         LOP23  -  TRUE HESSIAN EVALUATION MANAGER (OPTIONAL)
           USRHES- USER SUPPLIED HESSIAN EVALUATION (OPTIONAL)
      LOP21     -  FUNCTIONAL EVALUATION
         USRFNL -  USER SUPPLIED FUNCTIONAL EVALUATION
      LOP22     -  GRADIENT EVALUATION
         USRGRD -  USER SUPPLIED GRADIENT EVALUATION
   LOP20        -  ITERATION STEP MANAGER (ONE STEP PER CALL)
      LOP21     -  FUNCTIONAL EVALUATION
         USRFNL -  USER SUPPLIED FUNCTIONAL EVALUATION
      LOP22     -  GRADIENT EVALUATION
         USRGRD -  USER SUPPLIED GRADIENT EVALUATION
      LOP23     -  TRUE HESSIAN CALCULATION (OPTIONAL)
         USRHES -  USER SUPPLIED HESSIAN EVALUATION (OPTIONAL)
      LOP24     -  HESSIAN UPDATE ROUTINE (RANK 2)
         LOP12  -  INITIALIZE THE HESSIAN
      LOP26     -  STEP DIRECTION DETERMINATION
      LOP27     -  STEP LENGTH DETERMINATION
   LOP30        -  CONVERGENCE EVALUATION MANAGER
   LOP40        -  RESULTS DISPLAY MANAGER
```

3.1 Purposes Served by CRATER Routines

In this section we provide a brief statement of purpose for each routine in CRATER and the utility routines supporting it. This section is intended for quick reference only. More documentation detail is provided in Sections 3.2 and 3.3.

```
PURPOSE.....CRATER
    LABORATORY PROCESSOR FOR STUDYING LOCAL OPTIMIZATION

PURPOSE.....GTF
    GLOBAL MINIMIZATION TEST FUNCTION EVALUATION

PURPOSE.....HORNER
    EVALUATE A DEGREE (N-1) POLYNOMIAL WITH COEFFS. C AT X.

PURPOSE.....LOP05
    ACCEPT AND INTERPRET COMMANDS FOR LOCAL OPTIMIZATION

PURPOSE.....LOP06
    GET ONE USER INPUT VARIABLE OR COMMAND

PURPOSE.....LOP10
    CARRY OUT INITIALIZATION OPERATIONS FOR PROGRAM CRATER

PURPOSE.....LOP11
    INITIALIZE THE POSITION VECTOR FOR SELECTED ANALYSIS

PURPOSE.....LOP12
    INITIALIZE THE HESSIAN (JACOBIAN) MATRIX

PURPOSE.....LOP20
    CARRY OUT ONE OPTIMIZATION ITERATION STEP

PURPOSE.....LOP21
    EVALUATE THE SELECTED REAL-VALUED FUNCTIONAL

PURPOSE.....LOP22
    EVALUATE THE SELECTED GRADIENT FUNCTION

PURPOSE.....LOP23
    EVALUATE THE HESSIAN OF THE SELECTED FUNCTIONAL

PURPOSE.....LOP24
    UPDATE THE CURRENT APPROXIMATE HESSIAN MATRIX (RANK 2)

PURPOSE.....LOP26
    DETERMINE A STEP DIRECTION S IN THE U-V PLANE

PURPOSE.....LOP27
    MOVE POSITION X A REASONABLE DISTANCE ALONG S

PURPOSE.....LOP30
    CHECK THE CONVERGENCE AND SAVE INTERMEDIATE RESULTS

PURPOSE.....LOP40
    MANAGE ALL RESULT DISPLAY OPERATIONS FOR CRATER

PURPOSE.....LOP81
    MATRIX-VECTOR MULTIPLY,  U = H*V   (H IS NORMALLY A HESSIAN)

PURPOSE.....LOP91
    PROVIDE TEXT DATA FOR INTERACTIVE COMMUNICATION
```

```
PURPOSE.....LOP92
    PROVIDE DEFAULT VALUES FOR CONTROL VARIABLES OF CRATER

PURPOSE.....LOP94
    HOLD COMMON DECLARATIONS GERMANE TO MAN-MACHINE INTERACTION

PURPOSE.....LOP95
    HOLD COMMON DECLARATIONS GERMANE TO OPERATION OF CRATER.

PURPOSE.....LOP98
    DISPLAY THE CONTROL AND COMMUNICATION DATA (PARAMETERS)

PURPOSE.....LOP99
    GENERAL PURPOSE ERROR HANDLING ROUTINE

PURPOSE.....USRFNL
    CALCULATE A SPECIAL FUNCTIONAL FOR USE BY CRATER
```

3.2 Abstracts of CRATER Routines

In this section we provide brief discussions on how each routine in CRATER serves its purpose. Information on exactly how to use each routine is provided in Section 3.3.

```
ABSTRACT....CRATER
    FACILITIES FOR INTERACTIVELY SOLVING SEVERAL BUILT-IN TEST
PROBLEMS USING ANY OF SEVERAL BUILT-IN ALGORITHMS AND INITIAL
CONDITIONS ARE PROVIDED FOR EMPIRICAL STUDIES.  ADDITIONALLY,
LINK MECHANISMS FOR ANALYZING OTHER, USER SUPPLIED, PROBLEMS
ARE PROVIDED.
                    ------------
                    PROGRAM FLOW
                    ------------
INDENTATION IS USED TO INDICATE SUBROUTINE RELATIONSHIP

CRATER              -  EXECUTIVE PROCESSOR
   LOP05            -  USER COMMAND INTERPRETER
   LOP10            -  INITIALIZATION MANAGER
      LOP05         -  USER COMMAND INTERPRETER
      LOP11         -  INITIALIZE POSITION VECTOR
      LOP12         -  INITIALIZE HESSIAN (JACOBIAN) MATRIX
         LOP23      -  TRUE HESSIAN EVALUATION MANAGER (OPTIONAL)
           USRHES-  -  USER SUPPLIED HESSIAN EVALUATION (OPTIONAL)
      LOP21         -  FUNCTIONAL EVALUATION
         USRFNL     -  USER SUPPLIED FUNCTIONAL EVALUATION
      LOP22         -  GRADIENT EVALUATION
         USRGRD     -  USER SUPPLIED GRADIENT EVALUATION
   LOP20            -  ITERATION STEP MANAGER (ONE STEP PER CALL)
      LOP21         -  FUNCTIONAL EVALUATION
         USRFNL     -  USER SUPPLIED FUNCTIONAL EVALUATION
      LOP22         -  GRADIENT EVALUATION
         USRGRD     -  USER SUPPLIED GRADIENT EVALUATION
      LOP23         -  TRUE HESSIAN CALCULATION (OPTIONAL)
         USRHES     -  USER SUPPLIED HESSIAN EVALUATION (OPTIONAL)
      LOP24         -  HESSIAN UPDATE ROUTINE (RANK 2)
         LOP12      -  INITIALIZE THE HESSIAN
      LOP26         -  STEP DIRECTION DETERMINATION
      LOP27         -  STEP LENGTH DETERMINATION
   LOP30            -  CONVERGENCE EVALUATION MANAGER
   LOP40            -  RESULTS DISPLAY MANAGER

                    ------------------------------------
                    PARAMETERS  (LABELED COMMON /LOPPRM/)
                    ------------------------------------
 1   1  NPARM      TOTAL NO. OF PARAMETERS   (69)
 2   2  NIPARM     NO. OF INTEGER PARAMETERS (39)
 3   3  NRPARM     NO. OF REAL PARAMETERS    (27)
** INTEGER PARAMETERS **
 4   1  STCASE     OPTION FOR INITIAL POSITION VECTOR
 5   2  FNCASE     OPTION FOR FUNCTIONAL
 6   3  IHCASE     OPTION FOR INITIAL HESSIAN
 7   4  JCCASE     OPTION FOR HESSIAN (JACOBIAN) UPDATE PROCEDURE
 8   5  ALCASE     OPTION FOR ALGORITHM TO BE USED
 9   6  PRCASE     OPTION FOR PRINTED OUTPUT VOLUME (SEE BELOW)
10   7  PLCASE     OPTION FOR PLOTTED OUTPUT
11   8  LSCASE     OPTION FOR LINE SEARCH TECHNIQUE
12   9  X2CASE     OPTION FOR     * NOT ASSIGNED *
13  10  X3CASE     OPTION FOR     * NOT ASSIGNED *
14  11  NV         PROBLEM SIZE (NO. INDEPENDENT VARIABLES)
15  12  MXITR      MAX. ITERATION STEPS TO BE TAKEN
```

```
16 13  NLOP27    MAX. ITERATIONS TO BE USED FOR STEP LENGTH CALC.
17 14  LIMX1     LIMIT PARAMETER  * NOT ASSIGNED *
18 15  LIMX2     LIMIT PARAMETER  * NOT ASSIGNED *
19 16  NFEV      CURRENT NO. OF FUNCTIONAL EVALUATIONS
20 17  NGEV      CURRENT NO. OF GRADIENT EVALUATIONS
21 18  NHEV      CURRENT NO. OF HESSIAN EVALUATIONS
22 19  NHUP      CURRENT NO. OF HESSIAN UPDATES
23 20  NHMP      CURRENT NO. OF HESSIAN-VECTOR MULTIPLIES
24 21  NITR      CURRENT NO. OF ITERATIONS
25 22  I1STAT    NO. OF DEGENERATE PLANES (SEE LOP28)
26 23  I2STAT    OPERATIONAL STATISTIC PARAMETER  * NOT ASSIGNED *
27 24  I3STAT    OPERATIONAL STATISTIC PARAMETER  * NOT ASSIGNED *
28 25  I4STAT    OPERATIONAL STATISTIC PARAMETER  * NOT ASSIGNED *
29 26  I5STAT    OPERATIONAL STATISTIC PARAMETER  * NOT ASSIGNED *
30 27  I6STAT    OPERATIONAL STATISTIC PARAMETER  * NOT ASSIGNED *
31 28  LBCOMX    BLANK COMMON LOC. OF X (POSITION)
32 29  LBCOMG    BLANK COMMON LOC. OF G (GRADIENT)
33 30  LBCOMH    BLANK COMMON LOC. OF H (HESSIAN)
34 31  LBCOM1    BLANK COMMON LOC. OF STEP VECTOR
35 32  LBCOM2    BLANK COMMON LOC. OF WORKSPACE VECTOR
36 33  LBCOM3    BLANK COMMON LOC. OF WORKSPACE VECTOR
37 34  LBCOM4    BLANK COMMON LOC. OF WORKSPACE VECTOR
38 35  LBCOM5    BLANK COMMON LOC. OF FNL VALUES
39 36  LBCOM6    BLANK COMMON LOC. OF GRADIENT NORMS
40 37  LBCOM7    BLANK COMMON LOC. OF COS(THETA(G,S))
41 38  LBCOM8    BLANK COMMON LOC. OF STEP NORMS
42 39  LBCOM9    BLANK COMMON LOC. OF   * NOT ASSIGNED *
** REAL PARAMETERS **
43  1  ACCEPT    CONVERGENCE CRITERIA (2***-K, WHERE K BITS ARE FREE)
44  2  ALPHA     COEFFICIENTS FROM LOP16 FOR DETERMINING
45  3  BETA         THE CURRENT STEP DIRECTION
46  4  TOLPAC    TOLERENCE MULTIPLIER CUBIC LINE SEARCH - LOP27
47  5  FCONV     ANTICIPATED CONVERGED FUNCTION VALUE
48  6  RLPRM3    * NOT ASSIGNED *
49  7  RLPRM4    * NOT ASSIGNED *
50  8  RLPRM5    ACCEPTANCE CRITERIA FOR QUADRATIC LINE SEARCH - LOP27
51  9  RLPRM6    POSITION NORM (IF CALCULATED - LOP30)
52 10  F         CURRENT AND TWO PREVIOUS VALUES
53 11  F1           OF THE FUNCTIONAL
54 12  F2
55 13  GN        CURRENT AND TWO PREVIOUS VALUES
56 14  GN1          OF THE GRADIENT NORM
57 15  GN2
58 16  GS        CURRENT AND TWO PREVIOUS VALUES
59 17  GS1          GRADIENT-STEP INNER PRODUCT
60 18  GS2
61 19  SN        CURRENT AND TWO PREVIOUS VALUES
62 20  SN1          OF THE STEP VECTOR NORM
63 21  SN2
64 22  X1H       CURRENT AND TWO PREVIOUS VALUES
65 23  X1H1         * NOT ASSIGNED *
66 24  X1H2
67 25  X2H       CURRENT AND TWO PREVIOUS VALUES
68 26  X2H1         * NOT ASSIGNED *
69 27  X2H2

ABSTRACT....LOP05
    THE INDEX OF THE USER COMMAND IN THE COMMAND TREE (AT THE CURRENT
```

LEVEL) IS DETERMINED FOR THE CALLING PROGRAM. THIS INDEX IS ASSUMED TO INDICATE TO THE CALLING PROGRAM WHAT IT IS EXPECTED TO DO.

ABSTRACT....LOP06
A USER MAY INPUT SEVERAL ITEMS ON ONE LINE. THIS INFORMATION IS STACKED IN A COMMON ARRAY /LOPINP/ AND THE ENTRIES ARE PULLED OFF BY LOP06, ONE FOR EACH CALL. IF THE STACK IS EMPTY, LOP06 CALLS FOR MORE INPUT.

ABSTRACT....LOP10
THIS ROUTINE CARRIES OUT INITIALIZATION COMMANDS ISSUED BY A USER IN ACCORDANCE WITH THE COMMAND LANGUAGE TREE OF THE CRATER OPTIMIZATION PROGRAM.

ABSTRACT....LOP11
INITIALIZE THE POSITION VECTOR BY VARIOUS MEANS DEPENDING UPON THE PROBLEM BEING SOLVED. PROBLEM SPECIFICATION IS DETERMINED BY THE INDEX PARAMETER STCASE.

STCASE	INITIALIZATION
1	STANDARD FOR ROSENBROCK BANANA
6	RANDOM
7	ZERO

ABSTRACT....LOP12
INITIALIZE THE HESSIAN MATRIX ACCORDING TO THE CRITERION GIVEN IN PARAMETER IHCASE.

IHCASE	INTERPRETATION
1	ANALYTICAL HESSIAN (ROUTINE LOP23)
2	IDENTITY MATRIX
3	READ FROM AUX. STORAGE

ABSTRACT....LOP20
DETERMINE THE STEP DIRECTION AND LENGTH, MOVE (THE INDEPENDENT VARIABLE VECTOR) X BY THAT STEP, AND ACCORDINGLY UPDATE THE HESSIAN, GRADIENT AND FUNCTIONAL VALUES. THE HESSIAN IS UPDATED EITHER BY ANALYTIC CALCULATION (IF PARAMETER UCASE=2) OR BY A RANK TWO UPDATE FORMULA (OTHERWISE).

ABSTRACT....LOP21
THIS ROUTINE PROVIDES A COLLECTION OF TEST FUNCTIONALS FOR USE IN STUDYING THE BEHAVIOR OF LOCAL OPTIMIZATION ALGORITHMS. THIS DECK IS DESIGNED WITH TWO INTERNAL SUBROUTINES (IN THE EASY MSC SENSE) THAT EVALUATE THE CORRESPONDING GRADIENTS AND HESSIANS. THEY ARE CALLED LOP22 AND LOP23.

CURRENTLY, THE FUNCTIONALS DEFINED ARE AS FOLLOWS

CASE	FUNCTIONAL
1	ROSENBROCK BANANA (STANDARD)
2	SIMPLE ELLIPSOID
3	SPECIAL (USER SUPPLIED)
4	POWELL FUNCTION

ABSTRACT....LOP22
SEE ABSTRACT FOR LOP21

ABSTRACT....LOP23
SEE ABSTRACT FOR LOP21

ABSTRACT....LOP24
THE FULL HESSIAN OR INVERSE HESSIAN IS UPDATED USING THE RANK 2 DFP (DAVIDON-FLETCHER-POWELL) FORMULAS. SEE, E.G., J.E.DENNIS,JR. AND J.J.MORE, QUASI-NEWTON METHODS, SIAM REVIEW, 19,1 (1977) 46,89.

IN PRACTICAL APPLICATIONS (WITH SPARSE H), H SHOULD NOT BE DIRECTLY MODIFIED. INSTEAD, THE VECTORS S AND H (AND SCALERS A AND B) SHOULD BE RETAINED FOR USE IN SUBSEQUENT CALCULATIONS OF H*X FOR ARBITRARY X.

ABSTRACT....LOP26
DETERMINE S = AL*U + BT*V SUCH THAT THE FUNCTIONAL EXTREMUM OCCURS AT A POINT NEARER TO X+S THAN X. PRESENT ALGORITHM APPLIES TO U = G AND V = H*G. WHERE G IS THE CURRENT GRADIENT.

ABSTRACT....LOP27
FOR THE CUBIC SEARCH, THE STEP LENGTH IS ACCEPTED WHEN THE FUNCTION VALUE DOES NOT EXCEED TOLFAC*F0, WHERE TOLFAC IS A PARAMETER AND F0 IS THE INITIAL FUNCTION VALUE. FOR QUADRATIC SEARCH, A ZERO OF THE PARTIAL DERIVATIVE OF THE FUNCTION IN THE STEP DIRECTION IS SOUGHT UNDER THE CONTROL OF PARAMETER RLPRMS.

ABSTRACT....LOP30
DESIRED STATISTICS ARE RECORDED AND A CONVERGENCE CHECK IS MADE. THE CONVERGENCE CRITERION IS THAT THE NORM OF THE GRADIENT IS NEGLIGIBLE IN COMPARISON WITH THE ACCEPTANCE PARAMETER ACCEPT.

ABSTRACT....LOP40
USES THE CRATER COMMAND INTERPRETER TO DECIPHER USER COMMANDS AND CARRIES OUT THOSE COMMANDS.

ABSTRACT....LOP81
SPECIAL MATRIX-VECTOR MULTIPLY ROUTINE FOR LOCAL OPTIMIZATION. THIS ROUTINE TAKES INTO ACCOUNT THE SPECIAL STRUCTURE OF H USED IN OPTIMIZATION, I.E., H IS A BASIC (SPARSE) MATRIX WITH A SEQUENCE OF ASSOCIATED RANK-1 UPDATE VECTORS.

CURRENTLY, H IS JUST A FULL MATRIX WITH THE UPDATES EXPLICITLY INCLUDED.

ABSTRACT....LOP91
THIS ESTABLISHES THE TEXT FOR INTERACTIVE COMMUNICATION BETWEEN THE OPTIMIZATION PROGRAM CRATER AND A USER.

ABSTRACT....LOP92
THIS ESTABLISHES ALL OF THE DEFAULT VALUES FOR CONTROL VARIABLES USED IN PROGRAM CRATER.

ABSTRACT....LOP98
SYSTEMATICALLY DISPLAYS THE CONTENTS OF THE KEY LABELLED COMMON BLOCKS.

ABSTRACT....LOP99
AN INTERNAL TABLE IS MAINTAINED WHICH INDICATES WHAT ACTION IS TO BE TAKEN FOR EACH OF THE LOP ROUTINES. THIS ROUTINE CARRIES OUT THE INDICATED ACTION

3.3 Directions for Using CRATER Routines

In this section we specify the parameters and other considerations involved in using each routine in CRATER and the utility routines supporting it.

```
USAGE ......CRATER
    SIMPLY EXECUTE CRATER AND INTERACTIVELY SUPPLY COMMANDS AND
DATA AS REQUESTED.  THE TREE STRUCTURE OF THE COMMAND LANGUAGE
IS EXPLAINED IN RESPONSE TO THE HELP COMMAND.

    THE COMMANDS APPROPRIATE FOR THE CURRENT POSITION IN THE COMMAND
TREE ARE DISPLAYED AND THE USER HAS THE FOLLOWING OPTIONS
  1. TYPE ONE OF THE COMMANDS.
  2. TYPE THE WORD HELP FOR GUIDANCE.
  3. TYPE THE WORD BACKUP TO GET TO THE PREVIOUS POSITION IN THE TREE.
  4. TYPE THE WORD STOP TO TERMINATE THE EXECUTION.

    IN ORDER TO DO LOCAL MINIMIZATION ON A USER SUPPLIED FUNCTIONAL,
THE USER MUST SUPPLY SUBROUTINES WITH THE FOLLOWING CALLING SEQUENCES

          CALL USRFNL(N,F,X,W)        - FUNCTIONAL EVALUATION
          CALL USRGRD(N,G,X,W)        - GRADIENT EVALUATION
          CALL USRHES(N,H,X,W)        - HESSIAN EVALUATION (OPTIONAL)

WHERE  F  IS THE SCALAR FUNCTIONAL VALUE AT POSITION X
       G  IS THE VECTOR GRADIENT AT POSITION X  (LENGTH N)
       H  IS THE N BY N HESSIAN MATRIX AT POSITION X
       N  IS THE NO. INDEPENDENT VARIABLES IN THE FUNCTIONAL
       X  IS THE CURRENT POSITION VECTOR (INDEPENDENT VARIABLE)
       W  IS THE WORKSPACE (SCRATCH) VECTOR (LENGTH N)

   THE DETAILS OF THE PROGRAM FLOW ARE ALMOST ENTIRELY CONTROLLED BY
VALUES ASSIGNED TO PARAMETERS IN LABELLED COMMON (SEE ABSTRACT).
THESE MAY BE CHANGED BY SPECIFYING THE TYPE, INDEX AND VALUE, E.G.,
INTEGER,2,3 TO SET THE SECOND INTEGER TO THE VALUE 3.  THIS TYPE OF
WORK IS DONE AT THE PARAMETER NODE OF THE COMMAND TREE.

   COMMON VARIABLES AND ARRAYS ARE ESTABLISHED IN EXTERNAL PROCED-
URES(COMDECKS OR INCLUDE DECKS) LOP94 AND LOP95, AND IN EASY PROCED-
URE NITNOT.  THESE PROCEDURES MUST BE MADE AVAILABLE FOR THE COMP-
ILATION OF THE REST OF THE CRATER ROUTINES.

   THE AMOUNT OF PRINTOUT IS CONTROLLED BY A PRINT LEVEL ESTABLISHED
BY MEANS OF A PRINT COMMAND.  EACH LEVEL (0 TO 10) INCLUDES THE OUTPUT
OF LOWER LEVELS.  A ROUGH DESCRIPTION OF THE OUTPUT INTRODUCED AT
EACH LEVEL IS GIVEN BELOW.  THE NUMBERS IN PARENTHESES INDICATE
FROM WHICH ROUTINE THE OUTPUT COMES.

 0. SUMMARY DATA (40) AND FATAL ERRORS (99)
 1. FINAL POSITION (SOLUTION) (40), NOTE NONDESCENT STEPS (27)
 2. CONVERGENCE HISTORY (40), NONCONVERGED STEP MULTIPLIERS (27)
 3. NOTE OCCURRENCE OF COMPLEX ROOTS IN QUADRATIC LINE SEARCH (27)
 4. DISPLAY INITIAL POSITION VECTOR (11)
 5.
 6. FUNCTIONAL VALUE AS CALCULATED (21), DEGENERATE PLANES (26)
 7. GRADIENT (22) AND POSITION (20) VECTORS AS CALCULATED
 8. STEP VECTOR AS CALCULATED (26), ERROR ACTION CONTROLS (98)
 9. HESSIAN AS CALCULATED (23), COMMAND DATA (98)
10. LINE SEARCH DETAIL (27), CONVERSATIONAL DATA (98)

USAGE ......LOP05
```

```
    CALL LOP05(INDEX)
ARG.    TYPE    PURPOSE
INDEX   INT.    INDEX OF THE CURRENT COMMAND AMONG THOSE APPROPRIATE
                FOR THE CURRENT POSITION IN THE COMMAND TREE. (OUTPUT)
                ( = 0  IF USER RESPONSE IS BACKUP)
                INPUT INDEX=-1 FORCES AN AUTOMATIC BACKUP OPERATION

    THE COMMANDS APPROPRIATE FOR THE CURRENT POSITION IN THE COMMAND
TREE ARE DISPLAYED AND THE USER HAS THE FOLLOWING OPTIONS
  1. TYPE ONE OF THE COMMANDS.
  2. TYPE THE WORD HELP FOR GUIDANCE.
  3. TYPE THE WORD BACKUP TO GET TO THE PREVIOUS POSITION IN THE TREE.
  4. TYPE THE WORD STOP TO TERMINATE THE EXECUTION.

USAGE ......LOP06
    CALL LOP06(IVAR,RVAR,TYPE)
ARG.   TYPE    PURPOSE
IVAR   INT.    IF INPUT IS ALPHA OR INTEGER, IT IS RETURNED HERE
RVAR   REAL    IF INPUT IS REAL, IT IS RETURNED HERE
TYPE   INT.    -1,ALPHA    0,INT.    1,REAL        (OUTPUT)

USAGE ......LOP10
    CALL LOP10      (NO ARGUMENTS)

    REACTS TO INITIALIZATION COMMANDS FROM A USER

USAGE ......LOP11
    CALL LOP11(N,X)
ARG.   TYPE    PURPOSE
 N     INT.    NO. OF INDEPENDENT VARIABLES (SYSTEM SIZE)
 X     REAL    POSITION VECTOR TO BE INITIALIZED

USAGE ......LOP12
    CALL LOP12(N,H,X,W)
ARG.   TYPE    PURPOSE
 N     INT.    DIMENSION OF THE HESSIAN
 H     REAL    HESSIAN MATRIX (N BY N)
 X     REAL    POSITION VECTOR (INDEPENDENT VARIABLE)        LENGTH N
 W     REAL    WORKSPACE VECTOR                              LENGTH N

USAGE ......LOP20
    CALL LOP20(N,X,G,H,W)
ARG.   TYPE    PURPOSE
 N     INT.    PROBLEM SIZE (NO. INDEPENDENT VARIABLES)
 X     REAL    CURRENT POSITION VECTOR              (LENGTH N)
 G     REAL    CURRENT GRADIENT VECTOR              (LENGTH N)
 H     REAL    CURRENT HESSIAN MATRIX               (N BY N)
 W     REAL    WORK SPACE MATRIX                    (N BY 4)

USAGE ......LOP21
    CALL LOP21(N,FN,X,W)

ARG.    TYPE    PURPOSE
 N      INT.    NO. OF INDEPENDENT VARIABLES (DIMENSION OF DOMAIN)
 FN     REAL    RESULTING FUNCTION VALUE
 X      REAL    INDEPENDENT VARIABLE VECTOR (DIMENSION N)
 W      REAL    WORK SPACE (DIMENSION N)
```

```
USAGE ......LOP22
    CALL LOP22(N,G,X,W)
ARG.    TYPE    PURPOSE
 N      INT.    LENGTH OF G, X AND W  (PROBLEM DIMENSION)
 G      REAL    RESULTING GRADIENT VECTOR
 X      REAL    INDEPENDENT VARIABLE VECTOR
 W      REAL    WORK SPACE (LENGTH N)

USAGE ......LOP23
    CALL LOP23(N,H,X,W)
ARG.    TYPE    PURPOSE
 N      INT.    NO. OF INDEPENDENT VARIABLES
 H      REAL    N BY N HESSIAN MATRIX (OUTPUT)
 X      REAL    INDEPENDENT VARIABLE VECTOR
 W      REAL    WORK SPACE OF LENGTH N

USAGE ......LOP24
    CALL LOP24(N,H,S,G,W)
ARG.    TYPE    PURPOSE
 N      INT.    PROBLEM SIZE (NO. INDEPENDENT VARIABLES)
 H      REAL    N BY N, FULL HESSIAN MATRIX
 S      REAL    STEP VECTOR (LENGTH N)
 G      REAL    CURRENT GRADIENT VECTOR (IMMEDIATELY REPLACED BY G-W)
 W      REAL    PREVIOUS GRADIENT VECTOR AND WORK SPACE (SET TO H*G)

USAGE ......LOP26
    CALL LOP26(N,S,U,V,H)
ARG.    TYPE    PURPOSE
 N      INT.    PROBLEM SIZE
 S      REAL    STEP VECTOR PRODUCED HERE FOR NEXT APPROX.   (LENGTH N)
 U      REAL    COMPONENT OF PLANE ON WHICH TO SEARCH FOR S  (LENGTH N)
 V      REAL    COMPONENT OF PLANE ON WHICH TO SEARCH FOR S  (LENGTH N)
 H      REAL    N BY N HESSIAN MATRIX (CURRENTLY FULL)

USAGE ......LOP27
    CALL LOP27(N,X,S,G,W)
ARG.    TYPE    PURPOSE
 N      INT.    PROBLEM DIMENSION
 X      REAL    CURRENT POSITION (INDEPENDENT VARIABLE)   (LENGTH N)
 S      REAL    STEP DIRECTION FOR NEW POSITION           (LENGTH N)
 G      REAL    GRADIENT VECTOR
 W      REAL    WORK SPACE                                (LENGTH N*2)

    THE TECHNIQUE FOR THE LINE SEARCH IS CONTROLLED BY THE INTEGER
 PARAMETER LSCASE AS FOLLOWS:
          I 1 - CUBIC (MY FORMULA)
          I 2 - CUBIC (MY FORMULA) WITH 1ST POINT EXTRAPOLATION
LSCASE = <  3 - QUADRATIC
          I 4 - CUBIC (DAVIDON FORMULA)  WITH 1ST POINT EXTRAPOLATION
          I 5 - CUBIC (DAVIDON FORMULA)

    THIS ROUTINE USED PARAMETERS ALPHA, TOLFAC AND RLPRM5 ALSO
 (SEE THE ABSTRACT).

USAGE ......LOP30
    CALL LOP30(CONV)
ARG.    TYPE    PURPOSE
CONV    LOG.    SET TRUE IF AND ONLY IF THE PROCESS HAS CONVERGED
                AS REQUIRED

USAGE ......LOP40
    CALL LOP40   (NO ARGUMENTS)

RESPONDS TO DISPLAY COMMANDS ISSUED BY THE USER

USAGE ......LOP81
    CALL LOP81(N,U,H,V)
ARG.    TYPE    PURPOSE
 N      INT.    DIMENSION OF SYSTEM
```

```
 U      REAL   VECTOR RESULT,  U = H*V
 H      REAL   N BY N MATRIX (HESSIAN) FOR MULTIPLICATION
 V      REAL   MULTIPLICAND VECTOR

USAGE ......LOP91
    THE LOADER (MAP) DIRECTIVES FOR PROGRAM  CRATER  MUST INCLUDE
 PROVISIONS FOR THIS BLOCK DATA.

 IN A UNIVAC MAP, USE THE FORM
                  IN  LOP91

USAGE ......LOP92
    THE LOADER (MAP) DIRECTIVES FOR PROGRAM  CRATER  MUST INCLUDE
 PROVISIONS FOR THIS BLOCK DATA.

 IN A UNIVAC MAP, USE THE FORM
                  IN  LOP92

USAGE ......LOP98
    CALL LOP98    (NO ARGUMENTS)

ROUTINE IS MAINLY USED FOR DEBUGGING.

USAGE ......LOP99
    CALL LOP99(NO)
ARG.    TYPE    PURPOSE
 NO     INT.    INDEX OF LOP ROUTINE IN WHICH AN ERROR OCCURRED
```

4. COMMAND INTERPRETATION

In this section we discuss the specific commands implemented in the POL (problem oriented language) of CRATER. This discussion draws heavily on the material presented in Section 2 and is probably unintelligible for the reader who has not first read that Section. A general familiarity with Section 3 will be helpful for the reading of this section but is not quite as crucial.

Both the tree structure and the command words used in CRATER are provided in Table 2. As mentioned in the introduction, only the first four letters of each command are required from the user. For each command word C, the "offspring" command words (command word options at the next level of the tree that are associated with C) appear indented and directly under it. Table 1 is constructed by CRATER in response to the commands DISPLAY, COMMANDS (or DISP, COMM) issued at tree level 1. Note that several command words such as PRINT, READ, DISPLAY, etc. are repeated. They appear only once in the command word list COMMND but are associated with several nodes of the tree.

CRATER indicates the current level in the tree whenever it lists the command options for the user. For example,

```
2 PLEASE TYPE ONE OF THE FOLLOWING COMMANDS ...
FUNCTION        PRINT           POSITION        HESSIAN
PARAMETER       EVALUATE        METHOD          SEARCH
```

is a list of level 2 options associated with INITIALIZE. If a user has lost track of what options he has selected up to a certain point, he can back up to level 1 (using one less BACKUP command than the current level number) to get to level 1, and issue commands DISPLAY OPTIONS. A typical response is as follows:

TABLE 2. Command Table for CRATER

```
 1  .  INITIALIZE
 2  .  .  PRINT
 3  .  .  FUNCTION
 4  .  .  .  ROSENBROC
 5  .  .  .  ELLIPSOID
 6  .  .  .  SPECIAL
 7  .  .  .  POWELL
 8  .  .  .  WOOD
 9  .  .  POSITION
10  .  .  .  ROSENBROC
11  .  .  .  RANDOM
12  .  .  .  ZERO
13  .  .  .  READ
14  .  .  .  POWELL
15  .  .  .  WOOD
16  .  .  .  RAYLEIGH
17  .  .  HESSIAN
18  .  .  .  INITIAL
19  .  .  .  .  ANALYTIC
20  .  .  .  .  IDENTITY
21  .  .  .  .  READ
22  .  .  .  UPDATE
23  .  .  .  .  ANALYTIC
24  .  .  .  .  DFP
25  .  .  .  .  BFGS
26  .  .  METHOD
27  .  .  .  PLS1
28  .  .  .  DFP
29  .  .  .  BFGS
30  .  .  SEARCH
31  .  .  .  CUBIC
32  .  .  .  EXCUBIC
33  .  .  .  QUADRATIC
34  .  .  PARAMETER
35  .  .  .  INTEGER
36  .  .  .  REAL
37  .  .  .  DISPLAY
38  .  .  EVALUATE
39  .  ITERATE
40  .  DISPLAY
41  .  .  RESULTS
42  .  .  .  PRINT
43  .  .  .  PLOT
44  .  .  .  .  TERMINAL
45  .  .  .  .  SYSTEM
46  .  .  OPTIONS
47  .  .  COMMANDS
48  .  .  STATE
```

```
---------------
CURRENT OPTIONS
---------------

1  POSITION     WOOD
2  FUNCTION     WOOD
3  INITIAL      HESSIAN      IDENTITY
4  UPDATE       HESSIAN      DFP
5  PRINT          9
6  METHOD       DFP
7  SEARCH       EXCUBIC
```

As discussed in Section 2, two classes of command are used, viz: transitional and terminal. The terminal commands reside at the ends of the branches and certain of them require input data immediately following them. Note that any command in Table 2 that does not have another command directly beneath and to the right (indented) of it is a terminal command. Thus, for example, PRINT, ANALYTIC, and ITERATE are terminal commands. Prompting is provided for terminal commands requiring input data, such as READ.

4.1 *Command Semantics*

The interpretation of most of the commands is self evident. Here we shall run through the commands in Table 1, pointing out special considerations as we go.

4.1.1 Initialization

The first command to be entered is usually INITIALIZE, however, a user might enter

DISPLAY OPTIONS BACK INITIALIZE

in order to find out what the default options of CRATER are before initializing. Except for EVALUATE, which should be the last command used during initialization, the order in which the initialization commands are imposed is flexible. EVALUATE establishes the function, gradient and Hessian values at the initial position and thus requires that they be previously defined.

There are three standard test functions (ROSENBROCK, ELLIPSOID and POWELL) built in and a link SPECIAL to a user defined function. The terminal commands ELLIPSOID and SPECIAL require the number of unknowns in the problem

to be entered as data (free-field integer). Since it only makes sense to select one function, CRATER generates a BACKUP command internally after a selection is made for the convenience of the user. This innovation applies also to POSITION, INITIAL, UPDATE, METHOD, SEARCH, and PLOT commands.

The PRINT command sets the amount of operational detail to be printed out during the analysis. It must be followed by a print level (integer) ranging from 0 to 10. The effects of the various print levels are described in the CRATER USAGE description in Section 3.3. Typically, the print level is set high (8 - 10) for the first couple iterations and then lowered to (0 - 5) for the rest.

The POSITION command should follow the FUNCTION command. The three standard initial positions are built in corresponding to the standard, built in functions. In addition, a random or zero initial position can be generated internally and the user has the option of entering a special initial position after the READ command. If the size of the function has not been established before entering POSITION, it does not know how large the initial vector should be. A user can obtain a variety of random initial positions by repeating the command sequence POSITION RANDOM.

The method for constructing the HESSIAN must be established both initially and for update purposes during the analysis. The Davidon-Fletcher-Powell (DFP) and Broyden-Fletcher-Goldfarb-Shanno (BFGS) update formulas are built in, both for updating the Hessian and the inverse Hessian. For this study, an analytical initial Hessian is used with either the DFP or BFGS update formula. Periodically, during the analysis, the analytical Hessian is reconstructed by means of the command sequence

INITIALIZE, EVALUATE, BACK

and then the iteration is continued. It is possible to have that command sequence generated internally by the iteration portion of the program (on the basis of a suitable decision heuristic) but this has not been done.

Most of the operational "state" variables are maintained in a parameter list described in the CRATER documentation of Section 3.2. The PARAMETER command gives the user access to these parameters both to see what values they have (DISPLAY) and to change them (INTEGER and REAL). For example, to set

the 4th floating point parameter to 5000, one may enter

REAL 4,5.E3

or some FORTRAN equivalent representation of 5000. It is essential to understand the purposes of the parameters (see Sec. 3.2 under CRATER) before doing any modifications.

The methods currently implemented are the planar search with one dimensional line search (PLSI) descent method and versions of the famous Davidon-Fletcher-Powell (DFP) and Boyden-Fletcher-Goldfarb-Shanno (BFGS) quasi-Newton methods. In the current implementation, the Hessian updates are applied directly to the approximate Hessians rather than being held in product form. Consequently, it is not appropriate to use an analytical initial Hessian with either the DFP or the BFGS algorithms since the requisite matrix inversion algorithm is not provided.

A choice of the Davidon cubic interpolation line search (CUBIC) without extrapolation or with it (EXCUBIC), or a quadratic line search (QUADRATIC) is provided. If EXCUBIC is selected, then the expected value of the converged function must be supplied as data.

4.1.2 Iteration

The iteration process is trivial to invoke in comparison with initialization. The user simply enters ITERATE followed by the maximum number of iteration steps to be taken. As mentioned earlier, it is convenient to initially set the print level high and iterate only 2 or 3 times in order to see that the process is going properly. The user can then return to initialize and lower the print level if everything is ok, or else change the initial conditions in order to improve things.

4.1.3 Display

Besides intermediate results that can be obtained during the iteration process by setting the appropriate print level, CRATER provides a variety of summary results via the DISPLAY command. Values for the function, the gradient norm, the step norm and inner product of each new gradient with the previous step (0 for perfect iteration) are tabulated during iteration and can be printed or plotted (on Univac computers) in the display mode. The plotting system is machine dependent and would undoubtedly have to be modified for different computing environments. Terminal plotting, of course, is only applicable if the user has a graphics terminal.

In addition to computational results, select options chosen during initialization and the complete command table (see Table 1) can also be generated. Should something go wrong with the computation, the program state variables may also be displayed for debugging purposes. This output tends to be rather long and is seldom needed.

4.2 Batch Processing

The problem oriented language of CRATER provides a convenient means for setting up batch jobs as well as interactive ones. The following is typical of such a job in a fairly readable form.

```
HELP                          $     WE FIRST ILLUSTRATE THE HELP INSTRUCTION.
INITIALIZE                    $  THIS IS A TYPICAL PROBLEM DEFINITION IN THE
   PRINT 10                   $  CRATER POL (PROBLEM ORIENTED LANGUAGE). THE
   FUNCTION ROSENBROCK        $  PRINT LEVEL IS SET HIGH TO OBTAIN ALL OUT-
   POSITION ROSENBROCK        $  PUT DURING INITIALIZATION.  THE STANDARD
   HESSIAN                    $  ROSENBROCK FUN. IS TO BE SOLVED USING ITS
      INITIAL ANALYTIC        $  STANDARD INITIAL POSITION VECTOR. THE INIT-
      UPDATE  ANALYTIC        $  IAL AND UPDATE HESSIANS ARE TO BE ANALYTIC,
      BACK                    $  DEMONSTRATING THE BASIC ALGORITHM DISCUSSED
   METHOD PLS1                $  IN TEXT. THE PLANAR SEARCH 1 (PLS1) ALGORITHM
   PARAMETER                  $  IS TO BE USED. REAL PARAMETER 4 (TOLFAC) IS
      REAL 4,5000.            $  SET TO 5000. IN ORDER TO FREE ALL MONOTON-
      BACK                    $  ICITY CONSTRAINTS.  WE SHALL USE THE CUBIC
   SEARCH CUBIC               $  SEARCH IN THEIS SETTING.
   EVALUATE                   $
   PARAMETER                  $        AFTER THE INITIAL FUNCTION, GRADIENT
      DISPLAY                 $  AND HESSIAN EVALUATION, WE DISPLAY THE PAR-
      BACK                    $  AMETERS FOR DOUBLE CHECKING LATER. NOTE
   BACK                       $  THAT WE ALWAYS ENTER BACK TO RETURN TO THE
DISPLAY                       $  PREVIOUS TREE LEVEL (INDENTATION LEVEL). WE
   OPTIONS                    $  CHOOSE TO OBTAIN A SUMMARY OF THE OPTIONS
   BACK                       $  SELECTED ON THE OUTPUT FOR HANDY REFERENCE.
ITERATE 1                     $  NEXT WE CARRY OUT ONE ITERATION WITH LOTS
INITIALIZE                    $  OF PRINTOUT AND THEN GO BACK TO INITIALIZE
   PRINT 5                    $  IN ORDER TO REDUCE THE PRINT LEVEL.
   BACK                       $
ITERATE 200                   $  NOW WE SOLVE THE PROBLEM ALLOWING NO MORE
DISPLAY                       $  THAN 200 STEPS, AND DISPLAY THE RESULTS ON
   RESULTS                    $  THE PRINTER (AS OPPOSED TO PLOTTING THEM).
      PRINT                   $  NOTE THAT THE LINE END FLAG $ IS NOT RE-
      STOP                       QUIRED AFTER THE STOP COMMAND.
```

The same job in more compact (and less readable) form appears as follows:

```
HELP INIT PRIN 10 FUNC ROSE POSI ROSE HESS INIT ANAL UPDA ANAL BACK $
METH PLS1 PARA REAL 4,5000. BACK SEAR CUBI EVAL PARA DISP BACK BACK $
DISP OPTI BACK ITER   1  INIT PRIN 5  BACK ITER 200  DISP RESU PRIN $
STOP
```

It is obviously necessary to have Table 1 handy along with a familiarity of Sec. 4.1 in order to set up such a job. However, once such a set up is made for a particular problem, it proves very convenient for executing a number of studies.

By leaving off the stop and insuring that the last line does not end in $, a packaged command sequence similar to the above setup is also very helpful for interactive studies. Simply introduce the command sequence initially and have control returned to an interactive terminal afterwards for further processing.

4.3 Examples

In this section, we provide sample results obtained from two run streams. The first is taken directly from an interactive session at a terminal. The commands issued by the user are underlined for emphasis.

```
$GOT GA.
 HELLO, I AM A CONVERSATIONAL PROGRAM FOR LOCAL FUNCTIONAL MINIMIZATION.
 IF YOU NEED HELP, PLEASE TYPE  HELP.   OTHERWISE ...
  1 PLEASE TYPE ONE OF THE FOLLOWING COMMANDS ...
 INITIALIZE       ITERATE          DISPLAY
DISP
  2 PLEASE TYPE ONE OF THE FOLLOWING COMMANDS ...
 RESULTS          OPTIONS          COMMANDS         STATE
OPTIONS

                  ---------------
                  CURRENT OPTIONS
                  ---------------

   1  POSITION     RANDOM
   2  FUNCTION     SPECIAL
   3  INITIAL      HESSIAN      ANALYTIC
   4  UPDATE       HESSIAN      BFGS
   5  PRINT
                     5
   6  METHOD       PLS1
   7  SEARCH       QUADRATIC
  2 PLEASE TYPE ONE OF THE FOLLOWING COMMANDS ...
 RESULTS          OPTIONS          COMMANDS         STATE
BACK INIT
```

```
  2 PLEASE TYPE ONE OF THE FOLLOWING COMMANDS ...
 PRINT              FUNCTION           POSITION           HESSIAN
 METHOD             SEARCH             PARAMETER          EVALUATE
FUNC
  3 PLEASE TYPE ONE OF THE FOLLOWING COMMANDS ...
 ROSENBROC          ELLIPSOID          SPECIAL            POWELL
 WOOD
SPEC

 PLEASE TYPE THE SIZE (DIMENSION) OF YOUR FUNCTIONAL.
 4
  2 PLEASE TYPE ONE OF THE FOLLOWING COMMANDS ...
 PRINT              FUNCTION           POSITION           HESSIAN
 METHOD             SEARCH             PARAMETER          EVALUATE
POSI
  3 PLEASE TYPE ONE OF THE FOLLOWING COMMANDS ...
 ROSENBROC          RANDOM             ZERO               READ
 POWELL             WOOD               RAYLEIGH
RAYLINGTON
      INITIAL POSITION - RAYLEIGH
   1.00000+00   4.71405-01   4.71405-01   4.71405-01
  2 PLEASE TYPE ONE OF THE FOLLOWING COMMANDS ...
 PRINT              FUNCTION           POSITION           HESSIAN
 METHOD             SEARCH             PARAMETER          EVALUATE
EVAL BACK
      FORM OF INITIAL HESSIAN - ANALYTICAL
  1 PLEASE TYPE ONE OF THE FOLLOWING COMMANDS ...
 INITIALIZE         ITERATE            DISPLAY
ITER 30

 *** LOP24 - AN UPDATE COEFFICIENT IS ZERO.  A.P =   0.000      0.000
     RE-INITIALIZE THE HESSIAN.
     FORM OF INITIAL HESSIAN - ANALYTICAL
 (5,5) =  3.7291674-03 AT -4.2687112-01 - NOT CONVERGED
 (5,5) = -2.0617467-04 AT  5.2478673-01 - NOT CONVERGED
  1 PLEASE TYPE ONE OF THE FOLLOWING COMMANDS ...
 INITIALIZE         ITERATE            DISPLAY
DISP RESULTS
  3 PLEASE TYPE ONE OF THE FOLLOWING COMMANDS ...
 PRINT              PLOT
PRINT

                    -------------
                    FINAL RESULTS
                    -------------

     FUNCTIONAL VALUE          1.0000000+00
     GRADIENT NORM             1.6295352-07
     STEP NORM                 2.3020643-07
     ACCEPTANCE CRITERION      3.2000000+01
     STEP LENGTH FACTOR        2.0000000+00
```

```
      NO. FUNCTIONAL EVALS.        14
      NO. GRADIENT EVALS.          34
      NO. HESSIAN EVALS.            2
      NO. HESSIAN UPDATES          13
      NO. HES-VECTOR MULTS         39
      NO. ITERATIONS               13
      NO. DEGEN. PLANES             0

                    ------------------------------
                    SOLUTION (POSITION OF MINIMUM)
                    ------------------------------

 9.5356751-01 -3.6442101-08  2.6754551-08 -1.2008801-08

                    --------------------
                    CONVERGENCE HISTORY
                    --------------------

ITER  FUNCTIONAL     GRADIENT  COS(THETA(G,S))     STEP
   0  1.300000+00  1.715809+00  1.202284-08  1.290935+00
   1  1.664911+00  2.218740+00  0.000000     0.000000
   2  1.601733+00  2.915311+00  1.897474-02  6.741414-01
   3  1.214343+00  1.308314+00  3.712347-07  4.484947-01
   4  1.041974+00  6.183373-01 -7.361932-06  3.346284-01
   5  1.004502+00  1.997154-01 -4.704744-03  2.194259-01
   6  1.000091+00  3.247124-02 -1.403343-04  4.472325-02
   7  1.000002+00  3.490150-03  7.225236-05  9.935069-03
   8  1.000000+00  6.995709-04  8.676902-02  1.413664-03
   9  1.000000+00  1.939640-04 -2.517868-01  1.168739-04
  10  1.000000+00  3.488491-05  4.004605-02  5.255073-05
  11  1.000000+00  4.753469-06  8.729343-04  7.711701-06
  12  1.000000+00  1.167751-06 -2.406693-02  1.677726-06
  13  1.000000+00  1.629595-07 -5.353420-02  2.302064-07
 3 PLEASE TYPE ONE OF THE FOLLOWING COMMANDS ...
PRINT            PLOT
BACK OPTIONS

                    ---------------
                    CURRENT OPTIONS
                    ---------------

  1  POSITION     RAYLEIGH
  2  FUNCTION     SPECIAL
  3  INITIAL      HESSIAN      ANALYTIC
  4  UPDATE       HESSIAN      BFGS
  5  PRINT
                    5
  6  METHOD       PLS1
  7  SEARCH       QUADRATIC
 2 PLEASE TYPE ONE OF THE FOLLOWING COMMANDS ...
RESULTS          OPTIONS          COMMANDS          STATE
STOP

RUN TERMINATED
```

The second example is the result of submitting the run stream at the beginning of Section 4.2 in a batch processing mode. In this instance, the user commands were not displayed in the output. Can you tell where they would have appeared in interactive mode? There is an echo option for batch mode which includes a display of the commands.

```
@XQT CA.
 HELLO, I AM A CONVERSATIONAL PROGRAM FOR LOCAL FUNCTIONAL MINIMIZATION.
 IF YOU NEED HELP, PLEASE TYPE  HELP.   OTHERWISE ...
  1 PLEASE TYPE ONE OF THE FOLLOWING COMMANDS ...
 INITIALIZE      ITERATE          DISPLAY
 OUR CONVERSATION IS ORGANIZED AS A TREE  STRUCTURE.   AT EACH LEVEL YOU
 WILL HAVE DIFFERENT COMMAND  OPTIONS TO DICTATE MY SUBSEQUENT  ACTIONS.
 AT EACH LEVEL I SHALL TELL YOU THE CURRENT  LEVEL  NUMBER AND THE VALID
 COMMANDS. YOU SELECT ONE AND I WILL CARRY IT OUT.  YOU MAY MISSPELL THE
 COMMANDS AS LONG AS THE FIRST FOUR CHARACTERS ARE CORRECT.

 WHEN YOU HAVE  COMPLETED  WORK AT A BRANCH OF THE TREE.  USE THE BACKUP
 COMMAND TO GO BACK TO THE PREVIOUS (HIGHER) LEVEL BRANCH.  FOR BREVITY,
 I DO NOT REPEATEDLY PRINT BACKUP AS A COMMAND OPTION.  THE SAME BREVITY
 CONVENTION APPLIES TO THE HELP AND STOP COMMANDS.

 THE EVALUATION PHASE OF INITIALIZATION IS ALWAYS REQUIRED AT LEAST ONCE
 BEFORE THE SOLUTION (ITERATION) PHASE.  IT SHOULD BE DONE AFTER MOST OF
 THE OTHER INITIALIZATION WORK IS DONE.

 GOOD LUCK.
     INITIAL POSITION - ROSENBROCK STANDARD
  -1.20000+00   1.00000+00
 GRADIENT
 -2.1560000+02 -8.7999999+01

 F =  2.4200000+01
     FORM OF INITIAL HESSIAN - ANALYTICAL

 HESSIAN
   1.330000+03
   4.800000+02   2.000000+02

 THE INTEGER PARAMETERS ARE
    1    1    1    1    1   10    1    1    1    1
    2  100    5    0    0    1    1    1    0    0
    0    0    0    0    0    0    0    1    3    5
    9   11   13   15   17 1262 2507 3752 4997

 THE REAL PARAMETERS ARE
  3.200000+01  0.000000      0.000000      5.000000+03  0.000000
  0.000000     0.000000      1.280000+02   0.000000     2.420000+01
  1.000000+06  1.000000+06   2.328677+02   1.000000+06  1.000000+06
  4.693321-01  1.000000+06   1.000000+06   1.562050+00  1.000000+06
  1.000000+06  1.000000+06   1.000000+06   1.000000+06  1.000000+06
  1.000000+06  1.000000+06
```

```
                    ---------------
                    CURRENT OPTIONS
                    ---------------

  1  POSITION     ROSENBROC
  2  FUNCTION     ROSENBROC
  3  INITIAL      HESSIAN      ANALYTIC
  4  UPDATE       HESSIAN      ANALYTIC
  5  PRINT         10
  6  METHOD       PLS1
  7  SEARCH       CUBIC
STEP
 2.4742007-02  3.8061216-01

F =  4.7317856+00
GRADIENT
-4.6415982+00 -1.2383759-01

POSITION
-1.1752580+00  1.3806122+00

HESSIAN
  1.107233+03
  4.701032+02   2.000000+02

                    -------------
                    FINAL RESULTS
                    -------------

    FUNCTIONAL VALUE          0.0000000
    GRADIENT NORM             0.0000000
    STEP NORM                 2.4607389-06
    ACCEPTANCE CRITERION      3.2000000+01
    STEP LENGTH FACTOR        5.0000000+03
    NO. FUNCTIONAL EVALS          8
    NO. GRADIENT EVALS.           8
    NO. HESSIAN EVALS.            8
    NO. HESSIAN UPDATES           0
    NO. HES-VECTOR MULTS         14
    NO. ITERATIONS                7
    NO. DEGEN. PLANES             0

                    ------------------------------
                    SOLUTION (POSITION OF MINIMUM)
                    ------------------------------
 1.0000000+00  1.0000000+00

                    ------------------
                    CONVERGENCE HISTORY
                    ------------------
ITER  FUNCTIONAL     GRADIENT  COS(THET(G,S))     STEP
   0  2.420000+01  2.328677+02  4.693321-01  1.562050+00
   1  4.731786+00  4.643250+00 -9.146012-02  3.814155-01
   2  1.403741+03  1.363379+03  8.328175-01  4.943733+00
   3  5.727937-02  4.786486-01 -1.039726-04  3.747062+00
   4  3.281071-01  2.561674+01  1.175821-01  4.357235-01
   5  1.554799-07  7.355650-03 -4.906501-01  5.655543-02
   6  2.692291-12  5.475962-05 -2.711999-02  7.997023-04
   7  0.000000     0.000000      0.000000     2.460739-06

RUN TERMINATED
```