A SURVEY AND COMPARISON OF CONJUGATE GRADIENT METHODS FOR OPTIMIZATION

By

MEISHAN CHENG

Bachelor of Science

Zhongshan(Yet-San) University

Guangzhou, Guangdong, P.R. of China

1985

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
December, 1993

OKLAHOMA STATE UNIVERSITY

A SURVEY AND COMPARISON OF CONJUGATE GRADIENT METHODS FOR OPTIMIZATION

Thesis Approved:

J Chandler
Thesis Adviser

[illegible]

John Wolfe

[illegible]
Dean of the Graduate College

PREFACE

This paper presents a comparison of various Conjugate Gradient methods for solving optimization problems. These are of practical use for minimizing functions of very many variables because they do not require the storage of any two-dimensional matrices.

The comparisons are given for (1) four traditional conjugate gradient methods, including the Fletcher-Reeves method, the Polak-Ribiere method, the Beale-Sorenson method and the Perry method; (2) three different restart methods which include a traditional restart method, Powell's restart method and Shanno's restart method; (3) different line search methods which include the Brent line search method and the Nash line search method. All methods are implemented using FORTRAN 77, and compared with each other based on the number of function evaluations and the number of gradient evaluations. Numerical results on a set of test problems indicate that various CG methods are relatively equivalent.

I would like to express my sincere gratitude to my major adviser, Dr. John P. Chandler, who introduced me to this interesting subject, and constantly gave me intelligent guidance. Many thanks also to the other committee members, Dr. K. M. George and Dr. J. Wolfe, for their helpful advisement and suggestions.

Special thanks are due to my wife, Jingwen Chen, for her moral support. My deepest appreciation is extended to my parents whose encouragement and understanding were invaluable throughout the study.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

This thesis studies Conjugate Gradient methods for solving unconstrained optimization problems which are to minimize a real valued nonlinear function $f(x)$, where $f \in C^2$ and $x \in R^n$. Although we mainly concentrate on unconstrained minimization problems, this is not as restrictive as it may seem because constrained minimization problems can be reduced to unconstrained minimization problems or a system of unconstrained minimization problems. Obviously, maximum points of f are minimum points of $-f$, so maximization problems can be changed to minimization problems.

To solve unconstrained optimization problems, we usually apply either Direct Search methods or Gradient Direction methods. Direct Search methods are usually based on a sequential examination of trial solutions which by simple comparisons give an indication of a direction for further searches. They do not require derivative calculations but only function value evaluation. In general, Direct Search methods converge somewhat more slowly than Gradient Direction methods. Gradient Direction methods require calculating partial derivatives and gradients but they have a faster rate of convergence than Direct Search methods. Gradient Direction methods can generally be classified as four types of methods: steepest descent methods, the Newton-Raphson method, quasi-Newton (QN) methods and conjugate gradient (CG) methods.

Steepest descent methods are old and obvious. Because the direction of steepest ascent of $f(x)$ at a point x_0 is given by $\nabla f(x_0)$, the vector $-\nabla f(x_0)$ gives the direction of steepest descent. Along the line in the direction of steepest descent, we can minimize the function $f(x)$. Although the method of steepest descent is useful for a large class of well-conditioned problems, it has been shown that the methods can be extremely slow. For example, when the steepest descent method is applied to the simple function $f(x, y) = 0.0001x^2 + y^2$ of two variables with $(x_1, y_1) = (1, 0.001)$ as the initial point, it takes over 6900 linear minimizations to obtain its minimum point correct to six decimal places. Obviously, this is an unsatisfactory situation[2],[9].

In the Newton-Raphson method, we start with an estimate x_1 and the second order Taylor expansion of $f(x)$ at x_1. This yields a quadratic approximation $F_1(x)$ of $f(x)$ about x_1. The minimum point x_2 of $F_1(x)$ is taken to be the next estimate of the minimum point of $f(x)$. Repeating this process yields successive estimates x_1, x_2, x_3, ... , which normally converge quadratically. The Newton-Raphson method therefore can be viewed as a method based on successive minimizations of quadratic functions $F_1(x)$, $F_2(x)$, $F_3(x)$, ... The minimum point x_{k+1} of $F_k(x)$ is given by the formula

$$x_{k+1} = x_k - H_k \nabla f(x_k)$$

where H_k is the inverse of the Hessian $f''(x_k)$ of $f(x)$ at x_k and $\nabla f(x_k)$ is its gradient.

The well-known quasi-Newton method can be regarded as a modification of the Newton-Raphson method under the condition of the Hessian matrix of $f(x)$ being positive definite. A quasi-Newton method generates a sequence of points $x_{(i)}$ based on the iteration

$$x_{(i+1)} = x_{(i)} - \alpha_{(i)} H_{(i)} \nabla f(x_{(i)} \qquad i \geq 0$$

where $\alpha_{(i)}$ is chosen according to $f(x_{(i)} + \alpha_{(i)} d_{(i)}) = min\{f(x_{(i)} + \alpha d_{(i)})\}$ or at least $f(x_{(i+1)}) < f(x_{(i)})$. $H_{(i)}$ is an estimate of the inverse of the Hessian matrix of $f(x)$ and is updated at each iteration [16]. Quasi-Newton methods are regarded as the fastest approach for unconstrained optimization problems so far.

While the rate of convergence of quasi-Newton methods is quite rapid in most cases, they do require large storage compared to conjugate gradient methods which do not have to approximate the inverse Hessian matrix. For an unconstrained minimization of an n-variable function, the requirement of storage for a quasi-Newton method is $O(n^2)$ while for a conjugate gradient method it is $O(n)$. Thus CG methods have an obvious advantage over QN methods of much lower storage requirement when we deal with very large dimensions (i.e. n is very large).

In this thesis, a survey is given of several CG methods which use different search directions and restart criterion. A comparison is also given of those CG methods. The reliability and robustness of a method are important factors for the method to be regarded as a good method.

In Chapter II, a brief review of existing conjugate gradient methods will be given.

In Chapter III, a study will be given of search directions, line search routines, and restart criteria.

In Chapter IV, the numerical results for the different CG methods will be shown.

In Chapter V, I will make some conclusions based on the previous chapters.

Finally, a set of test problems and a source program which implements the different CG methods and the comparison among them will be put into appendices.

CHAPTER II

CONJUGATE GRADIENT METHODS

All conjugate gradient algorithms are of the form given below. For every $x \in R^n$, let F(x) be defined as:

$$F(x) = \{d \in R^n | \nabla f(x) d < 0\}$$

Conjugate gradient method prototype:

Step 0. Select a $x_{(0)} \in R^n$ and set i=0.

Step 1. Compute $\nabla f(x_{(i)})$.

Step 2. If $\nabla f(x_{(i)}) = 0$, stop; else compute a $d_{(i)} \in F(x_{(i)})$ and go to step 3.

Step 3. Compute a $\alpha_{(i)} > 0$ such that $f(x_{(i)} + \alpha_{(i)} d_{(i)}) = min_{\alpha}\{f(x_{(i)} + \alpha d_{(i)})\}$.

Step 4. Set $x_{(i+1)} = x_{(i)} + \alpha_{(i)} d_{(i)}$, set i=i+1, and go to step 1.

As shown in this prototype, a new search direction $d_{(i)}$ must be chosen in step 2. In step 3, a line search routine is needed to find the minimum point along the search direction $d_{(i)}$. A CG iteration can be generally defined by

$$x_{(i+1)} = x_{(i)} + \alpha_{(i)} d_{(i)} \tag{1}$$

where $\alpha_{(i)}$ is chosen to minimize f(x) along the search direction $d_{(i)}$ using a line search routine (I will use the same definition of $d_{(i)}$ through the rest of this thesis) . In general, a CG method requires a formula to generate the search directions $d_{(i)}$ and a line search routine [1].

Numerous conjugate gradient methods have been developed since CG methods were first introduced by Hestenes and Stiefel [2]. Most of those CG methods differ mainly in the formula for the new search direction $d_{(i)}$.

Search Direction

The first conjugate gradient method was proposed by Hestenes and Stiefel [2] for minimizing a quadratic function. It generates mutually conjugate directions to find the minimum point of an n-variable positive definite quadratic function in $m \leq n$ steps, if no roundoff errors occur.

Applying the idea of Hestenes and Stiefel method to general nonlinear function, the well-known Fletcher-Reeves method [6] uses an alternative formula for choosing the search direction $d_{(i)}$ sequence

$$d_{(i+1)} = -g_{(i+1)} + \beta_{(i)} d_{(i)} \qquad i \geq 0 \tag{2}$$

where $g_{(i)} = \nabla f(x_{(i)}$ and $\beta_{(i)} = g^T_{(i+1)} g_{(i+1)} / g^T_{(i)} g_{(i)}$. An exact line search routine is required by the Fletcher-Reeves method.

The well-known Polak-Ribiere method [16] differs from the Fletcher-Reeves method only in the choice of $\beta_{(i)}$. The formula to obtain $\beta_{(i)}$ is

$$\beta_{(i)} = g^T_{(i+1)}(g_{(i+1)} - g_{(i)}) / g^T_{(i)} g_{(i)} \tag{3}$$

An exact line search routine is also required by the Polak and Ribiere method.

The formula for $\beta_{(i)}$ in the Beale-Sorenson method [5] is

$$\beta_{(i)} = g^T_{(i+1)}(g_{(i+1)} - g_{(i)}) / d^T_{(i)}(g_{(i+1)} - g_{(i)}). \tag{4}$$

Unlike the above three methods, in which $\alpha_{(i)}$ is chosen to minimize $f(x_{(i)} + \alpha d_{(i)})$, the Perry method [7] proposed a modified formula of (1) to choose the search direction $d_{(i)}$ sequence

$$d_{(i+1)} = -g_{(i+1)} + ((y_{(i)} - \alpha_{(i)} d_{(i)})^T g_{(i+1)} / y_{(i)} d_{(i)}) d_{(i)} \tag{5}$$

where $y_{(i)} = g_{(i+1)} - g(i)$. An inexact line search routine (the well-known quadratic interpolation method) can be applied in the Perry method. Perry notes that with exact line searches, his method is identical to the Polak and Ribiere method. He tests his new method against the Fletcher-Reeves method and the Polak-Ribiere method without exact searches, and notes that his method produces a significant increase in computational efficiency for his choice of test functions. However, if we define $p_{(i)}$ as

$$p_{(i)} = \alpha_{(i)} d_{(i)}$$

then (5) can be rewritten as

$$d_{(i+1)} = -(I - \frac{p_{(i)} y_{(i)}^T}{y_{(i)}^T p_{(i)}} + \frac{p_{(i)} p_{(i)}^T}{p_{(i)}^T p_{(i)}}) g_{(i+1)} = -Q_{(i+1)} g_{(i+1)} \tag{6}$$

Shanno [12] points out that a major difficulty with the Polak-Ribiere method and Fletcher-Reeves method which is not corrected by Perry method is that the matrix $Q_{(i+1)}$ defined by (6) is not symmetric and hence not positive definite. Thus the direction $d_{(i+1)}$ is not necessarily a descent direction and numerical instability can still result.

In a more recent paper, Le [17] proposed a new formula for search direction $d_{(i)}$:

$$d_{(i+1)} = z_{(i+1)} - x_{(i+1)} \tag{7}$$

with $\quad z_{(i+1)} = w_{(i+1)} + \gamma_{(i+1)} d_{(i)}$

and $\quad w_{(i+1)} = x_{(i+1)} + \beta_{(i+1)}(-g_{(i+1)})$,

where $\gamma_{(i+1)}$ minimizes $f(w_{(i+1)} + \gamma d_{(i)})$, and $\beta_{(i+1)}$ is chosen such that $w_{(i+1)}$ approximates the point on the other side of the valley along $-g_{(i+1)}$ with the same function value as $f(x_{(i+1)})$.

In Le's method, a special inexact line search routine which uses only function evaluations but does take advantage of gradient information already available at the beginning of each iteration is required.

Restart Search Direction and Procedure

Fletcher and Reeves [6] recommend restarting the conjugate gradient method using the steepest descent direction every n or (n+1) iterations. We call this a traditional restart here. There seems to be general agreement that occasional restarting is very helpful in practice. Cowder and Wolfe [18] give an example to show that without restarting, the rate of convergence of traditional conjugate gradient methods can be only linear and Powell [19] shows that without restarts a linear rate of convergence is usual when there are more than two variables. If we want to improve that rate of convergence, it is necessary to study restarting procedures.

A disadvantage of restarting by searching along the steepest descent direction (i.e. $-\nabla f(x)$) is that the immediate reduction in the objective function $f(x)$ is usually less than it would be without the restart. Therefore we would like to find a conjugate gradient method that does not restrict the search direction of a restarting iteration to the steepest descent direction.

Beale [5] first proposed a double update scheme restart criterion which has the desired convergence rate properties, but allows the restart step to take into account $d_{(k-1)}$ while we derive the new search direction $d_{(k+1)}$. To make the sequence of search directions $d_{(t)}$, $d_{(t+1)}$, $d_{(t+2)}$, ... mutually conjugate, it is shown in Beale's paper [5] that form

$$d_{(k+1)} = -g_{(k+1)} + \beta_{(k+1)} d_{(k)} + \gamma_{(k+1)} d_{(t)} \tag{8}$$

and where

$$\beta_{(k+1)} = \frac{g^T_{(k+1)} y_{(k)}}{d^T_{(k)} y_{(k)}}$$

$$\gamma_{(k+1)} = \frac{g^T_{(k+1)} y_{(k)}}{d^T_{(t)} y_{(t)}}$$

for $k = t+1, t+2, ..., t+n-1$. Here again $y_{(k)} = g_{(k+1)} - g_{(k)}$.

Although McGuire and Wolfe [20] tried Beale's method and found their numerical results disappointing, Powell [8] suggests that Beale's method is still suitable if we use Beale's restart formula (8) every n steps or whenever

$$|g_{(k+1)}^T g_{(k)}| \geq 0.2||g_{(k+1)}||^2 \tag{9}$$

Powell shows some limited computational results to demonstrate the advantages of his method.

Later, Shanno [12] combined Beale's and Powell's restart criteria into a double update scheme. His method modified g with a positive definite matrix which best estimates the inverse Hessian without adding anything to storage requirements. His double update is accomplished by

$$\bar{H}_{(k)} = I - \frac{p_{(t)}y_{(t)}^T + y_{(t)}p_{(t)}^T}{p_{(t)}^T y_{(t)}} + (1 + \frac{y_{(t)}^T y_{(t)}}{p_{(t)}^T y_{(t)}})\frac{p_{(t)}^T p_{(t)}}{p_{(t)}^T y_{(t)}} \tag{10}$$

where again $y_{(k)} = g_{(k+1)} - g_{(k)}$, $p_{(k)} = \alpha_{(k)}d_{(k)}$, I is the n by n identity matrix, and

$$\bar{H}_{(k+1)} = \bar{H}_{(k)} - \frac{p_{(k)}y_{(k)}^T \bar{H}_{(k)} + \bar{H}_{(k)}y_{(k)}p_{(k)}^T}{p_{(k)}^T y_{(k)}} + (1 + \frac{y_{(k)}^T \bar{H}_{(k)} y_{(k)}}{p_{(k)}^T y_{(k)}})\frac{p_{(k)}^T p_{(k)}}{p_{(k)}^T y_{(k)}} \tag{11}$$

and defining $d_{(k+1)}$ by

$$d_{(k+1)} = -H_{(k+1)}g_{(k+1)} \tag{12}$$

Note that no matrix needs to be actually stored, because (12) can be rewritten as

$$d_{(k+1)} = -\bar{H}_{(k)}g_{(k+1)} + \frac{p_{(k)}^T g_{(k+1)}}{p_{(k)}^T y_{(k)}}\bar{H}_{(k)}y_{(k)} - (1 + \frac{y_{(k)}^T \bar{H}_{(k)} y_{(k)}}{p_{(k)}^T y_{(k)}}\frac{p_{(k)}^T g_{(k+1)}}{p_{(k)}^T y_{(k)}} - \frac{y_{(k)}^T \bar{H}_{(k)} g_{(k+1)}}{p_{(k)}^T y_{(k)}})p_{(k)} \tag{13}$$

where the vectors $\bar{H}_{(k)}g_{(k+1)}$ and $\bar{H}_{(k)}y_{(k)}$ are defined by

$$\bar{H}_{(k)}g_{(k+1)} = g_{(k+1)} - \frac{p_{(t)}^T g_{(k+1)}}{p_{(t)}^T y_{(t)}}y_{(t)} + (1 + \frac{y_{(t)}^T y_{(t)}}{p_{(t)}^T y_{(t)}}\frac{p_{(t)}^T g_{(k+1)}}{p_{(t)}^T y_{(t)}} - \frac{y_{(t)}^T g_{(k+1)}}{p_{(t)}^T y_{(t)}})p_{(t)} \tag{14}$$

and

$$\bar{H}_{(k)}y_{(k)} = y_{(k)} - \frac{p_{(t)}^T y_{(k)}}{p_{(t)}^T y_{(t)}}y_{(t)} + (1 + \frac{y_{(t)}^T y_{(t)}}{p_{(t)}^T y_{(t)}}\frac{p_{(t)}^T y_{(k)}}{p_{(t)}^T y_{(t)}} - \frac{y_{(t)}^T y_{(t)}}{p_{(t)}^T y_{(t)}})p_{(t)} \tag{15}$$

Shanno's algorithm seems to converge faster than previous CG methods based on his set of selected test functions: Wood's function, Powell's function, Fletcher and Powell's trigonometric function with number of variables n = 5, 10, and 15.

It should be mentioned that the above restart search directions generally require more effort to find line minima with the same convergence tolerance at each step than do the Polak-Ribiere, the Fletcher-Reeves, the Beale-Sorenson, and the Perry methods.

CHAPTER III

LINE SEARCH

For most multi-dimensional optimization methods, the line search procedure plays an important roles. As shown in the prototype in the previous chapter, in every CG iteration, CG methods generate an updated search direction and require a line search along the search direction to find the minimum point in this direction. For a fixed search direction d, the line search problem which searches for the minimal point along d can be regarded as finding a value of α which approximately minimizes the following one-dimension problem

$$\phi(\alpha) = f(x + \alpha d) \tag{16}$$

where the initial point x and direction $d \neq 0$ are given.

There exist many line search methods. They can be classified as two types: (1) methods using only function evaluations; (2) methods using both gradient and function values. The merits of line search method type (1) compared to line search method type (2) actually depend on the problem's dimension. As stated by Le [17], for a small dimension problem, we cannot tell which type of method is better. However, line search method type (1) becomes superior for large dimension problem, since the additional information generated by a gradient evaluation can hardly offset the expense of generating it.

As stated in previous chapter, CG methods have the advantage of smaller storage requirements over other methods in large dimension minimization problems. In order to keep this advantage and for the reason stated above, we prefer to use line search methods of type (1) in CG methods. We introduce two line search methods, the Brent line search method and the Nash line search method, which both use only function evaluations, in the rest of this chapter. [illegible]

Brent Line Search Method

The Brent line search method [4] is a combination of golden section search and successive parabolic interpolation. If f has a continuous second derivative which is positive at the minimum (which is not at the end-points of the interval) then, ignoring rounding errors, convergence is superlinear, and usually the order is at least 1.3247....

Here we give an outline of procedure LOCBAC which demonstrates the main ideas of Brent line search method. For more detail, the reader can refer to the Appendix, where the method is described formally by the subroutine LOCBAC.

Outline of LOCBAC:

At a typical step there are six significant points a, b, u, v, w, and x, not all distinct. The positions of these point change during the subroutine. Initially (a, b) is the interval on which f is defined, and

$$v = w = x = a + gnum(b - a) \tag{17}$$

a and b are the endpoints of the interval in which a local minimum lies. $gnum = \frac{3-\sqrt{5}}{2} = 0.381966$ is the magic number for golden section search. x is the point with the least function value among all the points having been evaluated. w is the point with next lowest function value. v is the previous value of w. u is the last point at which f was evaluated. The tolerance tol is a combination of relative tolerance eps and an absolute tolerance t, i.e.

$$tol = eps|x| + t \tag{18}$$

Once we define the tolerance tol, we have the stop criterion

$$max(x - a, b - a) \leq 2 * tol$$

For $m = \frac{a+b}{2}$, we can have the equivalent condition

$$|x - m| \leq 2 * tol - \frac{b-a}{2} \tag{19}$$

Given a, b, x, eps and t , LOCBAC can be expressed as below.

Let $m = \frac{a+b}{2}$, $tol = eps|x| + t$, $fu = f(u)$, $fv = f(v)$, $fw = f(w)$, and $fx = f(x)$.

```
Step 1. Test stopping criterion (19):
        if it is satisfied, then terminate;
        else
            set $p = q = r = 0$,
            go to Step 2.

Step 2. Fit parabola:
        set $r = (x - w)(f(x) - f(v))$;
        set $q = (x - v)(f(x) - f(w))$;
        set $p = (x - v)q - (x - w)r$;
        if $q > 0$ then p=-p, else q=-q;
        set $r = e$ and $e = d$;
        if($|p| < \frac{1}{2}|qr|$ and $p < min(a - x, b - x)$) go to Step 3;
        else go to Step 4.

Step 3. Parabolic interpolation step:
        set $d = \frac{p}{q}$ and $u = x + d$;

Step 4. Golden section step:
        if($x < m$) set $epoint = b$, else set $epoint = a$;
        set $u = epoint - x$ and set $d = gnum * d$;

Step 5. Update $a$, $b$, $v$, $w$, and $x$:
        if $fu < fx$
```

```
then
    if $u < x$ then set $b = x$, else set $a = x$;
    set $v = w$ and $fv = fw$;
    set $w = x$ and $fw = fx$;
    set $x = u$ and $fx = fu$;
else
    if $u < x$ then set $a = u$, else set $b = u$;
    if $fu < fw$ and $w = x$ then
       set $v = w$ and $fv = fw$;
       set $w = u$ and $fw = fu$
    else if $fu \leq fv$ or $v = x$ or $v = w$ then
       set $v = u$ and $fv = fu$;
Go to Step 1.
```

Nash Line Search Method [3]

Instead of giving the search interval $[a, b]$, let's consider the case that only a starting position x and a step h are given. If $f(x+h) < f(x)$, the step will be called a success, otherwise it will be called a failure. In case of success, it is obvious that we will replace u by $u + h$, increase the step size h, and try again. However, in case of failure, we have to reduce the step size h and/or change its sign, since either the step size h is too large or we are searching in the wrong direction. To adjust the step size h, we use two adjustment factors $A1$ and $A2$ which are also called success-failure factors. To enlarge the step size h upon success, we update h as $h = A2 * h$. To reduce h upon failure, we update h as $h = A1 * h$. In the present implementation, I choose the success-failure factors $(A2, A1)$ as $(1.5, -0.25)$. Other choices of $(A2, A1)$ such as $(2, -0.25)$ and $(3, -0.5)$, are also as efficient as $(1.5, -0.25)$. Although one choice may be slightly more efficient in some specific problems, they are the same in general problems.

It should be pointed out that the success-failure algorithm only compares the function values. When we step further to consider the changing rate of the function values respect to the step size h, we come up with the idea of using the function values and their associated points to generate some simple functions to describe approximately the curve of the function nearby. This idea leads to interpolation and extrapolation. One of the most popular interpolation methods is parabolic interpolation which is using polynomials of degree 2 to interpolate a given function $f(x)$ among the points at which function value has been computed.

Notice that the success-failure algorithm keeps searching upon success and stops upon failure. We have only three cases when we apply parabolic interpolation after the success-failure process. We define $b0$, $b1$, $b2$ as the latest three point selected in the success-failure process and $P0$, $P1$ and $P2$ are their function values accordingly.

(i) initial point, success, failure $(b0, b1, b2)$;

(ii) initial point, failure, failure $(b1, b0, b2)$;

(iii) success, success, failure $(b0, b1, b2)$.

For all three cases above, $b1$ is the point at which function f has the lowest function value $P1$, i.e. $P1 \leq P0$ and $P1 \leq P2$.

Assume $SMSTEP = b1 - b0$, $STEP = b2 - b1$, and X is the distance between the minimum of the interpolated parabola and $b1$. The parabolic interpolation formula we apply here is

$$X = 0.5 \frac{(P0 - P1) * STEP^2 - (P - P1) * SMSTEP^2}{(P0 - P1) * STEP - (P - P1) * SMSTEP} \tag{20}$$

Combining the success-failure algorithm with inverse interpolation, we come up with the Nash line search method. The Nash line search method has linear convergence. The flow charts of this line search method are shown in Figure 1, Figure 2 and Figure 3.

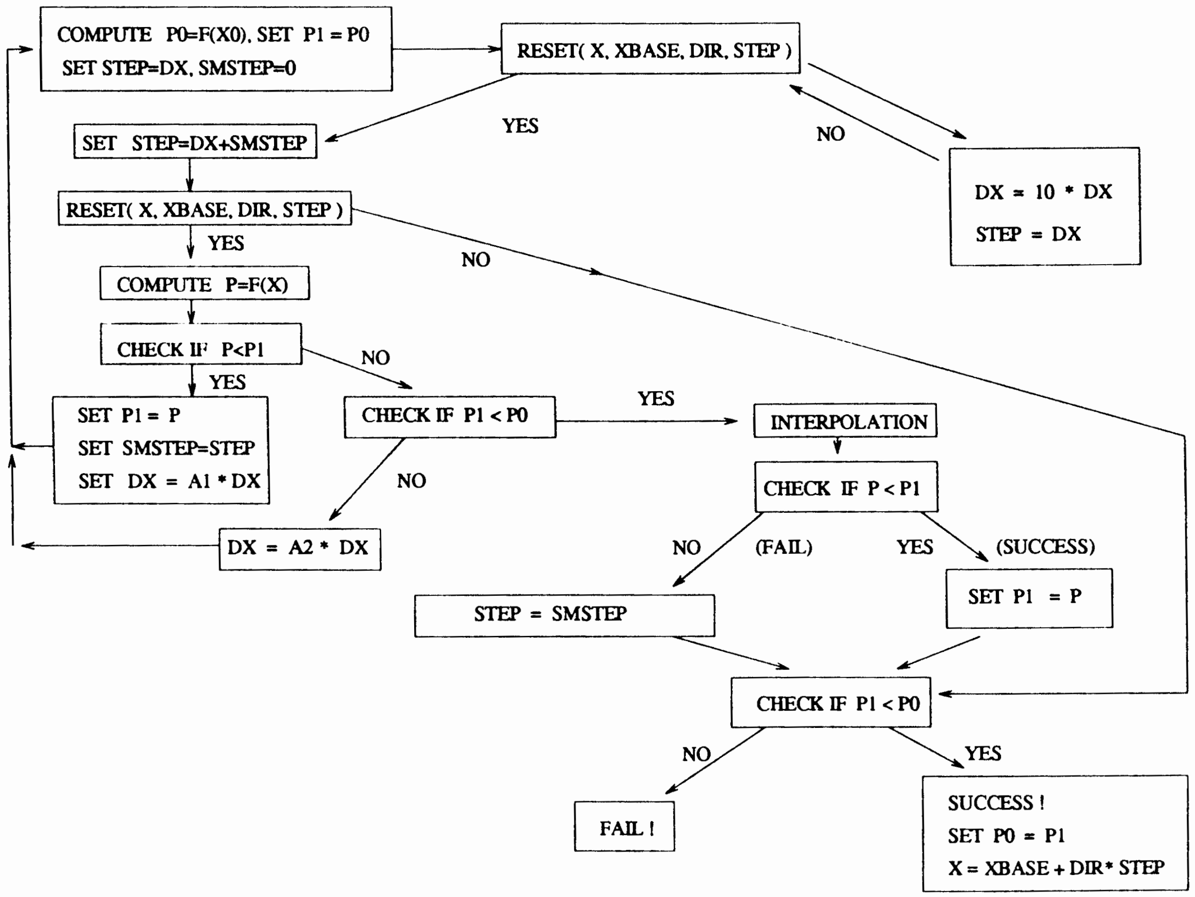

Figure 1. Flow Chart for Nash Line Search

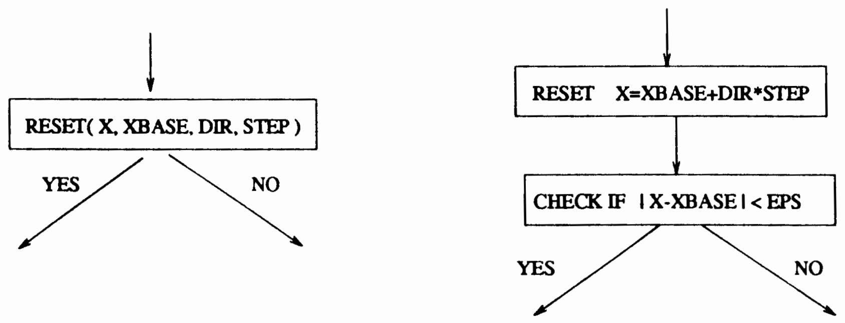

Figure 2. Flow Chart for Reset Subroutine

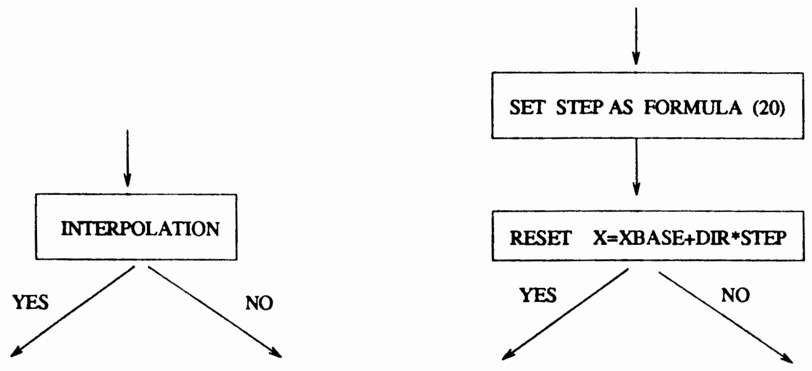

Figure 3. Flow Chart for Interpolation Subroutine

CHAPTER IV

NUMERICAL EXPERIMENTS

The test code CGCOMP examined in this chapter is a subroutine package which implements several CG methods in FORTRAN 77. It is written with line search options, strategy options, and restart options. Line search options include the Brent line search method and the Nash line search method. Strategy options include four traditional CG methods, the Fletcher-Reeves method, the Polak-Ribiere method, the Beale-Sorenson method and the Perry method. Restart options include a traditional restart method, Powell's restart method, and Shanno's restart method. The Brent line search subroutine which I use in my code was originally modified by Chandler from Brent's version [4].

Stopping criteria

There exist two kinds of stopping criteria. One is to use the step size Dx, the other is to use the gradient $\nabla F(x)$. The first criterion is to decide to stop or not based on the condition below

$$|Dx| \leq eps * ||x|| + t \tag{21}$$

where Dx is the size of the change to be made in the x vector in the first step of the line search. $||x||$ is the Euclidean norm of the x vector. eps is the relative tolerance and t is the absolute tolerance which are the same as formula (18). For some reason, this criterion is not working very well in my implementation, therefore I choose the second criterion which uses $\nabla F(x)$:

$$||\nabla F(x)|| \leq t \tag{22}$$

where t is defined as above.

Notice that the gradient $\nabla F(x)$ is scale dependent. However, this scale dependency may result in stopping of the minimizing process of the CG methods without approaching the minimal point close enough. To prevent this happening, we would like to use the scaled gradient when we use the second stopping criterion. Marquardt [21] states in his paper that for nonlinear least squares problems, we can compute its scaled gradient as below. In a typical nonlinear least squares problem in the field of curve-fitting, there is a set of m data points

$$\{(t_k, y_k) | k = 1, \ldots, m\},$$

and a modeling function $f(t_k, x)$ is given where x is an n-dimensional parameter vector to be determined such that the sum of squares of residuals

$$S(x) = \sum_{k=1}^{m} (y_k - f(t_k, x))^2$$

is minimized. The error $y_k - f(t_k, x)$ is called the k-th residual $r_k(x)$. Assume $x = (x_1, x_2, ..., x_n)$, and

$$P = (P_{ij})_{n \times n} = (\frac{\partial f(t_i, x)}{\partial x_j})_{n \times n}$$

Then the unscaled gradient is

$$\nabla S(x) = (\frac{\partial S(x)}{\partial x_j})_n$$

where $\frac{\partial S(x)}{\partial x_j} = 2 \sum_{k=1}^{m} (y_k - f(t_k, x)) * P_{kj}$.

Now we can scale the unscaled gradient by dividing every element $\frac{\partial S(x)}{\partial x_j}$, $j = 1$, 2, ..., n by the square root of the $i \times i$ element of matrix $P^T P$

$$Scaled\ gradient = (\frac{\frac{\partial S(x)}{\partial x_j}}{\sqrt{(P^T P)_{jj}}})_n$$

This scaled gradient is invariant under $x \rightarrow C * x$; C is a nonzero constant here. Since the test functions I chose are all nonlinear least square problems, I chose the second stopping criterion using this scaled gradient and tolerance $t = 10^{-5}$ in the stopping test formula (22).

Restart criteria

As stated in chapter II, the traditional restart method is to restart the conjugate gradient method in the steepest descent direction every n or (n+1) iterations. Powell's and Shanno's restart methods both use one of the traditional CG directions as the starting search direction at the beginning. Once the restart criterion (9) is satisfied or it is the end of a cycle of n iterations, they will generate their restart direction based on the current search direction and the previous search direction d_t. The following $(n-1)$ nonrestart subsequent search directions will be generated based on d_t, the search direction before restart, and the updated previous search direction. Notice that the above n search directions lead to n nonrestart CG iterations. We know that the traditional CG direction used as the starting search direction will not be applied on the next $(n-1)$ nonrestart iterations [8] [12]. In my experiment, it happens, though not often, that the line search fails to find a minimal point along Powell's restart direction and Shanno's restart direction. In this case, we will restart from the original CG direction. If it also can not make progress along the original CG direction, we will restart the conjugate gradient method by the steepest descent direction.

All of the numerical results are obtained in double precision on a SUN (SPARC IPX) station, using f77, the Sun FORTRAN compiler. All of the numerical tests are carried out on the same set of test problems that are described below.

Test Problems

The comparison of various CG methods requires a suitable set of test problems. A relatively large collection of 13 test functions used and referenced by (Moré *et al.*, 1981) is used in my numerical experiments. For easy reference, this set is listed in Appendix A with the following format:

Function number. Name of the function (reference)

- Dimensions (m, n)
- Function definition $(r_k(x))$
- Standard starting point $(x^{(0)})$
- Minima $(x^*,\ ||r(x^*)||^2)$

The test set represents various problems, including an almost linear problem (function 13), problems involving exponentials and triangles (function 2,3,4, 10, 11, and 12), a singular problem (function 5) and data-fitting problems (function 2, 3, 7, and 8).

Notice that $F(x)$ and $\nabla F(x)$ often contain common subexpressions such as exponential and trigonometric subexpressions. It is efficient to evaluate such subexpressions only once. In our program, any evaluation of $J(x)$ is always preceded by an evaluation of $r(x)$, and the evaluations of $F(x)$ and $\nabla F(x)$ of each problem are organized in a single subroutine by using ENTRY statements. Thus the values of the common subexpressions, calculated during the evaluation of $F(x)$, can be stored by a SAVE statement, and can be used in the calculation of $\nabla F(x)$ for the same value of x.

To prevent destructive overflows of exponential functions due to too large a steplength, my package has a subroutine CHDEXP to check the exponent in the subprograms of function evaluation and gradient before evaluation of the power. If the exponent is greater than a preset bound, then the subprogram returns a flag $IOVER$ to indicate the need to reduce the steplength.

Results and Analysis

Various CG methods using different strategies, different restart methods, and different line search methods were tested on the same set of test problems described in the previous section and in Appendix A, using the same tolerance 10^{-5} for the scaled gradient test. The results are shown in TABLE I through TABLE IV, where NPROB, NFV, and NGV are the problem number, number of function evaluations and number of gradient evaluations respectively. N is the dimension of the function, M is the number of data points used in the function. METHCG is defined as

1=Fletcher-Reeves method

2=Polak-Ribiere method

3=Beale-Sorenson method

4=Perry method

and LINEOP=2 indicates that the Brent line search method is used , LINEOP=1 indicates the use of the Nash line search.

From TABLE I, TABLE II and TABLE III in which the same (Brent) line search method is applied, I propose the following points based on the comparison of the number of function evaluations (NFV) and the number of gradient evaluations (NGV).

(1) With the traditional restart method, the four traditional CG methods: the Fletcher-Reeves method, the Polak-Ribiere method, the Beale-Sorenson method, and the Perry method, are relatively equivalent. Each of these methods has better performances over some test functions and worse performances over the other test functions when it is compared to the others. There is no evidence to show any of them is superior to the others. Especially, the statement that the Fletcher-Reeves method [3] is inferior to other existing CG methods is not supported here (see TABLE I).

(2) Powell's and Shanno's restart methods using different traditional CG direction as starting direction have different performances. I believe that this phenomenon comes from the fact that these two restart methods use not only the

TABLE I

NUMERICAL RESULT USING TRADITIONAL RESTART

METHCG	LINEOP	NPROB	N	M	NFV	NGV
1	2	1	2	2	140	42
1	2	2	5	33	3654	751
1	2	3	11	65	1443	437
1	2	4	3	3	160	45
1	2	5	4	4	212	58
1	2	6	2	2	80	17
1	2	7	3	15	102	23
1	2	8	4	11	208	50
1	2	9	31	31	725	174
1	2	10	3	5	52	12
1	2	11	2	5	43	13
1	2	12	4	5	125	34
1	2	13	10	10	36	9
2	2	1	2	2	166	48
2	2	2	5	33	5876	1226
2	2	3	11	65	1316	399
2	2	4	3	3	159	45
2	2	5	4	4	193	48
2	2	6	2	2	72	14
2	2	7	3	15	95	22
2	2	8	4	11	183	48
2	2	9	31	31	1596	387
2	2	10	3	5	37	9
2	2	11	2	5	45	13
2	2	12	4	5	128	36
2	2	13	10	10	45	12
3	2	1	2	2	137	44
3	2	2	5	33	7093	1457
3	2	3	11	65	1306	385
3	2	4	3	3	256	75
3	2	5	4	4	116	31
3	2	6	2	2	45	9
3	2	7	3	15	113	28
3	2	8	4	11	177	42
3	2	9	31	31	1562	421
3	2	10	3	5	50	11
3	2	11	2	5	41	10
3	2	12	4	5	127	36
3	2	13	10	10	62	15
4	2	1	2	2	135	44
4	2	2	5	33	7020	1435
4	2	3	11	65	986	291
4	2	4	3	3	353	17
4	2	5	4	4	116	32
4	2	6	2	2	28	7
4	2	7	3	15	83	20
4	2	8	4	11	348	86
4	2	9	31	31	2098	528
4	2	10	3	5	51	11
4	2	11	2	5	41	10
4	2	12	4	5	117	33
4	2	13	10	10	54	15

TABLE II

NUMERICAL RESULT USING POWELL'S RESTART

METHCG	LINEOP	NPROB	N	M	NFV	NGV
1	2	1	2	2	212	55
1	2	2	5	33	2813	6985
1	2	3	11	65	2111	588
1	2	4	3	3	186	42
1	2	5	4	4	547	134
1	2	6	2	2	88	19
1	2	7	3	15	168	43
1	2	8	4	11	717	180
1	2	9	31	31	7514	1842
1	2	10	3	5	160	40
1	2	11	2	5	65	19
1	2	12	4	5	264	72
1	2	13	10	10	111	30
2	2	1	2	2	321	72
2	2	2	5	33	3775	853
2	2	3	11	65	1604	439
2	2	4	3	3	316	71
2	2	5	4	4	264	66
2	2	6	2	2	119	25
2	2	7	3	15	317	64
2	2	8	4	11	662	152
2	2	9	31	31	4572	1147
2	2	1 0	3	5	93	20
2	2	1 1	2	5	62	17
2	2	1 2	4	5	158	43
2	2	1 3	10	10	165	38
3	2	1	2	2	117	35
3	2	2	5	33	112	262
3	2	3	11	65	738	216
3	2	4	3	3	4800	796
3	2	5	4	4	110	28
3	2	6	2	2	50	10
3	2	7	3	15	83	19
3	2	8	4	11	113	31
3	2	9	31	31	2458	651
3	2	10	3	5	74	15
3	2	11	2	5	41	10
3	2	12	4	5	100	29
3	2	13	10	10	114	24
4	2	1	2	2	129	33
4	2	2	5	33	165	271
4	2	3	11	65	758	230
4	2	4	3	3	158	40
4	2	5	4	4	118	29
4	2	6	2	2	57	11
4	2	7	3	15	99	21
4	2	8	4	11	146	39
4	2	9	31	31	2743	756
4	2	10	3	5	79	16
4	2	11	2	5	41	10
4	2	12	4	5	100	29
4	2	13	10	10	86	19

TABLE III

NUMERICAL RESULT USING
SHANNO'S RESTART

METHCG	LINEOP	NPROB	N	M	NFV	NGV
1	2	1	2	2	146	37
1	2	2	5	33	1148	290
1	2	3	11	65	1587	451
1	2	4	3	3	824	147
1	2	5	4	4	313	72
1	2	6	2	2	51	11
1	2	7	3	15	122	32
1	2	8	4	11	1385	357
1	2	9	31	31	2434	669
1	2	10	3	5	139	26
1	2	11	2	5	40	12
1	2	12	4	5	969	27
1	2	13	10	10	51	11
2	2	1	2	2	146	38
2	2	2	5	33	2132	52
2	2	3	11	65	1697	483
2	2	4	3	3	735	150
2	2	5	4	4	446	16
2	2	6	2	2	64	13
2	2	7	3	15	153	36
2	2	8	4	11	366	90
2	2	9	31	31	3370	940
2	2	10	3	5	144	32
2	2	11	2	5	32	9
2	2	12	4	5	636	156
2	2	13	10	10	51	11
3	2	1	2	2	146	37
3	2	2	5	33	554	153
3	2	3	11	65	1164	332
3	2	4	3	3	5586	995
3	2	5	4	4	214	55
3	2	6	2	2	52	10
3	2	7	3	15	119	24
3	2	8	4	11	801	195
3	2	9	31	31	947	249
3	2	10	3	5	83	19
3	2	11	2	5	39	9
3	2	12	4	5	572	150
3	2	13	10	10	77	18
4	2	1	2	2	139	34
4	2	2	5	33	557	160
4	2	3	11	65	1875	514
4	2	4	3	3	***	***
4	2	5	4	4	282	72
4	2	6	2	2	56	11
4	2	7	3	15	121	26
4	2	8	4	11	964	248
4	2	9	31	31	1439	374
4	2	10	3	5	81	17
4	2	11	2	5	39	9
4	2	12	4	5	513	129
4	2	13	10	10	32	7

TABLE IV

NUMERICAL RESULT USING
TRADITIONAL RESTART

METHCG	LINEOP	NPROB	N	M	NFV	NGV
1	1	1	2	2	227	31
1	1	2	5	33	68823	7343
1	1	3	11	65	2105	403
1	1	4	3	3	160	28
1	1	5	4	4	415	66
1	1	6	2	2	69	10
1	1	7	3	15	154	20
1	1	8	4	11	239	37
1	1	9	31	31	945	147
1	1	10	3	5	172	27
1	1	11	2	5	71	10
1	1	12	4	5	167	30
1	1	13	10	10	55	8
2	1	1	2	2	213	31
2	1	2	5	33	26549	2748
2	1	3	11	65	1830	377
2	1	4	3	3	235	46
2	1	5	4	4	266	41
2	1	6	2	2	83	12
2	1	7	3	15	154	24
2	1	8	4	11	484	79
2	1	9	31	31	3086	449
2	1	10	3	5	87	11
2	1	11	2	5	46	7
2	1	12	4	5	178	35
2	1	13	10	10	66	9
3	1	1	2	2	212	28
3	1	2	5	33	75277	7918
3	1	3	11	65	1713	343
3	1	4	3	3	438	83
3	1	5	4	4	337	62
3	1	6	2	2	63	9
3	1	7	3	15	102	17
3	1	8	4	11	303	43
3	1	9	31	31	3030	452
3	1	10	3	5	86	11
3	1	11	2	5	46	7
3	1	12	4	5	127	27
3	1	13	10	10	72	11
4	1	1	2	2	212	28
4	1	2	5	33	71764	7594
4	1	3	11	65	1446	281
4	1	4	3	3	165	31
4	1	5	4	4	191	30
4	1	6	2	2	73	9
4	1	7	3	15	155	23
4	1	8	4	11	525	82
4	1	9	31	31	2918	426
4	1	10	3	5	74	9
4	1	11	2	5	46	7
4	1	12	4	5	194	36
4	1	13	10	10	66	9

latest search direction but also the previous search direction when they generate the new restart direction and following nonrestart subsequent direction. They do not abandon the second derivative information that is found by the search along the previous search direction (see TABLE II and TABLE III).

(3) Powell's restart method using the Beale-Sorenson direction or the Perry direction as the starting search direction performs better than that using the Fletcher-Reeves direction or the Polak-Ribiere direction over the whole set of test functions. Powell's restart method using the Beale-Sorenson direction or the Perry direction as the starting search direction also performs better than the traditional restart method with the corresponding search direction. However, Powell's restart method using the Fletcher-Reeves direction or the Polak-Ribiere direction as the starting search direction performs worse than the traditional restart method using the corresponding search direction over most of the test functions (see TABLE II).

(4) There is no evidence to show that Shanno's restart method using a traditional CG direction as starting search direction is superior or inferior to the traditional restart method with the corresponding search direction. There is also no evidence to show that Shanno's restart method using one of the traditional CG directions as starting search direction is superior or inferior to that using another of the traditional CG direction. Shanno's restart method using the Fletcher-Reeves direction or the Polak-Ribiere direction as the starting search direction performs better than Powell's restart method using the corresponding traditional CG direction as starting search direction over most of the test functions. Shanno's restart method using the Perry direction as the starting search direction has trouble in solving the problem of function 4 (Helical valley function) within the limit of maximum number of function evaluations 10^6

(I use *** to indicate this). However it has come to a point quite close to the minimal point (see TABLE I, II, III).

Based on TABLE I and TABLE IV, I give the preference to the Brent line search method as the better line search method. With the traditional restart method, all four traditional CG methods with the Brent line search method perform better than that with the Nash line search method over all the test functions except function 1 (Rosenbrock function). It is mainly because the Brent line search method has superlinear convergent rate while the Nash line search method has only a linearly convergent rate. The numerical results collected in TABLE IV also verify the statement (1) above that the four traditional CG methods are relatively equivalent.

CHAPTER V

CONCLUSIONS

This study gives a comparison among four traditional CG methods, the Fletcher-Reeves method, the Polak-Ribiere method, the Beale-Sorenson method and the Perry method. A comparison is also made of the restart methods using the four traditional methods to generate the starting direction. The numerical experiment results lead to the following conclusion:

- The four traditional CG methods themselves are relatively equivalent.
- Powell's restart method using the Beale-Sorenson or the Perry direction as starting search direction is to be tried first when we apply CG method to solve a nonlinear minimization problem.

Further research might be done in the following aspects:

- In practice, it is often the case that CG methods are applied to a very large dimension problem. It is very desirable to test CG methods on large dimension problems.
- It is desirable to find a method which can obtain a scaled gradient of a general nonlinear function (not just a sum of squares) so that the scaled gradient applied to the stopping test does not depend on the scale of the function value.

A SELECTED BIBLIOGRAPHY

1. Polak, E., Computational Methods in Optimization, Mathematics in Science and Engineering, Vol. 77, Academic Press, New York and London, 1971.

2. Hestenes, M., Conjugate Direction Methods in Optimization, Springer-Verlag, 1980.

3. Nash, J. C., Compact Numerical Methods for Computers: Linear Algebra and Function Minimization, John Wiley & Sons, New York,1979.

4. Brent, R. P., Algorithms for Minimization Without Derivatives, Prentice-Hall, Englewood Cliffs, New Jersey,1973.

5. Beale, E. M. L., A Derivation of Conjugate Gradients, in F. A. Lootsma, ed., Numerical Methods for Nonlinear Optimization, Academic Press , New York, 1972.

6. Fletcher, R. and Reeves, C. M., Function Minimization by Conjugate Gradients, The Computer Journal, Vol. 7, pp. 149-154, 1964.

7. Perry, A., A Modified Conjugate Gradient Algorithm, Operations Research, Vol. 26, pp. 1073-1078, 1978.

8. Powell, M. J. D., Restart Procedures for the Conjugate Gradient Method, Mathematical Programming, Vol. 12, pp. 241-254, 1977.

9. Kowalik, J. and Osborne, M. R., Methods for Unconstrained Optimization Problems, American Elsevier Publishing Company, New York, 1968.

10. Davidon, W. C., Optimally Conditioned Optimization Algorithms Without Line Searches, Mathematical Programming, Vol. 9, pp. 1-30, 1975.

11. Fletcher, R. and Powell, W. J. D., A Rapidly Convergent Descent Method for Minimization, The Computer Journal, Vol. 6, pp. 163-168, 1963.

12. Shanno, D. F. and Phua, K. H., Matrix Conditioning and Non-linear Optimization, Mathematical Programming, Vol. 14, pp. 149-160, 1978.

13. Powell, M. J. D., An Efficient Method for Finding the Minimum of a Function of Several Variables Without Calculating Derivatives, The Computer Journal, Vol. 7, pp. 155-162,1964.

14. Sorenson, H. W., Comparison of Some Conjugate Direction Procedures for Function Minimization, Journal of the Franklin Institute, Vol. 288, No. 6, pp. 421-441, 1969.

15. Box, M. J., A Comparison of Several Current Optimization Methods and the Use of Transformation in Constrained Problems, The Computer Journal, Vol. 9, pp. 67-77, 1966.

16. Polak, E. and Ribiere, G., Note sur la Convergences Des Methods Conjuges, Rev, Fr. Inr. Rech. Oper. Vol. 16, pp. 35-43, 1969.

17. Le, D., A Fast and Robust Unconstrained Optimization Method Requiring Minimum Storage, Mathematical Programming, Vol. 32, pp. 41-68, 1985.

18. Crowder, H. P. and Wolfe, P., Linear convergence of the conjugate gradient method, IBM Journal of Research and Development, Vol. 16, pp. 431-433, 1972.

19. Powell, M. J. D., Some convergence properties of the conjugate gradient method, Mathematical Programming, Vol. 11, pp. 42-49, 1976.

20. McGuire, M. F. and Wolfe, P., Evaluating a restart procedure for conjugate gradients, Report RC-4382, IBM Research Center, Yorktown Heights.

21. Marquardt, D. W., Solution of nonlinear chemical engineering models, Chemical Engineering Progress, Vol. 55, No. 5, pp. 65-70, 1959.

APPENDIX A

TEST FUNCTIONS

1. Rosenbrock function (Rosenbrock, 1960)

   - $m = n = 2$
   - $r_1(x) = 10(x_2 - x_1^2)$
     $r_2(x) = 1 - x_1$
   - $x^{(0)} = (-1.2, 1)$
   - $||r^*||^2 = 0$ at $(1, 1)$

2. Osborne 1 function (Osborne, 1972)

   - $n = 5$, $m = 33$
   - $r_i(x) = y_i - (x_1 + x_2 e^{-10(i-1)x_4} + x_3 e^{-10(i-1)x_5})$,
     where the y_i's are as in TABLE V.
   - $x^{(0)} = (0.5, 1.5, -1, 0.01, 0.02)$
   - $||r^*||^2 = 5.46489 \ldots 10^{-5}$

TABLE V

DATA FOR FUNCTION 2

i	y_i	i	y_i	i	y_i	i	y_i	i	y_i
1	0.844	8	0.850	15	0.628	22	0.490	29	0.424
2	0.908	9	0.818	16	0.603	23	0.478	30	0.420
3	0.932	10	0.784	17	0.580	24	0.467	31	0.414
4	0.925	11	0.751	18	0.558	25	0.457	32	0.411
5	0.925	12	0.718	19	0.538	26	0.448	33	0.406
6	0.908	13	0.685	20	0.522	27	0.438		
7	0.881	14	0.658	21	0.506	28	0.431		

3. Osborne 2 function (Osborne, 1972)

- $n = 11, m = 65$
- $r_i(x) = y_i - (x_1 e^{-\frac{i-1}{10}x_5} + x_2 e^{-(\frac{i-1}{10}-x_9)^2 x_6} + x_3 e^{-(\frac{i-1}{10}-x_{10})^2 x_7} + x_4 e^{-(\frac{i-1}{10}-x_{11})^2 x_8})$, where the y_i's are as in TABLE VI.

TABLE VI

DATA FOR FUNCTION 3

i	y_i	i	y_i	i	y_i	i	y_i	i	y_i
1	1.366	14	0.665	27	0.612	40	0.429	53	0.597
2	1.191	15	0.616	28	0.558	41	0.523	54	0.625
3	1.112	16	0.606	29	0.533	42	0.562	55	0.739
4	1.013	17	0.602	30	0.495	43	0.607	56	0.710
5	0.991	18	0.626	31	0.500	44	0.653	57	0.729
6	0.885	19	0.651	32	0.432	45	0.672	58	0.720
7	0.831	20	0.724	33	0.395	46	0.708	59	0.636
8	0.847	21	0.649	34	0.375	47	0.633	60	0.581
9	0.786	22	0.649	35	0.372	48	0.668	61	0.428
10	0.725	23	0.694	36	0.391	49	0.645	62	0.292
11	0.746	24	0.644	37	0.396	50	0.632	63	0.162
12	0.679	25	0.624	38	0.405	51	0.591	64	0.098
13	0.608	26	0.661	39	0.428	52	0.559	65	0.054

- $x^{(0)} = (1.3, 0.65, 0.65, 0.7, 0.6, 3, 5, 7, 2, 4.5, 5.5)$
- $||r^*||^2 = 4.01377\ldots 10^{-2}$

4. Helical valley function (Fletcher and Powell, 1963)

   - $m = n = 3$
   - $r_1(x) = 10(x_3 - 10t)$
     $r_2(x) = 10(\sqrt{x_1^2 + x_2^2} - 1)$
     $r_3(x) = x_3$
     where
     $$t = \begin{cases} \frac{1}{2\pi}\arctan\frac{x_2}{x_1}, & x_1 > 0 \\ \text{sign}(0.25, x_2), & x_1 = 0 \\ \frac{1}{2\pi}\arctan\frac{x_2}{x_1} + 0.5, & x_1 < 0 \end{cases}$$
   - $x^{(0)} = (-1, 0, 0)$
   - $||r^*||^2 = 0$ at $(1, 0, 0)$

5. Powell singular function (Powell, 1962)

   - $m = n = 4$
   - $r_1(x) = x_1 + 10x_2$
     $r_2(x) = \sqrt{5}(x_3 - x_4)$
     $r_3(x) = (x_2 - 2x_3)^2$
     $r_4(x) = \sqrt{10}(x_1 - x_4)^2$
   - $x^{(0)} = (3, -1, 0, 1)$
   - $||r^*||^2 = 0$ at $(0, 0, 0, 0)$

6. Freudenstein and Roth function (Freudenstein and Roth, 1963)

   - $m = n = 2$

- $r_1(x) = -13 + x_1 + ((5 - x_2)x_2 - 2)x_2$
  $r_2(x) = -29 + x_1 + ((1 + x_2)x_2 - 14)x_2$
- $x^{(0)} = (0.5, -2)$
- $||r^*||^2 = 0$ at $(5, 4)$
  $||r^*||^2 = 48.9842\ldots$ at $(11.41\ldots, -0.8968\ldots)$

7. Bard function (Bard, 1970)

- $n = 3$, $m = 15$
- $r_i(x) = y_i - \left(x_1 + \dfrac{i}{(16-i)x_2 + \min(i, 16-i)x_3}\right)$,
  where the y_i's are as in TABLE VII.
- $x^{(0)} = (1, 1, 1)$
- $||r^*||^2 = 8.21487\ldots 10^{-3}$
  $||r^*||^2 = 17.4286\ldots$ at $(0.8406\ldots, -\infty, -\infty)$

TABLE VII

DATA FOR FUNCTION 7

i	y_i	i	y_i	i	y_i	i	y_i	i	y_i
1	0.14	4	0.25	7	0.35	10	0.58	13	1.34
2	0.18	5	0.29	8	0.39	11	0.73	14	2.10
3	0.22	6	0.32	9	0.37	12	0.96	15	4.39

8. Kowalik and Osborne function (Kowalik and Osborne, 1968)

- $m = 4$, $n = 11$
- $r_i(x) = y_i - \dfrac{x_1 u_i(u_i + x_2)}{u_i(u_i + x_3) + x_4}$,
  where the y_i's and the u_i's are as in TABLE VIII.

- $x^{(0)} = (0.25, 0.39, 0.415, 0.39)$
- $||r^*||^2 = 3.07505\ldots 10^{-4}$
  $||r^*||^2 = 1.02734\ldots 10^{-3}$ at $(+\infty, -14.07\ldots, -\infty, -\infty)$

TABLE VIII

DATA FOR FUNCTION 8

i	y_i	u_i	i	y_i	u_i	i	y_i	u_i
1	0.1957	4.0000	5	0.0844	0.2500	9	0.0323	0.0833
2	0.1947	2.0000	6	0.0627	0.1670	10	0.0235	0.0714
3	0.1735	1.0000	7	0.0456	0.1250	11	0.0246	0.0625
4	0.1600	0.5000	8	0.0342	0.1000			

9. Watson function (Kowalik and Osborne, 1968)

- $2 \leq n \leq 31$, $m = 31$
- $$r_i(x) = \begin{cases} \sum_{j=2}^{n}(j-1)\left(\frac{i}{29}\right)^{j-2} x_j - \left(\sum_{j=1}^{n}\left(\frac{i}{29}\right)^{j-1} x_j\right)^2 - 1, & i \leq 29 \\ x_1, & i = 30 \\ x_2 - x_1^2 - 1, & i = 31 \end{cases}$$
- $x^{(0)} = (0, \ldots, 0)$
- $||r^*||^2 = 2.28767\ldots 10^{-3}$ if $n = 6$
  $||r^*||^2 = 1.39976\ldots 10^{-6}$ if $n = 9$
  $||r^*||^2 = 4.72238\ldots 10^{-10}$ if $n = 12$

10. Box three-dimensional function (Box, 1966)

- $n = 3$, $m \geq n$
- $r_i(x) = e^{-0.1ix_1} - e^{-0.1ix_2} - x_3(e^{-0.1i} - e^{-i})$

- $x^{(0)} = (0, 10, 20)$
- $||r^*||^2 = 0$ at $(1, 10, 1), (10, 1, -1)$ and wherever $x_1 = x_2$ and $x_3 = 0$

11. Jennrich and Sampson function (Jennrich and Sampson, 1968)

- $n = 2$,$m \geq n$
- $r_i(x) = 2 + 2i - (e^{ix_1} + e^{ix_2})$
- $x^{(0)} = (0.3, 0.4)$
- $||r^*||^2 = 124.362\ldots$ at $x_1 = x_2 = 0.2578\ldots$ for $m = 10$

12. Brown and Dennis function (Brown and Dennis, 1971)

- $n = 4$, $m \geq n$
- $r_i(x) = (x_1 + \frac{i}{5}x_2 - e^{\frac{i}{5}})^2 + (x_3 + x_4 \sin\frac{i}{5} - \cos\frac{i}{5})^2$
- $x^{(0)} = (25, 5, -5, -1)$
- $||r^*||^2 = 85822.2\ldots$ if $m = 20$

13. Brown almost-linear function (Brown, 1969)

- $n = m$
- $r_i(x) = \begin{cases} x_i + \sum_{j=1}^{n} x_j - (n+1), & 1 \leq i < n \\ \prod_{j=1}^{n} x_j - 1, & i = n \end{cases}$
- $x^{(0)} = (\frac{1}{2}, \ldots, \frac{1}{2})$
- $||r^*||^2 = 0$ at $(\alpha, \ldots, \alpha, \alpha^{1-n})$,
  where $\alpha \neq 1$ and α satisfies $n\alpha^n - (n+1)\alpha^{n-1} + 1 = 0$
  $||r^*||^2 = 1$ at $(0, \ldots, 0, n+1)$

APPENDIX B

PROGRAM LISTING

```
C
C   CGCOMP                         WRITTEN BY MEISHAN CHENG
C
C       IMPLEMENTING VARIOUS CONJUGATE GRADIENT METHODS
C
C
      INTEGER METHCG,MAXCYC,MAXNFV,MINLOC,NOREST,NTRY,
     *  NTRACC,NTRACX,LPCG,KFLAGC,NCYCLE,ITERCG,MAXFWD,
     *  NFMAXB,NFMAXL,KFLAGD,KONNEW,LP,MASK,MASKT,NV
C
      DOUBLE PRECISION DX,DXSAVE,RELTLC,ABSTLC,GRNORM,
     *  RELTLD,ABSTLD,DECMAX,FBINCR,FBDECR
      DOUBLE PRECISION GRAD,GRPREV,GRSAVE,RLDFMX,X,XDIR,
     *  XDSAVE,XMAX,XMIN,XSAVE,XBASE
      DOUBLE PRECISION HUGE
C
      DIMENSION X(40),XSAVE(40),GRAD(40),GRSAVE(40),
     *  GRPREV(40),XDIR(40),XDSAVE(40),XBASE(40),
     *  XMAX(40),XMIN(40),MASK(40),MASKT(40),
     *  HHG(40),HHY(40).XTDIR(40),YT(40)
C
      COMMON /COMCG/ DX,DXSAVE,RELTLC,ABSTLC,GRNORM,NFV,NGV,
     *  RELTLD,ABSTLD,DECMAX,FBINCR,FBDECR,LINEOP,METHRT,
     *  METHCG,MAXCYC,MAXNFV,MINLOC,NOREST,NTRY,NTRACC,
     *  NTRACX,LPCG,KFLAGC.NCYCLE,ITERCG,MAXFWD,NFMAXB,
     *  NFMAXL,KFLAGD,KONNEW
C
      COMMON /NP/NPROB
C
      COMMON /IFLAG1/IOVER
      COMMON /IFLAG2/IUNDER
      COMMON /NTLSQ/MT
      COMMON /LO/LOUT
C
      PRINT *,' METHOD              # OF PROBLEM  # OF VARIBLE & LINEOP & RESTART'
      PRINT *,'1=FLETCHER-REEVES  1=Rosenbrock          2           1       0=Traditional'
      PRINT *,'2=POLAK-RIBIERE    2=Osborne I           5           2       1=Shanno'
      PRINT *,'3=BEALE-SORENSON   3=Osborne II         11                   2=Powell'
      PRINT *,'4=PERRY            4=Helical valley      3'
      PRINT *,'                   5=Powell Singlar      4'
      PRINT *,'                   6=Freudentein & Roth  2'
      PRINT *,'                   7=Brad                3'
      PRINT *,'                   8=Kowalik & Osborne   4'
      PRINT *,'                   9=Meyer               3'
      PRINT *,'                  10=Watson              2~31'
      PRINT *,'                  11=Box 3-DIM           3'
      PRINT *,'                  12=Jennrich & Samspson 2'
      PRINT *,'                  13=Brown & Dennis      4'
      PRINT *,'                  14=Brown almost-linear option'
C
      HUGE=1.0D30
      IN=5
     READ(IN,5)METHCG,NPROB,NV,LINEOP,METHRT
    5 FORMAT(I2,I3,I3,I2,I2)
```

```
C
      CALL INIT(NV,X)
C
      CALL CGSET
C
      LP=LPCG
C
      DO 10 J=1,NV
        MASK(J)=0
        XMAX(J)=HUGE
        XMIN(J)=-HUGE
   10   CONTINUE
C
      CALL GRCHEK (NV,X,MASK,GRAD,GRSAVE,LP,RLDFMX)
C
      CALL CG (NV,X,XSAVE,GRAD,GRSAVE,GRPREV,XDIR,XDSAVE,
     *  XBASE,XMAX,XMIN,MASK,MASKT,HHG,HHY,XTDIR,YT)
C
      WRITE(LOUT,30)METHCG,METHRT,LINEOP,NPROB,NFV,NGV
   30 FORMAT(I2,9X,I2,7X,I2,6X,I3,5X,I6,7X,I6)
C
   50 CONTINUE
C
   60 CONTINUE
C
      CLOSE(LOUT)
      STOP
      END
C
C
C
C     ***************************************************************
           SUBROUTINE  INIT  (N,X)
C     ***************************************************************
C
      IMPLICIT REAL*8 (A-H,O-Z)
C
      DIMENSION X(N)
C
      COMMON /NP/NPROB
C
      GOTO (1,2,3,4,5,6,7,8,9,10,11,12,13,14)NPROB
C
    1 CALL INIT1(N,X)
          GO TO 99
C
    2 CALL INIT2(N,X)
          GO TO 99
C
    3 CALL INIT3(N,X)
          GO TO 99
C
    4 CALL INIT4(N,X)
          GO TO 99
C
    5 CALL INIT5(N,X)
          GO TO 99
C
    6 CALL INIT6(N,X)
          GO TO 99
C
    7 CALL INIT7(N,X)
          GO TO 99
C
    8 CALL INIT8(N,X)
          GO TO 99
C
```

```
   9 CALL INIT9(N,X)
        GO TO 99
C
  10 CALL INIT10(N,X)
        GO TO 99
C
  11 CALL INIT11(N,X)
        GO TO 99
C
  12 CALL INIT12(N,X)
        GO TO 99
C
  13 CALL INIT13(N,X)
        GO TO 99
C
  14 CALL INIT14(N,X)
        GO TO 99
C
  99 RETURN
     END

C
C    **************************************************************
          SUBROUTINE  FCN   (N,X,Y)
C    **************************************************************
C
     IMPLICIT REAL*8 (A-H,O-Z)
C
     DIMENSION X(N)
C
     COMMON /COMCG/ DX,DXSAVE,RELTLC,ABSTLC,GRNORM,NFV,NGV,
    *  RELTLD,ABSTLD,DECMAX,FBINCR,FBDECR,LINEOP,METHRT,
    *  METHCG,MAXCYC,MAXNFV,MINLOC,NOREST,NTRY,NTRACC,
    *  NTRACX,LPCG,KFLAGC,NCYCLE,ITERCG,MAXFWD,NFMAXB,
    *  NFMAXL,KFLAGD,KONNEW
C
     COMMON /NP/NPROB
     COMMON /IFLAG1/IOVER
C
     IOVER=0
C
     GOTO (1,2,3,4,5,6,7,8,9,10,11,12,13,14)NPROB
C
   1 CALL FCN1(N,X,Y)
        GO TO 99
C
   2 CALL FCN2(N,X,Y)
        GO TO 99
C
   3 CALL FCN3(N,X,Y)
        GO TO 99
C
   4 CALL FCN4(N,X,Y)
        GO TO 99
C
   5 CALL FCN5(N,X,Y)
        GO TO 99
C
   6 CALL FCN6(N,X,Y)
        GO TO 99
C
   7 CALL FCN7(N,X,Y)
        GO TO 99
C
   8 CALL FCN8(N,X,Y)
        GO TO 99
C
```

```
  9 CALL FCN9(N,X,Y)
      GO TO 99
C
 10 CALL FCN10(N,X,Y)
      GO TO 99
C
 11 CALL FCN11(N,X,Y)
      GO TO 99
C
 12 CALL FCN12(N,X,Y)
      GO TO 99
C
 13 CALL FCN13(N,X,Y)
      GO TO 99
C
 14 CALL FCN14(N,X,Y)
      GO TO 99
C
 99 NFV=NFV+1
C
    RETURN
    END
C
C
C    ****************************************************************
        SUBROUTINE  FGRAD  (N,X,GDUM)
C    ****************************************************************
C
    IMPLICIT REAL*8 (A-H,O-Z)
C
    DIMENSION X(N),GDUM(N)
C
    COMMON /COMCG/ DX,DXSAVE,RELTLC,ABSTLC,GRNORM,NFV,NGV,
   *  RELTLD,ABSTLD,DECMAX,FBINCR,FBDECR,LINEOP,METHRT,
   *  METHCG,MAXCYC,MAXNFV,MINLOC,NOREST,NTRY,NTRACC,
   *  NTRACX,LPCG,KFLAGC,NCYCLE,ITERCG,MAXFWD,NFMAXB,
   *  NFMAXL,KFLAGD,KONNEW
C
    COMMON /NP/NPROB
C
    GOTO (1,2,3,4,5,6,7,8,9,10,11,12,13,14)NPROB
C
  1 CALL GRAD1(N,X,GDUM)
      GO TO 99
C
  2 CALL GRAD2(N,X,GDUM)
      GO TO 99
C
  3 CALL GRAD3(N,X,GDUM)
      GO TO 99
C
  4 CALL GRAD4(N,X,GDUM)
      GO TO 99
C
  5 CALL GRAD5(N,X,GDUM)
      GO TO 99
C
  6 CALL GRAD6(N,X,GDUM)
      GO TO 99
C
  7 CALL GRAD7(N,X,GDUM)
      GO TO 99
C
  8 CALL GRAD8(N,X,GDUM)
      GO TO 99
C
  9 CALL GRAD9(N,X,GDUM)
```

```
      GO TO 99
C
 10 CALL GRAD10(N,X,GDUM)
      GO TO 99
C
 11 CALL GRAD11(N,X,GDUM)
      GO TO 99
C
 12 CALL GRAD12(N,X,GDUM)
      GO TO 99
C
 13 CALL GRAD13(N,X,GDUM)
      GO TO 99
C
 14 CALL GRAD14(N,X,GDUM)
      GO TO 99
C
 99 NGV=NGV+1
C
    RETURN
    END
C
C
C
C    ****************************************************************
          FUNCTION  FSGRAD   (N,X,GDUM)
C    ****************************************************************
C
    IMPLICIT REAL*8 (A-H,O-Z)
C
    DIMENSION X(N),GDUM(N),P(100,40),Q(40,40)
C
    COMMON /NP/NPROB
C
    GRNORM=0.0
    GOTO (1,2,3,4,5,6,7,8,9,10,11,12,13,14)NPROB
C
  1 CALL SGRAD1(N,X,GDUM,P,Q,GRNORM)
      GO TO 99
C
  2 CALL SGRAD2(N,X,GDUM,P,Q,GRNORM)
      GO TO 99
C
  3 CALL SGRAD3(N,X,GDUM,P,Q,GRNORM)
      GO TO 99
C
  4 CALL SGRAD4(N,X,GDUM,P,Q,GRNORM)
      GO TO 99
C
  5 CALL SGRAD5(N,X,GDUM,P,Q,GRNORM)
      GO TO 99
C
  6 CALL SGRAD6(N,X,GDUM,P,Q,GRNORM)
      GO TO 99
C
  7 CALL SGRAD7(N,X,GDUM,P,Q,GRNORM)
      GO TO 99
C
  8 CALL SGRAD8(N,X,GDUM,P,Q,GRNORM)
      GO TO 99
C
  9     GO TO 99
C
 10 CALL SGRAD10(N,X,GDUM,P,Q,GRNORM)
      GO TO 99
C
 11 CALL SGRAD11(N,X,GDUM,P,Q,GRNORM)
```

```
      GO TO 99
C
  12 CALL SGRAD12(N,X,GDUM,P,Q,GRNORM)
      GO TO 99
C
  13 CALL SGRAD13(N,X,GDUM,P,Q,GRNORM)
      GO TO 99
C
  14 CALL SGRAD14(N,X,GDUM,P,Q,GRNORM)
      GO TO 99
C
  99 FSGRAD=GRNORM
C=    WRITE (*,*),FSGRAD
C
     RETURN
     END
C
C    ***************************************************************
          SUBROUTINE  PROB1   (N,X,Y,G,P,Q)
C    ***************************************************************
C
     IMPLICIT REAL*8 (A-H,O-Z)
C
     DIMENSION X(N),G(N),P(100,40),Q(40,40)
C
     COMMON /IFLAG1/IOVER
C
     ENTRY  INIT1(N,X)
       X(1)=-1.2D0
       X(2)=1.0D0
     RETURN
C
     ENTRY  FCN1(N,X,Y)
C
       IOVER=0
       IF(NORMF(N,X).GT.1.0D9) IOVER=1
       Y=(1.0D1*(X(2)-X(1)**2))**2+(1.0D0-X(1))**2
     RETURN
C
     ENTRY GRAD1(N,X,G)
C
       IOVER=0
       IF(NORMF(N,X).GT.1.0D9) IOVER=1
C
       G(1)=-2*1.0D2*(X(2)-X(1)**2)*2*X(1)
   &           -2*(1.0D0-X(1))
       G(2)=2*1.0D2*(X(2)-X(1)**2)
     RETURN
C
     ENTRY SGRAD1(N,X,G,P,Q,Y)
C
       P(1,1)=-2.0D1*X(1)
       P(1,2)=1.0D1
       P(2,1)=-1.0D0
       P(2,2)=0.0D0
       M=2
       CALL SGN(N,M,G,P,Q,Y)
     RETURN
C
     END
C
C
C    ***************************************************************
          SUBROUTINE  SGN   (N,M,G,P,Q,SGNORM)
C    ***************************************************************
C
     IMPLICIT REAL*8 (A-H,O-Z)
```

```
C
      DIMENSION G(N),P(100,40),Q(40,40)
C
      DO 20 I=1,N
       DO 10 J=1,N
        Q(I,J)=0.0D0
        DO 5 K=1,M
         Q(I,J) = Q(I,J) + P(K,I)*P(K,J)
    5   CONTINUE
   10  CONTINUE
   20 CONTINUE
C
      SUM=0.0D0
      DO 30 J=1,N
          SUM=SUM+(G(J)/Q(J,J))**2
   30 CONTINUE
      SGNORM=DSQRT(SUM)
C
      RETURN
      END
C
C    ***************************************************************
          SUBROUTINE  PROB2  (N,X,Y,G)
C    ***************************************************************
C
      IMPLICIT REAL*8 (A-H,O-Z)
C
      DIMENSION X(N),G(N),YI(33),P(100,40),Q(40,40)
      SAVE YI
C
      DATA YI/8.44D-1,9.08D-1,9.32D-1,9.36D-1,9.25D-1,9.08D-1,8.81D-1,
     &       8.5D-1, 8.18D-1,7.84D-1,7.51D-1,7.18D-1,6.85D-1,6.58D-1,
     &       6.28D-1,6.03D-1,5.8D-1, 5.58D-1,5.38D-1,5.22D-1,5.06D-1,
     &       4.9D-1, 4.78D-1,4.67D-1,4.57D-1,4.48D-1,4.38D-1,4.31D-1,
     &       4.24D-1,4.2D-1, 4.14D-1,4.11D-1,4.06D-1/
C
      COMMON /IFLAG1/IOVER
C
      ENTRY INIT2(N,X)
       X(1)=0.5D0
       X(2)=1.5D0
       X(3)=-1.0D0
       X(4)=1.0D-2
       X(5)=2.0D-2
      RETURN
C
C
      ENTRY FCN2(N,X,Y)
         SUM=0.0D0
         DO 20 I=1,33
          TI=1.0D1*DFLOAT(I-1)
          F= X(1)+ X(2)*CHDEXP(-X(4)*TI)+X(3)*CHDEXP(-X(5)*TI)
          SUM=SUM+(YI(I)-F)**2
   20     CONTINUE
         Y=SUM
C
      RETURN
C
C
      ENTRY GRAD2(N,X,G)
C
         DO 40 I=1,N
           SUM=0.0D0
           DO 30 J=1,33
             TJ=1.0D1*DFLOAT(J-1)
             F= X(1)+ X(2)*CHDEXP(-X(4)*TJ)+X(3)*CHDEXP(-X(5)*TJ)
            IF(I.EQ.1) THEN
            F_GRAD=1.0D0
          ELSE IF(I.EQ.2) THEN
            F_GRAD=CHDEXP(-X(4)*TJ)
          ELSE IF(I.EQ.3) THEN
            F_GRAD=CHDEXP(-X(5)*TJ)
          ELSE IF(I.EQ.4) THEN
            F_GRAD=-TJ*X(2)*CHDEXP(-X(4)*TJ)
          ELSE IF(I.EQ.5) THEN
            F_GRAD=-TJ*X(3)*CHDEXP(-X(5)*TJ)
          ENDIF
          SUM=SUM-2*(YI(J)-F)*
     &              F_GRAD
   30     CONTINUE
         G(I)=SUM
   40   CONTINUE
C
C     F= X(1) + X(2)*CHDEXP(-X(4)*TJ) + X(3)*CHDEXP(-X(5)*TJ)
C
      RETURN
C
      ENTRY SGRAD2(N,X,G,P,Q,Y)
C
        M=33
        DO 60 J=1,M
         TJ=1.0D1*DFLOAT(J-1)
         DO 50 I=1,N
C            F= X(1)+ X(2)*CHDEXP(-X(4)*TJ)+X(3)*CHDEXP(-X(5)*TJ)
          IF(I.EQ.1) THEN
            F_GRAD=1.0D0
          ELSE IF(I.EQ.2) THEN
            F_GRAD=CHDEXP(-X(4)*TJ)
          ELSE IF(I.EQ.3) THEN
            F_GRAD=CHDEXP(-X(5)*TJ)
          ELSE IF(I.EQ.4) THEN
            F_GRAD=-TJ*X(2)*CHDEXP(-X(4)*TJ)
          ELSE IF(I.EQ.5) THEN
            F_GRAD=-TJ*X(3)*CHDEXP(-X(5)*TJ)
          ENDIF
           P(J,I)=F_GRAD
   50     CONTINUE
   60   CONTINUE
        CALL SGN(N,M,G,P,Q,Y)
       RETURN
C
      END
C
C
C    ****************************************************************
         SUBROUTINE  PROB3 (N,X,Y,G)
C    ****************************************************************
C
      IMPLICIT REAL*8 (A-H,O-Z)
C
      DIMENSION X(N),G(N),YI(65),P(100,40),Q(40,40)
C
      DATA YI/1.366D0,1.191D0,1.112D0,1.013D0,9.91D-1,8.85D-1,8.31D-1,
     &      8.47D-1,7.86D-1,7.25D-1,7.46D-1,6.79D-1,6.08D-1,6.55D-1,
     &      6.16D-1,6.06D-1,6.02D-1,6.26D-1,6.51D-1,7.24D-1,6.49D-1,
     &      6.49D-1,6.94D-1,6.44D-1,6.24D-1,6.61D-1,6.12D-1,5.58D-1,
     &      5.33D-1,4.95D-1,5.00D-1,4.23D-1,3.95D-1,3.75D-1,3.72D-1,
     &      3.91D-1,3.96D-1,4.05D-1,4.28D-1,4.29D-1,5.23D-1,5.62D-1,
     &      6.07D-1,6.53D-1,6.72D-1,7.08D-1,6.33D-1,6.68D-1,6.45D-1,
     &      6.32D-1,5.91D-1,5.59D-1,5.97D-1,6.25D-1,7.39D-1,7.10D-1,
     &      7.29D-1,7.20D-1,6.36D-1,5.81D-1,4.28D-1,2.92D-1,1.62D-1,
     &      9.8D-2, 5.4D-2/
      SAVE YI
C
```

```
      COMMON /IFLAG1/IOVER
C
      ENTRY INIT3(N,X)
       X(1)=1.3D0
       X(2)=6.5D-1
       X(3)=6.5D-1
       X(4)=7.0D-1
       X(5)=6.0D-1
       X(6)=3.0D0
       X(7)=5.0D0
       X(8)=7.0D0
       X(9)=2.0D0
       X(10)=4.5D0
       X(11)=5.5D0
      RETURN
C
C
      ENTRY FCN3(N,X,Y)
C
        NOFD=65
        SUM=0.0D0
        DO 20 I=1,NOFD
         T=(I-1)/1.0D1
         F= X(1)*CHDEXP(-X(5)*T) + X(2)*CHDEXP(-(T-X(9))**2 *X(6)) +
     &      X(3)*CHDEXP(-(T-X(10))**2 *X(7)) +
     &      X(4)*CHDEXP(-(T-X(11))**2 *X(8))

         SUM=SUM+(YI(I)-F)**2
  20     CONTINUE
        Y=SUM
      RETURN
C
C
      ENTRY GRAD3(N,X,G)
        NOFD=65
        DO 40 I=1,N
          SUM=0.0D0
          DO 30 J=1,NOFD
            T=(J-1)/1.0D1
            F= X(1)*CHDEXP(-X(5)*T) +
     &        X(2)*CHDEXP(-(T-X(9))**2 *X(6)) +
     &        X(3)*CHDEXP(-(T-X(10))**2 *X(7)) +
     &        X(4)*CHDEXP(-(T-X(11))**2 *X(8))
C
            IF(I.EQ.1) THEN
              F_GRAD=CHDEXP(-X(5)*T)
            ELSE IF(I.EQ.2) THEN
              F_GRAD=CHDEXP(-(T-X(9))**2 *X(6))
            ELSE IF(I.EQ.3) THEN
              F_GRAD=CHDEXP(-(T-X(10))**2 *X(7))
            ELSE IF(I.EQ.4) THEN
              F_GRAD=CHDEXP(-(T-X(11))**2 *X(8))
            ELSE IF(I.EQ.5) THEN
              F_GRAD=-T*X(1)*CHDEXP(-X(5)*T)
            ELSE IF(I.EQ.6) THEN
              F_GRAD=-(T-X(9))**2 *  X(2)*
     &             CHDEXP(-(T-X(9))**2 *X(6))
            ELSE IF(I.EQ.7) THEN
              F_GRAD=-(T-X(10))**2 * X(3)*
     &             CHDEXP(-(T-X(10))**2 *X(7))
            ELSE IF(I.EQ.8) THEN
              F_GRAD=-(T-X(11))**2 * X(4)*
     &             CHDEXP(-(T-X(11))**2 *X(8))
            ELSE IF(I.EQ.9) THEN
              F_GRAD=2*X(6)*(T-X(9)) * X(2)*
     &             CHDEXP(-(T-X(9))**2 *X(6))
            ELSE IF(I.EQ.10) THEN
          F_GRAD=2*X(7)*(T-X(10)) * X(3)*
   &             CHDEXP(-(T-X(10))**2 *X(7))
         ELSE IF(I.EQ.11) THEN
          F_GRAD=2*X(8)*(T-X(11)) * X(4)*
   &             CHDEXP(-(T-X(11))**2 *X(8))
         ENDIF
C
         SUM=SUM-2*(YI(J)-F)*
   &            F_GRAD
  30    CONTINUE
       G(I)=SUM
  40   CONTINUE
C
   RETURN
C
   ENTRY SGRAD3(N,X,G,P,Q,Y)
      M=65
      DO 60 J=1,M
        T=(J-1)/1.0D1
        DO 50 I=1,N
         IF(I.EQ.1) THEN
           F_GRAD=CHDEXP(-X(5)*T)
         ELSE IF(I.EQ.2) THEN
           F_GRAD=CHDEXP(-(T-X(9))**2 *X(6))
         ELSE IF(I.EQ.3) THEN
           F_GRAD=CHDEXP(-(T-X(10))**2 *X(7))
         ELSE IF(I.EQ.4) THEN
           F_GRAD=CHDEXP(-(T-X(11))**2 *X(8))
         ELSE IF(I.EQ.5) THEN
           F_GRAD=-T*X(1)*CHDEXP(-X(5)*T)
         ELSE IF(I.EQ.6) THEN
           F_GRAD=-(T-X(9))**2 *  X(2)*
   &             CHDEXP(-(T-X(9))**2 *X(6))
         ELSE IF(I.EQ.7) THEN
           F_GRAD=-(T-X(10))**2 * X(3)*
   &             CHDEXP(-(T-X(10))**2 *X(7))
         ELSE IF(I.EQ.8) THEN
           F_GRAD=-(T-X(11))**2 * X(4)*
   &             CHDEXP(-(T-X(11))**2 *X(8))
         ELSE IF(I.EQ.9) THEN
           F_GRAD=2*X(6)*(T-X(9)) * X(2)*
   &             CHDEXP(-(T-X(9))**2 *X(6))
         ELSE IF(I.EQ.10) THEN
          F_GRAD=2*X(7)*(T-X(10)) * X(3)*
   &             CHDEXP(-(T-X(10))**2 *X(7))
         ELSE IF(I.EQ.11) THEN
          F_GRAD=2*X(8)*(T-X(11)) * X(4)*
   &             CHDEXP(-(T-X(11))**2 *X(8))
         ENDIF
C
         P(J,I)=F_GRAD
  50    CONTINUE
  60   CONTINUE
      CALL SGN(N,M,G,P,Q,Y)
C
   RETURN
C
   END
C
C
C    *****************************************************************
        SUBROUTINE  PROB4   (N,X,Y,G)
C    *****************************************************************
C
C   PROBLEM 4: HELICAL VALLEY FUCTION (FLETCHER AND POWELL,1963)
C   ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
C
```

```
      IMPLICIT REAL*8 (A-H,O-Z)
C
      DIMENSION X(N),G(N),P(100,40),Q(40,40)
C
      COMMON /IFLAG1/IOVER
C
      ENTRY  INIT4(N,X)
        X(1) = -1.0D0
        X(2) = 0.0D0
        X(3) = 0.0D0
      RETURN
C
      ENTRY  FCN4(N,X,Y)
C
        IOVER=0
        IF(NORMF(N,X).GT.1.0D9) IOVER=1
C
        TRI = 8.0D0*DATAN(1.0D0)
        TMP1 = DSIGN(2.5D-1,X(2))
        IF(X(1) .GT. 0.0D0) TMP1 = DATAN(X(2)/X(1))/TRI
        IF(X(1) .LT. 0.0D0) TMP1 = DATAN(X(2)/X(1))/TRI + 0.5D0
        TMP2 = DSQRT(X(1)**2+X(2)**2)

        Y1 = 1.0D1*(X(3)-1.0D1*TMP1)
        Y2 = 1.0D1*(TMP2-1.0D0)
        Y3 = X(3)
        Y=Y1**2+Y2**2+Y3**2
      RETURN
C
      ENTRY GRAD4(N,X,G)
C
        IOVER=0
        IF(NORMF(N,X).GT.1.0D9 .OR. X(1).EQ. 0.0D0) THEN
          IOVER=1
          RETURN
        ENDIF
C
        TRI = 8.0D0*DATAN(1.0D0)
        TMP1 = DSIGN(2.5D-1,X(2))

        IF(X(1) .GT. 0.0D0) TMP1 = DATAN(X(2)/X(1))/TRI
        IF(X(1) .LT. 0.0D0) TMP1 = DATAN(X(2)/X(1))/TRI + 0.5D0
        IF(X(1) .NE. 0.0D0) DTMP1= 1.0D0/( TRI*(1.0D0+(X(2)/X(1)))**2) )
        TMP2 = DSQRT(X(1)**2+X(2)**2)

        Y1 = 1.0D1*(X(3)-1.0D1*TMP1)
        Y2 = 1.0D1*(TMP2-1.0D0)
        Y3 = X(3)
C
        G(1) = 2.0D0*Y1*( -1.0D2*DTMP1*X(2)/(-X(1)**2) ) +
     &         2.0D0*Y2*1.0D1*X(1)/TMP2
        G(2) = 2.0D0*Y1*( -1.0D2*DTMP1/X(1) ) +
     &         2.0D0*Y2*1.0D1*X(2)/TMP2
        G(3) = 2.0D0*Y1*1.0D1 + 2.0D0*Y3
C
      RETURN
C
      ENTRY SGRAD4(N,X,G,P,Q,Y)
C
        M=3
C
        TRI = 8.0D0*DATAN(1.0D0)
        TMP1 = DSIGN(2.5D-1,X(2))
C
        IF(X(1) .GT. 0.0D0) TMP1 = DATAN(X(2)/X(1))/TRI
        IF(X(1) .LT. 0.0D0) TMP1 = DATAN(X(2)/X(1))/TRI + 0.5D0
        IF(X(1) .NE. 0.0D0) DTMP1= 1.0D0/( TRI*(1.0D0+(X(2)/X(1)))**2) )
```

```
       TMP2 = DSQRT(X(1)**2+X(2)**2)
C
       P(1,1)= -1.0D2*DTMP1*X(2)/(-X(1)**2)
       P(1,2)= -1.0D2*DTMP1/X(1)
       P(1,3)= 1.0D1
C
       P(2,1)= 1.0D1*X(1)/TMP2
       P(2,2)= 1.0D1*X(2)/TMP2
       P(2,3)= 0.0D0
C
       P(3,1)= 0.0D0
       P(3,2)= 0.0D0
       P(3,3)= 1.0D0
       CALL SGN(N,M,G,P,Q,Y)
C
     RETURN

     END
C
C
C    ****************************************************************
          SUBROUTINE  PROB5  (N,X,Y,G)
C    ****************************************************************
C
C    PROBLEM 5: POWELL SINGULAR FUCTION (POWELL,1962)
C    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
C
    IMPLICIT REAL*8 (A-H,O-Z)
C
    DIMENSION X(N),G(N),P(100,40),Q(40,40)
C
    COMMON /IFLAG1/IOVER
C
    ENTRY  INIT5(N,X)
       X(1) = 3.0D0
       X(2) = -1.0D0
       X(3) = 0.0D0
       X(4) = 1.0D0
    RETURN
C
    ENTRY  FCN5(N,X,Y)
C
      IOVER=0
      IF(NORMF(N,X).GT.1.0D9) IOVER=1
C
      Y1 = X(1)+1.0D1*X(2)
      Y2 = DSQRT(5.0D0)*(X(3)-X(4))
      Y3 = (X(2)-2.0D0*X(3))**2
      Y4 = DSQRT(1.0D1)*(X(1)-X(4))**2
      Y=Y1**2+Y2**2+Y3**2+Y4**2
    RETURN
C
    ENTRY GRAD5(N,X,G)
C
      IOVER=0
      IF(NORMF(N,X).GT.1.0D9) IOVER=1
C
      Y1 = X(1)+1.0D1*X(2)
      Y2 = DSQRT(5.0D0)*(X(3)-X(4))
      Y3 = (X(2)-2.0D0*X(3))**2
      Y4 = DSQRT(1.0D1)*(X(1)-X(4))**2
C
      G(1) = 2.0D0*Y1 + 2.0D0*Y4*DSQRT(1.0D1)*2.0D0*(X(1)-X(4))
      G(2) = 2.0D0*Y1*1.0D1 + 2.0D0*Y3*2.0D0*(X(2)-2.0D0*X(3))
      G(3) = 2.0D0*Y2*DSQRT(5.0D0) + 2.0D0*Y3*(-4.0D0)*
   &         (X(2)-2.0D0*X(3))
      G(4) = 2.0D0*Y2*(-DSQRT(5.0D0)) + 2.0D0*Y4*DSQRT(1.0D1)*
     &         (-2.0D0)*(X(1)-X(4))
      RETURN
C
     ENTRY SGRAD5(N,X,G,P,Q,Y)
C
      M=4
C
      P(1,1)= 1.0D0
      P(1,2)= 1.0D1
      P(1,3)= 0.0D0
      P(1,4)= 0.0D0
C
      P(2,1)= 0.0D0
      P(2,2)= 0.0D0
      P(2,3)= DSQRT(5.0D0)
      P(2,4)= -DSQRT(5.0D0)
C
      P(3,1)= 0.0D0
      P(3,2)= 2.0D0*(X(2)-2.0D0*X(3))
      P(3,3)= (-4.0D0)*
     &         (X(2)-2.0D0*X(3))
      P(3,4)= 0.0D0
C
      P(4,1)= DSQRT(1.0D1)*2.0D0*(X(1)-X(4))
      P(4,2)= 0.0D0
      P(4,3)= 0.0D0
      P(4,4)= DSQRT(1.0D1)*
     &         (-2.0D0)*(X(1)-X(4))
C
      CALL SGN(N,M,G,P,Q,Y)
      RETURN
      END
C
C
C
C    **************************************************************
         SUBROUTINE  PROB6   (N,X,Y,G)
C    **************************************************************
C
C    PROBLEM 6: FREUDENSTEIN AND ROTH FUNCTION (1963)
C    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
C
     IMPLICIT REAL*8 (A-H,O-Z)
C
     DIMENSION X(N),G(N),P(100,40),Q(40,40)
C
     COMMON /IFLAG1/IOVER
C
     ENTRY  INIT6(N,X)
        X(1) = 0.5D0
        X(2) = -2.0D0
     RETURN
C
     ENTRY  FCN6(N,X,Y)
C
        IOVER=0
        IF(NORMF(N,X).GT.1.0D9) IOVER=1
C
        Y1 = -1.3D1+X(1)+((5.0D0-X(2))*X(2)-2.0D0)*X(2)
        Y2 = -2.9D1+X(1)+((1.0D0+X(2))*X(2)-1.4D1)*X(2)
        Y=Y1**2+Y2**2
     RETURN
C
     ENTRY GRAD6(N,X,G)
C
        IOVER=0
        IF(NORMF(N,X).GT.1.0D9) IOVER=1
```

```
C
        Y1 = -1.3D1+X(1)+((5.0D0-X(2))*X(2)-2.0D0)*X(2)
        Y2 = -2.9D1+X(1)+((1.0D0+X(2))*X(2)-1.4D1)*X(2)
C
        G(1) = 2.0D0*Y1 + 2.0D0*Y2
        G(2) = 2.0D0*Y1*(1.0D1*X(2)-3.0D0*X(2)**2-2.0D0) +
     &        2.0D0*Y2*(2.0D0*X(2)+3.0D0*X(2)**2-1.4D1)
      RETURN
C
      ENTRY SGRAD6(N,X,G,P,Q,Y)
C
      M=2
C
      P(1,1)= 1.0D0
      P(1,2)= 1.0D1*X(2)-3.0D0*X(2)**2-2.0D0
C
      P(2,1)= 1.0D0
      P(2,2)= 2.0D0*X(2)+3.0D0*X(2)**2-1.4D1
C
      CALL SGN(N,M,G,P,Q,Y)
      END
C
C
C     ****************************************************************
          SUBROUTINE  PROB7  (N,X,Y,G)
C     ****************************************************************
C
C     PROBLEM 7: BRAD (1963)
C     ^^^^^^^^^^^^^^^^^^^^^^
C
      IMPLICIT REAL*8 (A-H,O-Z)
C
      DIMENSION X(N),G(N),YDAT(15),P(100,40),Q(40,40)
C
      COMMON /IFLAG1/IOVER
C
      DATA YDAT/1.4D-1,1.8D-1,2.2D-1,2.5D-1,2.9D-1,3.2D-1,3.5D-1,3.9D-1,
     &     3.7D-1,5.8D-1,7.3D-1,9.6D-1,1.34D0,2.1D0,4.39D0/
      SAVE YDAT
      DFLOAT(IVAR)=IVAR
C
      ENTRY  INIT7(N,X)
        CALL VSET(N,X,1.0D0)
      RETURN
C
      ENTRY  FCN7(N,X,Y)
C
        IOVER=0
        IF(NORMF(N,X).GT.1.0D9.OR.(X(2).EQ.0.0D0
     &                  .AND.X(3).EQ.0.0D0)) THEN
              IOVER=1
              RETURN
        ENDIF
C
        Y=0.0D0
        DO 10 I = 1,15
          TMP1 = DFLOAT(I)
          TMP2 = DFLOAT(16-I)
          TMP3 = TMP1
          IF(I.GT.8) TMP3=TMP2
          YI = YDAT(I) - (X(1) + TMP1/(X(2)*TMP2 + X(3)*TMP3))
          Y=Y+YI**2
 10     CONTINUE
      RETURN
C
      ENTRY GRAD7(N,X,G)
C
```

```
        IOVER=0
        IF(NORMF(N,X).GT.1.0D9.OR.(X(2).EQ.0.0D0
     &                .AND.X(3).EQ.0.0D0)) THEN
            IOVER=1
            RETURN
        ENDIF
C
        CALL VSET(N,G,0.0D0)
        DO 20 I = 1,15
          TMP1 = DFLOAT(I)
          TMP2 = DFLOAT(16-I)
          TMP3 = TMP1
          IF(I.GT.8) TMP3=TMP2
          YI = YDAT(I) - (X(1) + TMP1/(X(2)*TMP2 + X(3)*TMP3))
          G(1) = G(1) - 2.0D0*YI
          G(2) = G(2) + 2.0D0*YI*(TMP1*TMP2/
     &          (X(2)*TMP2+X(3)*TMP3)**2 )
          G(3) = G(3) + 2.0D0*YI*(TMP1*TMP3/
     &          (X(2)*TMP2+X(3)*TMP3)**2 )
   20   CONTINUE
      RETURN
C
      ENTRY SGRAD7(N,X,G,P,Q,Y)
C
       M=15
C
       DO 30 I = 1,M
          TMP1 = DFLOAT(I)
          TMP2 = DFLOAT(16-I)
          TMP3 = TMP1
          IF(I.GT.8) TMP3=TMP2
C
          P(I,1) = 1.0D0
          P(I,2) = (TMP1*TMP2/
     &          (X(2)*TMP2+X(3)*TMP3)**2 )
          P(I,3) = (TMP1*TMP3/
     &          (X(2)*TMP2+X(3)*TMP3)**2 )
   30 CONTINUE
C
      CALL SGN(N,M,G,P,Q,Y)
      RETURN
C
      END
C
C     ***************************************************************
          SUBROUTINE  PROB8  (N,X,Y,G)
C     ***************************************************************
C
C     PROBLEM 8: KOWALIK AND OSBORNE FUNCTION (1968)
C     ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
C
      IMPLICIT REAL*8 (A-H,O-Z)
C
      DIMENSION X(N),G(N),V(11),YDAT(11),P(100,40),Q(40,40)
      SAVE V,YDAT
C
      COMMON /IFLAG1/IOVER
C
      DATA V/4.0D0,2.0D0,1.0D0,5.0D-1,2.5D-1,1.67D-1,1.25D-1,
     &     1.0D-1,8.33D-2,7.14D-2,6.25D-2/
     &   YDAT/1.957D-1,1.947D-1,1.735D-1,1.6D-1,8.44D-2,6.27D-2,
     &     4.56D-2,3.42D-2,3.23D-2,2.35D-2,2.46D-2/
C
      ENTRY  INIT8(N,X)
       X(1) = 2.5D-1
       X(2) = 3.9D-1
       X(3) = 4.15D-1
      X(4) = 3.9D-1
    RETURN
C
    ENTRY  FCN8(N,X,Y)
C
      IOVER=0
      DO 5 I = 1,11
       IF(NORMF(N,X).GT.1.0D9.OR.
    &     (V(I)*(V(I)+X(3))+X(4).EQ.0.0D0)) THEN
            IOVER=1
            RETURN
       ENDIF
  5    CONTINUE
C
      Y=0.0D0
      DO 10 I = 1,11
        TMP1 = V(I)*(V(I)+X(2))
        TMP2 = V(I)*(V(I)+X(3))+X(4)
        YI = YDAT(I) - X(1)*TMP1/TMP2
        Y=Y+YI**2
 10    CONTINUE
    RETURN
C
    ENTRY GRAD8(N,X,G)
C
      IOVER=0
      DO 15 I = 1,11
       IF(NORMF(N,X).GT.1.0D9.OR.
    &     (V(I)*(V(I)+X(3))+X(4).EQ.0.0D0)) THEN
            IOVER=1
            RETURN
       ENDIF
 15    CONTINUE
C
      CALL VSET(N,G,0.0D0)
      DO 20 I = 1,11
        TMP1 = V(I)*(V(I)+X(2))
        TMP2 = V(I)*(V(I)+X(3))+X(4)
        YI = YDAT(I) - X(1)*TMP1/TMP2
        G(1) = G(1) - 2.0D0*YI*TMP1/TMP2
        G(2) = G(2) - 2.0D0*YI*X(1)*V(I)/TMP2
        G(3) = G(3) + 2.0D0*YI*X(1)*TMP1*V(I)/(TMP2**2)
        G(4) = G(4) + 2.0D0*YI*X(1)*TMP1/(TMP2**2)
 20    CONTINUE
    RETURN
C
    ENTRY SGRAD8(N,X,G,P,Q,Y)
C
      M=11
C
      DO 30 I = 1,M
        TMP1 = V(I)*(V(I)+X(2))
        TMP2 = V(I)*(V(I)+X(3))+X(4)
C
        P(I,1) = TMP1/TMP2
        P(I,2) = X(1)*V(I)/TMP2
        P(I,3) = X(1)*TMP1*V(I)/(TMP2**2)
        P(I,4) = X(1)*TMP1/(TMP2**2)
 30    CONTINUE
    CALL SGN(N,M,G,P,Q,Y)
C
    RETURN
    END
C
C
C     ******************************************************************
        SUBROUTINE  PROB9  (N,X,Y,G)
```

```
C     *****************************************************************
C
C     PROBLEM 9: MEYER (1970)
C     ^^^^^^^^^^^^^^^^^^^^^^^^
C
      IMPLICIT REAL*8 (A-H,O-Z)
C
      DIMENSION X(N),G(N),YDAT(16)
C
      COMMON /IFLAG1/IOVER
C
      DATA YDAT/3.478D4,2.861D4,2.365D4,1.963D4,1.637D4,1.372D4,1.154D4,
     &     9.744D3,8.261D3,7.03D3,6.005D3,5.147D3,4.427D3,3.82D3,
     &     3.307D3,2.872D3/
      SAVE YDAT
C     DFLOAT(IVAR)=IVAR
C
      ENTRY  INIT9(N,X)
        X(1) = 2.0D-2
        X(2) = 4.0D3
        X(3) = 2.5D2
      RETURN
C
      ENTRY  FCN9(N,X,Y)
C
        IOVER=0
        Y=0.0D0
        DO 10 I = 1,16
          TMP1 = 5.0D0*DFLOAT(I) + 4.5D1 + X(3)
          IF(TMP1.EQ.0.0D0)  THEN
            IOVER=1
            RETURN
          ENDIF
          TMP2 = CHDEXP(X(2)/TMP1)
          YI = X(1)*TMP2 - YDAT(I)
          Y=Y+YI**2
 10     CONTINUE
      RETURN
C
      ENTRY GRAD9(N,X,G)
C
        CALL VSET(N,G,0.0D0)
        IOVER=0
        DO 20 I = 1,16
          TMP1 = 5.0D0*DFLOAT(I) + 4.5D1 + X(3)
          IF(TMP1.EQ.0.0D0)  THEN
            IOVER=1
            RETURN
          ENDIF
          TMP2 = CHDEXP(X(2)/TMP1)
          YI = X(1)*TMP2 - YDAT(I)
          G(1) = G(1) + 2.0D0*YI*TMP2
          G(2) = G(2) + 2.0D0*YI*X(1)*TMP2/TMP1
          G(3) = G(3) + 2.0D0*YI*X(1)*TMP2*(-X(2)/(TMP1**2))
 20     CONTINUE
      RETURN
      END
C
C
C
C     *****************************************************************
        SUBROUTINE  PROB10  (N,X,Y,G)
C     *****************************************************************
C
C     PROBLEM 10: WATSON FUNCTION (KOWALIK AND OSBORNE,1968)
C     ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
C
```

```
      IMPLICIT REAL*8 (A-H,O-Z)
C
      DIMENSION X(N),G(N),P(100,40),Q(40,40)
C
      COMMON /IFLAG1/IOVER
      COMMON /NTLSQ/M
C       DFLOAT(IVAR)=IVAR
C
      ENTRY  INIT10(N,X)
        CALL VSET(N,X,0.0D0)
      RETURN
C
      ENTRY  FCN10(N,X,Y)
C
        IOVER=0
        IF(NORMF(N,X).GT.1.0D9) IOVER=1
C
        Y=0.0D0
        DO 30 I = 1,31
          DIV = DFLOAT(I)/2.9D1
          S1 = 0.0D0
          DDX = 1.0D0
          DO 10 J = 2,N
            S1 = S1 + DFLOAT(J-1)*DDX*X(J)
            DDX = DIV*DDX
  10      CONTINUE
          S2 = 0.0D0
          DDX = 1.0D0
          DO 20 J = 1,N
            S2 = S2 + DDX*X(J)
            DDX = DIV*DDX
  20      CONTINUE
          YI = S1 - S2**2 - 1.0D0
          IF(I.EQ.30) YI = X(1)
          IF(I.EQ.31) YI = X(2) - X(1)**2 - 1.0D0
          Y = Y+YI**2
  30    CONTINUE

      RETURN
C
      ENTRY GRAD10(N,X,G)
C
        IOVER=0
        IF(NORMF(N,X).GT.1.0D9) IOVER=1
C
        CALL VSET(N,G,0.0D0)
        DO 70 I = 1,31
          DIV = DFLOAT(I)/2.9D1
          S1 = 0.0D0
          DDX = 1.0D0
          DO 40 J = 2,N
            S1 = S1 + DFLOAT(J-1)*DDX*X(J)
            DDX = DIV*DDX
  40      CONTINUE
          S2 = 0.0D0
          DDX = 1.0D0
          DO 50 J = 1,N
            S2 = S2 + DDX*X(J)
            DDX = DIV*DDX
  50      CONTINUE
          YI = S1 - S2**2 - 1.0D0
          IF(I.EQ.30) YI = X(1)
          IF(I.EQ.31) YI = X(2) - X(1)**2 - 1.0D0
C
          DO 60 K = 1,N
            IF(I.LE.29) THEN
              G(K) = G(K) + 2.0D0*YI*
```

```
     &                  ( DFLOAT(K-1)*DIV**(K-2)
     &                   - 2.0D0*S2*DIV**(K-1) )
            ELSE IF(I.EQ.30.AND.K.EQ.1) THEN
              G(K) = G(K) + 2.0D0*YI
            ELSE IF(I.EQ.31) THEN
              IF(K.EQ.1) G(K) = G(K)+2.0D0*YI*(-2.0D0*X(1))
              IF(K.EQ.2) G(K) = G(K) + 2.0D0*YI
            ENDIF
   60     CONTINUE
C
   70   CONTINUE
      RETURN
C
      ENTRY SGRAD10(N,X,G,P,Q,Y)
C
        M=31
C
        DO 110 I = 1,M
          DIV = DFLOAT(I)/2.9D1
          S2 = 0.0D0
          DDX = 1.0D0
          DO 90 J = 1,N
            S2 = S2 + DDX*X(J)
            DDX = DIV*DDX
   90     CONTINUE
          DO 100 K = 1,N
            IF(I.LE.29) THEN
              P(I,K) = ( DFLOAT(K-1)*DIV**(K-2)
     &                   - 2.0D0*S2*DIV**(K-1) )
            ELSE IF(I.EQ.30) THEN
              P(I,K) = 0.0D0
              IF(K.EQ.1) P(I,K) = 1.0D0
            ELSE IF(I.EQ.31) THEN
              IF(K.EQ.1) P(I,K) = -2.0D0*X(1)
              IF(K.EQ.2) P(I,K) = 1.0D0
              IF(K.GE.3) P(I,K) = 0.0D0
            ENDIF
  100     CONTINUE
C
  110   CONTINUE
C
      CALL SGN(N,M,G,P,Q,Y)
C
      RETURN
C
      END
C
C
C   *****************************************************************
        SUBROUTINE  PROB11   (N,X,Y,G)
C   *****************************************************************
C
C    PROBLEM 11: BOX 3-DIMMENTION FUNCTION (1966) M > 3
C    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
C
      IMPLICIT REAL*8 (A-H,O-Z)
C
      DIMENSION X(N),G(N),P(100,40),Q(40,40)
C
      COMMON /IFLAG1/IOVER
      COMMON /NTLSQ/MT
C
      ENTRY  INIT11(N,X)
        X(1) = 0.0D0
        X(2) = 1.0D1
        X(3) = 2.0D1
C
```

```
      MT=5
      RETURN
C
      ENTRY  FCN11(N,X,Y)
C
         Y=0.0D0
         DO 10 I = 1,MT
           TEMP = DFLOAT(I)
           TMP1 = TEMP/1.0D1
           YI = CHDEXP(-TMP1*X(1))-CHDEXP(-TMP1*X(2))
     &          + (CHDEXP(-TEMP) - CHDEXP(-TMP1))*X(3)
           Y=Y+YI**2
   10    CONTINUE
      RETURN
C
      ENTRY GRAD11(N,X,G)
C
         CALL VSET(N,G,0.0D0)
         DO 20 I = 1,MT
           TEMP = DFLOAT(I)
           TMP1 = TEMP/1.0D1
           YI = CHDEXP(-TMP1*X(1))-CHDEXP(-TMP1*X(2))
     &          + (CHDEXP(-TEMP) - CHDEXP(-TMP1))*X(3)
C
           G(1) = G(1) + 2.0D0*YI*(-TMP1)*CHDEXP(-TMP1*X(1))
           G(2) = G(2) - 2.0D0*YI*(-TMP1)*CHDEXP(-TMP1*X(2))
           G(3) = G(3) + 2.0D0*YI*(CHDEXP(-TEMP) - CHDEXP(-TMP1))
C
   20    CONTINUE
      RETURN
C
      ENTRY SGRAD11(N,X,G,P,Q,Y)
C
       M=MT
C
       DO 30 I = 1,M
           TEMP = DFLOAT(I)
           TMP1 = TEMP/1.0D1
C
           P(I,1) =(-TMP1)*CHDEXP(-TMP1*X(1))
           P(I,2) =(-TMP1)*CHDEXP(-TMP1*X(2))
           P(I,3) = (CHDEXP(-TEMP) - CHDEXP(-TMP1))
C
   30  CONTINUE
C
      CALL SGN(N,M,G,P,Q,Y)
      RETURN
C

      END
C
C
C
C    ****************************************************************
         SUBROUTINE  PROB12   (N,X,Y,G)
C    ****************************************************************
C
C    PROBLEM 12: JENNRICH AND SAMPSON FUNCTION (1968) M > 2
C    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
C
      IMPLICIT REAL*8 (A-H,O-Z)
C
      DIMENSION X(N),G(N),P(100,40),Q(40,40)
C
      COMMON /IFLAG1/IOVER
      COMMON /NTLSQ/MT
C
```

```
      ENTRY  INIT12(N,X)
        X(1) = 0.3D0
        X(2) = 0.4D0
        MT=5
      RETURN
C
      ENTRY  FCN12(N,X,Y)
C
        Y=0.0D0
        DO 10 I = 1,MT
          TEMP = DFLOAT(I)
          YI = 2.0D0 + 2.0D0*TEMP - CHDEXP(TEMP*X(1))
     &          - CHDEXP(TEMP*X(2))
          Y=Y+YI**2
  10    CONTINUE
      RETURN
C
      ENTRY GRAD12(N,X,G)
C
        CALL VSET(N,G,0.0D0)
        DO 20 I = 1,MT
          TEMP = DFLOAT(I)
          YI = 2.0D0 + 2.0D0*TEMP - CHDEXP(TEMP*X(1))
     &          - CHDEXP(TEMP*X(2))
C
          G(1) = G(1) - 2.0D0*YI*TEMP*CHDEXP(TEMP*X(1))
          G(2) = G(2) - 2.0D0*YI*TEMP*CHDEXP(TEMP*X(2))
C
  20    CONTINUE
      RETURN
C
      ENTRY SGRAD12(N,X,G,P,Q,Y)
C
        M=MT
        DO 30 I = 1,MT
          TEMP = DFLOAT(I)
          P(I,1) = TEMP*CHDEXP(TEMP*X(1))
          P(I,2) = TEMP*CHDEXP(TEMP*X(2))
  30    CONTINUE
      CALL SGN(N,M,G,P,Q,Y)
      RETURN
C
      END
C
C     ****************************************************************
          SUBROUTINE  PROB13   (N,X,Y,G)
C     ****************************************************************
C
C     PROBLEM 13: BROWN AND DENNIS FUNCTION (1969) M>=N=4
C     ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
C
      IMPLICIT REAL*8 (A-H,O-Z)
C
      DIMENSION X(N),G(N),P(100,40),Q(40,40)
C
      COMMON /IFLAG1/IOVER
      COMMON /NTLSQ/MT
C
      ENTRY  INIT13(N,X)
        X(1) = 2.5D1
        X(2) = 5.0D0
        X(3) = -5.0D0
        X(4) = -1.0D0
        MT=5
      RETURN
C
      ENTRY  FCN13(N,X,Y)
```

```
C
       Y=0.0D0
       DO 10 I = 1,MT
         TEMP = DFLOAT(I)/5.0D0
         TMP1 = X(1) + TEMP*X(2) - CHDEXP(TEMP)
         TMP2 = X(3) + DSIN(TEMP)*X(4) - DCOS(TEMP)
         YI = TMP1**2 + TMP2**2
         Y=Y+YI**2
  10   CONTINUE
     RETURN
C
     ENTRY GRAD13(N,X,G)
C
       CALL VSET(N,G,0.0D0)
       DO 20 I = 1,MT
         TEMP = DFLOAT(I)/5.0D0
         TMP1 = X(1) + TEMP*X(2) - CHDEXP(TEMP)
         TMP2 = X(3) + DSIN(TEMP)*X(4) - DCOS(TEMP)
         YI = TMP1**2 + TMP2**2
C
         G(1) = G(1) + 2.0D0*YI*2.0D0*TMP1
         G(2) = G(2) + 2.0D0*YI*2.0D0*TMP1*TEMP
         G(3) = G(3) + 2.0D0*YI*2.0D0*TMP2
         G(4) = G(4) + 2.0D0*YI*2.0D0*TMP2*DSIN(TEMP)
C
  20   CONTINUE
      RETURN
C
     ENTRY SGRAD13(N,X,G,P,Q,Y)
C
       M=MT
       DO 30 I = 1,MT
         TEMP = DFLOAT(I)/5.0D0
         TMP1 = X(1) + TEMP*X(2) - CHDEXP(TEMP)
         TMP2 = X(3) + DSIN(TEMP)*X(4) - DCOS(TEMP)
C
         P(I,1) = 2.0D0*TMP1
         P(I,2) = 2.0D0*TMP1*TEMP
         P(I,3) = 2.0D0*TMP2
         P(I,4) = 2.0D0*TMP2*DSIN(TEMP)
  30   CONTINUE
      CALL SGN(N,M,G,P,Q,Y)
      RETURN
C
      END
C
C   ******************************************************************
        SUBROUTINE  PROB14   (N,X,Y,G)
C   ******************************************************************
C
C   PROBLEM 14: BROWN ALMOST-LINEAR FUNCTION (1969) M=N
C   ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
C
     IMPLICIT REAL*8 (A-H,O-Z)
C
     DIMENSION X(N),G(N),P(100,40),Q(40,40)
C
     COMMON /IFLAG1/IOVER
     COMMON /NTLSQ/MT
C
     ENTRY  INIT14(N,X)
       CALL VSET(N,X,0.5D0)
       MT=N
     RETURN
C
     ENTRY  FCN14(N,X,Y)
       IOVER=0
      IF(NORMF(N,X).GT.1.0D9) IOVER=1
C
      Y=0.0D0
      SUM = -DFLOAT(N+1)
      PROD = 1.0D0
      DO 10 J = 1,MT
        SUM = SUM + X(J)
        PROD = PROD*X(J)
 10   CONTINUE
      DO 20 I = 1,N
        YI = X(I) + SUM
        IF(I.EQ.N) YI = PROD - 1.0D0
        Y=Y+YI**2
 20   CONTINUE
    RETURN
C
    ENTRY GRAD14(N,X,G)
C
      CALL VSET(N,G,0.0D0)
      SUM = -DFLOAT(N+1)
      PROD = 1.0D0
      DO 30 J = 1,N
        SUM = SUM + X(J)
        PROD = PROD*X(J)
 30   CONTINUE
      DO 50 I = 1,MT
        YI = X(I) + SUM
        IF(I.EQ.N) YI = PROD - 1.0D0
        DO 40 K = 1,N
          IF(I.LT.MT) THEN
            G(K) = G(K) + 2.0D0*YI
            IF(K.EQ.I) G(K) = G(K) + 2.0D0*YI
          ELSE
            G(K) = G(K) + 2.0D0*YI* PROD/X(K)
          ENDIF
 40     CONTINUE
 50   CONTINUE

    RETURN
C
    ENTRY SGRAD14(N,X,G,P,Q,Y)
C
      M=N
C
      PROD = 1.0D0
      DO 60 J = 1,N
        PROD = PROD*X(J)
 60   CONTINUE
C
      DO 70 I = 1,M
        DO 80 K = 1,N
          IF(I.LT.MT) THEN
            P(I,K) = 1.0D0
            IF(K.EQ.I) P(I,K) = 2.0D0
          ELSE
            P(I,K) =  PROD/X(K)
          ENDIF
 80     CONTINUE
 70   CONTINUE
    CALL SGN(N,M,G,P,Q,Y)
    RETURN
C
    END
C
C
C-----------------------------------------------------------------------
    DOUBLE PRECISION FUNCTION NORMF(N,X)
```

```
C-----------------------------------------------------------------------
      IMPLICIT REAL*8 (A-H,O-Z)
C
      DIMENSION X(N)

      SUM=0.0D0
      DO 10 J=1,N
        SUM=SUM+X(J)**2
 10   CONTINUE
      NORMF=DSQRT(SUM)
C
      RETURN
      END
C
C-----------------------------------------------------------------------
      DOUBLE PRECISION FUNCTION CHDEXP(ARG)
C-----------------------------------------------------------------------
      IMPLICIT REAL*8 (A-H,O-Z)
C
      COMMON /IFLAG1/IOVER
      COMMON /IFLAG2/IUNDER
C
      ARGMAX=70.0D0
C
      IF(ARG.GT.ARGMAX) THEN
       ARG=ARGMAX
       IOVER=1
      ENDIF
      IF(ARG.LT.-ARGMAX) THEN
       ARG=-ARGMAX
       IUNDER=1
      ENDIF
C
      CHDEXP=DEXP(ARG)
C
      RETURN
      END
C
C
C
C-----------------------------------------------------------------------
      SUBROUTINE  VSET(N,A,SCALE)
C-----------------------------------------------------------------------
      IMPLICIT REAL*8 (A-H,O-Z)
C
      DIMENSION A(N)
C
      DO 10 I=1,N
        A(I)=SCALE
 10   CONTINUE
C
      RETURN
      END
C
C***********************************************************************
      SUBROUTINE CGSET
C
      INTEGER METHCG,MAXCYC,MAXNFV,MINLOC,NOREST,NTRY,
     *  NTRACC,NTRACX,LPCG,KFLAGC,NCYCLE,ITERCG,MAXFWD,
     *  NFMAXB,NFMAXL,KFLAGD,KONNEW
C
      DOUBLE PRECISION DX,DXSAVE,RELTLC,ABSTLC,GRNORM,
     *  RELTLD,ABSTLD,DECMAX,FBINCR,FBDECR
C
      COMMON /COMCG/ DX,DXSAVE,RELTLC,ABSTLC,GRNORM,NFV,NGV,
     *  RELTLD,ABSTLD,DECMAX,FBINCR,FBDECR,LINEOP,METHRT,
     *  METHCG,MAXCYC,MAXNFV,MINLOC,NOREST,NTRY,NTRACC,
     * NTRACX,LPCG,KFLAGC,NCYCLE,ITERCG,MAXFWD,NFMAXB,
     * NFMAXL,KFLAGD,KONNEW
C
      COMMON /NP/NPROB
C
      LPCG=6
      DX=0.01D0
      RELTLC=1.0D-7
      ABSTLC=1.0D-20
      NFV=0
      NGV=0
      MAXCYC=100000
      MAXNFV=1000000
      MINLOC=1
      NOREST=10
      NTRY=2
      NTRACC=0
      NTRACX=0
      RELTLD=1.0D-7
      ABSTLD=1.0D-20
      DECMAX=100.0D0
      FBINCR=3.0D0
      FBDECR=3.0D0
      MAXFWD=4
      NFMAXB=20
      NFMAXL=8
C
      RETURN
C
C  END CGSET
C
      END
C
C*******************************************************************
      SUBROUTINE GRCHEK (NV,X,MASK,GRAD,GRADIF,LP,RLDFMX)
C
C  GRCHEK 1.0        SEPTEMBER 1992
C
C  CHECK THE ANALYTICAL GRADIENT VECTOR USING FINITE
C  DIFFERENCES.
C
C  J. P. CHANDLER, COMPUTER SCIENCE DEPARTMENT,
C    OKLAHOMA STATE UNIVERSITY
C
C
C
      IMPLICIT REAL*8 (A-H,O-Z)
C
      INTEGER MASK,NV,LP,J,IOPT
C
      DOUBLE PRECISION X,GRAD,GRADIF,RLDFMX,RELDFK,RZERO,
     * XSAVE,DX,RLDF,DENOM,DABS,DMAX1,
     * FXA,FXB
C
      COMMON /NP/NPROB
C
      DIMENSION X(1),MASK(1),GRAD(1),GRADIF(1)
C
      RELDFK=1.0D-5
C
      RZERO=0.0D0
C
      NFV=0
      NGV=0
C
      DO 10 J=1,NV
C
```

```
        IF(MASK(J).NE.0) THEN
          GRADIF(J)=RZERO
          GO TO 10
        ENDIF
C
        XSAVE=X(J)
C
C  ADD AN ABSDFK PARAMETER FOR DIFFERENCING IF DESIRED:
C     DX=RELDFK*DABS(X(J))+ABSDFK
C  (THE VALUE OF ABSDFK WILL DEPEND ON THE SCALING OF X(J).)
C
        DX=RELDFK*DABS(X(J))
        IF(DX.EQ.RZERO) DX=RELDFK
        X(J)=XSAVE+DX
        IOPT=-1
        CALL FCN (NV,X,FXA)
        X(J)=XSAVE-DX
        CALL FCN (NV,X,FXB)
        X(J)=XSAVE
        GRADIF(J)=(FXA-FXB)/(DX+DX)
   10   CONTINUE
C
      WRITE(LP,20)(GRADIF(J),J=1,NV)
   20 FORMAT(/' NUMERICAL GRADIENT ='/(1X,1PG15.7,4G15.7))
C
      IOPT=0
      CALL FCN (NV,X,FX)
      CALL FGRAD (NV,X,GRAD)
C
      WRITE(LP,30)(GRAD(J),J=1,NV)
   30 FORMAT(/' ANALYTICAL GRADIENT ='/(1X,1PG15.7,4G15.7))
C
      RLDFMX=RZERO
      DO 40 J=1,NV
        IF(MASK(J).NE.0) GO TO 40
        DENOM=DMAX1(DABS(GRAD(J)),DABS(GRADIF(J)))
        IF(DENOM.EQ.RZERO) DENOM=1.0D0
        RLDF=(GRAD(J)-GRADIF(J))/DENOM
        RLDFMX=DMAX1(RLDFMX,DABS(RLDF))
   40   CONTINUE
C
      WRITE(LP,50)RLDFMX
   50 FORMAT(//' MAXIMUM RELATIVE DIFFERENCE =',1PG15.7)
C
      RETURN
C
C  END GRCHEK
C
      END
C
C
C***********************************************************************
      SUBROUTINE CG (NV,X,XSAVE,GRAD,GRSAVE,GRPREV,XDIR,
     *   XDSAVE,XBASE,XMAX,XMIN,MASK,MASKT,HHG,HHY,XTDIR,YT)
C
C  CG 1.0        AUGUST 1992
C
C  FUTURE IMPROVEMENTS:  SCALE X(*) AND GRAD(*)
C
C  UNCONSTRAINED MINIMIZATION BY CONJUGATE GRADIENT METHODS
C
C  ORIGINALLY  WITTEN BY  J. P. CHANDLER, COMPUTER SCIENCE DEPARTMENT,
C    OKLAHOMA STATE UNIVERSITY
C
C  MODIFIED  BY  MEISHAN CHENG
C
C    DX  -- THE SIZE OF THE CHANGE TO BE MADE IN THE X(*)
```

```
C              VECTOR IN THE FIRST STEP OF THE LINE
C              SEARCH
C
C    NTRACC  -- A PRINT CONTROL SWITCH,
C
C              =-1 TO PRINT ONLY ERROR MESSAGES,
C
C              =0 TO PRINT ON ENTRY AND EXIT,
C
C              =1 TO PRINT ALSO AT "UNUSUAL RESTARTS",
C
C              =2 TO PRINT AT ALL RESTARTS,
C
C              =3 TO PRINT AT EACH ITERATION,
C
C              =4 TO PRINT AFTER EACH CALL TO FUN
C
C    NTRACX  -- A SWITCH TO CONTROL THE PRINTING OF X(*):
C
C              =-1 NOT TO PRINT X(*),
C
C              =0 TO PRINT THE FINAL X(*),
C
C              =1 TO PRINT ALSO THE INITIAL X(*),
C
C              =2 TO PRINT ALSO AT "UNUSUAL RESTARTS",
C
C              =3 TO PRINT AT ALL RESTARTS,
C
C              =4 TO PRINT AT THE END OF EACH ITERATION
C
C    CALL FUN (NV,X,FX,GRAD,IOPT,NFV,NGV)
C
C    IOPT   -- =-1 TO COMPUTE ONLY FX=F(X(*)),
C
C            = 0 TO COMPUTE BOTH FX AND GRAD(*),
C
C            =+1 TO COMPUTE ONLY GRAD(*)
C
C********************************
C
      IMPLICIT REAL*8 (A-H,O-Z)
C
      INTEGER METHCG,MAXCYC,MAXNFV,MINLOC,NOREST,NTRY,
     *  NTRACC,NTRACX,LPCG,KFLAGC,NCYCLE,ITERCG,MAXFWD,
     *  NFMAXB,NFMAXL,KFLAGD,KONNEW
      INTEGER NV,LP,IOPT,J,KOUNT,KTRYSW,MAXIT,MRESTA,MNORES,
     *  KPATH,JFIRST,MASK,MASKT
C
      DOUBLE PRECISION DX,DXSAVE,RELTLC,ABSTLC,GRNORM,
     *  RELTLD,ABSTLD,DECMAX,FBINCR,FBDECR
      DOUBLE PRECISION X,XSAVE,GRAD,GRSAVE,GRPREV,XDIR,
     *  XDSAVE,XMAX,XMIN,XBASE,ZZ,SDENOM,SNUMER,
     *  ZABS,ARG,DABS,ZSQRT,DSQRT,FX,RZERO,
     *  SUM,XNORM,SUMG
C
      DIMENSION X(1),XSAVE(1),GRAD(1),GRSAVE(1),GRPREV(1),
     *  XDIR(1),XDSAVE(1),XBASE(1),XMAX(1),XMIN(1),MASK(1),
     *  MASKT(1),HHG(1),HHY(1),XTDIR(1),YT(1)
C
      COMMON /COMCG/ DX,DXSAVE,RELTLC,ABSTLC,GRNORM,NFV,NGV,
     *  RELTLD,ABSTLD,DECMAX,FBINCR,FBDECR,LINEOP,METHRT,
     *  METHCG,MAXCYC,MAXNFV,MINLOC,NOREST,NTRY,NTRACC,
     *  NTRACX,LPCG,KFLAGC,NCYCLE,ITERCG,MAXFWD,NFMAXB,
     *  NFMAXL,KFLAGD,KONNEW
C
      COMMON /NP/NPROB
      COMMON /LO/LOUT
C
      ZABS(ARG)=DABS(ARG)
      ZSQRT(ARG)=DSQRT(ARG)
C
      FSKIP=1.0D0
C
      RZERO=0.0D0
      LP=LPCG
      MAXIT=NV+1
C
      NCYCLE=1
      NFV=0
      NGV=0
      MRESTA=0
      MNORES=0
C
      IOPT=0
      CALL FCN(NV,X,FX)
      CALL FGRAD (NV,X,GRAD)
C
      JFIRST=1
      JRST=0
C     JRTING=0
C
C  BEGIN THE NEXT CYCLE OF ITERATIONS.
C  EACH ITERATION CONSISTS OF A LINE SEARCH ALONG A DIRECTION
C  VECTOR AND THE COMPUTATION OF A NEW DIRECTION VECTOR.
C
   10 SUMG=RZERO
      DO 20 J=1,NV
        IF(MASK(J).NE.0 .OR.
     *     X(J).EQ.XMAX(J) .AND. GRAD(J).LT.RZERO .OR.
     *     X(J).EQ.XMIN(J) .AND. GRAD(J).GT.RZERO)
     *     GRAD(J)=RZERO
        XDIR(J)=-GRAD(J)
        SUMG=SUMG+GRAD(J)**2
   20   CONTINUE
      GRNORM=ZSQRT(SUMG)
C
      IF(NTRACC.GE.0 .AND. JFIRST.EQ.1) WRITE(LP,30)NV,NTRACC,
     *  NTRACX,METHCG,NOREST,NTRY,RELTLC,ABSTLC,MINLOC,DX,FX,
     *  GRNORM
   30 FORMAT(//'1. ENTER SUBROUTINE CG.'//6X,'NV =',I11,6X,
     *  'NTRACC =',I3,6X,'NTRACX =',I3//6X,'METHCG =',I2,6X,
     *  'NOREST =',I11,6X,'NTRY =',I11//6X,'RELTLC =',
     *  1PG15.7,6X,'ABSTLC =',G15.7//6X,'MINLOC =',I5,
     *  6X,'DX =',G15.7//6X,'FX =',G15.7,'GRNORM =',G15.7//
     *  ' ')
      IF(NTRACX.GE.1 .AND. JFIRST.EQ.1)
     *  WRITE(LP,40)(X(J),J=1,NV)
   40 FORMAT(/' X(*)=',1PG15.7,4G15.7/(6X,5G15.7))
C
      IF(NTRACC.GE.3 .AND. JFIRST.NE.1) WRITE(LP,50)NCYCLE,
     *  GRNORM,DX,NFV,NGV,MRESTA,MNORES,FX
   50 FORMAT(/' CG.   BEGIN CYCLE NUMBER',I10/6X,'GRNORM =',
     *  1PG15.7,6X,'DX =',G15.7/6X,'NFV =',I8,'  NGV =',
     *  I8/6X,'MRESTA =',I10,6X,'MNORES =',I10,6X,
     *  'FX =',G15.7)
C
      JFIRST=0
C
      ITERCG=1
C
C  START A NEW CYCLE OF ITERATIONS, OR RESTART.
C
   60 KOUNT=0
```

```
      KTRYSW=0
C
C  BEGIN THE NEXT ITERATION.
C
   70 IF(LINEOP.EQ.1) THEN
         CALL LSEARCH(NV,X,XBASE,FX,XDIR,GRAD,GRPREV)
      ELSE
         CALL ONEDIR (NV,X,XBASE,XDIR,GRAD,GRPREV,XMAX,XMIN,
     *   MASK,MASKT,FX)
      ENDIF
C
C  COMPUTE THE NEXT DIRECTION VECTOR, XDIR(*).
C
C  FIRST, PROJECT ALL VECTORS ONTO ANY ACTIVE CONSTRAINTS.
C
      DO 80 J=1,NV
        IF(MASK(J).NE.0 .OR.
     *     X(J).EQ.XMAX(J) .AND. GRAD(J).LE.RZERO .OR.
     *     X(J).EQ.XMIN(J) .AND. GRAD(J).GE.RZERO) THEN
          GRAD(J)=RZERO
          GRPREV(J)=RZERO
          XDIR(J)=RZERO
        ENDIF
   80   CONTINUE
C
      IF(JRST.EQ.1) GO TO 121
C
   85 SNUMER=RZERO
      SDENOM=RZERO
C
      IF(METHCG.EQ.1) THEN
C
C  METHCG .EQ. 1 :  METHOD OF FLETCHER AND REEVES
C
        DO 90 J=1,NV
          SNUMER=SNUMER+GRAD(J)**2
          SDENOM=SDENOM+GRPREV(J)**2
   90     CONTINUE
C
      ELSE IF(METHCG.EQ.2) THEN
C
C  METHCG .EQ. 2 :  METHOD OF POLAK AND RIBIERE
C
        DO 100 J=1,NV
          SNUMER=SNUMER+GRAD(J)*(GRAD(J)-GRPREV(J))
          SDENOM=SDENOM+GRPREV(J)**2
  100     CONTINUE
C
      ELSE IF(METHCG.EQ.3) THEN
C
C  METHCG .EQ. 3 :  METHOD OF HESTENES AND STIEFEL,
C                   AND SORENSON
C
        DO 110 J=1,NV
          SNUMER=SNUMER+GRAD(J)*(GRAD(J)-GRPREV(J))
          SDENOM=SDENOM+XDIR(J)*(GRAD(J)-GRPREV(J))
  110     CONTINUE
C
      ELSE
C
C  METHCG .EQ. 4 :  METHOD OF PERRY
C
        XNORM=RZERO
        XDIRNORM=RZERO
        DO 115 J=1,NV
          XNORM=XNORM+(X(J)-XBASE(J))**2
          XDIRNORM=XDIRNORM+XDIR(J)**2
  115     CONTINUE
      AK=DSQRT(XNORM)/DSQRT(XDIRNORM)
      DO 120 J=1,NV
        SNUMER=SNUMER+GRAD(J)*(GRAD(J)-GRPREV(J)-AK*XDIR(J))
        SDENOM=SDENOM+XDIR(J)*(GRAD(J)-GRPREV(J))
  120     CONTINUE
     ENDIF
C
     IF(SDENOM.EQ.RZERO) THEN
       IF(KFLAGD.EQ.0) KFLAGD=-5
       GO TO 150
     ENDIF
C
C  METHRT=2:  METHOD OF POWELL RESTART SCHEME
C
 121 IF(METHRT.EQ.2.AND. JRST.EQ.1) THEN
C
       YKGK1=RZERO
       DKYK=RZERO
       YTGK1=RZERO
       DTYT=RZERO
       DO 122 J=1,NV
        YKGK1=YKGK1+(GRAD(J)-GRPREV(J))*GRAD(J)
        DKYK=DKYK + XDIR(J)*(GRAD(J)-GRPREV(J))
        YTGK1=YTGK1+YT(J)*GRAD(J)
        DTYT=DTYT + XTDIR(J)*YT(J)
 122   CONTINUE
C
       IF(DKYK.EQ.RZERO .OR. DTYT.EQ.RZERO) THEN
         PRINT *,"=== RESTART WITH -GRAD(*)"
         JRST=0
C          JRTING=0
         GO TO 10
       ENDIF
C
       DO 123 J=1,NV
        XDIR(J)= -GRAD(J) + YKGK1*XDIR(J)/DKYK +
   &     YTGK1*XTDIR(J)/DTYT
 123   CONTINUE
C
       XDIRNORM=RZERO
       DO 124 J=1,NV
         XDIRNORM=XDIRNORM+XDIR(J)**2
 124   CONTINUE
       IF(XDIRNORM.EQ.RZERO) THEN
         PRINT *," XDIR = RZERO RESTART WITH -G"
         GO TO 10
       ENDIF
C
       IF(ITERCG.GE.NV-1) THEN
C          JRTING=0
         JRST=0
         ITERCG=1
         GO TO 70
       ENDIF
C
       GO TO 150
     ENDIF
C
C  METHRT=1: METHOD OF SHANNO (COMBINE BEALE AND POWELL RESTART SCHEME)
C
     IF(METHRT.EQ.1.AND. JRST.EQ.1) THEN
C
       XNORM=RZERO
       XDIRNORM=RZERO
C
       DO 125 J=1,NV
```

```
        IF(XDIRNORM.EQ.RZERO) THEN
           PRINT *," XDIR = RZERO RESTART WITH -G"
           GO TO 10
        ENDIF
C
        IF(ITERCG.GE.NV-1) THEN
C=          JRTING=0
          JRST=0
          ITERCG=1
          GO TO 70
        ENDIF
C
        GO TO 150
      ENDIF

C
C  KEEP Yt, Dt,  AND At TO BE USED TO COMPUTE Pt in my implementation:
C
        XNORM=RZERO
        XTDIRNORM=RZERO
        DO 138 J=1,NV
          YT(J)=GRAD(J)-GRPREV(J)
          XTDIR(J)=XDIR(J)
          XNORM=XNORM+(X(J)-XBASE(J))**2
          XTDIRNORM=XTDIRNORM+XTDIR(J)**2
 138   CONTINUE
        IF(XTDIRNORM.EQ.RZERO) GO TO 10
        AT=DSQRT(XNORM)/DSQRT(XTDIRNORM)
C
      ZZ=SNUMER/SDENOM
C
      DO 140 J=1,NV
        XDIR(J)=ZZ*XDIR(J)-GRAD(J)
 140   CONTINUE
C
 150 SUMG=RZERO
      SUMD=RZERO
      DO 160 J=1,NV
        SUMG=SUMG+GRAD(J)**2
        SUMD=SUMD+XDIR(J)**2
 160   CONTINUE
      GRNORM=ZSQRT(SUMG)
      XDNORM=ZSQRT(SUMD)
C
C  DAMP OUT RADICAL CHANGES IN DX, ESPECIALLY DECREASES.
C
      DX=ZSQRT(DXSAVE)*ZSQRT(ZABS(DX))
      IF(DX.LT.DXSAVE/DECMAX) DX=DXSAVE/DECMAX
C
      IF(NTRACC.GE.3) WRITE(LP,170)NCYCLE,ITERCG,NFV,NGV,
     *  FX,NFBRAC,NEVALL,KFLAGB,KFLAGL,GRNORM,XDNORM,DX
 170 FORMAT(/' ONEDIR.',6X,'NCYCLE =',I10,6X,
     *  'END OF ITERATION',I10/
     *  6X,'NFV =',I8,6X,'NGV =',I8,6X,'FX =',G15.7/
     *  6X,'NFBRAC =',I3,6X,'NEVALL =',I2,
     *  6X,'KFLAGB =',I3,6X,'KFLAGL =',I3/
     *  6X,'GRNORM =',G15.7,3X,'XDNORM =',G15.7,3X,'DX =',
     *  G15.7)
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
      IF(NTRACX.GE.4) WRITE(LP,40)(X(J),J=1,NV)
C
      SUM=RZERO
      DO 180 J=1,NV
        SUM=SUM+X(J)**2
 180   CONTINUE
```

```
      XNORM=ZSQRT(SUM)
C
      IF(FSGRAD(NV,X,GRAD).LT.1.0D-5) THEN
        IF(NTRACC.GE.0) WRITE(LP,190)
 190    FORMAT(/' SUBROUTINE CG CONVERGED NORMALLY.')
        KFLAGC=0
        GO TO 360
      ENDIF
C
      ITERCG=ITERCG+1
C
      IF(METHRT.GT.0.AND.JRST.EQ.0) THEN
C                                                          2
C  RESTART (POWELL) EVERY NV STEPS OR WHENEVER |g'  g |>=0.2*||g  ||
C                                          k+1 k        k+1
        IF(ITERCG.GE.MAXIT) GO TO 195
        GGR=RZERO
        GNORM=RZERO
        DO 193 J=1,NV
          GGR  =GGR  + GRAD(J)*GRPREV(J)
          GNORM=GNORM + GRAD(J)*GRAD(J)
 193    CONTINUE
        IF(DABS(GGR).LT.2.0D-1*GNORM) GO TO 70

 195    PRINT *,"RESTART WITH SHANNO or POWELL RESTART CRITERION"
        JRST=1
        ITERCG=1

        NCYCLE=NCYCLE+1
        GO TO 70
      ENDIF
C
C
      IF(KFLAGD.LE.-1) THEN
C
C  SUBROUTINE ONEDIR COULD NOT COMPLETE ITS FUNCTION NORMALLY.
C
        IF(JRST.EQ.1) JRTFAIL=1
        KPATH=1
        GO TO 320
      ENDIF
C
      IF(ITERCG.GT.MAXIT) GO TO 340
      GO TO 70
C
C
 320 IF(NTRACC.GE.0) WRITE(LP,330)KPATH,KFLAGD,NCYCLE,ITERCG,
    *   KONNEW,NFV,NGV,DX,FX,GRNORM
 330 FORMAT(/' UNUSUAL RESTART IN SUBROUTINE CG.'/
    *  6X,'KPATH =',I2,6X,'KFLAGD =',I3,6X,'NCYCLE =',I10,
    *  6X,'ITERCG =',I10/
    *  6X,'KONNEW =',I7,6X,'NFV =',I8,6X,'NGV =',I8/
    *  6X,'DX =',G15.7,3X,'FX =',G15.7,3X,'GRNORM =',G15.7)
C
C  A CYCLE OF ITERATIONS HAS BEEN COMPLETED, OR HAS TERMINATED
C  PREMATURELY.  START A NEW CYCLE.
C
 340 NCYCLE=NCYCLE+1
      IF(JRTFAIL.EQ.1) THEN
         JRST=0
         JRTFAIL=0
         GO TO 85
        ENDIF
      IF(NCYCLE.LE.MAXCYC) GO TO 10
C
      KFLAGC=-1
      IF(NTRACC.GE.0) WRITE(LP,350)MAXCYC
 350 FORMAT(/' MAXCYC =',I9,' EXCEEDED IN SUBROUTINE CG.')
C
 360 IF(NTRACC.GE.0) WRITE(LP,370)KFLAGC,NCYCLE,NFV,NGV,
   *  MRESTA,MNORES,DX,FX,GRNORM
 370 FORMAT(//' EXIT FROM SUBROUTINE CG.',6X,'KFLAGC =',I3//
   *  6X,'NCYCLE =',I11,6X,'NFV =',I8,6X,'NGV =',I8//
   *  6X,'MRESTA =',I8,6X,'MNORES =',I8//
   *  6X,'DX =',G15.7,3X,'FX =',G15.7,3X,'GRNORM =',G15.7)
    IF(NTRACX.GE.0) WRITE(LP,40)(X(J),J=1,NV)
C
C=     WRITE(LOUT,380)FX,GRNORM
C=  380 FORMAT(/'FX =',G15.7,3X,' GRNORM = ',G15.7)
C     IF(NTRACX.GE.0) WRITE(LOUT,40)(X(J),J=1,NV)
C
    RETURN
C
C  END CG
C
    END
C
C
C
C **************************************************************************
    SUBROUTINE LSEARCH(N,X,XBASE,FX,XDIR,GRAD,GRPREV)
C **************************************************************************
C
C   NASH LINE SEARCH METHOD                         MODIFIED BY MEISHAN CHENG
C
C  USING SUCCESS/FAILURE LINE SEARCH METHOD
C         IFLAG=0 IF FAILURE AND NEED TO STOP
C         IFLAG=-1 IF FAILURE AND NEED TO TRY ONE MORE ITERATION
C         IFLAG=1 IF SUCCESS
C
    IMPLICIT REAL*8 (A-H,O-Z)
C
    DOUBLE PRECISION FX,FXSAVE,XDIR,X,GRAD,GRPREV
C
    COMMON /COMCG/ DX,DXSAVE,RELTLC,ABSTLC,GRNORM,NFV,NGV,
   *  RELTLD,ABSTLD,DECMAX,FBINCR,FBDECR,LINEOP,METHRT,
   *  METHCG,MAXCYC,MAXNFV,MINLOC,NOREST,NTRY,NTRACC,
   *  NTRACX,LPCG,KFLAGC,NCYCLE,ITERCG,MAXFWD,NFMAXB,
   *  NFMAXL,KFLAGD,KONNEW

C
    COMMON /IFLAG1/IOVER
    COMMON /IFLAG2/IUNDER
C
    DIMENSION X(1),XBASE(1),XDIR(1),GRAD(1),GRPREV(1)
C
    RZERO=0.0D0
    UNITR=1.0D0
C
    KFLAGD=0
C
    SUMD=RZERO
    DO 5 J=1,N
      XBASE(J)=X(J)
      GRPREV(J)=GRAD(J)
  5   CONTINUE
C
    DX=DABS(DX)
    DXSAVE=DX
C
C  BEGIN LINEAR SEARCH
C
C  SET THE SUCCESS/FIALURE PARAMETERS
C
```

```
      A1=1.5D0
      A2=-2.0D-1
C
C  LET SMSTEP=0, THE POSITION ALONG THE SEARCH DIRECTION OF THE LOWEST
C  POINT FOUND SO FAR
C
      SMSTEP=RZERO
      FXSAVE=FX

C  IT IS TRIAL STEP LENTH
C
   90 STEP=DX
      CALL RESETB(N,X,XBASE,XDIR,ICOUNT,STEP)
C
  110 IF(ICOUNT.GT.N) THEN
      DX=DX*1.0D1
      STEP=DX
      CALL RESETB(N,X,XBASE,XDIR,ICOUNT,STEP)
      ENDIF
C
  120 STEP=SMSTEP+DX
C
  130 CALL RESETB(N,X,XBASE,XDIR,ICOUNT,STEP)
      IF(ICOUNT.EQ.N) GO TO 240
C
  140 CALL FCN(N,X,P)
C
C                             FUN EVALUATION 'SUCCESS'
  150 IF(IOVER.EQ.0 .AND. P.LT.FXSAVE) THEN
C                             ACTION IN CASE OF 'SUCCESS'
        FXSAVE=P
        SMSTEP=STEP
        DX=A1*DX
C
C                             ACTION IN CASE OF 'FAILURE'.
  160 ELSEIF(FXSAVE.LT.FX) THEN
        GO TO 180
      ELSE
        DX=A2*DX
      ENDIF
C                             CONTINUE TO SEARCH
      GO TO 120
C
C  PARABOLIC INVERSE INTERPOLATION
C
C  IF THE FAILURE HAS OCCURED DUE TO NON-COMPUTABILITY OF THE FUNCTION
C  GOTO STEP 23 TO RESET PARAMETERS TO THOSE AT LOWEST POINT SO FAR.
C
  180 IF(IOVER.EQ.1) GO TO 230
C
  190 STEP=0.5D0*((FXSAVE-FX)*(STEP**2)-(P-FX)*(SMSTEP**2))/
     &    ((FXSAVE-FX)*STEP-(P-FX)*SMSTEP)
C
  200 CALL RESETB(N,X,XBASE,XDIR,ICOUNT,STEP)
      IF(ICOUNT.EQ.N) GO TO 230
C
  210 CALL FCN(N,X,P)
C
C IF THE FUNCTION CANNOT BE COMPUTED, GO TO 230 TO RESET X
C ELSE IT IS END OF LINE SEARCH, GOTO 240 TO SEE IF IT IS SUCCESSFUL
C
  220  IF(IOVER.EQ.0 .AND. P.LT.FXSAVE) THEN
        FXSAVE=P
        SMSTEP=STEP
        GO TO 240
      ENDIF
C
```

```
  230 STEP=SMSTEP
      CALL RESETB(N,X,XBASE,XDIR,ICOUNT,STEP)
C
C UNSUCCESSFUL SEARCH ALONG THE STEEPEST DESCENT DIRECTION
C INDICATES CONVERGECE
C
  240 IF(FXSAVE.GE.FX) THEN
        KFLAGD=-4
        DX=DABS(SMSTEP)
        RETURN
      ENDIF
C
C SUCCESSFUL SEARCH
C SAVE THE NEW LOWEST FUNCTION VALUE
C
  260 FX=FXSAVE
      DX=DABS(SMSTEP)
      KFLAGD=0
C
  300 CALL FGRAD(N,X,GRAD)
      RETURN
      END
C
C ***************************************************************************
      SUBROUTINE RESETB(N,X,XBASE,XDIR,ICOUNT,STEP)
C ***************************************************************************
C
      IMPLICIT REAL*8 (A-H,O-Z)
C
      DIMENSION X(1),XBASE(1),XDIR(1)
C
      ICOUNT=0
C
      DO 5 I=1,N
       X(I)=XBASE(I)+STEP*XDIR(I)
       IF(X(I).EQ.XBASE(I)) THEN
         ICOUNT=ICOUNT+1
       ENDIF
    5 CONTINUE
C
      RETURN
      END
C
C
C
C*************************************************************************
      SUBROUTINE ONEDIR (NV,X,XBASE,XDIR,GRAD,GRPREV,
     *  XMAX,XMIN,MASK,MASKT,FX)
C
C  ONEDIR 1.0        SEPTEMBER 1992
C
C  J. P. CHANDLER, COMPUTER SCIENCE DEPARTMENT,
C    OKLAHOMA STATE UNIVERSITY
C
C  CARRIES OUT A LINE SEARCH FOR NONLINEAR OPTIMIZATION BY
C  CONJUGATE GRADIENT METHODS, AND COMPUTES THE NEXT DIRECTION
C  VECTOR XDIR(*).
C
C
C    KFLAGD  -- A CONDITION CODE, RETURNED
C
C              = 0 IF SUBROUTINE ONEDIR OPERATED
C                NORMALLY,
C
C              =-1 IF SUBROUTINE BRACMN FAILED TO BRACKET
C                A MINIMUM OR TO REDUCE THE VALUE OF FX,
C
```

```
C              =-2 IF SUBROUTINE LOCBAC FAILED TO CONVERGE
C                WITHIN NFMAXB ITERATIONS,
C
C              =-3 IF THE BEST VALUE OF XX WAS LESS THAN
C                ZERO,
C
C              =-4 IF FX WAS NOT REDUCED IN ONEDIR,
C
C              =-5 IF THE DENOMINATOR FOR COMPUTING THE
C                NEXT DIRECTION VECTOR WAS EQUAL TO ZERO,
C
C              =-6 IF A NEW CONSTRAINT BECAME ACTIVE
C
      IMPLICIT REAL*8 (A-H,O-Z)
C
      INTEGER MASK,MASKT,NV
      INTEGER METHCG,MAXCYC,MAXNFV,MINLOC,NOREST,NTRY,
     *  NTRACC,NTRACX,LPCG,KFLAGC,NCYCLE,ITERCG,MAXFWD,
     *  NFMAXB,NFMAXL,KFLAGD,IOPT,J,KFLAGB,KFLAGL,LP,
     *  NEVALL,NFBRAC,NFLOC,NTRACB,KONOLD,KONNEW
C
      DOUBLE PRECISION DX,RELTLC,ABSTLC,GRNORM,
     *  RELTLD,ABSTLD,DECMAX,FBINCR,FBDECR
      DOUBLE PRECISION X,XDIR,GRAD,GRPREV,XBASE,XMAX,XMIN
      DOUBLE PRECISION A,B,DABS,DSQRT,DXSAVE,FBESAV,FXBEST,
     *  FFB,FP,FR,FXSAVE,FX,P,R,REALSV,RZERO,
     *  SUMD,XXBEST,XDNORM,XFB,XX,
     *  XXFAC,ZABS,ZSQRT,XXSAVE,ZSIGN,ARGB,DSIGN,UNITR,
     *  DXHOLD
C
      DIMENSION X(1),XDIR(1),GRAD(1),GRPREV(1),
     *  XBASE(1),XMAX(1),XMIN(1),MASK(1),MASKT(1)
C
      DIMENSION XFB(2),FFB(2),REALSV(8)
C
      COMMON /COMCG/ DX,DXSAVE,RELTLC,ABSTLC,GRNORM,NFV,NGV,
     *  RELTLD,ABSTLD,DECMAX,FBINCR,FBDECR,LINEOP,METHRT,
     *  METHCG,MAXCYC,MAXNFV,MINLOC,NOREST,NTRY,NTRACC,
     *  NTRACX,LPCG,KFLAGC,NCYCLE,ITERCG,MAXFWD,NFMAXB,
     *  NFMAXL,KFLAGD,KONNEW
C
      COMMON /NP/NPROB
C
      ZABS(ARG)=DABS(ARG)
      ZSQRT(ARG)=DSQRT(ARG)
      ZSIGN(ARG,ARGB)=DSIGN(ARG,ARGB)
C
      RZERO=0.0D0
      UNITR=1.0D0
C
      LP=LPCG
C
      NTRACB=0
      IF(NTRACC.GE.4) NTRACB=2
C
      KFLAGD=0
C
      SUMD=RZERO
      DO 10 J=1,NV
        XBASE(J)=X(J)
        GRPREV(J)=GRAD(J)
        SUMD=SUMD+XDIR(J)**2
   10   CONTINUE
C
      XDNORM=ZSQRT(SUMD)
      IF(XDNORM.EQ.RZERO) XDNORM=UNITR
C
```

```
      DX=ZABS(DX)
      DXSAVE=DX
C
C********************************
C
C  SEARCH ALONG THE LINE
C     X(*) = XBASE(*) + XX * (XDIR(*)/XDNORM)
C  FOR A LOCAL MINIMUM OF FX AS A FUNCTION OF XX.
C  FIRST CALL BRACMN TO BRACKET THE MINIMUM.
C
      KFLAGB=0
      XX=RZERO
      FXSAVE=FX
      DX=ZABS(DX)
      KONNEW=0
      NEVALL=0
      KFLAGL=-99
C
   20 XXSAVE=XX
      DXHOLD=DX
      CALL BRACMN (KFLAGB,XX,FX,DX,FBINCR,FBDECR,MAXFWD,NFMAXB,
     *  NTRACB,LP,NFBRAC,XXBEST,FXBEST,XFB,FFB)
C
      IF(KONNEW.GT.0 .AND. ZABS(XX).GT.ZABS(XXSAVE) .AND.
     *  ZSIGN(UNITR,XX).EQ.ZSIGN(UNITR,XXSAVE)) THEN
        DX=DXHOLD
        KFLAGD=-6
        GO TO 60
      ENDIF
C
      IF(KFLAGB.GE.1) THEN
        XXFAC=XX/XDNORM
        CALL ONESTP (NV,X,XBASE,XDIR,XXFAC,XMAX,XMIN,
     *     MASK,MASKT,KONOLD,KONNEW)
        IF(KONNEW.GT.0) DX=DXHOLD
        IOPT=-1
        CALL FCN (NV,X,FX)
        GO TO 20
      ENDIF
C
      IF(KFLAGB.LE.-4) THEN
        KFLAGD=-1
        GO TO 60
      ENDIF
C
C  SUBROUTINE BRACMN HAS SUCCEEDED IN BRACKETTING A LOCAL
C  MINIMUM OF FX, OR ELSE THE LINE SEARCH HAS HIT A CONSTRAINT.
C  CALL SUBROUTINE LOCBAC TO REFINE THE POSITION OF THE
C  MINIMUM.
C
      IF(XFB(1).LT.XFB(2)) THEN
        P=XFB(1)
        FP=FFB(1)
        R=XFB(2)
        FR=FFB(2)
      ELSE
        P=XFB(2)
        FP=FFB(2)
        R=XFB(1)
        FR=FFB(1)
      ENDIF
C
      A=P
      B=R
      KFLAGL=3
      NFLOC=2
      IF(FP.LT.FR) THEN
```

```
          REALSV(1)=P
          REALSV(2)=FP
          REALSV(3)=R
          REALSV(4)=FR
        ELSE
          REALSV(1)=R
          REALSV(2)=FR
          REALSV(3)=P
          REALSV(4)=FP
        ENDIF
        REALSV(5)=REALSV(3)
        REALSV(6)=REALSV(4)
        REALSV(8)=R-P
        REALSV(7)=REALSV(8)
C
   30 CALL LOCBAC (A,B,RELTLD,ABSTLD,NFMAXL,KFLAGL,
     *   XX,FX,NFLOC,REALSV)
C
        FBESAV=FXBEST
C
        IF(KFLAGL.GE.1) THEN
C
          XXFAC=XX/XDNORM
          CALL ONESTP (NV,X,XBASE,XDIR,XXFAC,XMAX,XMIN,
     *      MASK,MASKT,KONOLD,KONNEW)
C
          IOPT=-1
          IF(NEVALL.EQ.MINLOC-1) IOPT=0
          IF(IOPT.EQ.-1) THEN
             CALL FCN (NV,X,FX)
          ELSE
             CALL FCN (NV,X,FX)
             CALL FGRAD(NV,X,GRAD)
          ENDIF
          NEVALL=NEVALL+1
C
          IF(NTRACC.GE.4) WRITE(LP,40)NCYCLE,ITERCG,KFLAGL,NEVALL,
     *      IOPT,NFLOC,FX,FXSAVE,FXBEST
   40     FORMAT(/' LOCBAC.',6X,'NCYCLE =',I10,6X,'ITERCG =',I10,
     *       6X,'KFLAGL =',I3/6X,'NEVALL =',I3,6X,'IOPT =',I3,
     *       6X,'NFLOC =',I3,6X,'FX =',1PG15.7/6X,'FXSAVE =',
     *       G15.7,6X,'FXBEST =',G15.7)
C
          IF(FX.LT.FXBEST) THEN
            XXBEST=XX
            FXBEST=FX
          ENDIF
C
          IF(NEVALL.LT.MINLOC .OR. FXBEST.GE.FXSAVE) GO TO 30
C
        ELSE IF(KFLAGL.LE.-1) THEN
C
          KFLAGD=-2
C
        ENDIF
C
        XX=XXBEST
C
        IF(NTRACC.GE.4) WRITE(LP,50)A,B,KFLAGL,XX,FX,NEVALL
   50 FORMAT(/' LOCBAC FINIS.',6X,'A =',1PG15.7,6X,'B =',G15.7,
     *   6X,'KFLAGL =',I3/
     *   6X,'XX =',G15.7,6X,'FX =',G15.7,6X,'NEVALL =',I4)
C
        IF(KFLAGL.GE.0 .AND. FX.EQ.FXBEST .AND. IOPT.EQ.0)
     *   GO TO 90
C
C  GRAD(*) IS NOT THE GRADIENT EVALUATED AT XXBEST.
```

```
C  GO BACK TO THE BEST KNOWN POINT.
C
  60 XXFAC=XX/XDNORM
     CALL ONESTP (NV,X,XBASE,XDIR,XXFAC,XMAX,XMIN,
    *   MASK,MASKT,KONOLD,KONNEW)
C
C  COMPUTE THE GRADIENT VECTOR GRAD(*) AT THE BEST
C  KNOWN POINT, AND RECOMPUTE FX THERE ALSO.
C
     IF(NTRACC.GE.3) WRITE(LP,70)NCYCLE,ITERCG,KFLAGL,NFBRAC,
    * NEVALL,IOPT,FXSAVE,FBESAV,FX,FXBEST
  70 FORMAT(/' EXTRA CALL TO SUBROUTINE FUN IN ONEDIR.'/
    *  6X,'NCYCLE =',I10,6X,'ITERCG =',I10,6X,'KFLAGL =',I3/
    *  6X,'NFBRAC =',I3,6X,'NEVALL =',I3,6X,'IOPT =',I3/
    *  6X,'FXSAVE =',1PG15.7,6X,'FBESAV =',G15.7/
    *  6X,'FX =',G15.7,6X,'FXBEST =',G15.7)
C
     IOPT=0
     CALL FCN (NV,X,FX)
     CALL FGRAD (NV,X,GRAD)
C
     IF(NTRACC.GE.3) WRITE(LP,80)FX
  80 FORMAT(6X,'NEW VALUE OF FX =',1PG15.7)
C
  90 IF(FX.GE.FXSAVE) THEN
       KFLAGD=-4
     ELSE IF(XX.LE.RZERO) THEN
       KFLAGD=-3
     ENDIF
C
C
     RETURN
C
C  END ONEDIR
C
     END
C
C
C**********************************************************************
     SUBROUTINE ONESTP (NV,X,XBASE,XDIR,XXFAC,XMAX,XMIN,
    *              MASK,MASKT,KONOLD,KONNEW)
C
C  ONESTP 1.0        SEPTEMBER 1992
C
C  TAKE ONE STEP FOR SUBROUTINE ONEDIR, SATISFYING ALL
C  CONSTRAINTS XMAX(*) AND XMIN(*).
C
     IMPLICIT REAL*8 (A-H,O-Z)
C
     INTEGER NV,MASK,MASKT,KONOLD,KONNEW,
    *  J,JFACEQ,JFMIN,JFSIGN,KONSAV
C
     DOUBLE PRECISION X,XBASE,XDIR,XMAX,XMIN,XXFAC,
    *  DABS,FAC,FACMIN,RZERO,ZABS
C
     DIMENSION X(1),XBASE(1),XDIR(1),XMAX(1),XMIN(1),
    *  MASK(1),MASKT(1)
C
     COMMON /NP/NPROB
C
     ZABS(ARG)=DABS(ARG)
C
     RZERO=0.0D0
C
C  SET X(J) AND MASKT(J) FOR EVERY XBASE(J) THAT IS ON A
C  CONSTRAINT SURFACE AND FOR WHICH THE PROPOSED STEP WOULD
C  LIE ON THE WRONG SIDE OF A CONSTRAINT OR WOULD LIE EXACTLY
```

```
C  ON THE CONSTRAINT SURFACE ITSELF.
C  THESE CONSTRAINTS ARE CALLED "OLD ACTIVE CONSTRAINTS".
C  KONOLD COUNTS THE NUMBER OF SUCH CONSTRAINTS.
C
      KONOLD=0
      DO 10 J=1,NV
C
        MASKT(J)=MASK(J)
        IF(MASK(J).NE.0) GO TO 10
C
        IF(XMAX(J).LT.XMIN(J)) THEN
          XMAX(J)=XBASE(J)
          XMIN(J)=XBASE(J)
        ENDIF
C
        IF(XBASE(J).GT.XMAX(J)) XBASE(J)=XMAX(J)
        IF(XBASE(J).LT.XMIN(J)) XBASE(J)=XMIN(J)
        X(J)=XBASE(J)
C
        IF(XBASE(J).EQ.XMAX(J) .AND. XXFAC*XDIR(J).GE.RZERO
     *     .OR.
     *     XBASE(J).EQ.XMIN(J) .AND. XXFAC*XDIR(J).LE.RZERO)
     *       THEN
          MASKT(J)=1
          KONOLD=KONOLD+1
        ENDIF
C
   10   CONTINUE
C
C  COMPUTE THE LONGEST STEP THAT CAN BE TAKEN WITHOUT
C  VIOLATING A CONSTRAINT XMAX(J) OR XMIN(J).
C  ANY CONSTRAINT ON WHICH THE RESULTING POINT WILL LIE, AND
C  WHICH IS NOT AN OLD ACTIVE CONSTRAINT, IS A NEW ACTIVE
C  CONSTRAINT.  KONNEW COUNTS THE NUMBER OF SUCH CONSTRAINTS.
C
      KONNEW=0
C
   20 JFMIN=0
      JFACEQ=0
      FACMIN=XXFAC
      DO 30 J=1,NV
        IF(MASKT(J).NE.0) GO TO 30
        X(J)=XBASE(J)+XXFAC*XDIR(J)
        JFSIGN=0
        FAC=FACMIN
C
        IF(X(J).GE.XMAX(J)) THEN
          JFSIGN=1
          IF(XDIR(J).NE.RZERO) FAC=(XMAX(J)-XBASE(J))/XDIR(J)
          X(J)=XMAX(J)
        ENDIF
C
        IF(X(J).LE.XMIN(J)) THEN
          JFSIGN=-1
          IF(XDIR(J).NE.RZERO) FAC=(XMIN(J)-XBASE(J))/XDIR(J)
          X(J)=XMIN(J)
        ENDIF
C
        IF(JFSIGN.NE.0) THEN
          JFACEQ=J
          IF(ZABS(FAC).LT.ZABS(FACMIN)) THEN
            JFMIN=J
            FACMIN=FAC
          ENDIF
        ENDIF
C
   30   CONTINUE
```

```
C
      KONSAV=KONNEW
      IF(JFMIN.GT.0) THEN
        MASKT(JFMIN)=1
        KONNEW=KONNEW+1
        XXFAC=FACMIN
      ELSE IF(JFACEQ.GT.0) THEN
        MASKT(JFACEQ)=1
        KONNEW=KONNEW+1
      ENDIF
C
C  BECAUSE THE DISTANCES FROM XBASE(*) TO TWO OR MORE NEW
C  ACTIVE CONSTRAINTS COULD BE EQUAL, AND BECAUSE OF THE
C  POSSIBILITY OF ROUNDOFF ERROR IN THE EXPRESSIONS
C  XBASE(J)+XXFAC*XDIR(J)  AND  (XMAX(J)-XBASE(J))/XDIR(J) ,
C  ETC., WE NEED TO GO BACK AND CHECK FOR ADDITIONAL POSSIBLE
C  NEW ACTIVE CONSTRAINTS ANYTIME A NEW CONSTRAINT HAS BECOME
C  ACTIVE.  ONLY RARELY, HOWEVER, WILL KONNEW EXCEED ONE.
C
      IF(KONNEW.GT.KONSAV) GO TO 20
C
C  TAKE THE PROPER STEP FOR ALL X(J) THAT ARE NOT ON A
C  CONSTRAINT SURFACE.
C
      DO 40 J=1,NV
        IF(MASKT(J).EQ.0) X(J)=XBASE(J)+XXFAC*XDIR(J)
C
C  IT SHOULD BE IMPOSSIBLE FOR X(J) TO VIOLATE A CONSTRAINT.
C  MAKE THE CODE BULLETPROOF, HOWEVER.
C
        IF(X(J).GT.XMAX(J)) X(J)=XMAX(J)
        IF(X(J).LT.XMIN(J)) X(J)=XMIN(J)
   40   CONTINUE
C
      RETURN
C
C  END ONESTP
C
      END
C $PRINTOFF
C
C
C*******************************************************************
      SUBROUTINE BRACMN (KFLAGB,X,FX,DX,FBINCR,FBDECR,MAXFWD,
     *             NFMAXB,NTRACB,LP,NF,XBEST,FBEST,
     *             XFB,FFB)
C
C  BRACMN 1.0        AUGUST 1992
C
C  A.N.S.I. STANDARD FORTRAN 77
C
C  J. P. CHANDLER, COMPUTER SCIENCE DEPARTMENT,
C    OKLAHOMA STATE UNIVERSITY
C
C  SUBROUTINE BRACMN ATTEMPTS TO BRACKET A LOCAL MINIMUM IN
C  THE FUNCTION F(X).
C  BRACMN USES REVERSE COMMUNICATION TO GET VALUES OF F(X).
C
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C  INPUT QUANTITIES....  KFLAGB,X,FX,DX,MAXFWD,NFMAXB,NTRACB,LP
C
C  OUTPUT QUANTITIES...  KFLAGB,NF,XBEST,FBEST,KFB,XFB(*),FFB(*)
C
C
C    KFLAGB -- A SWITCH VARIABLE.
C             THE USER MUST SET KFLAGB=0 BEFORE THE FIRST
```

```
C            CALL TO SUBROUTINE BRACMN, FOR EACH PROBLEM
C            TO BE SOLVED.
C
C          ON EXIT FROM SUBROUTINE BRACMN, THE VALUE OF
C            KFLAGB HAS THE FOLLOWING MEANING...
C
C          = 1 TO 6 WHEN SUBROUTINE BRACMN REQUIRES THE
C            CALLING MODULE TO COMPUTE A VALUE OF
C            FX=F(X) FOR THE GIVEN VALUE OF X,
C
C          =-1 IF A LOWER VALUE OF F(X) WAS FOUND AND A
C            LOCAL MINIMUM WAS BRACKETTED,
C
C          =-2 IF A LOCAL MINIMUM OF F(X) WAS BRACKETTED
C            BUT A LOWER VALUE OF F(X) WAS NOT FOUND
C            (THE USER CAN NOW CALL SUBROUTINE LOCBAC TO
C            MINIMIZE F(X)),
C
C          =-3 IF A LOWER VALUE OF F(X) WAS FOUND BUT A
C            LOCAL MINIMUM OF F(X) WAS NOT BRACKETTED
C            (THE USER SHOULD EITHER INCREASE NFMAXB OR
C            EXAMINE F(X) TO SEE WHETHER OR NOT IT HAS
C            A LOCAL MINIMUM),
C
C          =-4 IF A LOWER VALUE OF F(X) WAS NOT FOUND AND
C            A LOCAL MINIMUM OF F(X) WAS NOT BRACKETTED
C            (THE USER SHOULD EITHER INCREASE NFMAXB OR
C            DECREASE MAXFWD OR EXAMINE F(X))
C
C   X     -- ON EXIT WITH KFLAGB .GE. 1,
C            X CONTAINS A VALUE FOR WHICH SUBROUTINE
C            BRACMN NEEDS THE VALUE OF FX=F(X).
C            THE USER SHOULD COMPUTE FX AND CALL BRACMN
C            AGAIN, SENDING IT THAT VALUE OF FX.
C
C   FX    -- ON THE INITIAL ENTRY WITH KFLAGB=0, AND ON ANY
C            ENTRY WITH KFLAGB .GE. 1, FX MUST FURNISH
C            THE VALUE OF FX=F(X) FOR THE CURRENT VALUE
C            OF X
C
C   DX    -- DX IS THE SIZE OF THE FIRST STEP TO BE TAKEN
C            IN THE SEARCH
C
C   MAXFWD -- THE MAXIMUM NUMBER OF STEPS TO BE TAKEN IN THE
C            "FORWARD" (+DX) DIRECTION BEFORE TRYING A
C            STEP IN THE "BACKWARD" (-DX) DIRECTION
C            (SUGGESTION: TRY MAXFWD = 3 TO 6)
C
C   NFMAXB -- AN UPPER LIMIT ON THE NUMBER OF TIMES THE
C            FUNCTION F(X) IS TO BE EVALUATED
C            (SUBROUTINE BRACMN MAY RUN OVER THIS LIMIT
C            SLIGHTLY IF IT APPEARS ADVANTAGEOUS)
C
C   NTRACB -- =2 TO OBTAIN DIAGNOSTIC OUTPUT FROM BRACMN,
C
C          =1 TO OBTAIN ONLY FINAL OUTPUT,
C
C          =0 TO OBTAIN NO OUTPUT EXCEPT ERROR MESSAGES
C
C   LP    -- THE LOGICAL UNIT NUMBER FOR PRINTED OUTPUT
C
C   XBEST -- ON EXIT, XBEST CONTAINS THE BEST KNOWN VALUE
C            OF X
C
C   FBEST -- ON EXIT, FBEST CONTAINS THE BEST (LEAST) VALUE
C            OF F(X) THAT HAS BEEN FOUND
C
```

```
C   XFB(*) --  AN ARRAY OF TWO X VALUES
C
C   FFB(*) --  AN ARRAY OF TWO VALUES OF FX=F(X)
C
C
C  USAGE...
C
C  THE USER MUST DIMENSION THE ARRAYS XFB(2) AND FFB(2).
C
C  FOR EACH PROBLEM TO BE SOLVED, THE USER MUST
C    1) INITIALIZE KFLAGB=0,
C    2) SET X TO THE STARTING POINT FOR THE SEARCH,
C    3) SET FX TO THE VALUE OF F(X),
C    4) SET DX, MAXFWD, NFMAXB, NTRACB, AND LP TO THE
C         DESIRED VALUES,
C    5) PROCEED TO CALL SUBROUTINE BRACMN IN A LOOP AS LONG
C         AS KFLAGB .GE. 1 .
C
C  WHENEVER SUBROUTINE BRACMN RETURNS KFLAGB .GE. 1, THE USER
C  COMPUTES THE VALUE OF FX=F(X) FOR THE VALUE OF X RETURNED BY
C  BY BRACMN, AND CALLS SUBROUTINE BRACMN AGAIN.
C
C  WHEN BRACMN RETURNS KFLAGB .LT. ZERO, IT HAS FINISHED THE
C  SEARCH.
C
C  BETWEEN CALLS TO SUBROUTINE BRACMN FOR A GIVEN PROBLEM,
C  THE USER MUST NOT (REPEAT, ****  MUST NOT  ****) ALTER THE
C  VALUES OF KFLAGB, X, DX, NF, XBEST, FBEST, XFB(*), OR FFB(*).
C  TO ALTER ANY OF THESE QUANTITIES IS TO INVITE DISASTER.
C
C*********************************
C
      INTEGER KFLAGB,MAXFWD,NFMAXB,NTRACB,LP,NF,
     *  J,KFB,JFB
C
      DOUBLE PRECISION X,FX,DX,XBEST,FBEST,XFB(2),FFB(2),
     *  FBINCR,FBDECR
C
      COMMON /NP/NPROB
C
      KFB=1
      IF(KFLAGB.GE.5) KFB=2
C
      IF(KFLAGB.LT.0 .OR. KFLAGB.GT.6 .OR. DX.EQ.0.0D0) THEN
        WRITE(LP,10)KFLAGB,DX
   10   FORMAT(//' ERROR IN INPUT TO SUBROUTINE BRACMN....'//
     *    5X,'KFLAGB =',I11,9X,'DX =',1PG15.7)
        STOP
      ENDIF
C
      IF(NTRACB.GE.2) WRITE(LP,20)KFLAGB,DX,X,FX
   20 FORMAT(/' BRACMN.   KFLAGB =',I2,6X,'DX =',1PG15.7/
     *  6X,'X =',G15.7,6X,'FX =',G15.7)
C
      IF(NTRACB.GE.2 .AND. KFLAGB.GE.1)
     *  WRITE(LP,30)NF,KFB,XBEST,FBEST,XFB(1),FFB(1),
     *    XFB(2),FFB(2)
   30 FORMAT(6X,'NF =',I3,4X,'KFB =',I2,4X,'XBEST =',1PG15.7,
     *  3X,'FBEST =',G15.7/6X,'XFB(1) =',G15.7,6X,'FFB(1) =',
     *  G15.7/6X,'XFB(2) =',G15.7,6X,'FFB(2) =',G15.7)
C
      IF(KFLAGB.GE.1) GO TO 50
C
C  KFLAGB .EQ. 0 .
C  THIS IS THE INITIAL ENTRY FOR THIS PROBLEM.
C
      XBEST=X
     FBEST=FX
C
     DO 40 J=1,2
       XFB(J)=X
       FFB(J)=FX
  40   CONTINUE
C
     KFLAGB=1
     NF=0
     GO TO 240
C
C  KFLAG .GE. 1 .
C
  50 NF=NF+1
C
     GO TO (60,80,120,170,180,200),KFLAGB
C
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C  STATE 1:   KFLAGB .EQ. 1   AND   KFB .EQ. 1
C
C  TWO FUNCTION VALUES, FBEST AND FX, ARE NOW AVAILABLE.
C
C  IF (FX-FBEST) STATE 4, STATE 2, STATE 3
C
C
  60 IF(FX-FBEST) 90,70,100
C
  70 KFLAGB=2
     GO TO 230
C
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C  STATE 2:   KFLAGB .EQ. 2   AND   KFB .EQ. 1
C
C  SEARCHING IN THE FORWARD DIRECTION, ALL FUNCTION VALUES
C  FOUND SO FAR ARE EQUAL TO FBEST.
C
C  IF (FX-FBEST) STATE 4, STATE 2, STATE 5
C
C
  80 IF(FX-FBEST)90,230,110
C
  90 KFLAGB=4
     GO TO 220
C
 100 KFLAGB=3
C
 110 XFB(1)=X
     FFB(1)=FX
     GO TO 160
C
C * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
C
C  STATE 3:   KFLAGB .EQ. 3   AND   KFB .EQ. 1
C
C  SEARCHING IN THE FORWARD DIRECTION, ALL FUNCTION VALUES HAVE
C  BEEN GREATER THAN FBEST.
C
C  IF (FX-FBEST) FINISHED, STATE 5, STATE 3
C
C
 120 IF(FX-FBEST)130,150,140
C
 130 XBEST=X
     FBEST=FX
     GO TO 260
```

```
C
 140 XFB(1)=X
     FFB(1)=FX
     IF(NF.LT.MAXFWD .AND. NF.LT.NFMAXB) GO TO 160
C
 150 KFLAGB=5
     KFB=2
     DX=XBEST-X
     GO TO 240
C
 160 DX=DX/FBDECR
     GO TO 240
C
C ********************************
C
C  STATE 4:   KFLAGB .EQ. 4   AND   KFB .EQ. 1
C
C  SEARCHING IN THE FORWARD DIRECTION, A VALUE OF F(X) HAS BEEN
C  FOUND THAT IS LESS THAN FBEST.
C
C  IF (FX-FBEST) STATE 4, FINISHED, FINISHED
C
C
 170 IF(FX-FBEST)220,250,250
C
C ********************************
C
C  STATE 5:   KFLAGB .EQ. 5   AND   KFB .EQ. 2
C
C  SEARCHING IN THE BACKWARD DIRECTION, EITHER BRACMN HAS JUST
C  STARTED IN THIS DIRECTION OR ALL FUNCTION VALUES HAVE BEEN
C  EQUAL TO FBEST.
C
C  IF (FX-FBEST) STATE 6, STATE 5, FINISHED
C
C
 180 IF(FX-FBEST)210,230,190
C
 190 XFB(2)=X
     FFB(2)=FX
     GO TO 270
C
C ********************************
C
C  STATE 6:   KFLAGB .EQ. 6   AND   KFB .EQ. 2
C
C  SEARCHING IN THE BACKWARD DIRECTION, A VALUE OF F(X) HAS
C  BEEN FOUND THAT IS LESS THAN FBEST.
C
C  IF (FX-FBEST) STATE 6, FINISHED, FINISHED
C
C
 200 IF(FX-FBEST) 220,250,250
C
 210 KFLAGB=6
C
C  FX .LT. FBEST .
C  SHIFT THE ITERATES, INCREASE THE STEP SIZE, AND SEARCH
C  FARTHER IN THE SAME DIRECTION.
C
 220 XFB(KFB)=X
     FFB(KFB)=FX
     JFB=3-KFB
     XFB(JFB)=XBEST
     FFB(JFB)=FBEST
     XBEST=X
     FBEST=FX
```

```
      IF(NF.GE.NFMAXB) GO TO 280
C
  230 DX=DX*FBINCR
      IF(NF.GE.NFMAXB) GO TO 290
C
  240 X=XBEST+DX
      IF(X.EQ.XBEST) GO TO 230
C
C  RETURN TO THE CALLING MODULE TO GET THE NEXT VALUE
C  FOR FX=F(X) .
C
      RETURN
C
  250 XFB(KFB)=X
      FFB(KFB)=FX
C
C*********************************
C
C  AN OPTIMUM RESULT HAS BEEN ACHIEVED:
C    A LOWER VALUE OF F(X) HAS BEEN FOUND, AND A LOCAL MINIMUM
C    HAS BEEN BRACKETTED.
C
  260 KFLAGB=-1
      GO TO 300
C
C  A LOCAL MINIMUM HAS BEEN BRACKETTED, BUT A LOWER VALUE OF
C  F(X) HAS NOT BEEN FOUND.
C  THE USER MAY WISH TO CALL SUBROUTINE LOCBAC TO LOCATE
C  THE POSITION OF THE MINIMUM MORE ACCURATELY.
C
  270 KFLAGB=-2
      GO TO 300
C
C  A LOWER VALUE OF F(X) HAS BEEN FOUND, BUT A LOCAL MINIMUM
C  COULD NOT BE BRACKETTED WITHOUT EXCEEDING NFMAXB EVALUATIONS.
C
  280 KFLAGB=-3
      GO TO 300
C
C  A LOCAL MINIMUM HAS NOT BEEN BRACKETTED AND A LOWER VALUE OF
C  F(X) HAS NOT BEEN FOUND.
C
  290 KFLAGB=-4
C
C  SUBROUTINE BRACMN HAS FINISHED THIS PROBLEM, FOR BETTER
C  OR FOR WORSE.
C
  300 X=XBEST
      FX=FBEST
C
      IF(NTRACB.GE.1) WRITE(LP,310)NF,KFLAGB,DX,
     * XBEST,FBEST,XFB(1),FFB(1),XFB(2),FFB(2)
  310 FORMAT(/' BRACMN FINIS.',6X,'NF =',I5,6X,'KFLAGB =',I3,
     *  6X,'DX =',1PG15.7/6X,'XBEST =',
     *  G15.7,6X,'FBEST =',G15.7/6X,'XFB(1) =',G15.7,6X,
     *  'FFB(1) =',G15.7/6X,'XFB(2) =',G15.7,6X,'FFB(2) =',
     *  G15.7)
C
      RETURN
C
C  END BRACMN
C
      END
C$PRINTOFF
C$XREFOFF
C
C
```

```
C******************************************************************
      SUBROUTINE LOCBAC (A,B,RELTOL,ABSTOL,NFMAXL,KFLAGL,
     *                  X,FX,NF,REALSV)
C
C  LOCBAC 1.1        NOVEMBER 1991
C
C  A.N.S.I. STANDARD FORTRAN 77
C
C  LOCBAC FINDS AN APPROXIMATION X TO THE POINT IN THE
C  INTERVAL (A,B) AT WHICH A GIVEN FUNCTION F(X) ATTAINS
C  A LOCAL MINIMUM.
C  LOCBAC USES REVERSE COMMUNICATION TO GET VALUES OF F.
C
C  J. P. CHANDLER, COMPUTER SCIENCE DEPARTMENT,
C     OKLAHOMA STATE UNIVERSITY,
C     STILLWATER, OKLAHOMA 74078
C
C  SUBROUTINE LOCBAC IS ADAPTED FROM FUNCTION LOCALM IN
C     "ALGORITHMS FOR MINIMIZATION WITHOUT DERIVATIVES"
C     BY RICHARD P. BRENT (PRENTICE-HALL 1973)
C
C  THE METHOD USED IN LOCBAC IS A COMBINATION OF GOLDEN SECTION
C  SEARCH AND SUCCESSIVE PARABOLIC INTERPOLATION.
C  IF F(X) HAS A CONTINUOUS SECOND DERIVATIVE WHICH IS POSITIVE
C  AT THE MINIMUM, AND THE MINIMUM IS IN  A .LT. X .LT. B,
C  THEN, IGNORING ROUNDING ERRORS, CONVERGENCE IS SUPERLINEAR,
C  AND USUALLY THE ORDER OF CONVERGENCE IS AT LEAST AS LARGE AS
C  1.324717957245... (THE REAL ROOT OF X**3=X+1).
C  CONVERGENCE TO A RELATIVE ACCURACY OF 1.0D-7 USUALLY
C  REQUIRES ABOUT EIGHT TO TWELVE ITERATIONS.
C
C********************************
C
C  INPUT QUANTITIES....  A,B,RELTOL,ABSTOL,NFMAXL,KFLAGL,(FX)
C
C  OUTPUT QUANTITIES...  A,B,X,FX,KFLAGL,NF
C
C  SAVE AREA...........  REALSV(*)
C
C
C     A      -- THE LEFT ENDPOINT OF THE INTERVAL STILL
C                TO BE SEARCHED
C
C     B      -- THE RIGHT ENDPOINT OF THE INTERVAL STILL
C                TO BE SEARCHED
C
C     RELTOL --  A RELATIVE CONVERGENCE TOLERANCE FOR X
C
C     ABSTOL --  AN ABSOLUTE CONVERGENCE TOLERANCE FOR X
C
C     NFMAXL --  AN UPPER LIMIT ON THE NUMBER OF TIMES
C                THE FUNCTION F(X) IS TO BE EVALUATED
C
C     X      -- ON EXIT WITH KFLAGL .GE. 1,
C                X CONTAINS A VALUE FOR WHICH SUBROUTINE
C                LOCBAC NEEDS THE VALUE OF FX=F(X).
C                THE USER SHOULD COMPUTE FX AND
C                CALL LOCBAC AGAIN, SENDING IT THAT VALUE.
C
C              ON EXIT WITH KFLAGL .LE. 0,
C                X CONTAINS THE BEST ESTIMATE AVAILABLE OF
C                THE POSITION OF THE LOCAL MINIMUM OF F(X).
C
C     FX     -- ON THE INITIAL ENTRY WITH KFLAGL=0,
C                AND ON EXIT WITH KFLAGL .GE. 1,
C                FX DOES NOT CONTAIN ANY USEFUL INFORMATION.
C
```

```
C          ON ENTRY WITH KFLAGL .GE. 1,
C            THE USER MUST FURNISH THE VALUE OF FX=F(X)
C            FOR THE GIVEN VALUE OF X.
C
C          ON EXIT WITH KFLAGL .LE. 0,
C            FX RETURNS THE VALUE OF F(X).
C
C   KFLAGL -- A SWITCH VARIABLE.
C            THE USER MUST SET KFLAGL=0 BEFORE THE FIRST
C            CALL TO SUBROUTINE LOCBAC, FOR EACH PROBLEM
C            TO BE SOLVED.
C
C          THE VALUE OF KFLAGL ON RETURNING FROM
C            SUBROUTINE LOCBAC HAS THE FOLLOWING
C            MEANING...
C
C          =1 WHEN SUBROUTINE LOCBAC REQUIRES THE CALLING
C            MODULE TO COMPUTE A VALUE OF FX=F(X) FOR
C            THE GIVEN INITIAL VALUE OF X,
C
C          =2 WHEN LOCBAC WANTS A VALUE OF FX AT THE
C            GIVEN VALUE OF X FOR A PARABOLIC
C            INTERPOLATION STEP,
C
C          =3 WHEN LOCBAC WANTS A VALUE OF FX AT THE
C            GIVEN VALUE OF X FOR A GOLDEN SECTION STEP,
C
C          =0 IF NORMAL CONVERGENCE OCCURRED ACCORDING TO
C            THE RELTOL AND ABSTOL CRITERIA,
C
C          =-1 IF CONVERGENCE ACCORDING TO THE RELTOL AND
C            ABSTOL CRITERIA DID NOT OCCUR, BUT THE
C            PROCESS DID CONVERGE TO FULL MACHINE
C            PRECISION
C            (THIS CANNOT BE COUNTED UPON TO HAPPEN IN
C            EVERY CASE, AND INDICATES THAT RELTOL
C            AND/OR ABSTOL ARE TOO SMALL),
C
C          =-2 IF CONVERGENCE FAILED BECAUSE MORE THAN
C            NFMAXL FUNCTION EVALUATIONS WERE REQUIRED
C
C   NF   -- THE NUMBER OF FUNCTION EVALUATIONS THAT HAVE
C            BEEN COMPLETED SO FAR
C            (WHEN FX IS COMPUTED, THE VALUE OF NF
C            IS INCREMENTED AT THE NEXT ENTRY
C            INTO SUBROUTINE LOCBAC)
C
C   REALSV(*) -- A "SAVE" ARRAY OF EIGHT DOUBLE PRECISION
C              (OR REAL) ELEMENTS
C
C
C USAGE...
C
C THE USER MUST DIMENSION THE ARRAY REALSV(*).
C
C FOR EACH PROBLEM TO BE SOLVED, THE USER MUST
C   1) INITIALIZE A AND B TO THE ENDPOINTS OF THE INTERVAL
C        TO BE SEARCHED,
C   2) SET RELTOL AND ABSTOL (SEE BELOW),
C   3) SET NFMAXL,
C   4) INITIALIZE KFLAGL=0, AND
C   5) PROCEED TO CALL SUBROUTINE LOCBAC IN A LOOP AS LONG
C        AS KFLAGL .GE. 1 .
C
C WHENEVER SUBROUTINE LOCBAC RETURNS KFLAGL .GE. 1, THE USER
C COMPUTES THE VALUE OF FX=F(X) FOR THE VALUE OF X RETURNED BY
C LOCBAC, AND CALLS SUBROUTINE LOCBAC AGAIN.
```

```
C
C WHEN LOCBAC RETURNS KFLAGL .LE. 0, IT HAS FINISHED THE
C SEARCH.
C
C BETWEEN CALLS TO SUBROUTINE LOCBAC FOR A GIVEN PROBLEM,
C THE USER MUST NOT (REPEAT, **** MUST NOT ****) ALTER
C THE VALUES OF A, B, X, KFLAGL, NF, OR REALSV(*).
C TO ALTER ANY OF THESE QUANTITIES IS TO INVITE DISASTER.
C
C
C RELTOL AND ABSTOL DEFINE A TOLERANCE
C   TOL = RELTOL * ABS(X) + ABSTOL ,
C AND F(X) IS NEVER EVALUATED AT TWO POINTS CLOSER TOGETHER
C THAN TOL.
C
C IF F(X) IS DELTA-UNIMODAL (SEE BRENT) FOR SOME
C DELTA .LT. TOL, THEN X APPROXIMATES THE GLOBAL MINIMUM OF
C F(X) WITH AN ERROR THAT IS LESS THAN 3*TOL.  IF F(X) IS NOT
C DELTA-UNIMODAL ON (A,B), THEN X MAY APPROXIMATE A LOCAL BUT
C NONGLOBAL MINIMUM.
C
C IF F(X) IS SMOOTH AND THERE IS NO REASON TO BELIEVE THAT THE
C MINIMUM VALUE OF F(X) IS EXACTLY ZERO, SET RELTOL EQUAL TO
C SQRT(MACHINE EPSILON) OR LARGER.
C
C IF F(X) IS NOT SMOOTH OR THE MINIMUM VALUE OF F(X) IS
C BELIEVED TO BE ZERO, AND VERY HIGH ACCURACY IS DESIRED IN X,
C RELTOL MAY BE SET AS SMALL AS 2*(MACHINE EPSILON) BUT NOT
C SMALLER.
C
C IF THE INITIAL INTERVAL A .LE. X .LE. B INCLUDES THE ORIGIN,
C SET ABSTOL TO A SMALL POSITIVE VALUE.
C OTHERWISE, ABSTOL CAN BE SET EQUAL TO ZERO.
C
C IF A MINIMUM HAS ALREADY BEEN BRACKETTED, SUBROUTINE LOCBAC
C CAN BE CALLED IN A WAY THAT SAVES THREE FUNCTION
C EVALUATIONS.  SUPPOSE THAT THREE POINTS (P,FP), (Q,FQ), AND
C (R,FR) ARE KNOWN, SUCH THAT P .LT. Q .LT. R, FQ .LT. FP,
C FQ .LT. FR, AND THE RATIO (Q-P)/(R-P) IS NOT EXTREMELY CLOSE
C TO ZERO OR TO UNITY.
C THEN BEFORE ENTERING THE LOOP IN WHICH SUBROUTINE LOCBAC
C IS CALLED, EXECUTE ALSO THE FOLLOWING STATEMENTS...
C   A=P
C   B=R
C   X=Q
C   FX=FQ
C   KFLAGL=3
C   NF=2
C   IF(FP.LT.FR) THEN
C     REALSV(1)=P
C     REALSV(2)=FP
C     REALSV(3)=R
C     REALSV(4)=FR
C   ELSE
C     REALSV(1)=R
C     REALSV(2)=FR
C     REALSV(3)=P
C     REALSV(4)=FP
C   ENDIF
C   REALSV(5)=REALSV(3)
C   REALSV(6)=REALSV(4)
C   REALSV(8)=R-P
C   REALSV(7)=REALSV(8)
C (FOLLOWING A GOLDEN SECTION SEARCH, SET
C   REALSV(7)=REALSV(8)*0.381966011D0  INSTEAD.)
C
C*********************************
```

```
C
      DOUBLE PRECISION A,B,RELTOL,ABSTOL,X,FX,REALSV(8),
     *  ZABS,ARG,DABS,
     *  CGS,RZERO,RTWO,W,V,E,FW,FV,XMID,TOL,TWOTOL,
     *  R,Q,P,D,U,FU
C
      INTEGER NFMAXL,KFLAGL,NF
C
      COMMON /NP/NPROB
C
      ZABS(ARG)=DABS(ARG)
C
C  CGS IS A CONSTANT USED IN THE GOLDEN SECTION SEARCH
C  METHOD...
C
C     CGS = (3-SQRT(5))/2 = 2 - PHI = 1/(PHI + 1)
C
C  WHERE PHI IS THE GOLDEN RATIO.
C  THE VALUE OF THIS CONSTANT NEED NOT BE PRECISE.
C
      CGS=0.38196601D0
C
      RZERO=0.0D0
      RTWO=2.0D0
C
      IF(KFLAGL.GE.1) GO TO 10
C
C  KFLAGL .EQ. 0 .  COMPUTE THE FIRST VALUE OF X.
C
      X=A+CGS*(B-A)
C
C  RETURN TO THE CALLING MODULE TO GET THE FIRST VALUE
C  FOR FX=F(X) .
C
      KFLAGL=1
C
C  GIVE VALUES (FX, FW, AND FV ARE ARBITRARY) TO ALL OUTPUT
C  AND SAVED QUANTITIES.
C
      NF=0
C
      U=X
      W=X
      V=X
      FX=RZERO
      FW=RZERO
      FV=RZERO
      D=RZERO
      E=RZERO
C
      GO TO 110
C
C  KFLAGL .GE. 1 .
C
   10 NF=NF+1
C
C  UNPACK SAVED QUANTITIES.
C
      W=REALSV(3)
      V=REALSV(5)
      D=REALSV(7)
      E=REALSV(8)
C
      IF(KFLAGL.EQ.1) THEN
        FW=FX
        FV=FX
        GO TO 50
      ENDIF
C
C  KFLAGL .GE. 2 .  FU=F(U) HAS JUST BEEN COMPUTED IN FX.
C
      U=X
      FU=FX
C
      X=REALSV(1)
      FX=REALSV(2)
      FW=REALSV(4)
      FV=REALSV(6)
C
C  UPDATE A, B, V, W, AND X.
C
      IF(FU.GT.FX) GO TO 20
C
      IF(U.GE.X) THEN
        A=X
      ELSE
        B=X
      ENDIF
C                         STATEMENT NUMBER 140 ...
C                           (IN BRENT'S FUNCTION LOCALM)
      V=W
      FV=FW
      W=X
      FW=FX
      X=U
      FX=FU
C
      GO TO 40
C                         STATEMENT NUMBER 150 ...
   20 IF(U.LT.X) THEN
        A=U
      ELSE
        B=U
      ENDIF
C                         STATEMENT NUMBER 170 ...
C
      IF(FU.GT.FW .AND. W.NE.X) GO TO 30
C
      V=W
      FV=FW
      W=U
      FW=FU
C
      GO TO 40
C                         STATEMENT NUMBER 180 ...
C
   30 IF(FU.GT.FV .AND. V.NE.X .AND. V.NE.W) GO TO 40
C
      V=U
      FV=FU
C
   40 IF(NF.GT.NFMAXL) THEN
        KFLAGL=-2
        RETURN
      ENDIF
C
C  BEGIN THE MAIN LOOP.
C  AT THIS POINT,
C    1)  IF F(X) IS DELTA-UNIMODAL ON THE INTERVAL
C          A .LE. X .LE. B, THEN F(X) HAS A LOCAL MINIMUM
C          IN THIS INTERVAL;
C    2)  OF ALL THE POINTS AT WHICH F(X) HAS BEEN EVALUATED,
C          X IS THE ONE WITH THE LEAST VALUE OF F(X),
C          AND X IS THE POINT OF THE MOST RECENT
```

```
C        EVALUATION IF THERE IS A TIE;
C    3)  W IS THE POINT WITH THE NEXT LOWEST VALUE OF F(X);
C    4)  V IS THE PREVIOUS VALUE OF W; AND
C    5)  U IS THE LAST POINT AT WHICH F(X) HAS BEEN EVALUATED.
C
C                      STATEMENT NUMBER 10 ...
   50 XMID=A+(B-A)/RTWO
      TOL=RELTOL*ZABS(X)+ABSTOL
      TWOTOL=TOL+TOL
C
C  CHECK THE STOPPING CRITERIA.
C
      KFLAGL=0
      IF(ZABS(X-XMID).LE.TWOTOL-(B-A)/RTWO) RETURN
C
C  ADD A TEST FOR CONVERGENCE TO FULL MACHINE PRECISION,
C  IN CASE THE USER HAS SET RELTOL AND ABSTOL TOO SMALL.
C
      IF(XMID.LE.A .OR. XMID.GE.B) THEN
        KFLAGL=-1
        RETURN
      ENDIF
C
      R=RZERO
      Q=RZERO
      P=RZERO
C
      IF(ZABS(E).LE.TOL) GO TO 60
C
C  FIT A PARABOLA.
C  INTERPOLATE A QUADRATIC POLYNOMIAL
C  THROUGH THE THREE POINTS (V,FV), (W,FW), (X,FX).
C
      R=(X-W)*(FX-FV)
      Q=(X-V)*(FX-FW)
      P=(X-V)*Q-(X-W)*R
      Q=RTWO*(Q-R)
      IF(Q.GT.RZERO) THEN
        P=-P
      ELSE
        Q=-Q
      ENDIF
C                      STATEMENT NUMBER 30 ...
      R=E
      E=D
C                      STATEMENT NUMBER 40 ...
C
   60 IF(ZABS(P).GE.ZABS(Q*R/RTWO)) GO TO 70
      IF(P.LE.Q*(A-X) .OR. P.GE.Q*(B-X)) GO TO 70
C
C  TAKE A "PARABOLIC INTERPOLATION" STEP.
C  STEP TO THE MINIMUM OF THE INTERPOLATED QUADRATIC
C  POLYNOMIAL.
C
      KFLAGL=2
C
      D=P/Q
      U=X+D
C
C  F(X) MUST NOT BE EVALUATED TOO CLOSE TO A OR B.
C
      IF(U-A.GE.TWOTOL .AND. B-U.GE.TWOTOL) GO TO 80
C
      IF(X.LT.XMID) THEN
        D=TOL
      ELSE
        D=-TOL
```

```
    ENDIF
C
    GO TO 80
C
C  TAKE A "GOLDEN SECTION" STEP.
C  COMPUTE A POINT U THAT DIVIDES THE LONGER OF THE INTERVALS
C  (A,X) AND (X,B) IN THE RATIO OF ONE TO (1+SQRT(5))/2,
C  WITH THE SHORTER OF THESE TWO SUBINTERVALS ADJACENT TO X.
C
  70 KFLAGL=3
C                    STATEMENT NUMBER 60 ...
    IF(X.GE.XMID) THEN
      E=A-X
    ELSE
      E=B-X
    ENDIF
C                    STATEMENT NUMBER 80 ...
    D=CGS*E
C
C  F(X) MUST NOT BE EVALUATED TOO CLOSE TO X.
C
C                    STATEMENT NUMBER 90 ...
C
  80 IF(ZABS(D).LE.TOL) GO TO 90
    U=X+D
    GO TO 100
C                    STATEMENT NUMBER 100 ...
  90 IF(D.GT.RZERO) THEN
      U=X+TOL
    ELSE
      U=X-TOL
    ENDIF
C
C  ADD A TEST FOR CONVERGENCE TO FULL MACHINE PRECISION.
C
 100 IF(U.LE.A .OR. U.GE.B .OR. U.EQ.X) THEN
      KFLAGL=-1
      RETURN
    ENDIF
C
C  RETURN TO THE CALLING MODULE TO GET A VALUE FOR FX=F(X) OR
C  (AT   STATEMENT NUMBER 120 IN BRENT'S FUNCTION LOCALM)
C  FOR FU=F(U).
C  IN EITHER CASE, THE FUNCTION VALUE WILL BE SENT IN AS THE
C  VALUE OF THE PARAMETER FX.
C
C  SAVE INTERMEDIATE QUANTITIES AND RETURN.
C
 110 REALSV(1)=X
    X=U
    REALSV(2)=FX
    REALSV(3)=W
    REALSV(4)=FW
    REALSV(5)=V
    REALSV(6)=FV
    REALSV(7)=D
    REALSV(8)=E
C
    RETURN
C
C  END LOCBAC
C
    END
```

VITA

Meishan Cheng

Candidate for the Degree of

Master of Science

Thesis: A SURVEY AND COMPARISON OF CONJUGATE GRADIENT METHODS FOR OPTIMIZATION

Major Field: Computer Science

Biographical:

Personal Data: Born in Maoming, Guangdong, People's Republic of China, August 3, 1964, the son of Jingguang Cheng and Guogrong Li.

Education: Received the Bachelor of Science degree in Mathematics from Zhongshan (Yet-San) University, Guangzhou, Guangdong, People's Republic of China, in July, 1985; completed course requirements for the Master of Science in Management Engineering at South-China University of Technology in July, 1990; completed requirements for the Master of Science degree at Oklahoma State University in December, 1993.

Professional Experience: Software Engineer, Shenzhen Splendid China Development Co., Ltd, Shenzhen, Guangdong, People's Republic of China, September, 1985, to August, 1990.