A Brief Discussion of Nonlinear Constrained Optimization
and
A Proposal for a New Direct Search Method

Judy Shau Han Tam

A Thesis

in

The Department

of

Mathematics

Presented in Partial Fulfillment of the Requirements
for the Degree of Master of Science at
Concordia University
Montréal, Québec, Canada

March 1987

ISBN 0-315-35524-7

ABSTRACT

A Brief Discussion of Nonlinear Constrained Optimization
and
A Proposal for a New Direct Search Method

Judy Shau Han Tam

The concept of nonlinear constrained optimization is briefly discussed, and two existing Direct Search strategies, Rosenbrock's method (1960) and Box's Complex method (1965) are described. Both algorithms require a relatively long computation time, and a lot of storage space. Moreover, none of the methods offers a systematic search for a global optimum. In response, a new nonrandom Complex method (the Revised Complex method) has been developed for application to optimization problems characterized by nonlinear objective and constraint functions involving continuous optimization variables. Using a line-search technique to construct the initial complex, and a procedure Factor to find suitable values for the over-reflection factor R, the new method is believed to be more efficient than Box's strategy. Finally, the performance of the Revised Complex method with constrained problems, is compared with those of the original Complex method. These results suggest that the new method offers a good alternative for nonlinear optimization, especially when the problem consists of several local optima.

ACKNOWLEDGEMENT

Special thanks is due to my supervisor, Dr. S.T. Ali, for his guidance, patience and support during the period of preparation of this thesis.

I am also grateful to the Concordia University Mathematics Department for providing financial assistance during the completion of this work.

TABLE OF CONTENTS

(I) INTRODUCTION

1.1 Optimization

The concept of optimization is becoming pervasive in the modern world, appearing in engineering, economics, mathematics, physics and even the social sciences. It represents the basic principle underlying the analysis of many complex decision or allocation problems. Using this principle, one approaches a complex decision problem, involving the selection of values for a number of interrelated variables, by focusing attention on a single objective function which measures the quality of the decision. This objective is minimized (or maximized depending on the formulation) subject to various constraints which may limit the selection of decision variable values. It is possible to avoid having to optimize when there is only one way to carry out a task, because one then has no other alternative. However, when there exist two or more solutions, one should always choose the best way out. Unfortunately, one can never be sure whether his or her selection of a solution provides the optimal result if no optimization technique is employed. In response to this problem, numerous optimization models and techniques began to develop, and the growth of large and fast computing facilities has aided substantially in the advancement of this area.

1.2 Scope of this Work

With the lack of a universal method of optimization, various procedures, each having only a limited application to special cases, are made available. We will not attempt to list them all here. However it is understood that solutions for different optimization models ask for certain particular techniques; for instance, the calculus of variations and its extensions provide the mathematical basis of functional optimization. Problems of this type cannot be solved by the methods for minimizing (or maximizing) a function learned in introductory calculus, because instead of finding the values of one or several variables which yield the minimum (or maximum) of a function, one is required to find one or more complete functions which minimize (or maximize) an integral, whose value depends upon these functions. We shall not discuss further this type of problems which are concerned with finding optimal trajectories. The problem with which this paper is concerned is that of finding the minimum (or maximum) point of a deterministic mathematical function, where there exist restrictions or constraints as to what the permissible values of the independent variables are.

In fact, our subject matter is even more limited than has been indicated so far. Problems involving linear functions subject to linear constraints will not be discussed here. They give rise to what are termed linear programming problems, and linear programming has been deeply studied in operations research. Details of the techniques usually employed in this special situation are widely covered in almost all mathematical programming texts. Here, we are only concerned with the general nonlinear functions subject to one or more constraints. These constraints are all inequality constraints, which are assumed to apply to functions of the independent variables as well as to the independent variables themselves. Moreover, all the independent variables are assumed to be continuous variables, as opposed to variables which can only take on discrete values. Finally, functions of one dimension will not be discussed here. Numerous effective search schemes for this type of problems have been developed during the past few decades.

To conclude this section, we state that there actually exists a tremendous amount of optimization problems which are of different nature, and a universal optimizer does not exist, at least up to this point. In other words, there will always be problems for which a particular optimization method performs unexpectedly poorly. If there were indeed an "optimal" optimization strategy, all the others would long have been forgotten, and since all the methods presently known can only be used in particular areas of application, it is necessary for us first to identify what kind of optimization problems we are going to examine.

1.3 Statement of the Problem

First of all, let $\underline{x}$ be an n-dimensional vector of unknowns,

$$\underline{x} = (x_1, x_2, \ldots, x_n),$$

and F, G_j (j = 1, 2, ..., m) be some real valued functions of the variables $x_1, x_2, \ldots, x_n$. We note that

$$\underset{\text{with respect to } \underline{x}}{\text{Minimum}} \left\{ -F(\underline{x}) : \underline{x} \in X \right\} = \underset{\text{with respect to } \underline{x}}{\text{Maximum}} \left\{ F(\underline{x}) : \underline{x} \in X \right\}$$

where X is some subspace of the n-dimensional euclidian space $\mathbb{R}^n$ defined by a set of constraints, and we shall confine our attention throughout this paper to minimization. Thus the complete constrained optimization or nonlinear programming problem can be written as the following:

Minimize the objective function $F(\underline{x})$

subject to m inequality constraints $G_j(\underline{x}) \geqslant 0$,

for $j = 1, 2, \ldots, m$.

All other constraints can be reduced to this form.

1.4 Size of Problems

Those points which satisfy all the constraints are said to be feasible, and in their entirety these points constitute the feasible region. All other points are non-feasible, and constitute the non-feasible region. It should be noted that a feasible region does not necessarily exist, in which case the problem has no solution. However this paper will only be concerned with problems which have solutions. The optimization problem formulated above is considered to be multidimensional. One obvious measure of the complexity of this kind of problem is its size, measured in terms of its dimension or sometimes, the number of constraints. As might be expected, with advancing computing technology and with improved theorectical methods, the size of problems that can be effectively solved has been increasing. With present day computing capabilities, it is reasonable to distinguish between three classes of problems: small-scale problems having about five or fewer unknowns and constraints; intermediate-scale problems having from about five to a hundred variables, and large-scale problems having more than a hundred and perhaps thousands of variables and constraints. This classification is not entirely rigid, but it reflects the basic differences in approach that accompany problems of different size.

(II) MULTIDIMENSIONAL STRATEGIES - DIRECT SEARCH METHODS

2.1 Introduction

There have been frequent attempts to extend the basic ideas of one dimensional optimization procedures to several dimensions, and various methods are distinguished according to the kind of information they need, namely:

(i) Direct Search methods - which only need the objective function values $F(\underline{x})$,

(ii) Gradient methods - which also use the first partial derivatives, $\nabla F(\underline{x})$ (First Order Strategies),

(iii) Newton methods - which in addition make use of the second partial derivatives, $\nabla^2 F(\underline{x})$ (Second Order Strategies).

All the above numerical optimization techniques have certain features in common. Such techniques, which are referred to as iterative techniques, require an initial point $\underline{x}(0)$ to be specified, and proceed by generating a sequence of points $\underline{x}(k)$, $k = 1, 2, \ldots$, which represent improved approximations to the solution, that is

$$F(\underline{x}(k+1)) \leqslant F(\underline{x}(k)).$$

We will note here that it is usual to regard equal function values as improvements.

Iterative techniques can conveniently be studied with the aid of the recursion schemes employed by most multidimensional strategies. The formula is as follows:

$$\underline{x}(k+1) = \underline{x}(k)+s(k)\cdot\underline{d}(k),$$

where $\underline{d}(k)$ is an n-dimensional direction vector, and $s(k)$ is the step length moved along $\underline{d}(k)$. The determination of a suitable direction $\underline{d}(k)$ may involve several computations of the function F. Once $\underline{d}(k)$ has been chosen, F can be computed at one or more points along this direction, and from these results a suitable value for $s(k)$ can be found. Every strategy differs from the others with regard to the step size and search direction. Typically, the algorithm is repeated and a sequence of ever-improving points is generated that approaches a solution point $\underline{x}^*$. For nonlinear programming problems, the sequence generally does not ever exactly reach the solution point, but converges towards it. In operation, for nonlinear problems, the process is terminated whenever a point sufficiently close to the solution point, for practical purposes, is obtained. Among the three different iterative methods, Direct Search methods will be the only focus of this paper.

Direct Search strategies are those which do not require the explicit evaluation of any partial derivatives of the function, but instead rely solely on values of the objective function F, plus information gained from earlier iterations. Some of these methods in effect use the objective function values to obtain numerical approximations to the derivatives of the objective function, or to fit low order polynomials or surfaces through selected points. In most cases, the search directions and even the step sizes are fixed by a scheme of some sort, rather than in an optimal way. Thus there is always a risk of not being able to improve the objective function value at each step, and hence failures must accordingly be planned for such that something can be learnt from them. This "trial" character of search strategies has earned them the name of trial-and-error methods. Their attraction lies not so much in theorectical proofs of convergence or the rates of convergence, as in their simplicity and the fact that they have proved themselves useful in practice. Among the Direct Search methods, there are two which are the most important and effective strategies for constrained problems. There are the Rosenbrock's method (8) and the Complex strategy of Box (2, 3). Since the emphasis here will be placed on the latter strategy, we will therefore only look at the Rosenbrock's method briefly.

2.2 Rosenbrock's Method

Rosenbrock's idea was to remove the limitation on the number of search directions in the coordinate strategy so that the search steps can move parallel to the axes of a coordinate system which can in turn rotate in the space $\mathbb{R}^n$. One of the axes is set to point in the direction which appears most favourable. The remaining directions are fixed orthogonally to the first and are also mutually orthogonal. We will denote the n mutually orthonormal direction vectors at the k^{th} iteration by $\underline{d}(k,1)$, $\underline{d}(k,2)$, ..., $\underline{d}(k,n)$, and by s(1), s(2), ..., s(n) the respective step lengths associated with each of these directions. For the first iteration, the coordinate directions are usually used as the mutually orthonormal direction vectors.

Starting from a given initial point $\underline{x}(0)$ which satisfies all the constraints, a trial is made in each direction with the associated discrete initial step size. A success is recorded if the resulting function value is no greater than the current best value. This trial point will replace the current point, and s(i) will be multiplied by a positive factor $\alpha > 1$. Then the next search direction is carried out. If the trial brings an increase in the function value, it is considered to be a failure. In that case, the current point remains unchanged, and s(i) will be multiplied by a negative factor $-1 < \beta < 0$. (Rosenbrock found that the values $\alpha = 3$ and $\beta = -0.5$ to be suitable choices for the operational coefficients.) This process is repeated until a success followed by a failure has occurred along every direction. Following this first part of the search, the coordinate axes are rotated.

Now, new direction vectors are to be computed. The recursion formulae are as follows:

Let $\Delta(k,j)$ represent the distance covered in the direction $\underline{d}(k,j)$ in the k^{th} iteration, then

$$\underline{a}(i) = \sum_{j=i}^{n} \Delta(k,j) \cdot \underline{d}(k,j) \qquad \text{for } i = 1, 2, \ldots, n$$

so that $\underline{a}(1)$ represents the total progress made during the k^{th} iteration, and $\underline{a}(2)$ represents the progress made during the iteration excluding that made in direction $\underline{d}(k,1)$, and so on. Then the new set of mutually orthogonal unit vectors will be computed according to the following scheme.

$$\underline{d}(k+1,i) = \frac{\underline{w}(i)}{\|\underline{w}(i)\|} \qquad \text{for } i = 1, 2, \ldots, n$$

where

$$\underline{w}(i) = \begin{cases} \underline{a}(i) & \text{for } i = 1 \\ \underline{a}(i) - \sum_{j=1}^{i-1} (\underline{a}(i)^T \cdot \underline{d}(k+1,j)) \cdot \underline{d}(k+1,j) & \\ & \text{for } i = 2, 3, \ldots, n. \end{cases}$$

This procedure does not incorporate a convergence criterion. Rosenbrock's suggestions are that either a run be terminated after a specified number of function evaluations, or whenever the magnitude of $\underline{a}(1)$ is less than a specified value, and $\|\underline{a}(2)\| > 0.3\|\underline{a}(1)\|$ for each of several consecutive iterations. This condition ensures that the direction of total progress is changing - something that Rosenbrock regarded as a sure sign of the proximity of a minimum.

In his original publication in 1960, Rosenbrock had already given detailed rules as to how inequality constraints can be treated. His procedure for doing this can be viewed as a partial penalty function method, since the objective function is only altered in the neighbourhood of the boundaries. Immediately after each variation of the variables, the objective function value is tested. The following is one of the several suggestions of Rosenbrock. For constraints of the form $G_j \geq 0$ ($j = 1, 2, \ldots, m$), one constructs the extended objective function $F'(\underline{x})$ in the form:

$$F'(\underline{x}) = F(\underline{x}) + \sum_{j=1}^{m} \delta_j(\underline{x}) \cdot (f_j - F(\underline{x}))$$

in which

$$\delta_j(\underline{x}) = \begin{cases} 0 & \text{if } G_j(\underline{x}) \geq e \\ 3q - 4q^2 + 2q^3 & \text{if } 0 < G_j(\underline{x}) < e \\ 1 & \text{if } G_j(\underline{x}) \leq 0 \end{cases}$$

and

$$q = 1 - \frac{1}{e} \cdot G_j(\underline{x}).$$

f_j is the value of the objective function belonging to the last success of the search which did not fall in the region of the j^{th} boundary. As a reasonable value for the boundary zone, one can take $e = 10^{-4}$.

The entire algorithm of this method can be easily found in most nonlinear programming texts, and Rosenbrock's strategy has been widely used to good effect. In fact, numerical experiments by Schwefel (9) show that within a few iterations, the rotating coordinates become oriented in such a way that one of the axes points along the gradient direction. The strategy is thus able to follow sharp valleys in the topology of the objective function. It has however one major disadvantage when compared to other Direct Search methods - the orthogonalisation procedure is very costly. Computation time is relatively long and the algorithm requires a lot of storage space.

2.3 Complex Strategy of Box

After revealing the Rosenbrock's method, we now turn to the Complex strategy of Box. Box modified the Simplex method for unconstrained minimization, proposed by Nelder and Mead (7) to find constrained minima, and termed his constrained Simplex method the "Complex" method. The two most important differences to the Nelder-Mead strategy are the use of more vertices and the expansion of the polyhedron at each normal reflection. Both measures are intended to prevent shrinkage of the complex.

In this method, $N \geq n+1$ points are used. It is assumed that an initial point

$$\underline{x}(0) = (x_1(0),\ x_2(0),\ \ldots,\ x_n(0))$$

which satisfies all the m constraints is available. It will be one of the N vertices of the intended polyhedron. The remaining vertex points are fixed by a random process in which each vector inside the closed region defined by the explicit constraints has an equal probability of selection. The complete scheme for the construction of the initial complex is as follows:

$$\underline{x}(0,1) = \underline{x}(0),$$

$$\underline{x}(0,v) = \sum_{i=1}^{n} z_i \cdot \underline{e}_i \qquad \text{for } v = 2, 3, \ldots, N$$

where the z_i is an evenly distributed random number from the range (a_i, b_i) if constraints are given in the form $a_i \leqslant x_i \leqslant b_i$. Otherwise, we will take the range to be $(x_i(0)-0.5s, x_i(0)+0.5s)$, where s, for example equals one. A point so selected will necessarily satisfy the explicit constraints, but may not satisfy all the implicit constraints. If an implicit constraint is violated, the trial point is moved halfway towards the centroid of those points already selected (where the given initial point is always included). Ultimately a satisfactory point will be found (it is assumed that the feasible region is convex), and in this way, all the N points or vertices of the initial configuration can be constructed.

The function is then evaluated at each of these points, and the vertex of greatest function value is determined. This worst point will be replaced by a point R$\geqslant$1 times as far from the centroid of the remaining points as the reflection of the worst point in the centroid, the new point being collinear with the rejected point and the centroid of the retained vertices. If this trial point is also the worst, it is moved halfway towards the centroid of the remaining points to give a new trial point. The above procedure is repeated until some constraint is violated. If a trial vertex violates some of the constraints, then again a further trial point is constructed by a move halfway back towards the centroid of the remaining points. Ultimately a permissible point is found. (These moves could prove unsuccessful if the region were not convex.)

The author would like to mention here that in the original

Complex method, Box suggested if the trial vertex does not satisfy some constraint on some independent variable x_i, $i = 1, 2, \ldots, n$, then that particular variable is re-set to a value 0.000001 inside the appropriate limit. However, as Guin (4) has later discussed that this rule sometimes causes the method to obtain a false optimum if all points of the complex fall into this hyperplane. This happens especially when the optimum is near, but not upon the constraint. To alleviate this situation, it is therefore recommended that the above rule should be abandoned and that only the rule for moving halfway toward the centroid should be retained to deal with constraint violation.

The only stopping criterion built into the program is a conservative one, namely that the program shall stop when several consecutive values of the objective function are the same to computational accuracy. This means that the program will not terminate when there is any chance of further improvement in the function, but avoids fruitless machine time when the complex has shrunk to such a size that changes in the function are smaller than one digit in the least significant place. The usual method for checking that the global rather than a local minimum has been found is to restart the program from different points, and infer that if these all lead to the same solution, then this is indeed the global minimum. For constrained optimization, if the feasible region of the parameter space is small, then it is not an easy matter to find alternative starting points which satisfy all the constraints, and which differ substantially from each other. With the Complex method, there is no difficulty in using the same initial point, but different pseudo-random number sequence initiators to perform such a rough check as to whether the optimum is global. The ease with which this can be done is considered to be an advantage of the method.

Furthermore, the use of over-reflection by a factor $R>1$ enables rapid progress to be made when the initial point is remote from the optimum. It also causes a continual enlargement of the complex, and thus is able to compensate for the moves halfway towards the centroid. In other words, it is an aid towards maintaining the full dimensionality of the complex. The use of $N \geq n+1$ points also serves this purpose. However, the uncertainty in the selection of suitable values for R and N may pose difficulties in the entire optimization process. According to Box, it was decided to take $R = 1.3$ as reflection factor and $N = 2n$ as the number of vertices, the choice apparently not being critical. In practice, a good choice of R can improve the rate of convergence (as it will be shown later on), and the use of 2n points appears to be too costly, especially when the dimension is high. Moreover, Mitchell and Kaplan (6) found that the initial configuration of the complex would influence the results obtained. It is therefore better to place the vertices in a deterministic way rather than making a random choice.

Even with the uncertainty of R and N, Box found that the Complex method is likely to find a lower optimum than

Rosenbrock's method if the permissible region contains several local minima, as Rosenbrock's method will tend to converge to that local minimum which is "nearest" to the initial point in some sense. Moreover, the initial few over-reflections may well throw the program from end to end of the feasible region, i.e. the first few trials can scan the entire permitted region. However, no systematic search for alternative optima is made by Box. In response to this question and the construction of the initial complex in a more deterministic way, an attempt to modify the original Complex method is made here. From now on, we will call this new nonrandom Complex method as the Revised Complex method.

2.4 Revised Complex Method

The algorithm of the Revised Complex method will follow basically the idea of Box. However, in selecting the initial configuration of the complex and the over-reflection factor, a more deterministic procedure will be employed respectively. Moreover, the stopping criterion will be changed. Instead of using the conservative approach suggested by Box, we will test whether the difference between the objective function values at the new vertex of the present complex and at the worst point from the previous iteration is less than a prescribed limit. Hence, the Revised Complex method will require two input data: an initial feasible point

$$\underline{x}(0) = (x_1(0), x_2(0), \ldots, x_n(0)),$$

and a prescribed accuracy parameter E. However, unlike Box's strategy, this new method will only use $n+1$ points to set up the polyhedron.

The use of fewer vertices for the complex appears to be an advantage over Box's method. As Box himself observed that for $n>5$, a number of vertices $N = 2n$ is unnecessarily too high and requires more storage space than the Simplex method. (The Simplex method uses $n+1$ points to form the polyhedron.) This problem is also shared by Mitchell's nonrandom Complex method because Mitchell used $2n+1$ vertices to construct the complex. Thus, the Revised Complex method is able to reduce the number of objective function calls by using a smaller number of vertices, and the initial configuration of the complex is generated by the following procedure:

<u>STEP I</u>

(1) Denote the i^{th} point in the set of $n+1$ points by

$$\underline{x}(i) = (x_1(i), x_2(i), \ldots, x_n(i)).$$

Let $\underline{x}(1) = \underline{x}(0)$.

(2) Denote the explicit constraints on the independent variables by

$$a_i \leqslant x_i \leqslant b_i \qquad \text{for } i = 1, 2, \ldots, n.$$

Set $i = 1$.

(3) Let

$$\underline{x}(i+1) = (x_1(i), x_2(i), \ldots, x_{i-1}(i), x_i(i+1), x_{i+1}(i), \ldots, x_n(i)).$$

This point is the same as its previous point $\underline{x}(i)$ except in the i^{th} coordinate. $\underline{x}(i+1)$ is determined by the following algorithm $A : \mathbb{R}^{2n} \longrightarrow \mathbb{R}^{n}$.

$$A(\underline{x}(i), \underline{e}_i) = \{\underline{x}(i+1) : F(\underline{x}(i+1)) = \min_{a_i \leq s_i \leq b_i} F(\underline{x}'(i+1))$$

$$\text{where } \underline{x}'(i+1) = (x_1(i),\ x_2(i),\ \ldots,\ x_{i-1}(i),\ s_i,\ x_{i+1}(i),\ \ldots,\ x_n(i))\}.$$

(4) Evaluate all implicit constraint functions at $\underline{x}(i+1)$. If all are satisfied, move on to (5). If any implicit constraint is violated, then the new point is displaced stepwise towards the midpoint of the allowed vertices which have already been defined. In other words,

$$\underline{x}(i+1) \longleftarrow 0.5(\underline{x}(i+1) + \frac{1}{i}\sum_{j=1}^{i} \underline{x}(j)).$$

Repeat the process until $\underline{x}(i+1)$ is feasible.

(5) Reiterate (3) and (4) for $i = 2, 3, \ldots, n$. This gives a collection of n+1 points

$$\underline{x}(1),\ \underline{x}(2),\ \ldots,\ \underline{x}(n+1)$$

which is the initial complex.

The method proceeds to search for an optimum by repeatedly altering the complex, one point at a time, as in the original Complex strategy of Box. The details of the algorithm are as follows.

STEP II

(1) Determine the index w (worst vertex) such that

$$F(\underline{x}(w)) = \text{Max } \{F(\underline{x}(k)) : k = 1, 2, \ldots, n+1\}.$$

(2) Construct

$$\underline{NEWX} = \underline{AVE} + R(\underline{AVE} - \underline{x}(w)) \quad \text{where} \quad \underline{AVE} = \frac{1}{n}\sum_{\substack{k=1\\k\neq w}}^{n+1} \underline{x}(k).$$

(3) Evaluate all constraints at $\underline{NEWX}$. If all are satisfied, then go to STEP III; otherwise, carry out the following contraction procedure:

$$\underline{NEWX} \longleftarrow 0.5(\underline{AVE} + \underline{NEWX}).$$

Repeat the process until $\underline{NEWX}$ is feasible.

STEP III

If $F(\underline{NEWX}) < F(\underline{x}(k))$ for at least one $1 \leqslant k \neq w \leqslant n+1$, then set

$$\underline{x}(k) \longleftarrow \begin{cases} \underline{x}(k) & \text{for all } 1 \leqslant k \neq w \leqslant n+1 \\ \underline{NEWX} & \text{for } k = w, \end{cases}$$

else go back to STEP II:(3) - apply the contraction procedure.

STEP IV

If $(F(\underline{x}(w)) - F(\underline{NEWX})) < E$, where E = accuracy parameter, then determine the index b (best vertex) such that

$$F(\underline{x}(b)) = \text{Min}\ \{F(\underline{x}(k)) : k = 1, 2, \ldots, n+1\}.$$

End the search with the result $\underline{x}(b)$ and $F(\underline{x}(b))$. Otherwise, go back to STEP II.

The main difference of this method to the strategy of Box is that instead of using a fixed over-reflection factor R as suggested by Box, we attempt to find a suitable value

for R at each iteration. The idea is as follows:
We will begin the procedure Factor with n+1 vertices of the complex being generated inside the feasible region

$$\underline{x}(k) = (x(k,1), x(k,2), \ldots, x(k,n)) \qquad \text{for } k = 1, 2, \ldots, n+1.$$

Let $\underline{x}(w) = (x(w,1), x(w,2), \ldots, x(w,n))$ denote the worst vertex, then

$$AVE(i) = \sum_{\substack{k=1 \\ k \neq w}}^{n+1} \frac{x(k,i)}{n} \qquad \text{for } i = 1, 2, \ldots, n,$$

and

$$NEWX(i) = AVE(i) + R(AVE(i) - x(w,i)) \qquad \text{for } i = 1, 2, \ldots, n.$$

If we let

$$D(i) = AVE(i) - x(w,i),$$

then we have

$$NEWX(i) = AVE(i) + R \cdot D(i) \qquad \text{for } i = 1, 2, \ldots, n.$$

With F being our objective function, we get

$$F(x_1, x_2, \ldots, x_n) = F(AVE(1)+R \cdot D(1), AVE(2)+R \cdot D(2), \ldots, AVE(n)+R \cdot D(n)).$$

Since AVE(i) and D(i) are known (i = 1, 2, ..., n), hence $F(x_1, x_2, \ldots, x_n)$ can be rewritten as a function f_1 with only one variable R. Thus the dimension of our objective function has changed from n to one.

In order to avoid contraction, we will therefore only consider the reflection factor R which is bigger than or equal to one. The procedure for Factor is as follows:

(1) Starting from R = 1.

Set Value(1) = $f_1(R)$ and k = 2.

(2) Let R ⟵ R+s where s is some fixed step size.

(3) Set Value(2) = $f_1(R)$.

If Value(2) < Value(1), then set Value(1) ⟵ Value(2) and reiterate (2) for k = 3, 4, ...

Otherwise, R* = R-s and R* will then be our over-reflection factor for the given complex.

Thus we have approximated the value for R such that $f_1(R^*)$ is a minimum. We do not know whether this is a global minimum, but it does not fall into the interest of our objective, and therefore we will not discuss it further.

Now with a suitable value of R being found, we are able to generate the new point

$$\underline{NEWX} = \underline{AVE} + R(\underline{AVE} - \underline{x}(w)).$$

We will then check whether NEWX satisfies all the constraints, as illustrated in STEP II:(3) of the algorithm. If not, we will apply one contraction procedure

$$\underline{NEWX} \longleftarrow 0.5(\underline{AVE} + \underline{NEWX}).$$

This process will be repeated until all constraints are satisfied.

If this new point gives a better objective function value than the second worst point, then we will replace the worst point by <u>NEWX</u> (STEP III). In this manner, a new set of n+1 vertices is found. Similar to the above,

$$F(AVE(1)+R \cdot D(1),\ AVE(2)+R \cdot D(2),\ \ldots,\ AVE(n)+R \cdot D(n))$$

can be replaced by a new one dimensional function $f_2(R)$. We will note here that $f_2(R)$ will not be the same as $f_1(R)$. This is due to the fact that new points are being used; thus AVE(i) and D(i) will be different from those we have obtained before (i = 1, 2, ..., n). We will then carry out the procedure Factor again to find an optimum value of R for f_2. In other words, whenever a new set of vertices is obtained, we will use Factor to find a suitable value for R, and we will proceed with the entire algorithm until our accuracy bound is satisfied.

With this built in function Factor, it seems that we are able to find suitable values for R. However, there is one important remark about Factor that we must mention here. It is highly necessary that we prescribe an upper limit for R. This is due to the fact that in some cases, the one dimension function $f_i(R)$ appears to be a strictly decreasing function. This means that the procedure Factor will keep on repeating the entire algorithm unless we have prescribed a maximum value for R. We will denote this value as R'. (That is to say, the number of iterations, as denoted by the letter k in the procedure Factor, must be fixed before hand.) R' will therefore serve as a control unit for Factor.

Suppose that after the execution of the entire algorithm, a list of R being used is sorted out, we will denote these values of R as (R_1, R_2, ..., R_A), where A denotes the number of total iterations. Now let us look at this list closely. If there is no R_i (i = 1, 2, ..., A) exceeding the value of R', then we know that the program cannot converge at a faster rate or attain a lower optimum, if there exist more than one minimum. However, if one or more R_is (i = 1, 2, ..., A) reach the maximum level R', then there is a chance that the convergence rate can be improved further and we will be able to obtain a "better" optimum.

Without loss of generality, we will assume that R_i ($1 \leq i \leq A$) attains the maximum level R', and let us increase R' to R". We will now run the program again and check whether the algorithm converges at a faster rate. It is understood here that in this case, the new value of R_i will be between R' and R". In other words, we have $R' \leqslant R_i \leqslant R''$. If the new value of R_i remains unchanged, then this implies that the convergence rate of this program cannot be improved further, and with the given initial point, we will not be able to attain a lower optimum (if it exists). However, if $R' < R_i < R''$, then three cases could arise:

Case I : Rate of convergence increases and the minimum point remains unchanged

- in this case, no further changes of the upper limit for the reflection factor is necessary.

Case II : Rate of convergence decreases and the minimum point remains unchanged

- if we recall the procedure Factor, then this simply means that even though the new R_i gives a better values for the objective function $f_i(R)$, the new point being generated lies far outside the feasible region and requires more contraction to be carried out so that all constraints can be satisfied finally. In this case, the extra contraction steps imply that the whole feasible region has been scanned (in a rough sense), and since the optimum being sought remains unchanged, it is probably the global minimum.

Case III : Rate of convergence decreases and a lower optimum is obtained / Rate of convergence increases and a lower optimum is obtained

- in both cases, the important thing is that we are able to get a better objective function value. This is due to the fact that the new value of R_i enables the program to reach another optimum which is farther from the initial point than the previous optimum.

In the above three different cases, since $R' < R_i < R''$, any further increase in the upper limit of the over-reflection factor will bring no changes. Now let us consider what might happen when the new R_i takes on the value R''. This will be the same as in the beginning when R_i takes on the initial upper limit R', and a similar procedure should be carried out. In other words, we will again raise the maximum value of the reflection factor. However, we should always cease increasing this upper limit for R whenever the entire program appears to converge at a much slower rate and generates the same minimum. With this built in function Factor, we are then able to find suitable values for the over-reflection factor and to check whether it is possible to obtain a better optimum if there exist more than one, with the same given initial point. A comparison with the original Complex method of Box will enable us to see whether the Revised Complex method is a better one, but at this stage, we need first to show the entire algorithm indeed converges.

(III) GLOBAL CONVERGENCE OF THE REVISED COMPLEX METHOD

With the four steps stated in 2.4, our complete algorithm $M : \mathbb{R}^n \longrightarrow \mathbb{R}^n$ is in fact

$$M = M_4 M_3 M_2 M_3 M_2 \ldots M_3 M_2 M_1$$

where M_i is the corresponding algorithm of the i^{th} step (i = 1, 2, 3, 4). We will assume here that the feasible region is convex and F is unimodal. Then the global convergence of M can be proved by the following theorem.

<u>Theorem</u>: Let X be a nonempty closed convex set in $\mathbb{R}^n$, and let the nonempty set $Y \subset X$ be the solution set. Let $F : \mathbb{R}^n \longrightarrow \mathbb{R}$ be a unimodal continuous function, and let $M : \mathbb{R}^n \longrightarrow \mathbb{R}^n$ be a point-to-set map stated as above, then with any given $\underline{x} \in X$, we have $\underline{y} \in Y$ for $\underline{y} \in M(\underline{x})$.

Several theorems from Bazaraa and Shetty (1), and Luenberger (5) will be used here to complete the proof of global convergence. However before we prove the above theorem, let us first examine each of the four steps separately.

<u>STEP I</u>

Starting with one given initial point inside the feasible region, this step represents the algorithm $M_1 : \mathbb{R}^n \longrightarrow \mathbb{R}^{n(n+1)}$ where

$$M_1(\underline{x}) = (\underline{x}, AC^1(\underline{x}), AC^2AC^1(\underline{x}), \ldots, AC^nAC^{n-1}\ldots AC^1(\underline{x})).$$

$\underline{x} = (x_1, x_2, \ldots, x_n)$ is the first point of the initial configuration, and $AC^iAC^{i-1}\ldots AC^1(\underline{x})$ is the respective $i+1^{th}$ point where C^i and A are two different mappings (i = 1, 2, ..., n) which have the following properties:

(1) $C^i : \mathbb{R}^n \longrightarrow \mathbb{R}^{2n}$

$C^i(\underline{x}) = (\underline{x}, \underline{e}_i)$; $\underline{x} \in \mathbb{R}^n$ and $\underline{e}_i$ is the i^{th} unit vector $(i = 1, 2, \ldots, n)$.

(2) $A : \mathbb{R}^{2n} \longrightarrow \mathbb{R}^n$

$$A(\underline{x}, \underline{e}_i) = \left\{ \underline{y} : F(\underline{y}) = \min_{a \leq s \leq b} F(\underline{y}') \text{ where } \underline{y}' = (x_1, x_2, \ldots, x_{i-1}, s, x_{i+1}, \ldots, x_n) \text{ for some real numbers a and b} \right\}.$$

F is our objective function, and A represents the line-search algorithm.

We have to prove that the algorithm A is closed on $C^i(\underline{x})$, then the composite map AC^i is closed on $\underline{x}$ which implies M_1 is continuous. The mapping $A : \mathbb{R}^{2n} \longrightarrow \mathbb{R}^n$ can be re-defined in the following manner:

$$A(\underline{x}, \underline{d}) = \left\{ \underline{y} : F(\underline{y}) = \min_{r} F(\underline{y}') \text{ where } \underline{y}' = \underline{x} + r \cdot \underline{d};\ r \in \mathbb{R} \text{ and } \underline{d} \text{ is a direction vector} \right\}.$$

Since our objective function F is unimodal, hence there must exist such a $\underline{y}$. However $\underline{y}$ may not be unique (in the case where F has a "flat bottom"). We will verify that A is closed.

<u>Lemma</u>: Let $F : \mathbb{R}^n \longrightarrow \mathbb{R}$ be a unimodal continuous function, and let $A : \mathbb{R}^{2n} \longrightarrow \mathbb{R}^n$ be a point-to-set map stated as above, then A is closed.

<u>Proof</u>: (of Lemma)

Suppose that $\{\underline{x}_k\}$ and $\{\underline{d}_k\}$ are sequences with $\underline{x}_k \longrightarrow \underline{x}$, $\underline{d}_k \longrightarrow \underline{d}$. Suppose also that $\underline{y}_k \in A(\underline{x}_k, \underline{d}_k)$, and that $\underline{y}_k \longrightarrow \underline{y}$. We must show that $\underline{y} \in A(\underline{x}, \underline{d})$.

For each k, we have $\underline{y}_k = \underline{x}_k + r_k \cdot \underline{d}_k$ for some $r_k \in \mathbb{R}$.

<u>Proof</u>: (of Lemma - continued)

From this, we can write

$$r_k = \frac{y_k(1)-x_k(1)}{d_k(1)} = \frac{y_k(2)-x_k(2)}{d_k(2)} = \cdots = \frac{y_k(n)-x_k(n)}{d_k(n)}$$

if $d_k(i) \neq 0$, where the number in the parenthesis means the i^{th} element of the vector $\underline{x}_k$, $\underline{y}_k$ and $\underline{d}_k$ ($i = 1, 2, \ldots, n$). Now recall that $\underline{d} = \underline{e}_i$ where $\underline{e}_i$ is the i^{th} unit vector ($i = 1, 2, \ldots, n$). This implies

$$\lim_{k\to\infty} d_k(j) = 0 \quad \text{for } j = 1, 2, \ldots, n;\ j \neq i,$$

and

$$\lim_{k\to\infty} d_k(i) = 1.$$

We will therefore only consider the following:

$$r_k = \frac{y_k(i)-x_k(i)}{d_k(i)}.$$

Taking the limit of the above, we see that

$$r_k \longrightarrow \hat{\underline{r}} = \frac{y(i)-x(i)}{d(i)}.$$

It then follows that

$$y(j) = x(j) \quad (\text{since } d(j) = 0 \text{ for } j = 1, 2, \ldots, n;\ j \neq i),$$

$$y(i) = x(i)+\hat{r}\cdot d(i).$$

Thus

$$\underline{y} = \underline{x}+\hat{r}\cdot\underline{d}.$$

For each k and each r_k, we know that

Proof: (of Lemma - continued)

$$F(\underline{y}_k) \leq \underset{r}{\text{Min}}\ F(\underline{x}_k+r\cdot\underline{d}_k) \quad \text{(by the unimodality of F)},$$

i.e. for each k and each r, $0 \leq r < \infty$,

$$F(\underline{y}_k) \leqslant F(\underline{x}_k+r\cdot\underline{d}_k).$$

Letting $k \longrightarrow \infty$, we obtain

$$F(\underline{y}) \leqslant F(\underline{x}+r\cdot\underline{d}).$$

Thus

$$F(\underline{y}) = \underset{r}{\text{Min}}\ F(\underline{x}+r\cdot\underline{d}),$$

and hence

$$\underline{y} \in A(\underline{x}, \underline{d}).$$

STEP II

This step represents an algorithm which depends on the n+1 points of the complex. $M_2 : \mathbb{R}^{n(n+1)} \longrightarrow \mathbb{R}^n$. By the convexity of the feasible region, a new point NEWX will be generated under this algorithm map using the informations on the n+1 points that we have.

Observe that the map M_2 is continuous.

STEP III

Define the algorithm $M_3 : \mathbb{R}^n \longrightarrow \mathbb{R}^{n(n+1)}$.

Again by the convexity of the feasible region, a new set of n+1 points will be generated. In this set, n points are from the previous configuration, and only one is new. In other words, M_3(NEWX) will give a new set of n+1 points as long as F(NEWX) gives a value at least smaller than the one given by the second worst vertex.

STEP IV

This final step represents the map $M_4 : \mathbb{R}^{n(n+1)} \longrightarrow \mathbb{R}^n$. In this algorithm, a best point will be chosen among the n+1 points when the given accuracy bound is satisfied. This best point will then be our solution.

With the above four steps, our overall process M amounts merely to repeated applications of the composite algorithm M_3M_2 until a certain accuracy bound is satisfied. Now let us concentrate on the algorithm M_3M_2. We will represent M_3M_2 as M' where $M' : \mathbb{R}^{n(n+1)} \longrightarrow \mathbb{R}^{n(n+1)}$. Then after each iteration, n+1 points will be generated. Collecting the "best point" among these n+1 points (regardless whether the accuracy bound is satisfied), we will obtain a sequence of best points. We will denote this sequence as $\underline{b}_k$. To prove the theorem stated on page 26, it will be sufficient for us to show that as $k \longrightarrow \infty$, $\underline{b}_k \longrightarrow \underline{b}$ where $\underline{b}$ is the solution point.

Proof: (of Theorem)

Under the algorithm M', we know that $\{\underline{b}_k\}$ is bounded. Since every bounded sequence has a convergent subsequence, hence if we let $\underline{b}$ to be the limit of this sequence, then we want to show that $\underline{b}$ is indeed a solution.

Let $\{\underline{b}_{k'}\}$ be the convergent subsequence which converges to the limit $\underline{b}$.

Since F is continuous, then it follows that

$$\lim_{k\to\infty} F(\underline{b}_{k'}) = F(\underline{b}).$$

<u>Proof</u>: (of Theorem - continued)

This means that F is convergent with respect to this subsequence. Now we shall show that F is convergent with respect to the entire sequence. By the monotonicity of F on the sequence $\{\underline{b}_{k'}\}$, we know

$$F(\underline{b}_{k'}) - F(\underline{b}) \geqslant 0 \qquad \text{for all } k'.$$

Then by the convergence of F on the subsequence, there exists, for a given $\varepsilon > 0$, a number K such that

$$F(\underline{b}_{k'}) - F(\underline{b}) < \varepsilon \qquad \text{for all } k' > K.$$

Thus for all $k > K$,

$$F(\underline{b}_k) - F(\underline{b}) < \varepsilon$$

which means

$$F(\underline{b}_k) \longrightarrow F(\underline{b}) \text{ as } k \longrightarrow \infty.$$

Hence F is convergent with respect to the entire sequence.

Finally, we have to show that $\underline{b}$ is a solution.

Let us suppose that this is not the case. Then there exists a point $\underline{b}_k{*}$ such that

$$F(\underline{b}_k{*}) < F(\underline{b}).$$

However $\underline{b}$ is the limit of the sequence and we have

$$F(\underline{b}_{k+1}) \leqslant F(\underline{b}_k) \qquad \text{for all } k.$$

Thus

$$F(\underline{b}_k{*}) < F(\underline{b})$$

simply contradicts the fact that F is a descent funct-

Proof: (of Theorem - continued)

ion with respect to the sequence $\underline{b}_k$. Summing up from above, convergence of the algorithm M is therefore guaranteed.

Even though our proof for global convergence of the algorithm requires that the feasible region is convex and F is unimodal, however in practice, this algorithm still converges when the above two conditions are violated. This will be illustrated in the comparison test (Problem 3).

(IV) NUMERICAL COMPARISON OF THE REVISED AND ORIGINAL COMPLEX METHOD

4.1 Computer Used

Having proved the global convergence of the Revised Complex method, it is time for us to compare it with the original strategy of Box. The usual way to do is to refer to a minimum problem for which the known method fails to find a solution whereas the new proposal is successful. Or it is shown with reference to chosen examples that computation time and iterations can be saved by using the new version. In our case, the latter method will be employed, and since almost all iteration strategies require a considerable number of calculation steps, we need some kind of mechanical assistance.

The machine on which the numerical experiments will be carried out is a Televideo model 925 from the computer center of Concordia University. All computer codes for the problems of this paper are written in Fortran 77, and they are therefore machine independent. The codes are then extensively tested on the NOS timesharing system of the Control Data CYBER 830.

4.2 Termination Criterion

The two strategies will be judged by the computation time that they require to achieve a result, with a specified accuracy. The basic quantity for this purpose is the execution time of the central unit (Central Processor seconds = CP seconds). We will also look at the number of iterations since any of these measures is not entirely satisfactory. The computer time needed to execute an algorithm depends not only on its efficiency but also on the type of machine used, the character of the measured time, and the efficiency of coding. Also, the number of iterations cannot be used as the only measure of effectiveness of an algorithm because the effort per iteration may vary considerably from one procedure to another.

The termination criterion for both methods is based on testing whether the difference between the objective function values at the new point of the present complex and the worst point from the previous iteration is less than a prescribed limit E. (The original termination criterion of the strategy of Box has been changed here in carrying out the tests.) In this case, the difference between the average of the objective function values at the vertices of the second last iteration and of the last iteration is less than $E/(n+1)$. The calculation is as follows:

Let us suppose that in the second last iteration, the $n+1$ objective function values are $(a_1, a_2, \ldots, a_n, a_{n+1})$ and without loss of generality, let us suppose that the last value gives the worst result. Then in the last iteration, a new complex is generated which gives n objective function values $a_1, a_2, \ldots, a_n$ and one new objective function value a_{n+1}':

$$(a_1, a_2, \ldots, a_n, a_{n+1}').$$

Since

$$a_{n+1} - a_{n+1}' < E,$$

hence if

$$\frac{a_1+a_2+\ldots+a_n+a_{n+1}}{n+1} = A,$$

and

$$\frac{a_1+a_2+\ldots+a_n+a_{n+1}'}{n+1} = A',$$

then

$$A - A' = \frac{a_{n+1} - a_{n+1}'}{n+1}.$$

This means that

$$A - A' < \frac{E}{n+1}.$$

We will set $E = 10^{-10}$ for all problems. Moreover, in order to see how a nonrandom selection of the initial complex -- and the over-reflection factor will affect the rate of convergence, both strategies will use the same number of vertices to form the polyhedron. In all cases, n+1 points will be used. Finally, in Box's method, a fixed reflection factor R = 1.3 is employed in all tests.

4.3 Test Problems

The problems that we will carry out comparison are the following:

Problem 1

Objective function : $F(\underline{x}) = 100(x_2-x_1^2)^2+(x_1-1)^2$; $\underline{x}\in R^2$.

Constraints : $-3 \leqslant x_1 \leqslant 3$,

$-1.5 \leqslant x_2 \leqslant 4.5$.

Minimum : $\underline{x}^* = (1, 1)$; $F(\underline{x}^*) = 0$.

Start : $\underline{x}(0) = (-1.2, 1)$; $F(\underline{x}(0)) = 24.2$.

Problem 2

Objective function : $F(\underline{x}) = 2x_1^2+2x_2^2+x_3^2+2x_1x_2+2x_1x_3-8x_1-6x_2-4x_3+9$; $\underline{x}\in R^3$.

Constraints : $x_1 \geqslant 0$,

$x_2 \geqslant 0$,

$x_3 \geqslant 0$,

$-x_1-x_2-2x_3+3 \geqslant 0$.

Minimum : $\underline{x}^* = (4/3, 7/9, 4/9)$; $F(\underline{x}^*) = 1/9$.

Start : $\underline{x}(0) = (0.1, 0.1, 0.1)$;

$F(\underline{x}(0)) = 7.29$.

Problem 3

Objective function : $F(\underline{x}) = -x_1^2 - x_2^2$; $\underline{x} \in \mathbb{R}^2$.

Constraints : $x_1 \geqslant 0$,

$x_2 \geqslant 0$,

$-x_1 + x_2 + 4 \geqslant 0$,

$\frac{x_1}{3} - x_2 + 4 \geqslant 0$,

$x_1^2 + x_2^2 - 10x_1 - 10x_2 + 41 \geqslant 0$.

Minimum : $\underline{x}^* = (12, 8)$; $F(\underline{x}^*) = -208$.

Start : $\underline{x}(0) = (0, 0)$; $F(\underline{x}(0)) = 0$.

Besides the above global minimum, there are two local minima:

$\underline{MIN}_1 = (2.018, 4.673)$; $F(\underline{MIN}_1) = -25.91$.

$\underline{MIN}_2 = (6.293, 2.293)$; $F(\underline{MIN}_2) = -44.86$.

Problem 4

Objective function : $F(\underline{x}) = 3x_1^2+x_2^2-2x_1x_2-x_2$; $\underline{x}\in\mathbb{R}^2$.

Constraints : $0 \leq x_1 \leq 1$,

$0 \leq x_2 \leq 1$.

Minimum : $\underline{x}^* = (0.25,\ 0.75)$; $F(\underline{x}^*) = -0.375$.

Start : $\underline{x}(0) = (0.5,\ 0.5)$; $F(\underline{x}(0)) = 0$.

Besides the above global minimum, there are three local minima:

$\underline{MIN}_1 = (0,\ 0)$; $F(\underline{MIN}_1) = 0$,

$\underline{MIN}_2 = (0,\ 0.5)$; $F(\underline{MIN}_2) = -0.25$,

$\underline{MIN}_3 = (1/3,\ 1)$; $F(\underline{MIN}_3) = -1/3$.

4.4 Results of the Tests

Individual programs have been written (see Appendix) for each of the above four problems. All arrays and vectors specified in the dimension statements have been altered for each particular problem. After the codes had run, various results were obtained and were presented in the following five tables respectively.

Table 1 summarizes the results of the tests when the Complex strategy of Box is employed. Table 2, 3, 4 and 5 represent the test results when the Revised Complex method is used to solve Problem 1, 2, 3 and 4, where in each case, different upper limit for the over-reflection factor is employed.

Table 1

Complex Method of Box (R = 1.3)

Problem	Execution Time (CP, seconds)	Number of Iterations	Minimum ($\underline{x}^*$)	Objective Function Value $F(\underline{x}^*)$
1	0.646	144	(0.999696807, 0.999431389)	0.0000002339
2	0.964	195	(1.333334179, 0.777776230, 0.444444796)	0.1111111111
3	1.031	337	(6.292893219, 2.292893219)	-44.8578643762
4	0.334	85	(0.249994855, 0.749993568)	-0.3749999999

<u>Table 2</u> (Problem 1)

<u>Revised Complex Method (Upper Limit of R = R')</u>

R'	Execution Time (CP seconds)	Number of Iterations	Minimum ($\underline{x}^*$)	Objective Function Value $F(\underline{x}^*)$
2.9	0.282	59	$\begin{pmatrix} 1.000078043 \\ 1.000149624 \end{pmatrix}$	0.0000000103
3.9	0.278	56	$\begin{pmatrix} 1.000002761 \\ 1.000003590 \end{pmatrix}$	0.0000000004
4.9	0.301	63	$\begin{pmatrix} 1.000003125 \\ 1.000006323 \end{pmatrix}$	0.0000000000

Table 3 (Problem 2)

Revised Complex Method (Upper Limit of R = R')

R'	Execution Time (CP seconds)	Number of Iterations	Minimum ($\underline{x}^*$)	Objective Function Value $F(\underline{x}^*)$
2.9	0.640	98	1.333331446 0.777778028 0.444445263	0.1111111111
3.9	0.716	103	1.333335347 0.777777371 0.444443641	0.1111111111
4.9	0.760	101	1.333331069 0.777780478 0.444444225	0.1111111116
5.9	0.479	65	1.333293961 0.777779633 0.444463185	0.1111111209
6.9	0.704	97	1.333326931 0.777774518 0.444449275	0.1111111113

Table 4 (Problem 3)

Revised Complex Method (Upper Limit of R = R')

R'	Execution Time (CP seconds)	Number of Iterations	Minimum ($\underline{x}^*$)	Objective Function Value $F(\underline{x}^*)$
2.9	Time Limit exceeded	First Iteration is obtained	$\begin{pmatrix} 9.902343750 \\ 6.093750000 \end{pmatrix}$	-135.1902008057
3.9	0.691	113	$\begin{pmatrix} 6.292893219 \\ 2.292893219 \end{pmatrix}$	-44.8578643762
4.9	Time Limit exceeded	First Iteration is obtained	$\begin{pmatrix} 8.759765625 \\ 5.390625000 \end{pmatrix}$	-105.7923316956
5.9	Time Limit exceeded	First Iteration is obtained	$\begin{pmatrix} 10.029296875 \\ 6.171875000 \end{pmatrix}$	-138.6788368225
6.9	0.730	91	$\begin{pmatrix} 6.292893219 \\ 2.292893219 \end{pmatrix}$	-44.8578643760
7.9	1.115	126	$\begin{pmatrix} 12.000000000 \\ 8.000000000 \end{pmatrix}$	-208.0000000000
8.9	Time Limit exceeded	First Iteration is obtained	$\begin{pmatrix} 8.188476563 \\ 5.039062500 \end{pmatrix}$	-92.4432992935

Remark : Whenever the execution time exceeded the time limit, we considered the vector obtained in the last iteration to be the minimum.

Table 5 (Problem 4)

Revised Complex Method (Upper Limit of R = R')

R'	Execution Time (CP seconds)	Number of Iterations	Minimum ($\underline{x}^*$)	Objective Function Value $F(\underline{x}^*)$
2.9	0.315	47	$\begin{pmatrix} 0.249987056 \\ 0.749976666 \end{pmatrix}$	-0.3749999996

4.5 Discussion

From the above data, we can see that the Revised Complex method converges at a much faster rate than the Complex method of Box which uses a fixed over-reflection factor R = 1.3. In fact, let us look at the results for Problem 3 closely. In Box's method, after 337 iterations, only a local minimum is obtained, and there is no indication whether there exists a better optimum. However, in the Revised Complex method, the employment of R' = 2.9 gives us an objective function value which is already better than the one we have obtained from Box's method (even though the program has been terminated before it reaches the accuracy bound, and only the first iteration is being generated). Once we have increased the upper limit for the reflection factor to 7.9, the optimum we get not only yields a better objective function value but also satisfies the accuracy parameter. We ceased increasing further for R' = 8.9 because at this point, the program converged much, much slower than the previous case (when R' = 7.9), and was only able to give us a worst objective function value.

For Problem 4, both the Revised and the original Complex strategy were able to generate the global minimum. In the Revised Complex method, we do not need to increase the upper limit R' further to see whether a better optimum might be attained because in all 47 iterations we have, there was not even one iteration which employed a reflection factor greater than 2.9. This means that no matter what R' is, as long as it is greater than or equal to 1, the program will converge at the same speed and produce the same minimum. Even though both methods gave the same optimum, the Revised Complex method appears to perform much better, at least with the given initial point.

(V) CONCLUSION

With the above discussion, it seems that the Revised Complex method offers significant advantages over Box's strategy which involves a pseudo-random process and employs a fixed over-reflection factor. This new method appears to be not only more efficient but is also able to give information about the global optimum. In the case where there exist several optima, it is extremely important for us to be able to discover the global optimum among these local points. The entire process may be very costly, especially when one considers the case where an increase in the upper limit for R has led directly to a large reduction in the rate of convergence (as shown in the test : Problem 3). However, at this stage, it seems that we have no alternative but to compromise between reliability and speed.

In conclusion, the author would like to mention that both Rosenbrock's method and the Complex method have been used extensively for constrained optimization and appear to be very reliable, even though both methods require a relatively long computation time and a lot of storage space. (In the original Complex method, Box suggested that the suitable value for number of vertices in each configuration was 2n.) Therefore the new proposal is able to provide a good alternative, being able to converge at a faster speed and requiring less storage than is necessary with Box's method which uses 2n points to construct the polyhedron, especially when the dimension of the objective function is large. Moreover, without requiring a large preparation and computation time, a rather systematic search for alternative optima is made possible by increasing the upper limit of the over-reflection factor. However, limited computational experience reported in the present paper is insufficient to warrant a definitive claim of superiority of the Revised Complex method over other available methods.

BIBLIOGRAPHY

1. Bazaraa, Mokhtar S., and C.M. Shetty, Nonlinear Programming : Theory and Algorithms, John Wiley and Sons, Inc., Toronto, 1979, 229-243.

2. Box, M.J., A New Method of Constrained Optimization and A Comparison with Other Methods, The Computer Journal, 8(1965), 42-45.

3. Box, M.J., D. Davies, and W.H. Swann, Non-linear Optimization Techniques, Mathematical and Statistical Techniques for Industry, Monograph No. 5(1969), 49-54.

4. Guin, J.A., Modification of the Complex Method of Constrained Optimization, The Computer Journal, 10(1968), 416-417.

5. Luenberger, David G., Linear and Nonlinear Programming, 2nd ed., Addison-Wesley Publishing Co., Don Mills, Ontario, 1984, 196-232.

6. Mitchell, R.A., and J.L. Kaplan, Nonlinear Constrained Optimization by a Non-random Complex Method, Journal of Research of the National Bureau of Standards - C. Engineering and Instrumentation, 72(1968), 249-258.

7. Nelder, J.A., and R. Mead, <u>A Simplex Method for Function Minimization</u>, The Computer Journal, 7(1965), 308-313.

8. Rosenbrock, H.H., <u>An Automatic Method for Finding the Greatest or Least Value of a Function</u>, The Computer Journal, 3(1960), 175-184.

9. Schwefel, Hans-Paul, <u>Numerical Optimization of Computer Models</u>, John Wiley and Sons, Inc., Toronto, 1981, 155-233.

APPENDIX

Remark : The following are the individual programs for the test problems. Because of the high similarity of some programs, only several of the results presented on pages 40-44 are included. The program name indicates the test problem number and which method is used. For example, PROGRAM TEST1R means that Problem 1 is solved using the Revised Complex method; PROGRAM TEST2O means that Problem 2 is solved using the original Complex method. (There are two programs listed as PROGRAM TEST3R. The first one uses 6.9 as the upper limit for the reflection factor while the second program uses 7.9 for R'.)

```
      PROGRAM TESTIR     74/810  OPT=0,ROUND= A/ S/ M/-D,-DS     FTN 5.1+642        87/03/01. 14.58.40
DO=-LONG/-OT,ARG=-COMMON/-FIXED,CS= USER/-FIXED,DB=-TB/-SB/-SL/ ER/-ID/-PMD/-ST,-AL,PL=5000
FTN5,I=PROG1.

      PROGRAM TESTIR (INPUT, OUTPUT)
C
C     ***************************************************************
C     *  OBJECTIVE FUNCTION  : F(X1,X2) = 100*((X2-X1**2)**2)+(X1-1)**2  *
C     *  CONSTRAINTS         :    -3.GE.X1.LE.3                      *
C     *                         -1.5.GE.X2.LE.4.5                    *
C     *  MINIMUM             : MIN = (1,1)        ; F(1,1) = 0       *
C     *  START               : X(0) = (-1.2,1)    ; F(-1.2,1) = 24.2 *
C     ***************************************************************
C
      INTEGER I, K, L, J, N, A
      REAL X(0:3,2), S, SUM(2), AVE(2), NEWX(2), R, WORST, BEST, B, W,
     $     VALUE(3), VALNEW, TEST(2), E, D(2), SEARCH(11), BNEW(2)
      READ*, N
C
C     *****************************************************************
C     *  N IS THE DIMENSION OF THE SYSTEM.  IN THIS PROBLEM, N = 2 AND WE  *
C     *  WILL USE N+1 = 3 POINTS TO CONSTRUCT A POLYHEDRON (SINCE WE ONLY  *
C     *  HAVE THREE POINTS, A TRIANGLE WILL BE FORMED).                    *
C     *                                                                    *
C     *  A = NUMBER OF ITERATIONS                                          *
C     *****************************************************************
C
      DO 10 I=1, N
           READ*, X(0,I)
           X(1,I) = X(0,I)
   10 CONTINUE
C
C     *****************************************************************
C     *  S IS THE STEPSIZE THAT WE WILL USE WHEN CARRYING OUT LINE-SEARCH  *
C     *  ALGORITHM IN EACH COORDINATE DIRECTION.  THE VALUE OF S IS        *
C     *  DETERMINED BY THE FOLLOWING;                                      *
C     *  S = (UPPER BOUND OF X - LOWER BOUND OF X)/NUMBERS OF INTERVALS    *
C     *****************************************************************
C
      DO 70 K=2, N+1
           S = 0.6
           J = K-1
           IF (K.EQ.2) THEN
              X(K,1) = -3
              X(K,2) = X(J,2)
           ELSE
              X(K,1) = X(J,1)
              X(K,2) = -1.5
           ENDIF
           SEARCH(1) = F(X(K,1),X(K,2))
           BEST = SEARCH(1)
           DO 20 I=1, N
                BNEW(I) = X(K,I)
   20      CONTINUE
           DO 50 L=2, 11
                DO 30 I=1, N
                     IF ((J-I).EQ.0) THEN
                          X(K,I) = X(K,I)+S
                     ELSE
```

```
                       X(K,I) = X(J,I)
                    ENDIF
30             CONTINUE
               SEARCH(L) = F(X(K,1),X(K,2))
               IF (SEARCH(L).LT.BEST) THEN
                  BEST = SEARCH(L)
                  DO 40 I=1, N
                      BNEW(I) = X(K,I)
40                CONTINUE
               ENDIF
50         CONTINUE
           DO 60 I=1, N
               X(K,I) = BNEW(I)
60         CONTINUE
70     CONTINUE
       PRINT 80
80     FORMAT (5(/), 8X, 'X(K,1)', 15X, 'X(K,2)', 13X,
      $        'F(X(K,1),X(K,2))')
       DO 100 K=1, N+1
              PRINT 90, X(K,1), X(K,2), F(X(K,1),X(K,2))
90            FORMAT (4X, F12.9, 10X, F11.9, 10X, F16.10)
100    CONTINUE
C
C      ***********************************************************************
C      *  NOW WE HAVE GENERATED THREE POINTS WHICH ARE ALL INSIDE THE FEASIBLE  *
C      *  REGION.  THESE THREE POINTS ARE INDICATED BY X(K) = (X(K,1),X(K,2))   *
C      *  WHERE K = 1, 2, 3.  THE NEXT STEP IS TO FIND OUT THE WORST POINT      *
C      *  AMONG THESE THREE POINTS.  WORST MEANS THAT THE POINT WILL GIVE THE   *
C      *  LARGEST OBJECTIVE FUNCTION VALUE.  THEN WE WILL CONSTRUCT A NEW       *
C      *  POINT (NEWX(1),NEWX(2)) AS ILLUSTRATED IN THE DO-LOOP 170.            *
C      ***********************************************************************
C
       PRINT 110
110    FORMAT (5(/), 7X, 'NEWX(1)', 14X, 'NEWX(2)', 13X, 'VALNEW',
      $        7X, 'REFLEXION FACTOR')
       A = 0
       READ*, E
C
C      ********************************
C      *  E IS THE ACCURACY PARAMETER.  *
C      ********************************
C
120    DO 130 K=1, N+1
              VALUE(K) = F(X(K,1),X(K,2))
130    CONTINUE
       WORST = VALUE(1)
       W = 1
       DO 140 K=2, N+1
              IF (VALUE(K).GT.WORST) THEN
                 WORST = VALUE(K)
                 W = K
              ENDIF
140    CONTINUE
       TEST(1) = WORST
C
C      *************************************************************
C
```

```
      DO 160 I=1, N
            SUM(I) = 0
            DO 150 K=1, N+1
                  IF (K.NE.W) THEN
                     SUM(I) = SUM(I)+X(K,I)
                  ENDIF
150         CONTINUE
            AVE(I) = SUM(I)/N
            D(I) = AVE(I)-X(W,I)
160   CONTINUE
      R = FACTOR(AVE(1),AVE(2),D(1),D(2))
C
C     **********************************************************************
C     *  R IS CALLED THE OVER-REFLECTION FACTOR.  THE SELECTION OF THE VALUE  *
C     *  FOR R IS DETERMINED BY THE REAL FUNCTION FACTOR.  R MUST BE GREATER  *
C     *  THAN OR EQUAL TO 1.                                                  *
C     **********************************************************************
C
      DO 170 I=1, N
            NEWX(I) = AVE(I)+R*D(I)
170   CONTINUE
180   VALNEW = F(NEWX(1),NEWX(2))
      IF ((C1(NEWX(1)).EQ.1).AND.(C2(NEWX(2)).EQ.1)) THEN
         DO 200 K=1, N+1
               IF ((K.NE.W).AND.(VALNEW.LT.VALUE(K))) THEN
                  DO 190 I=1, N
                        X(W,I) = NEWX(I)
190               CONTINUE
                  GO TO 220
               ENDIF
200      CONTINUE
      ENDIF
      DO 210 I=1, N
            NEWX(I) = 0.5*(AVE(I)+NEWX(I))
210   CONTINUE
      GO TO 180
C
C     **********************************************************************
C     *  THE FOLLOWING STEP IS TO CHECK WHETHER THE VARIANCE OF THE OBJECTIVE  *
C     *  FUNCTION VALUES AT THE THREE VERTICES IS LESS THAN OUR PRESCRIBED     *
C     *  LIMIT.  THIS WILL BE THE CRITERION FOR ENDING THE MINIMUM SEARCH.     *
C     **********************************************************************
C
220   TEST(2) = VALNEW
      A = A+1
      PRINT 230, NEWX(1), NEWX(2), VALNEW, R
230   FORMAT (4X, F12.9, 9X, F11.9, 8X, F14.10, 11X, F3.1)
C
C     *****************************************************************
C
      IF ((TEST(1)-TEST(2)).LT.E) THEN
         DO 240 K=1, N+1
               VALUE(K) = F(X(K,1),X(K,2))
240      CONTINUE
         BEST = VALUE(1)
         B = 1
         DO 260 K=2, N+1
```

```
1                         IF (VALUE(K).LT.BEST) THEN
2                            BEST = VALUE(K)
2                            B = K
2                         ENDIF
1         250       CONTINUE
1                   PRINT 260, X(B,1), X(B,2), BEST
1         260       FORMAT (9(/), 6X, 'MINIMUM = ( ', F11.9, ', ', F11.9,
1               $         ')    ;     ', 'F(MINIMUM) = ', F12.10)
1                   PRINT*, ' '
1                   PRINT 270, A
1         270       FORMAT (8X, 'NUMBER OF ITERATIONS = ', I3)
1               ELSE
1                   GO TO 120
1               ENDIF
                STOP
                END
```

--VARIABLE MAP--(LO=A)

NAME	ADDRESS	BLOCK	PROPERTIES	TYPE	SIZE
A	1065B			INTEGER	
AVE	1101B			REAL	2
B	1110B			REAL	
BEST	1107B			REAL	
BNEW	1136B			REAL	2
D	1121B			REAL	2
E	1120B			REAL	
I	1060B			INTEGER	
J	1063B			INTEGER	
K	1061B			INTEGER	
L	1062B			INTEGER	
N	1064B			INTEGER	
NEWX	1103B			REAL	2
R	1105B			REAL	
S	1076B			REAL	
SEARCH	1123B			REAL	11
SUM	1077B			REAL	2
TEST	1116B			REAL	2
VALNEW	1115B			REAL	
VALUE	1112B			REAL	3
W	1111B			REAL	
WORST	1106B			REAL	
X	1066B			REAL	8

--PROCEDURES--(LO=A)

NAME	TYPE	ARGS	CLASS
C1	REAL	1	FUNCTION
C2	REAL	1	FUNCTION
F	REAL	2	FUNCTION
FACTOR	REAL	4	FUNCTION

--STATEMENT LABELS--(LO=A)

LABEL	ADDRESS	PROPERTIES	DEF
10	INACTIVE	DO-TERM	27
20	INACTIVE	DO-TERM	50
30	INACTIVE	DO-TERM	58
40	INACTIVE	DO-TERM	64
50	INACTIVE	DO-TERM	66
60	INACTIVE	DO-TERM	69
70	INACTIVE	DO-TERM	70
80	707B	FORMAT	72
90	710B	FORMAT	76
100	INACTIVE	DO-TERM	77
110	723B	FORMAT	89
120	315B		98
130	INACTIVE	DO-TERM	100
140	INACTIVE	DO-TERM	108
150	INACTIVE	DO-TERM	119
160	INACTIVE	DO-TERM	122
170	INACTIVE	DO-TERM	133
180	464B		134
190	INACTIVE	DO-TERM	140
200	INACTIVE	DO-TERM	143
210	INACTIVE	DO-TERM	147
220	563B		156
230	733B	FORMAT	159
240	INACTIVE	DO-TERM	166
250	INACTIVE	DO-TERM	174
260	740B	FORMAT	176
270	751B	FORMAT	180

```
--ENTRY POINTS--(LO=A)
 -NAME---ADDRESS--ARGS---

  TESTIR      14B      0

--STATISTICS--

  PROGRAM-UNIT LENGTH            1162B =  626
  CM STORAGE USED               63300B =  26304
  COMPILE TIME                  0.896 SECONDS
```

```
  FUNCTION F          74/810  OPT=0,ROUND= A/ S/ M/-D,-DS      FTN 5.1+642          87/03/01. 14.58.49
DO=-LONG/-OT,ARG=-COMMON/-FIXED,CS= USER/-FIXED,DB=-TB/-SB/-SL/ ER/-ID/-PMD/-ST,-AL,PL=5000
FTN5,I=PROG1.

                      REAL FUNCTION F(X1,X2)
                      REAL X1, X2
                      F = 100*((X2-X1**2)**2)+(X1-1)**2
                      RETURN
                      END

--VARIABLE MAP--(LO=A)
 -NAME---ADDRESS --BLOCK-----PROPERTIES-------TYPE---------SIZE

  F          25B                              REAL
  X1           1   DUMMY-ARG                  REAL
  X2           2   DUMMY-ARG                  REAL

--ENTRY POINTS--(LO=A)
 -NAME---ADDRESS--ARGS---

  F          7B     2

--STATISTICS--

  PROGRAM-UNIT LENGTH            30B =  24
  CM STORAGE USED            63300B =  26304
  COMPILE TIME               0.043 SECONDS
```

```
FUNCTION C1         74/810  OPT=0,ROUND= A/ S/ M/-D,-DS     FTN 5.1+642        87/03/81, 14.58.40
DO=-LONG/-OT,ARG=-COMMON/-FIXED,CS= USER/-FIXED,DB=-TB/-SB/-SL/ ER/-ID/-PMD/-ST,-AL,PL=5000
FTN5,I=PROG1.

                 REAL FUNCTION C1(X1)
                 REAL X1
                 IF ((X1.GE.-3).AND.(X1.LE.3)) THEN
                    C1 = 1
                 ELSE
                    C1 = 0
                 ENDIF
                 RETURN
                 END

--VARIABLE MAP--(LO=A)
 -NAME---ADDRESS --BLOCK-----PROPERTIES-------TYPE---------SIZE

  C1          32B                              REAL
  X1            1   DUMMY-ARG                  REAL

--ENTRY POINTS--(LO=A)
 -NAME---ADDRESS--ARGS---

  C1          7B      1

--STATISTICS--

  PROGRAM-UNIT LENGTH              35B =   29
  CM STORAGE USED              633000B =   26304
  COMPILE TIME                 0.048 SECONDS
```

```
      FUNCTION C2          74/810  OPT=0,ROUND= A/ S/ M/-D,-DS      FTN 5.1+642           87/03/01. 14.58.40
DO=-LONG/-OT,ARG=-COMMON/-FIXED,CS= USER/-FIXED,DB=-TB/-SB/-SL/ ER/-ID/-PMD/-ST,-AL,PL=5000
FTN5,I=PROG1.

          1                 REAL FUNCTION C2(X2)
          2                 REAL X2
          3                 IF ((X2.GE.-1.5).AND.(X2.LE.4.5)) THEN
    1     4                    C2 = 1
    1     5                 ELSE
    1     6                    C2 = 0
    1     7                 ENDIF
          8                 RETURN
          9                 END

--VARIABLE MAP--(LO=A)
 -NAME---ADDRESS --BLOCK-----PROPERTIES-------TYPE---------SIZE

  C2          33B                                REAL
  X2            1   DUMMY-ARG                    REAL

--ENTRY POINTS--(LO=A)
 -NAME---ADDRESS--ARGS---

  C2          7B      1

--STATISTICS--

  PROGRAM-UNIT LENGTH             36B =   30
  CM STORAGE USED              63300B = 26304
  COMPILE TIME                 0.047 SECONDS
```

```
 FUNCTION FACTOR      74/810  OPT=0,ROUND= A/ S/ M/-D,-DS      FTN 5.1+642         87/03/01. 14.58.40
DO=-LONG/-OT,ARG=-COMMON/-FIXED,CS= USER/-FIXED,DB=-TB/-SB/-SL/ ER/-ID/-PMD/-ST,-AL,PL=5000
FTN5,I=PROG1.

            REAL FUNCTION FACTOR(AVE1,AVE2,D1,D2)
            REAL AVE1, AVE2, D1, D2, S, R, F(2)
            INTEGER I
            R = 1
            F(1) = 100*(((AVE2+R*D2)-(AVE1+R*D1)**2)**2)+((AVE1+R*D1)-1)**2
            S = 0.1
            DO 10 I=2, 30
                  R = 1+(I-1)*S
                  F(2) = 100*(((AVE2+R*D2)-(AVE1+R*D1)**2)**2)+((AVE1+R*D1)
     $                -1)**2
                  IF (F(2).GT.F(1)) THEN
                     R = R-S
                     GO TO 20
                  ELSE
                     F(1) = F(2)
                  ENDIF
   10       CONTINUE
   20       FACTOR = R
            RETURN
            END

--VARIABLE MAP--(LO=A)
 -NAME---ADDRESS --BLOCK-----PROPERTIES-------TYPE----------SIZE     -NAME---ADDRESS --BLOCK-----PROPERTIES-------TYPE---------SIZE

  AVE1          1   DUMMY-ARG                  REAL                    FACTOR      76B                               REAL
  AVE2          2   DUMMY-ARG                  REAL                    I          103B                               INTEGER
  D1            3   DUMMY-ARG                  REAL                    R          100B                               REAL
  D2            4   DUMMY-ARG                  REAL                    S           77B                               REAL
  F          101B                              REAL              2

--STATEMENT LABELS--(LO=A)
 -LABEL-ADDRESS-----PROPERTIES----DEF

    10  INACTIVE   DO-TERM        17
    20     60B                    18

--ENTRY POINTS--(LO=A)
 -NAME---ADDRESS--ARGS---

  FACTOR      7B     4

--STATISTICS--

  PROGRAM-UNIT LENGTH            107B =   71
  CM STORAGE USED             63300B = 26304
  COMPILE TIME                0.139 SECONDS
```

2
-1.2
1
0.0000000001

X(K,1)	X(K,2)	F(X(K,1),X(K,2))
-1.200000000	1.000000000	24.2000000000
1.200000000	1.000000000	19.4000000000
1.200000000	1.500000000	.4000000000

NEWX(1)	NEWX(2)	VALNEW	REFLEXION FACTOR
1.275000000	1.257812500	13.6042785156	1.0
1.275000000	1.757812500	1.8229785156	1.0
1.235156250	1.052099609	1.6552362556	1.0
1.102734375	1.212524414	.0117764334	2.0
1.109472656	1.208343506	.0629973142	1.0
1.100234985	1.192336082	.0431017122	1.0
1.093496704	1.196519991	.0088027781	1.0
1.097585678	1.207568857	.0103481112	1.0
1.093742895	1.197421301	.0097785828	1.0
1.078152874	1.160530072	.0064596391	3.9
1.077966682	1.157633761	.0078748979	1.0
1.070296315	1.140370380	.0070060777	1.0
1.059864460	1.122454038	.0036574743	3.9
1.067721600	1.142621730	.0052587996	1.0
1.049432585	1.104537696	.0034861994	1.0
1.032425283	1.063981900	.0014200639	1.7
.995483985	.992593622	.0002782673	2.4
.996215441	.990162793	.0005352635	1.0
.977561711	.955076361	.0005337860	1.0
.984099519	.969752041	.0004271318	1.0
.991328344	.984435453	.0003709797	1.0
1.002704521	1.007270033	.0003531544	1.0
1.006887872	1.015434202	.0003198437	1.0
.992213700	.981791639	.0000930891	3.9
.999690118	.990190844	.0001917370	1.0
.989319132	.978094867	.0001161121	1.0
.990550105	.982927500	.0000153099	1.0
1.002444663	1.005324277	.0000243750	1.0
1.001630236	1.003043013	.0000083585	1.0
.998675919	.997067879	.0000089164	1.0
1.001060446	1.001837433	.0000091077	1.0
1.001516430	1.002775995	.0000090167	1.0
1.001837027	1.003449640	.0000087519	1.0
1.002115024	1.004040356	.0000082434	1.7
1.002033634	1.003916450	.0000065239	3.9
1.002509322	1.004913092	.0000075276	1.0
1.002443472	1.004827783	.0000063946	1.1
1.001162300	1.002250291	.0000025369	3.9
1.001303605	1.002634622	.0000021479	1.9
1.000910057	1.000930647	.0000000020	1.1
[illegible]	1.001024057	.0000010462	1.0
[illegible]	.999379067	.0000003307	1.0
.999520606	.999651007	.0000002767	1.0
.999870375	.999820958	[illegible]	1.0
1.000110606	1.000223175	.0000000307	1.0
1.000127063	1.000250050	.0000000164	1.0
1.000049641	1.000100300	.0000000130	1.0

.9999999972	.9999992281	.0000000100	1.0
.9999953360	.9999910960	.0000000039	1.0
.9999984373	.9999960734	.0000000006	1.0
1.0000024303	1.0000048182	.0000000006	1.0
1.0000002781	1.0000003590	.0000000004	1.0
1.0000020072	1.0000040624	.0000000005	1.0
1.0000036535	1.0000009100	.0000000004	1.0
1.0000003106	1.0000004207	.0000000004	1.0
1.0000000331	.9999990600	.0000000004	1.0

MINIMUM = (1.000002781, 1.000003590) ; F(MINIMUM) = .0000000004

NUMBER OF ITERATIONS = 56

```
      PROGRAM TEST10 (INPUT, OUTPUT)
C
C     ***************************************************************
C     * THIS PROGRAM WILL SOLVE THE SAME PROBLEM AS IN PROGRAM TEST1R, *
C     * HOWEVER, FOR THE FIRST PART OF THE PROGRAM WHICH GENERATES THE *
C     * INITIAL THREE POINTS, WE WILL NOT CARRY OUT THE LINE-SEARCH    *
C     * ALGORITHM.  INSTEAD WE WILL CONSTRUCT THE THREE POINTS IN THE  *
C     * FOLLOWING WAY:-                                                *
C     *                X(1,1) = X(0,1) ; X(1,2) = X(0,2)               *
C     *                X(K,1) = L1+S*(U1-L1) ; K=1, 2                  *
C     *                X(K,2) = L2+S*(U2-L2) ; K=1, 2                  *
C     * WHERE L1, L2 REPRESENT THE LOWER BOUND OF X1 AND X2, U1, U2    *
C     * REPRESENT THE UPPER BOUND , AND S IS A PSEUDO-RANDOM DEVIATE   *
C     * RECTANGULARLY DISTRIBUTED OVER THE INTERVAL (0,1).             *
C     *                                                                *
C     * MOREOVER, WE WILL USE A FIXED OVER-REFLECTION FACTOR; R = 1.3. *
C     ***************************************************************
C
      INTEGER I, J, K, M, A
      REAL X(0:3,2), L(2), SUM(2), AVE(2), NEWX(2), R, WORST, BEST, B,
     $     W, VALUE(3), VALNEW, TEST(2), E, S(2)
      READ*, N
      DO 10 I=1, N
            READ*, X(0,I)
            X(1,I) = X(0,I)
   10 CONTINUE
C
C     ***********************************************************
C     * L(I), S(I) ARE SOME RANDOM NUMBERS BETWEEN (0,1); I=1,2 *
C     ***********************************************************
C
      DO 20 I=1, N
            READ*, L(I), S(I)
   20 CONTINUE
      DO 30 K=2, N+1
            J = K-1
            X(K,1) = -3+L(J)*(3-(-3))
   30 CONTINUE
      DO 40 K=2, N+1
            J = K-1
            X(K,2) = -1.5+S(J)*(4.5-(-1.5))
   40 CONTINUE
      PRINT 50
   50 FORMAT (5(/), 8X, 'X(K,1)', 15X, 'X(K,2)', 13X,
     $        'F(X(K,1),X(K,2))')
      DO 70 K=1, N+1
            PRINT 60, X(K,1), X(K,2), F(X(K,1),X(K,2))
   60       FORMAT (4X, F12.9, 10X, F11.9, 10X, F16.10)
   70 CONTINUE
C
C     ***************************************************************
C     * NOW WE HAVE GENERATED THE INITIAL THREE POINTS.  THE REST OF THIS *
C     * PROGRAM WILL BE THE SAME AS IN PROGRAM TEST1R, EXCEPT HERE WE DO  *
C     * NOT HAVE THE REAL FUNCTION FACTOR.                                *
C     ***************************************************************
```

```
C
      PRINT 80
80    FORMAT (5(/), 7X, 'NEWX(1)', 14X, 'NEWX(2)', 13X, 'VALNEW',
     $        7X, 'REFLECTION FACTOR')
C
C     ****************************************************************
C
      A = 0
      READ*, R, E
90    DO 100 K=1, N+1
            VALUE(K) = F(X(K,1),X(K,2))
100   CONTINUE
      WORST = VALUE(1)
      W = 1
      DO 110 K=2, N+1
            IF (VALUE(K).GT.WORST) THEN
               WORST = VALUE(K)
               W = K
            ENDIF
110   CONTINUE
      TEST(1) = WORST
C
C     ****************************************************************
C
      DO 130 I=1, N
            SUM(I) = 0
            DO 120 K=1, N+1
                  IF (K.NE.W) THEN
                     SUM(I) = SUM(I)+X(K,I)
                  ENDIF
120         CONTINUE
            AVE(I) = SUM(I)/N
            NEWX(I) = AVE(I)+R*(AVE(I)-X(W,I))
130   CONTINUE
140   VALNEW = F(NEWX(1),NEWX(2))
      IF ((C1(NEWX(1)).EQ.1).AND.(C2(NEWX(2)).EQ.1)) THEN
         DO 160 K=1, N+1
               IF ((K.NE.W).AND.(VALNEW.LT.VALUE(K))) THEN
                  DO 150 I=1, N
                        X(W,I) = NEWX(I)
150               CONTINUE
                  GO TO 180
               ENDIF
160      CONTINUE
      ENDIF
      DO 170 I=1, N
            NEWX(I) = 0.5*(AVE(I)+NEWX(I))
170   CONTINUE
      GO TO 140
C
C     ****************************************************************
C
180   TEST(2) = VALNEW
      A = A+1
      PRINT 190, NEWX(1), NEWX(2), VALNEW, R
190   FORMAT (4X, F12.9, 9X, F11.9, 6X, F16.10, 11X, F3.1)
C
```

```
         C     ************************************************************
         C
               IF ((TEST(1)-TEST(2)).LT.E) THEN
 1                DO 200 K=1, N+1
 1                      VALUE(K) = F(X(K,1),X(K,2))
 1       200      CONTINUE
 1                BEST = VALUE(1)
 1                B = 1
 1                DO 210 K=2, N+1
 1                      IF (VALUE(K).LT.BEST) THEN
 2                         BEST = VALUE(K)
 2                         B = K
 2                      ENDIF
 1       210      CONTINUE
 1                PRINT 220, X(B,1), X(B,2), BEST
 1       220      FORMAT (5(/), 6X, 'MINIMUM = ( ', F11.9, ', ', F11.9,
 1            $          ')'  ,  ',  'F(MINIMUM) = ', F12.10)
 1                PRINT*, ' '
 1                PRINT 230, A
 1       230      FORMAT (6X, 'NUMBER OF ITERATIONS = ', I3)
 1             ELSE
 1                GO TO 90
 1             ENDIF
               STOP
               END
```

--VARIABLE MAP--(LO=A)

-NAME	---ADDRESS	--BLOCK	-----PROPERTIES	-------TYPE	---------SIZE
A	731B			INTEGER	
AVE	740B			REAL	2
B	755B			REAL	
BEST	754B			REAL	
E	765B			REAL	
I	725B			INTEGER	
J	726B			INTEGER	
K	727B			INTEGER	
L	742B			REAL	2
N	730B			INTEGER	

-NAME	---ADDRESS	--BLOCK	-----PROPERTIES	-------TYPE	---------SIZE
NEWX	750B			REAL	2
R	752B			REAL	
S	766B			REAL	2
SUM	744B			REAL	2
TEST	763B			REAL	2
VALNEW	762B			REAL	
VALUE	757B			REAL	3
W	756B			REAL	
WORST	753B			REAL	
X	732B			REAL	6

--PROCEDURES--(LO=A)

-NAME	------TYPE	--------ARGS	------CLASS-----
C1	REAL	1	FUNCTION
C2	REAL	1	FUNCTION
F	REAL	2	FUNCTION

--STATEMENT LABELS--(LO=A)

-LABEL-	-ADDRESS-----	-PROPERTIES----	-DEF
10	INACTIVE	DO-TERM	26
20	INACTIVE	DO-TERM	34
30	INACTIVE	DO-TERM	38
40	INACTIVE	DO-TERM	42
50	563B	FORMAT	44
60	562B	FORMAT	48
70	INACTIVE	DO-TERM	49
80	567B	FORMAT	50
90	177B		65
100	INACTIVE	DO-TERM	67
110	INACTIVE	DO-TERM	75
120	INACTIVE	DO-TERM	86
130	INACTIVE	DO-TERM	89
140	327B		90
150	INACTIVE	DO-TERM	96
160	INACTIVE	DO-TERM	99
170	INACTIVE	DO-TERM	103
180	420B		106
190	577B	FORMAT	111
200	INACTIVE	DO-TERM	118
210	INACTIVE	DO-TERM	126
220	604B	FORMAT	128
230	615B	FORMAT	132

--ENTRY POINTS--(LO=A)

-NAME---	-ADDRESS--	-ARGS---
TESTIO	14B	0

--STATISTICS--

PROGRAM-UNIT LENGTH	1000B = 512
CM STORAGE USED	63300B = 26304
COMPILE TIME	0.700 SECONDS

```
  FUNCTION F           74/810  OPT=0,ROUND= A/ S/ M/-D,-DS      FTN 5.1+642           87/03/01. 15.03.28
 DO=-LONG/-OT,ARG=-COMMON/-FIXED,CS= USER/-FIXED,DB=-TB/-SB/-SL/ ER/-ID/-PMD/-ST,-AL,PL=5000
 FTN5,I=PROG11.

                    REAL FUNCTION F(X1,X2)
                    REAL X1, X2
                    F = 100*((X2-X1**2)**2)+(X1-1)**2
                    RETURN
                    END

--VARIABLE MAP--(LO=A)
 -NAME---ADDRESS --BLOCK-----PROPERTIES-------TYPE---------SIZE

  F          25B                               REAL
  X1           1   DUMMY-ARG                   REAL
  X2           2   DUMMY-ARG                   REAL

--ENTRY POINTS--(LO=A)
 -NAME---ADDRESS--ARGS---

  F          7B     2

--STATISTICS--

  PROGRAM-UNIT LENGTH           30B =  24
  CM STORAGE USED           63300B =  26304
  COMPILE TIME              0.046 SECONDS
```

```
FUNCTION C1          74/810  OPT=0,ROUND= A/ S/ M/-D,-DS      FTN 5.1+642           87/03/01. 15.03.28
DO=-LONG/-OT,ARG=-COMMON/-FIXED,CS= USER/-FIXED,DB=-TB/-SB/-SL/ ER/-ID/-PMD/-ST,-AL,PL=5000
FTN5,I=PROB11.

                  REAL FUNCTION C1(X1)
                  REAL X1
                  IF ((X1.GE.-3).AND.(X1.LE.3)) THEN
                     C1 = 1
                  ELSE
                     C2 = 0
                  ENDIF
                  RETURN
                  END

--VARIABLE MAP--(LO=A)
 -NAME---ADDRESS --BLOCK-----PROPERTIES-------TYPE---------SIZE

  C1        32B                                REAL
  C2        33B                 *S*            REAL
  X1         1    DUMMY-ARG                    REAL

--ENTRY POINTS--(LO=A)
 -NAME---ADDRESS--ARGS---

  C1        7B     1

--STATISTICS--

  PROGRAM-UNIT LENGTH           36B =  30
  CM STORAGE USED           633008 =  26304
  COMPILE TIME              0.049 SECONDS
```

```
      FUNCTION C2         74/810  OPT=0,ROUND= A/ S/ M/-D,-DS      FTN 5.1+642         87/03/01. 15.03.28
     DO=-LONG/-OT,ARG=-COMMON/-FIXED,CS= USER/-FIXED,DB=-TB/-SB/-SL/ ER/-ID/-PMD/-ST,-AL,PL=5000
     FTN5,I=PROG11.

                         REAL FUNCTION C2(X2)
                         REAL X2
                         IF ((X2.GE.-1.5).AND.(X2.LE.4.5)) THEN
                            C2 = 1
                         ELSE
                            C2 = 0
                         ENDIF
                         RETURN
                         END

--VARIABLE MAP--(LO=A)
 -NAME---ADDRESS --BLOCK------PROPERTIES--------TYPE----------SIZE

  C2          33B                                REAL
  X2            1   DUMMY-ARG                    REAL

--ENTRY POINTS--(LO=A)
 -NAME---ADDRESS--ARGS---

  C2          7B      1

--STATISTICS--

  PROGRAM-UNIT LENGTH            36B =  30
  CM STORAGE USED             63300B =  26304
  COMPILE TIME                0.045 SECONDS
```

```
2
-1.2
1
0.498    0.336
0.791    0.469
1.3
0.0000000001
```

X(K,1)	X(K,2)	F(X(K,1),X(K,2))
-1.200000000	1.000000000	24.2000000000
-.030000000	.510000000	26.9791810000
1.746000000	1.314000000	301.4110914256

NEWX(1)	NEWX(2)	VALNEW	REFLECTION FACTOR
-.998662500	.664162500	15.0944961291	1.3
-1.121052041	.838623525	21.9824797742	1.3
-.877871722	.428203929	15.2291468043	1.3
-.789416702	.160010811	13.4261637129	1.3
-.812987844	.398034199	10.1975741082	1.3
-.599411625	.033631508	13.1637427299	1.3
-.713804877	.280801510	8.1097983151	1.3
-.978426785	.738710108	8.6021155383	1.3
-.888781484	.852463897	5.3533341457	1.3
-.571192784	.115302082	6.9190205397	1.3
-.748867308	.818148914	3.2404874973	1.3
-.978134858	.890808218	4.3478101107	1.3
-.833236871	.772098783	3.9882808424	1.3
-.547844132	.325732870	2.4813558051	1.3
-.528183100	.194338263	3.0517217200	1.3
-.400860718	.092281891	2.4319925591	1.3
-.404485247	.228054234	2.3879084520	1.3
-.308392585	.052534437	1.8931248355	1.3
-.327801045	.171515424	2.1751743754	1.3
-.205446853	-.038813183	2.1095518373	1.3
-.281184900	-.008517984	2.0500394298	1.3
-.378433298	.103378818	2.0414531087	1.3
-.372062027	.105400215	1.9915924321	1.3
-.293180818	.047245729	1.8220055552	1.3
-.208104281	-.022284788	1.8897493588	1.3
-.175843007	-.039589686	1.8775882275	1.3
-.268472338	.037774674	1.7268897314	1.3
-.315029625	.089182665	1.8190091685	1.3
-.289919083	-.061592992	1.7143409199	1.3
-.232612540	.024322362	1.6080556941	1.3
-.251898250	.049895569	1.5861751723	1.3
-.180291560	.005049719	1.4684678688	1.3
-.194821981	.031338024	1.4313984323	1.3
-.103882847	-.022758335	1.3303975663	1.3
-.108871523	.003302008	1.2363902875	1.3
-.048880088	-.036421186	1.7502263688	1.3
-.048175820	-.008501219	1.1057908385	1.3
-.098101821	.019384421	1.2117391403	1.3
-.022346316	.008223073	1.0511575678	1.3
.048131901	-.025519818	.9863042617	1.3
.087301898	-.008839441	.8600143580	1.3
.095715811	-.025435374	.9374245212	1.3
.150980666	-.006240535	.8051217749	1.3
.149230399	.015724014	.7280830454	1.3

.231206825	.022397273	.6875111013	1.3
.241744143	.051952175	.5791614310	1.3
.293179590	.051117686	.6209538643	1.3
.314603420	.089413886	.4789122390	1.3
.268411756	.083400505	.5481164007	1.3
.356188566	.131190721	.4163666898	1.3
.422464001	.145203842	.4437188538	1.3
.486479007	.201716895	.3858188135	1.3
.419864507	.193983965	.3678782606	1.3
.518210831	.241174041	.2980588409	1.3
.458050600	.227889373	.3355097579	1.3
.529203918	.260887739	.2583941829	1.3
.528123220	.252911138	.2902832819	1.3
.544852128	.277342458	.2452680272	1.3
.548604262	.290180244	.2153927368	1.3
.569509755	.313497045	.1970817714	1.3
.577523356	.333683690	.1784887787	1.3
.605902537	.367023528	.1553137003	1.3
.620577096	.398267141	.1612571700	1.3
.659671215	.448295471	.1282099430	1.3
.648868891	.417569564	.1244575024	1.3
.718007128	.518314205	.0808977681	1.3
.712829149	.493782223	.1030397574	1.3
.800939251	.618770459	.0913051291	1.3
.818845444	.663430475	.0378259122	1.3
.757130823	.580480451	.0642175987	1.3
.880393439	.759289098	.0392808183	1.3
.857134141	.721993732	.0365020932	1.3
.824206607	.671074580	.0377010255	1.3
.854858229	.718061549	.0372740790	1.3
.857287558	.722011156	.0370873937	1.3
.860200253	.727138607	.0362291611	1.3
.860540081	.727887686	.0354299320	1.3
.864656351	.734688387	.0350680053	1.3
.866625868	.738882287	.0339978079	1.3
.871122447	.745822293	.0335927898	1.3
.873207306	.749785375	.0322196742	1.3
.880668586	.762261832	.0319564272	1.3
.884494646	.769285307	.0303599155	1.3
.894764709	.780558222	.0308028172	1.3
.901283063	.798279677	.0294332859	1.3
.890450166	.780174043	.0281999744	1.3
.910650175	.815150879	.0279571686	1.3
.899597409	.798860080	.0254949387	1.3
.924199506	.839586348	.0269403552	1.3
.913521226	.822217249	.0226160752	1.3
.883627070	.770478677	.0241931516	1.3
.897243808	.795679911	.0193323584	1.3
.933664711	.858962053	.0207018859	1.3
.917967320	.833955835	.0143125750	1.3
.890687201	.786124155	.0170858179	1.3
.913512618	.820708105	.0108408189	1.3
.940334568	.890102128	.0112002841	1.3
.947768748	.892409184	.0080606334	1.3
.987636334	.822244114	.0087742821	1.3
.948147148	.894622756	.0029328095	1.3
.972515341	.939908676	.0042098052	1.3
.974368081	.949475208	.0006579218	1.3
.968271861	.916244581	.0022320983	1.3
.988241201	.987568175	.0004914588	1.3
1.018143686	1.030731935	.0005905163	1.3
1.021938327	1.043813705	.0005098058	1.3
1.082810081	1.087498117	.0003495861	1.3
.984466197	.981806401	.0004734724	1.3
1.008568851	1.021890218	.0004107845	1.3
1.008886871	1.021778801	.0003823784	1.3

1.002308261	1.000358732	.0003063123	1.3
.997729997	.997260216	.0003208439	1.3
.996382820	.994434908	.0002874049	1.3
1.001447300	1.004462797	.0002473621	1.3
.994502590	.990466101	.0002349086	1.3
1.000044710	1.001402749	.0001724805	1.3
.991847924	.994847541	.0001842304	1.3
.997823190	.996581902	.0000913857	1.3
1.000145758	1.017300844	.0001709397	1.3
1.000006122	1.014233240	.0000793482	1.3
.994734188	.999642705	.0000397420	1.3
1.004001249	1.009130864	.0000220328	1.3
.995027756	.999901589	.0000279147	1.3
1.001408605	1.002860121	.0000098138	1.3
1.003083888	1.007194997	.0000170436	1.3
1.001257520	1.002196411	.0000116989	1.3
1.000008808	1.000078418	.0000118788	1.3
1.000748498	1.001215283	.0000085280	1.3
1.001757370	1.003314774	.0000072114	1.3
1.000078461	1.001781409	.0000050827	1.3
1.002170858	1.004246243	.0000057164	1.3
1.001334836	1.002580092	.0000024767	1.3
.999836876	.999470311	.0000040859	1.3
1.000078021	1.000090332	.0000004093	1.3
1.001271300	1.002553998	.0000016257	1.3
.999815258	.998678400	.0000002449	1.3
.999515370	.999015451	.0000002690	1.3
.999808428	.999224094	.0000002396	1.3
.999714822	.999488410	.0000002342	1.3
.999630731	.999293117	.0000002357	1.3
.999806807	.999431369	.0000002339	1.3
.999707237	.999453084	.0000002341	1.3
.999883832	.999425223	.0000002341	1.3
.999881446	.999420253	.0000002341	1.3

MINIMUM = (.999806807, .999431369) ; F(MINIMUM) = .0000002339

NUMBER OF ITERATIONS = 144

```
PROGRAM TESTZR      74/810  OPT=0,ROUND= A/ S/ M/-D,-DS      FTN 5.1+642         87/03/04. 13.48.52
DO=-LONG/-OT,ARG=-COMMON/-FIXED,CS= USER/-FIXED,DB=-TB/-SB/-SL/ ER/-ID/-PMD/-ST,-AL,PL=5000
FTN5,I=PROG2.

      PROGRAM TESTZR (INPUT, OUTPUT)
C
C     *******************************************************************
C     *  OBJECTIVE FUNCTION : F(X1,X2,X3) = 2*(X1**2)+2*(X2**2)+X3**2+  *
C     *                                     2*X1*X2+2*X1*X3-8*X1-6*X2-  *
C     *                                     4*X3+9                      *
C     *  CONSTRAINTS        : X1.GE.0                                   *
C     *                       X2.GE.0                                   *
C     *                       X3.GE.0                                   *
C     *                       -X1-X2-2*X3+3.GE.0                        *
C     *  MINIMUM            : MIN = (4/3,7/9,4/9)  ; F(4/3,7/9,4/9) = 1/9  *
C     *  START              : X(0) = (0.1,0.1,0.1) ; F(0.1,0.1,0.1) = 7.29 *
C     *******************************************************************
C
      INTEGER I, K, L, J, N, A
      REAL X(0:4,3), S, SUM(3), AVE(3), NEWX(3), R, WORST, BEST, B, W,
     $     VALUE(4), VALNEW, D(3), E, TEST(2), SEARCH(15), BNEW(3)
      READ*, N
C
C     *******************************************************************
C     *  DIMENSION IN THIS PROBLEM IS THREE AND WE WILL USE FOUR POINTS TO *
C     *  CONSTRUCT OUR POLYHEDRON.                                      *
C     *******************************************************************
C
      DO 10 I=1, N
            READ*, X(0,I)
            X(1,I) = X(0,I)
   10 CONTINUE
C
C     *********************************************************
C     *  WE WILL USE A STEPSIZE OF 0.1 TO CARRY OUT THE LINE-SEARCH  *
C     *  ALGORITHM IN EACH COORDINATE DIRECTION.              *
C     *********************************************************
C
      DO 110 K=2, N+1
            S = 0.1
            J = K-1
            DO 20 I=1, N
                  IF ((J-I).EQ.0) THEN
                        X(K,I) = 0
                  ELSE
                        X(K,I) = X(J,I)
                  ENDIF
   20       CONTINUE
            SEARCH(1) = F(X(K,1),X(K,2),X(K,3))
            BEST = SEARCH(1)
            DO 30 I=1, N
                  BNEW(I) = X(K,I)
   30       CONTINUE
            DO 80 L=2, 15
                  DO 40 I=1, N
                        IF ((J-I).EQ.0) THEN
                              X(K,I) = X(K,I)+S
                        ELSE
                              X(K,I) = X(J,I)
```

```
                    ENDIF
40             CONTINUE
               SEARCH(L) = F(X(K,1),X(K,2),X(K,3))
               IF (SEARCH(L).LT.BEST) THEN
                  BEST = SEARCH(L)
                  DO 50 I=1, N
                        BNEW(I) = X(K,I)
50                CONTINUE
               ENDIF
60          CONTINUE
            DO 70 I=1, N
               X(K,I) = BNEW(I)
70          CONTINUE
80          IF (C(X(K,1),X(K,2),X(K,3)).LT.0) THEN
              DO 100 I=1, N
                   SUM(I) = 0
                   DO 90 L= 1, J
                         SUM(I) = SUM(I)+X(L,I)
90                 CONTINUE
                   X(K,I) = 0.5*(X(K,I)+SUM(I)/J)
100           CONTINUE
              GO TO 80
            ENDIF
110   CONTINUE
C
C     ******************************************************************
C
      PRINT 120
120   FORMAT (5(/), 8X, 'X(K,1)', 11X, 'X(K,2)', 10X, 'X(K,3)', 7X,
     $         'F(X(K,1),X(K,2),X(K,3))')
      DO 140 K=1, N+1
               PRINT 130, X(K,1), X(K,2), X(K,3), F(X(K,1),X(K,2),X(K,3))
130            FORMAT (5X, F11.8, 5X, F11.8, 5X, F11.8, 8X, F14.10)
140   CONTINUE
      PRINT 150
150   FORMAT (5(/), 6X, 'NEWX(1)', 7X, 'NEWX(2)', 7X, 'NEWX(3)', 8X,
     $         'VALNEW', 7X, 'REFLEXION FACTOR')
C
C     ******************************************************************
C
      A = 0
      READ*, E
160   DO 170 K=1, N+1
               VALUE(K) = F(X(K,1),X(K,2),X(K,3))
170   CONTINUE
      WORST = VALUE(1)
      W = 1
      DO 180 K=2, N+1
               IF (VALUE(K).GT.WORST) THEN
                  WORST = VALUE(K)
                  W = K
               ENDIF
180   CONTINUE
      TEST(1) = WORST
      DO 200 I=1, N
               SUM(I) = 0
               DO 190 K=1, N+1
```

```
                  IF (K.NE.W) THEN
                     SUM(I) = SUM(I)+X(K,I)
                  ENDIF
190           CONTINUE
              AVE(I) = SUM(I)/N
              D(I) = AVE(I)-X(W,I)
200     CONTINUE
        R = FACTOR(AVE(1),AVE(2),AVE(3),D(1),D(2),D(3))
        DO 210 I=1, N
              NEWX(I) = AVE(I)+R*D(I)
210     CONTINUE
C
C       ******************************************************************
C
220     VALNEW = F(NEWX(1),NEWX(2),NEWX(3))
        IF (C(NEWX(1),NEWX(2),NEWX(3)).GE.0) THEN
           DO 240 K=1, N+1
                IF ((K.NE.W).AND.(VALNEW.LT.VALUE(K))) THEN
                   DO 230 I=1, N
                        X(W,I) = NEWX(I)
230                CONTINUE
                   GO TO 260
                ENDIF
240        CONTINUE
        ENDIF
        DO 250 I=1, N
              NEWX(I) = 0.5*(AVE(I)+NEWX(I))
250     CONTINUE
        GO TO 220
C
C       ******************************************************************
C
260     TEST(2) = VALNEW
        A = A+1
        PRINT 270, NEWX(1), NEWX(2), NEWX(3), VALNEW, R
270     FORMAT (3X, F11.9, 3X, F11.9, 3X, F11.9, 2X, F14.10, 11X, F3.1)
C
C       ******************************************************************
C
        IF ((TEST(1)-TEST(2)).LT.E) THEN
           DO 280 K=1, N+1
                VALUE(K) = F(X(K,1),X(K,2),X(K,3))
280        CONTINUE
           BEST = VALUE(1)
           B = 1
           DO 290 K=2, N+1
                IF (VALUE(K).LT.BEST) THEN
                   BEST = VALUE(K)
                   B = K
                ENDIF
290        CONTINUE
           PRINT 300, X(B,1), X(B,2), X(B,3), BEST
300        FORMAT (5(/), 6X, 'MINIMUM = ( ', F11.9, ', ', F11.9, ', ',
     $          F11.9, ')    ;  ', 'F(MINIMUM) = ', F14.10)
           PRINT*, ' '
           PRINT 310, A
310        FORMAT (4X, 'NUMBER OF ITERATIONS = ', I3)
```

```
     170        ELSE
     171           GO TO 100
     172        ENDIF
     173        STOP
     174        END

--VARIABLE MAP--(LO=A)
-NAME---ADDRESS --BLOCK-----PROPERTIES-------TYPE----------SIZE    -NAME---ADDRESS --BLOCK-----PROPERTIES-------TYPE----------SIZE

 A        12120                            INTEGER                  NEWX      12410                             REAL          3
 AVE      12300                            REAL           3         R         12440                             REAL
 B        12470                            REAL                     S         12370                             REAL
 BEST     12400                            REAL                     SEARCH    12640                             REAL         18
 BNEW     13020                            REAL           3         SUM       12330                             REAL          3
 D        12500                            REAL           3         TEST      12620                             REAL          2
 E        12610                            REAL                     VALNEW    13550                             REAL
 I        12050                            INTEGER                  VALUE     12510                             REAL          4
 J        17100                            INTEGER                  W         12500                             REAL
 K        12000                            INTEGER                  WORST     12450                             REAL
 L        12070                            INTEGER                  X         17130                             REAL         18
 N        12110                            INTEGER
                                           INTEGER

--PROCEDURES--(LO=A)
-NAME------TYPE---------ARGS------CLASS-----

 C         REAL           3      FUNCTION
 F         REAL           3      FUNCTION
 FACTOR    REAL           6      FUNCTION

--STATEMENT LABELS--(LO=A)
-LABEL-ADDRESS-----PROPERTIES----DEF      -LABEL-ADDRESS-----PROPERTIES----DEF      -LABEL-ADDRESS-----PROPERTIES----DEF

   10  INACTIVE   DO-TERM        28          120   10220     FORMAT         84         220    6000                     127
   20  INACTIVE   DO-TERM        44          130   10330     FORMAT         88         230  INACTIVE   DO-TERM         133
   30  INACTIVE   DO-TERM        49          140  INACTIVE   DO-TERM        89         240  INACTIVE   DO-TERM         136
   40  INACTIVE   DO-TERM        57          150   10400     FORMAT         91         250  INACTIVE   DO-TERM         140
   50  INACTIVE   DO-TERM        63          160    4260                    96         260    6710                     145
   60  INACTIVE   DO-TERM        65          170  INACTIVE   DO-TERM       100         270   10510     FORMAT          148
   70  INACTIVE   DO-TERM        68          180  INACTIVE   DO-TERM       108         280  INACTIVE   DO-TERM         155
   80     7000                   69          190  INACTIVE   DO-TERM       118         290  INACTIVE   DO-TERM         163
   90  INACTIVE   DO-TERM        74          200  INACTIVE   DO-TERM       119         300   10570     FORMAT          165
  100  INACTIVE   DO-TERM        78          210  INACTIVE   DO-TERM       123         310   10710     FORMAT          169
  110  INACTIVE   DO-TERM        79

--ENTRY POINTS--(LO=A)
-NAME---ADDRESS--ARGS---

 TEST2R     140     0

--STATISTICS--

 PROGRAM-UNIT LENGTH         13330 =   731
 CM STORAGE USED            053000 = 27328
 COMPILE TIME               1.024 SECONDS
```

```
FUNCTION F          74/810  OPT=0,ROUND= A/ S/ M/-D,-DS     FTN 5.1+642          87/03/04. 13.48.52
DO=-LONG/-OT,ARG=-COMMON/-FIXED,CS= USER/-FIXED,DB=-TB/-SB/-SL/ ER/-ID/-PMD/-ST,-AL,PL=5000
FTNB,I=PROG2.

                          REAL FUNCTION F(X1,X2,X3)
                          REAL X1, X2, X3
                          F = 2*(X1**2)+2*(X2**2)+X3**2+2*X1*X2+2*X1*X3-8*X1-6*X2-4*X3+9
                          RETURN
                          END

--VARIABLE MAP--(LO=A)
 -NAME---ADDRESS --BLOCK-----PROPERTIES-------TYPE----------SIZE

  F          30B                                  REAL
  X1          1    DUMMY-ARG                      REAL
  X2          2    DUMMY-ARG                      REAL
  X3          3    DUMMY-ARG                      REAL

--ENTRY POINTS--(LO=A)
 -NAME---ADDRESS--ARGS---

  F          7B     3

--STATISTICS--

  PROGRAM-UNIT LENGTH               41B =  33
  CM STORAGE USED               83300B =  26304
  COMPILE TIME                   0.073 SECONDS
```

```
   FUNCTION C            74/810  OPT=0,ROUND= A/ S/ M/-D,-DS      FTN 5.1+642          07/03/84. 13.46.52
  DO=-LONG/-OT,ARG=-COMMON/-FIXED,CS= USER/-FIXED,DB=-TB/-SB/-SL/ ER/-ID/-PMD/-ST,-AL,PL=5000.
  FTNB,I=PROG2.

                    REAL FUNCTION C(X1,X2,X3)
                    REAL X1, X2, X3
                    C = -X1-X2-2*X3+3
                    RETURN
                    END

--VARIABLE MAP--(LO=A)
 -NAME---ADDRESS --BLOCK-----PROPERTIES--------TYPE---------SIZE

  C          24B                             REAL
  X1           1   DUMMY-ARG                 REAL
  X2           2   DUMMY-ARG                 REAL
  X3           3   DUMMY-ARG                 REAL

--ENTRY POINTS--(LO=A)
 -NAME---ADDRESS--ARGS---

  C          7B     3

--STATISTICS--

  PROGRAM-UNIT LENGTH           27B =   23
  CM STORAGE USED            63300B = 26304
  COMPILE TIME               0.047 SECONDS
```

```
   FUNCTION FACTOR      74/810  OPT=0,ROUND= A/ S/ M/-D,-DS      FTN 5.1+642          87/03/04. 13.48.52
DO=-LONG/-OT,ARG=-COMMON/-FIXED,CS= USER/-FIXED,DB=-TB/-SB/-SL/ ER/-ID/-PMD/-ST,-AL,PL=5000
FTN5,I=PROG2.

                    REAL FUNCTION FACTOR(AVE1,AVE2,AVE3,D1,D2,D3)
                    REAL AVE1, AVE2, AVE3, D1, D2, D3, S, R, F(2)
                    INTEGER I
                    R = 1
                    F(1) = 2*((AVE1+R*D1)**2)+2*((AVE2+R*D2)**2)+(AVE3+R*D3)**2+
                   $       2*(AVE1+R*D1)*(AVE2+R*D2)+2*(AVE1+R*D1)*(AVE3+R*D3)-
                   $       B*(AVE1+R*D1)-B*(AVE2+R*D2)-4*(AVE3+R*D3)+9
                    S = 0.1
                    DO 10 I=2, 50
                          R = 1+(I-1)*S
                          F(2) = 2*((AVE1+R*D1)**2)+2*((AVE2+R*D2)**2)+
                   $             (AVE3+R*D3)**2+2*(AVE1+R*D1)*(AVE2+R*D2)+
                   $             2*(AVE1+R*D1)*(AVE3+R*D3)-B*(AVE1+R*D1)-
                   $             B*(AVE2+R*D2)-4*(AVE3+R*D3)+9
                          IF (F(2).GT.F(1)) THEN
                             R = R-S
                             GO TO 20
                          ELSE
                             F(1) = F(2)
                          ENDIF
               10   CONTINUE
               20   FACTOR = R
                    RETURN
                    END

--VARIABLE MAP--(LO=A)
-NAME---ADDRESS --BLOCK-----PROPERTIES-------TYPE---------SIZE     -NAME---ADDRESS --BLOCK-----PROPERTIES-------TYPE---------SIZE

 AVE1         1   DUMMY-ARG                  REAL                    F         127B                                  REAL          2
 AVE2         2   DUMMY-ARG                  REAL                    FACTOR    124B                                  REAL
 AVE3         3   DUMMY-ARG                  REAL                    I         131B                                  INTEGER
 D1           4   DUMMY-ARG                  REAL                    R         126B                                  REAL
 D2           5   DUMMY-ARG                  REAL                    S         125B                                  REAL
 D3           6   DUMMY-ARG                  REAL

--STATEMENT LABELS--(LO=A)
 -LABEL-ADDRESS-----PROPERTIES----DEF

    10  INACTIVE   DO-TERM      21
    20    110B                  22

--ENTRY POINTS--(LO=A)
 -NAME---ADDRESS--ARGS---

 FACTOR       7B     6

--STATISTICS--

 PROGRAM-UNIT LENGTH          135B =  93
 CM STORAGE USED            63300B = 26304
 COMPILE TIME               0.215 SECONDS
```

3
0.1
0.1
0.1
0.0000000001

X(K,1)	X(K,2)	X(K,3)	F(X(K,1),X(K,2),X(K,3))
.100000000	.100000000	.100000000	7.2900000000
1.400000000	.100000000	.100000000	1.3100000000
1.400000000	.800000000	.100000000	.3300000000
1.183333333	.566666667	.350000000	.4680555556

NEWX(1)	NEWX(2)	NEWX(3)	VALNEW	REFLEXION FACTOR
1.634722222	.506111111	.204166667	.2460281636	1.0
1.400027778	.926388889	.277083333	.2164491705	1.0
1.558729167	.621875000	.154687500	.2581719293	1.0
1.599739583	.767187500	.267968750	.2063307297	1.0
1.535980903	.666927083	.302317708	.1572503078	1.5
1.499940321	.811924913	.325992839	.1679314276	1.0
1.613310515	.866825304	.354597982	.1913547565	1.0
1.537247498	.899310667	.380053033	.1537120158	1.0
1.390091855	.818781105	.382253689	.1281187153	1.4
1.400815025	.787440828	.399695303	.1306583668	1.0
1.414655347	.816770517	.382283642	.1293647483	1.0
1.331666656	.862880167	.390035390	.1307745354	1.0
1.368720089	.848400092	.385205357	.1306473066	1.0
1.418115366	.819312339	.380075806	.1293049309	1.0
1.427980249	.803220935	.379673790	.1291725752	1.0
1.437089237	.775891027	.389100802	.1258418189	5.8
1.355154130	.807267379	.413306893	.1171373695	5.9
1.366849256	.771823280	.417577968	.1178021912	1.8
1.373243233	.751073352	.431073346	.1157110905	1.0
1.328773096	.777036493	.436475258	.11[illegible]00532	1.0
1.341729916	.783500473	.433835528	.1129261643	5.0
1.344294053	.752171409	.444014115	.1155370248	1.0
1.377402973	.747460371	.436190063	.1141146751	1.0
1.326322436	.776000232	.446423063	.1113854148	3.0
1.349479057	.772980720	.438406714	.1115265967	1.0
1.320082718	.782810527	.442237617	.1122257478	1.0
1.329004920	.778802644	.445140945	.1111727777	3.7
1.337053315	.774230032	.444211509	.1112011128	1.0
1.321795807	.778520094	.449887431	.1113350851	1.0
1.330831482	.778146325	.445492953	.1111259741	3.3
1.337891955	.778826453	.442582018	.1111584218	1.0
1.330682521	.780272695	.444506173	.1[illegible]317213	1.0
1.334339459	.777929538	.443859634	.1111152603	1.4
1.330468954	.779271952	.445128562	.1111211003	1.0
1.333075409	.778625849	.445148926	.1111148232	1.0
1.332918703	.777909318	.444585530	.1111114882	1.3
1.334189166	.777042305	.444382063	.1111132775	1.0
1.333276297	.777100389	.444811240	.1111123974	1.0
1.333847368	.778079471	.444036963	.1111125888	1.0
1.332920601	.778021421	.444525835	.1111113715	1.0
1.332830625	.777577422	.444791845	.1111118700	1.0
1.332783918	.778203900	.444545905	.1111116805	1.0
1.332852468	.778161743	.444492891	.1111115855	1.0
1.333114133	.777840446	.444522193	.1111113898	1.1
1.333128489	.777685712	.444598437	.1111114218	1.0

1.333458856	.777858823	.444438031	.1111111944	3.0
1.333212570	.778001414	.444392930	.1111112334	1.0
1.333218514	.777909093	.444435717	.1111111569	3.0
1.333481670	.777779405	.444369425	.1111111558	1.0
1.333473235	.777878404	.444424119	.1111111844	1.0
1.333374211	.777819378	.444403184	.1111111291	1.0
1.333300580	.777815187	.444392104	.1111111540	1.0
1.333468973	.777801988	.444364498	.1111111500	1.0
1.333331047	.777878574	.444395180	.1111111376	1.0
1.333482240	.777751441	.444387138	.1111111441	1.0
1.333308064	.777833836	.444429038	.1111111222	1.2
1.333301657	.777867052	.44441563[illegible]	.1111111315	1.0
1.333324601	.777797754	.444438801	.1111111216	1.1
1.333309593	.777766927	.444431715	.1111111234	1.0
1.333293961	.777779633	.444463185	.1111111209	1.0
1.333278517	.777822148	.444449854	.1111111217	1.0
1.333298767	.777791347	.444455924	.1111111210	1.0
1.333311758	.777781436	.444453382	.1111111210	1.0
1.333297857	.777782437	.444459834	.1111111210	1.0
1.333292304	.777785232	.444461214	.1111111209	1.0

MINIMUM = (1.333293961, .777779633, .444463185) ; F(MINIMUM) = .1111111209

NUMBER OF ITERATIONS = 85

```
      PROGRAM TEST20      74/810  OPT=O,ROUND= A/ S/ M/-D,-DS      FTN 5.1+642        87/03/04. 13.44.48
DO=-LONG/-OT,ARG=-COMMON/-FIXED,CS= USER/-FIXED,DB=-TB/-SB/-SL/ ER/-ID/-PMD/-ST,-AL,PL=5000
FTN5,I=PROG22.

            PROGRAM TEST20 (INPUT, OUTPUT)
C
C     ************************************************************
C     *  THIS PROGRAM WILL SOLVE THE SAME PROBLEM AS IN PROGRAM TEST2R.  *
C     ************************************************************
C
            INTEGER I, J, K, N, S, A
            REAL X(0:4,3), SUM(3), AVE(3), NEWX(3), R, WORST, BEST, B, W,
     $           VALUE(4), VALNEW, TEST(2), E, L(9)
C
C     ********************************************************
C     *  L(I) IS SOME RANDOM NUMBER BETWEEN (0,1); I =1,2,...,9  *
C     ********************************************************
C
            READ*, N
            DO 10 I=1, N
                  READ*, X(0,I)
                  X(1,I) = X(0,I)
10          CONTINUE
            DO 20 I=1, 3*N
                  READ*, L(I)
20          CONTINUE
C
C     ****************************************************
C     *  WE WILL ASSUME THAT EVERY X LIES BETWEEN (0,1.5).  *
C     ****************************************************
C
            DO 40 I=1, N
                  DO 30 K=2, N+1
                        J = (K-1)*I
                        X(K,I) = L(J)*1.5
30                CONTINUE
40          CONTINUE
            DO 80 K=2, N+1
                  J = K-1
50                IF (C(X(K,1),X(K,2),X(K,3)).LT.0) THEN
                     DO 70 I=1, N
                           SUM(I) = 0
                           DO 60 S=1, J
                                 SUM(I) = SUM(I)+X(S,I)
60                         CONTINUE
                           X(K,I) = 0.5*(X(K,I)+SUM(I)/J)
70                   CONTINUE
                     GO TO 50
                  ENDIF
80          CONTINUE
C
C     ***************************************************************
C
            PRINT 90
90          FORMAT (5(/), 8X, 'X(K,1)', 10X, 'X(K,2)', 10X, 'X(K,3)', 4X,
     $        'F(X(K,1),X(K,2),X(K,3))')
            DO 110 K=1, N+1
                  PRINT 100, X(K,1), X(K,2), X(K,3), F(X(K,1),X(K,2),X(K,3))
100               FORMAT (4X, F11.9, 5X, F11.9, 5X, F11.9, 5X, F14.10)
```

```
110   CONTINUE
      PRINT 120
120   FORMAT (5(/), 8X, 'NEWX(1)', 7X, 'NEWX(2)', 7X, 'NEWX(3)', 8X,
     $        'VALNEW', 7X, 'REFLECTION FACTOR')
C
C     *******************************************************************
C
      A = 0
      READ*, R, E
130   DO 140 K=1, N+1
              VALUE(K) = F(X(K,1),X(K,2),X(K,3))
140   CONTINUE
      WORST = VALUE(1)
      W = 1
      DO 150 K=2, N+1
              IF (VALUE(K).GT.WORST) THEN
                 WORST = VALUE(K)
                 W = K
              ENDIF
150   CONTINUE
      TEST(1) = WORST
      DO 170 I=1, N
              SUM(I) = 0
              DO 160 K=1, N+1
                      IF (K.NE.W) THEN
                         SUM(I) = SUM(I)+X(K,I)
                      ENDIF
160           CONTINUE
              AVE(I) = SUM(I)/N
              NEWX(I) = AVE(I)+R*(AVE(I)-X(W,I))
170   CONTINUE
C
C     *******************************************************************
C
180   VALNEW = F(NEWX(1),NEWX(2),NEWX(3))
      IF (C(NEWX(1),NEWX(2),NEWX(3)).GE.0) THEN
         DO 200 K=1, N+1
                 IF ((K.NE.W).AND.(VALNEW.LT.VALUE(K))) THEN
                    DO 190 I=1, N
                            X(W,I) = NEWX(I)
190                 CONTINUE
                    GO TO 220
                 ENDIF
200      CONTINUE
      ENDIF
      DO 210 I=1, N
              NEWX(I) = 0.5*(AVE(I)+NEWX(I))
210   CONTINUE
      GO TO 180
C
C     *******************************************************************
C
220   TEST(2) = VALNEW
      A = A+1
      PRINT 230, NEWX(1), NEWX(2), NEWX(3), VALNEW, R
230   FORMAT (3X, F11.9, 3X, F11.9, 3X, F11.9, 2X, F14.10, 11X, F3.1)
C
```

```
          C      ************************************************************
          C
                 IF ((TEST(1)-TEST(2)).LT.E) THEN
1                   DO 240 K=1, N+1
1                           VALUE(K) = F(X(K,1),X(K,2),X(K,3))
1         240       CONTINUE
1                   BEST = VALUE(1)
1                   B = 1
1                   DO 250 K=2, N+1
1                           IF (VALUE(K).LT.BEST) THEN
2                              BEST = VALUE(K)
2                              B = K
2                           ENDIF
1         250       CONTINUE
1                   PRINT 260, X(B,1), X(B,2), X(B,3), BEST
1         260       FORMAT (5(/), 8X, 'MINIMUM = ( ', F11.8, ', ', F11.8, ', ',
1               $           F11.8, ' ) ', /, ' F(MINIMUM) = ', F14.10)
1                   PRINT*, ' '
1                   PRINT 270, A
1         270       FORMAT (4X, 'NUMBER OF ITERATIONS = ', I3)
1                ELSE
1                   GO TO 130
1                ENDIF
                 STOP
                 END
```

--VARIABLE MAP--(LO=A)

-NAME-	--ADDRESS	--BLOCK-	----PROPERTIES-	------TYPE-	--------SIZE
A	10428			INTEGER	
AVE	10658			REAL	3
B	10768			REAL	
BEST	10758			REAL	
E	11078			REAL	
I	10358			INTEGER	
J	10368			INTEGER	
K	10378			INTEGER	
L	11108			REAL	9
N	10408			INTEGER	
NEWX	10708			REAL	3
R	10738			REAL	
S	10418			INTEGER	
SUM	10028			REAL	3
TEST	11058			REAL	2
VALNEW	11048			REAL	
VALUE	11008			REAL	4
W	10778			REAL	
WORST	10748			REAL	
X	10438			REAL	15

--PROCEDURES--(LO=A)

-NAME-	-----TYPE-	-------ARGS-	-----CLASS-----
C	REAL	3	FUNCTION
F	REAL	3	FUNCTION

--STATEMENT LABELS--(LO=A)

-LABEL	-ADDRESS-	----PROPERTIES-	---DEF
10	INACTIVE	DO-TERM	19
20	INACTIVE	DO-TERM	22
30	INACTIVE	DO-TERM	32
40	INACTIVE	DO-TERM	33
50	1358		36
60	INACTIVE	DO-TERM	41
70	INACTIVE	DO-TERM	43
80	INACTIVE	DO-TERM	46
90	6538	FORMAT	51
100	6648	FORMAT	55
110	INACTIVE	DO-TERM	56
120	6718	FORMAT	58
130	2768		65
140	INACTIVE	DO-TERM	67
150	INACTIVE	DO-TERM	75

```
-LABEL-ADDRESS-----PROPERTIES----DEF      -LABEL-ADDRESS-----PROPERTIES----DEF      -LABEL-ADDRESS-----PROPERTIES----DEF

   160  INACTIVE   DO-TERM         83        200  INACTIVE   DO-TERM          99        240  INACTIVE   DO-TERM         118
   170  INACTIVE   DO-TERM         86        210  INACTIVE   DO-TERM         103        250  INACTIVE   DO-TERM         126
   180    431B                     90        220    522B                       106        260    711B     FORMAT          128
   190  INACTIVE   DO-TERM         96        230    703B     FORMAT          111        270    723B     FORMAT          132

--ENTRY POINTS--(LO=A)
 -NAME---ADDRESS--ARGS---

  TEST20     14B     0

--STATISTICS--

  PROGRAM-UNIT LENGTH          1142B =  610
  CM STORAGE USED            63300B = 26304
  COMPILE TIME                0.039 SECONDS
```

```
  FUNCTION F          74/81B  OPT=0,ROUND= A/ S/ M/-D,-DS      FTN 5.1+842.       87/03/04. 13.44.48
DO=-LONG/-OT,ARG=-COMMON/-FIXED,CS= USER/-FIXED,DB=-TB/-SB/-SL/ ER/-ID/-PMD/-ST,-AL,PL=5000
FTN5,I=PROB22.

                    REAL FUNCTION F(X1,X2,X3)
                    REAL X1, X2, X3
                    F = 2*(X1**2)+2*(X2**2)+X3**2+2*X1*X2+2*X1*X3-8*X1-6*X2-4*X3+9
                    RETURN
                    END

--VARIABLE MAP--(LO=A)
 -NAME---ADDRESS --BLOCK-----PROPERTIES-------TYPE---------SIZE

  F           30B                              REAL
  X1            1   DUMMY-ARG                  REAL
  X2            2   DUMMY-ARG                  REAL
  X3            3   DUMMY-ARG                  REAL

--ENTRY POINTS--(LO=A)
 -NAME---ADDRESS--ARGS---

  F           7B    3

--STATISTICS--

  PROGRAM-UNIT LENGTH          41B = 33
  CM STORAGE USED           63300B = 26304
  COMPILE TIME              0.081 SECONDS
```

```
      FUNCTION C            74/810  OPT=0,ROUND= A/ S/ M/-D,-DS     FTN 5.1+642         87/03/04. 13.44.48
DO=-LONG/-OT,ARG=-COMMON/-FIXED,CS= USER/-FIXED,DB=-TB/-SB/-SL/ ER/-ID/-PMD/-ST,-AL,PL=5000
FTN5,I=PROG22.

                    REAL FUNCTION C(X1,X2,X3)
                    REAL X1, X2, X3
                    C = -X1-X2-2*X3+3
                    RETURN
                    END

--VARIABLE MAP--(LO=A)
 -NAME---ADDRESS --BLOCK-----PROPERTIES-------TYPE---------SIZE

  C          24B                              REAL
  X1           1   DUMMY-ARG                  REAL
  X2           2   DUMMY-ARG                  REAL
  X3           3   DUMMY-ARG                  REAL

--ENTRY POINTS--(LO=A)
 -NAME---ADDRESS--ARGS---

  C          7B      3

--STATISTICS--

  PROGRAM-UNIT LENGTH              27B =  23
  CM STORAGE USED              63300B =  26304
  COMPILE TIME                 0.041 SECONDS
```

3
0.1
0.1
0.1
0.406
0.336
0.791
0.400
0.279
0.006
0.013
0.761
0.290
1.3
0.0000000001

X(K,1)	X(K,2)	X(K,3)	F(X(K,1),X(K,2),X(K,3))
.100000000	.100000000	.100000000	7.2900000000
.421250000	.381250000	.643250000	2.9954180825
.381562500	.452062500	.709312500	2.4871744102
.743718750	.665718750	.459593750	1.0952189932

NEWX(1)	NEWX(2)	NEWX(3)	VALNEW	REFLECTION FACTOR
.866651302	.904230002	.707009010	1.0732190094	1.3
.702898404	.745799529	.847113093	.7918448825	1.3
.906425376	.888325219	.900064035	.3929403060	1.3
.708080000	.732943190	.727310929	.8998421941	1.3
.900080502	.961674217	.402473010	.2952597623	1.3
.932384333	.872360000	.980162056	.2920292441	1.3
1.107528413	1.000000288	.313814430	.3427885755	1.3
1.000219007	1.002640834	.302070930	.3040108178	1.3
.944408715	.947400817	.844027390	.2751714063	1.3
.938008436	.928040837	.951130120	.2901384783	1.3
.927000005	.901474836	.900278612	.2862108509	1.3
.940783700	.901879746	.930062641	.2853417920	1.3
.930082705	.952053346	.952660576	.2722824982	1.3
.950183610	1.005704600	.910570906	.2842390340	1.3
.940474184	.961901400	.940742497	.2639178640	1.3
.936710420	.870006779	.900093074	.2744105755	1.3
.941308797	.924014936	.966130178	.2062798405	1.3
.945412184	.963827446	.945363344	.2000559575	1.3
.940821064	.942063065	.953000436	.2615009734	1.3
.940529243	.900198279	.970420372	.2083028411	1.3
.950404303	.940817700	.940712636	.2858313094	1.3
.956399203	.907658309	.967700395	.2534709743	1.3
.961610401	.902240934	.971001041	.2507837567	1.3
.900033729	.920176474	.966611490	.2471074927	1.3
.963874251	.904040587	.909500022	.2533327131	1.3
.973511001	.906084011	.970847731	.2446457075	1.3
.971182936	.941406614	.942420136	.2476327122	1.3
.981404005	.930588042	.943272394	.2384300058	1.3
.970003410	.950230050	.961046364	.2410095222	1.3
.981930814	.930411150	.902937095	.2356350359	1.3
.986330791	.900508316	.900950730	.2247061813	1.3
1.000615000	.922514420	.933009187	.2219500713	1.3
1.020321711	.922320000	.970071557	.2184904515	1.3
1.020402306	.925790407	.920042961	.2007782175	1.3
1.047611922	.904000230	.920030092	.1950151708	1.3

1.064004900	.070400105	.551500004	.1073001361	1.3
1.081527064	.065235430	.476110079	.1090724926	1.3
1.111587240	.352718218	.514790179	.1025502978	1.3
1.070154181	.700540004	.560040179	.1957277513	1.3
1.127400064	.732000015	.567120000	.1047825542	1.3
1.141829000	.824003715	.515453400	.1491027270	1.3
1.707020213	.700027070	.507592200	.1340072518	1.3
1.187770820	.823770079	.441747519	.1555097733	1.3
1.200512620	.822250043	.453773273	.1175994054	1.3
1.718000307	.720073204	.525114007	.1430401915	1.3
1.349540050	.706713041	.400545283	.1729122151	1.3
1.340700240	.814703200	.414387087	.1174530117	1.3
1.302617207	.790002005	.405000614	.1181570507	1.3
1.333733817	.825170750	.417357043	.1177716000	1.3
1.303284001	.820023063	.432193007	.1109002132	1.3
1.307340004	.810205750	.443910187	.1164035374	1.3
1.327304717	.810614671	.422400500	.1167023745	1.3
1.282722717	.823022430	.430070624	.1167601320	1.3
1.301305713	.812377000	.438054773	.1156192036	1.3
1.282936107	.817507720	.446413507	.1166300334	1.3
1.287741204	.807040291	.440505078	.1157014136	1.3
1.264147620	.800033170	.440307093	.1150605772	1.3
1.311280001	.795773442	.440791374	.1146349540	1.3
1.320034000	.801712000	.420004178	.1150510105	1.3
1.332141104	.700000200	.433020707	.1144118564	1.3
1.300012204	.770000007	.420000020	.1150478274	1.3
1.334150777	.700001000	.440303477	.1148959772	1.3
1.310104034	.700520401	.445673737	.1146042675	1.3
1.312571372	.798243073	.440021620	.1145647321	1.3
1.327204036	.761700079	.436604004	.1142932735	1.3
1.330051052	.700001322	.437161281	.1144537099	1.3
1.330007207	.702635372	.433103621	.1144200709	1.3
1.331500067	.770000047	.437100190	.1143047497	1.3
1.331040007	.702335710	.440097441	.1143619707	1.3
1.338202751	.770000101	.440540011	.1143530000	1.3
1.330064077	.702612007	[illegible]	.1143505490	1.3
1.327000014	.770472042	[illegible]	.1142900401	1.3
1.320077304	.703114170	[illegible]	.1142400700	1.3
1.324004230	.770000250	.430717524	.1142018101	1.3
1.331340400	.770020302	.437727957	.1142065031	1.3
1.330050703	.702750051	.434000000	.1142941410	1.3
1.324270001	.704206175	.439470005	.1141457945	1.3
1.320041032	.701040003	.434050050	.1141341517	1.3
1.331402404	.700072200	.434773001	.1141020020	1.3
1.320700130	.705000104	.430175730	.1139074513	1.3
1.320070403	.704072400	.431332719	.1141211193	1.3
1.320332010	.701034004	.434071047	.1139000773	1.3
1.321077003	.700110100	.430790750	.1140742030	1.3
1.317030077	.702301242	.440050003	.1139630507	1.3
1.327470070	.700400302	.430313301	.1137000210	1.3
1.317370004	.705000200	.443532737	.1139110000	1.3
1.313270006	.700037015	.442143501	.1137000058	1.3
1.322400016	.700401000	.440000012	.1135507515	1.3
1.320057200	.700200110	.430305000	.1135571100	1.3
1.311000030	.700005232	.443427046	.1133072074	1.3
1.300720040	.700107030	.430033532	.1131300004	1.3
1.314401709	.704370000	.445700016	.1132970034	1.3
1.312000020	.704101000	.444400723	.1127125100	1.3
1.327700000	.707437200	.441070020	.1120393057	1.3
1.304300400	.704000040	.432050403	.1120405700	1.3
1.300152040	.700000070	.443070150	.1117230705	1.3
1.300000720	.700170027	.433033110	.1122370330	1.3
1.327040000	.773400137	.447071033	.1120253377	1.3
1.300720040	.702307220	.441700016	.1112751100	1.3
1.314000113	.701704434	.461400073	.1119000003	1.3
1.320004721	.707013200	.440371270	.1113424200	1.3

1.374595602	.781473853	.446941252	.1111998155	1.3
1.330743645	.784312365	.447452207	.1111991853	1.3
1.339583831	.777310147	.441523014	.1111691860	1.3
1.331299968	.780777792	.443953748	.1111306927	1.3
1.336891884	.780581132	.441246052	.1111673244	1.3
1.330293257	.776464952	.442103617	.1111552074	1.3
1.333387319	.780551937	.443026921	.1111302831	1.3
1.333209575	.778409118	.444186423	.1111137931	1.3
1.331549880	.780473094	.443985411	.1111256151	1.3
1.333635747	.779182997	.443589378	.1111168827	1.3
1.332032806	.777799662	.445081933	.1111140009	1.3
1.333876548	.777157867	.444481237	.1111127279	1.3
1.332845468	.777335840	.444906189	.1111135750	1.3
1.334140501	.777526828	.444182361	.1111135267	1.3
1.333754058	.776992817	.444623902	.1111134028	1.3
1.334273687	.777190130	.444265301	.1111134004	1.3
1.333855486	.778845064	.444648207	.1111133511	1.3
1.334323344	.777157429	.444258233	.1111128925	1.3
1.333685896	.776875850	.444718894	.1111126786	1.3
1.334099540	.777348038	.444275486	.1111120879	1.3
1.333319741	.777088097	.444795681	.1111123730	1.3
1.333673479	.777095225	.444815487	.1111120272	1.3
1.333712784	.777568771	.444358409	.1111116315	1.3
1.333993981	.777418350	.444293214	.1111118665	1.3
1.333395452	.777377349	.444613319	.1111115641	1.3
1.333718458	.777688562	.444295652	.1111116351	1.3
1.333108291	.777709402	.444590479	.1111115080	1.3
1.333202092	.777462971	.444667040	.1111116077	1.3
1.333157684	.777508092	.444666708	.1111115638	1.3
1.333244374	.777620850	.444566902	.1111114133	1.3
1.333023648	.777765812	.444604592	.1111115384	1.3
1.333104477	.777822575	.444535725	.1111114986	1.3
1.333319733	.777654945	.444512079	.1111114066	1.3
1.333371803	.777686528	.444470318	.1111113558	1.3
1.333440841	.777544604	.444503893	.1111113607	1.3
1.333555070	.777638887	.444402516	.1111113942	1.3
1.333637528	.777582253	.444389789	.1111113660	1.3
1.333394807	.777559708	.444522463	.1111113268	1.3
1.333495613	.777693857	.444404909	.1111112974	1.3
1.333279830	.777688587	.444515387	.1111113284	1.3
1.333413847	.777596497	.444494686	.1111112341	1.3
1.333636159	.777523218	.444420268	.1111112752	1.3
1.333671728	.777662784	.444337694	.1111112484	1.3
1.333588634	.777577987	.444417666	.1111112202	1.3
1.333455018	.777728374	.444406187	.1111111781	1.3
1.333363803	.777615751	.444510048	.1111112208	1.3
1.333539591	.777688157	.444381097	.1111111870	1.3
1.333633147	.777702236	.444332292	.1111112098	1.3
1.333513854	.777795143	.444345384	.1111111873	1.3
1.333333474	.777790377	.444437931	.1111111751	1.3
1.333396378	.777702458	.444450489	.1111111730	1.3
1.333381714	.777732873	.444442838	.1111111458	1.3
1.333387967	.777689673	.444416156	.1111111605	1.3
1.333802440	.777685095	.444401216	.1111111488	1.3
1.333368824	.777682921	.444470582	.1111111447	1.3
1.333297960	.777768441	.444468731	.1111111438	1.3
1.333323464	.777737313	.444409533	.1111111457	1.3
1.333294683	.777732904	.444486051	.1111111448	1.3
1.333314448	.777723763	.444480852	.1111111381	1.3
1.333309630	.777722488	.444455394	.1111111375	1.3
1.333278026	.777787133	.444463859	.1111111417	1.3
[illegible]	.777720955	[illegible]	.1111111290	1.3
[illegible]	.777688114	[illegible]	.1111111308	1.3
[illegible]	[illegible]	.444441288	.1111111347	1.3
[illegible]	.777686128	.444463538	.1111111289	1.3
1.333487518	.777788824	.444441929	.1111111213	1.3

1.333368151	.777700822	.444462810	.1111111221	1.3
1.333390881	.777080792	.444455158	.1111111225	1.3
1.333391510	.777720797	.444439845	.1111111172	1.3
1.333386679	.777735336	.444438988	.1111111174	1.3
1.333412841	.777735711	.444425723	.1111111185	1.3
1.333383350	.777785336	.444425852	.1111111159	1.3
1.333353820	.777752835	.444446886	.1111111155	1.3
1.333362839	.777766739	.444435307	.1111111140	1.3
1.333350413	.777762882	.444433294	.1111111153	1.3
1.333337882	.777768937	.444448723	.1111111147	1.3
1.333347877	.777785920	.444433095	.1111111146	1.3
1.333348020	.777762013	.444444980	.1111111133	1.3
1.333372861	.777774963	.444426187	.1111111135	1.3
1.333368706	.777758195	.444437048	.1111111132	1.3
1.333363998	.777762863	.444438588	.1111111124	1.3
1.333358846	.777755810	.444443870	.1111111130	1.3
1.333368465	.777758881	.444437271	.1111111127	1.3
1.333354366	.777761470	.444442082	.1111111121	1.3
1.333364106	.777762012	.444436941	.1111111122	1.3
1.333350889	.777768776	.444440167	.1111111117	1.3
1.333348846	.777765874	.444443840	.1111111116	1.3
1.333333119	.777769590	.444448844	.1111111118	1.3
1.333329492	.777776520	.444446993	.1111111115	1.3
1.333348337	.777770800	.444440431	.1111111114	1.3
1.333329276	.777773888	.444448418	.1111111113	1.3
1.333321475	.777784216	.444447154	.1111111114	1.3
1.333334179	.777776230	.444444798	.1111111111	1.3
1.333321801	.777780487	.444448856	.1111111113	1.3

MINIMUM = (1.333334179, .777776230, .444444796) ; F(MINIMUM) = .1111111111

NUMBER OF ITERATIONS = 195

```
      PROGRAM TEST3R      74/810  OPT=0,ROUND= A/ S/ M/-D,-DS      FTN 5.1+642        87/03/04. 15.00.15
DO=-LONG/-OT,ARG=-COMMON/-FIXED,CS= USER/-FIXED,DB=-TB/-SB/-SL/ ER/-ID/-PMD/-ST,-AL,PL=5000
FTN5,I=PROG4.

      PROGRAM TEST3R (INPUT, OUTPUT)
C
C     ************************************************************************
C     *  OBJECTIVE FUNCTION : F(X1,X2) = -X1**2-X2**2                        *
C     *  CONSTRAINTS        : X1.GE.0                                        *
C     *                       X2.GE.0                                        *
C     *                       -X1+X2+4.GE.0                                  *
C     *                       X1/3-X2+4.GE.0                                 *
C     *                       X1**2+X2**2-10*X1-10*X2+41.GE.0                *
C     *  MINIMUM            : MIN = (12,8)      ;     F(12,8) = -208         *
C     *  START              : X(0) = (0,0)      ;     F(0,0) = 0             *
C     *                                                                      *
C     *  BESIDES THE ABOVE GLOBAL MINIMUM, THERE ARE TWO LOCAL MINIMA:       *
C     *               MIN1 = (2.018,4.673) ;     F(2.018,4.673) = -25.91     *
C     *               MIN2 = (6.293,2.293) ;     F(6.293,2.293) = -44.86     *
C     ************************************************************************
C
      INTEGER I, K, L, J, N, A
      REAL X(0:3,2), S, SUM(2), AVE(2), NEWX(2), R, WORST, BEST, B, W,
     $     VALUE(3), VALNEW, TEST(2), E, D(2), SEARCH(26), BNEW(2)
      READ*, N
      DO 10 I=1, N
           READ*, X(0,I)
           X(1,I) = X(0,I)
10    CONTINUE
      DO 100 K=2, N+1
            S = 0.5
            J = K-1
            DO 20 I=1, N
                 IF ((J-I).EQ.0) THEN
                    X(K,I) = 0
                 ELSE
                    X(K,I) = X(J,I)
                 ENDIF
20          CONTINUE
            SEARCH(1) = F(X(K,1),X(K,2))
            BEST = SEARCH(1)
            DO 50 L=2, 26
                 DO 30 I=1, N
                      IF ((J-I).EQ.0) THEN
                         X(K,I) = X(K,I)+S
                      ELSE
                         X(K,I) = X(J,I)
                      ENDIF
30               CONTINUE
                 SEARCH(L) = F(X(K,1),X(K,2))
                 IF (SEARCH(L).LT.BEST) THEN
                    BEST = SEARCH(L)
                    DO 40 I=1, N
                          BNEW(I) = X(K,I)
40                  CONTINUE
                 ENDIF
50          CONTINUE
            DO 60 I=1, N
                 X(K,I) = BNEW(I)
```

```
60          CONTINUE
70          IF ((C3(X(K,1),X(K,2)).LT.0).OR.(C4(X(K,1),X(K,2)).LT.0)
     $        .OR.(C5(X(K,1),X(K,2)).LT.0)) THEN
              DO 90 I=1, N
                    SUM(I) = 0
                    DO 80 L=1, J
                          SUM(I) = SUM(I)+X(L,I)
80                  CONTINUE
                    X(K,I) = 0.5*(X(K,I)+SUM(I)/J)
90            CONTINUE
              GO TO 70
            ENDIF
100   CONTINUE
C
C     ******************************************************************
C
      PRINT 110
110   FORMAT (5(/), 10X, 'X(K,1)', 12X, 'X(K,2)', 8X,
     $        'F(X(K,1),X(K,2))')
      DO 130 K=1, N+1
            PRINT 120, X(K,1), X(K,2), F(X(K,1), X(K,2))
120         FORMAT (4X, F13.9, 5X, F13.9, 5X, F16.10)
130   CONTINUE
      PRINT 140
140   FORMAT (5(/), 9X, 'NEWX(1)', 10X, 'NEWX(2)', 12X, 'VALNEW',
     $        11X, 'REFLEXION FACTOR')
C
C     ******************************************************************
C
      A = 0
      READ*, E
150   DO 160 K=1, N+1
            VALUE(K) = F(X(K,1),X(K,2))
160   CONTINUE
      WORST = VALUE(1)
      W = 1
      DO 170 K=2, N+1
            IF (VALUE(K).GT.WORST) THEN
              WORST = VALUE(K)
              W = K
            ENDIF
170   CONTINUE
      TEST(1) = WORST
C
C     ******************************************************************
C
      DO 190 I=1, N
            SUM(I) = 0
            DO 180 K=1, N+1
                  IF (K.NE.W) THEN
                    SUM(I) = SUM(I)+X(K,I)
                  ENDIF
180         CONTINUE
            AVE(I) = SUM(I)/N
            D(I) = AVE(I)-X(W,I)
190   CONTINUE
      R = FACTOR(AVE(1),AVE(2),D(1),D(2))
```

```
      DO 200 I=1, N
            NEWX(I) = AVE(I)+R*D(I)
200   CONTINUE
C
C     ****************************************************************
C
210   VALNEW = F(NEWX(1),NEWX(2))
      IF ((C1(NEWX(1)).GE.0).AND.(C2(NEWX(2)).GE.0).AND.(C3(NEWX(1),
     $   NEWX(2)).GE.0).AND.(C4(NEWX(1),NEWX(2)).GE.0).AND.(C5(NEWX(1),
     $   NEWX(2)).GE.0)) THEN
         DO 230 K=1, N+1
               IF ((K.NE.W).AND.(VALNEW.LT.VALUE(K))) THEN
                  DO 220 I=1, N
                        X(W,I) = NEWX(I)
220               CONTINUE
                  GO TO 250
               ENDIF
230      CONTINUE
      ENDIF
      DO 240 I=1, N
            NEWX(I) = 0.5*(AVE(I)+NEWX(I))
240   CONTINUE
      GO TO 210
C
C     ****************************************************************
C
250   TEST(2) = VALNEW
      A = A+1
      PRINT 260, NEWX(1), NEWX(2), VALNEW, R
260   FORMAT (4X, F13.9, 5X, F13.9, 5X, F16.10, 11X, F3.1)
C
C     ****************************************************************
C
      IF ((TEST(1)-TEST(2)).LT.E) THEN
         DO 270 K=1, N+1
               VALUE(K) = F(X(K,1),X(K,2))
270      CONTINUE
         BEST = VALUE(1)
         B = 1
         DO 280 K=2, N+1
               IF (VALUE(K).LT.BEST) THEN
                  BEST = VALUE(K)
                  B = K
               ENDIF
280      CONTINUE
         PRINT 290, X(B,1), X(B,2), BEST
290      FORMAT (5(/), 6X, 'MINIMUM = ( ', F13.9, ', ', F13.9,
     $   ')  ;  ', 'F(MINIMUM) = ', F16.10)
         PRINT*, ' '
         PRINT 300, A
300      FORMAT (6X, 'NUMBER OF ITERATIONS = ', I3)
      ELSE
         GO TO 150
      ENDIF
      STOP
      END
```

```
--VARIABLE MAP--(LO=A)
-NAME---ADDRESS --BLOCK-----PROPERTIES-------TYPE---------SIZE     -NAME---ADDRESS --BLOCK-----PROPERTIES-------TYPE---------SIZE

 A          1175B                            INTEGER                NEWX     1213B                                 REAL         2
 AVE        1211B                            REAL             2     R        1215B                                 REAL
 B          1220B                            REAL                   S        1206B                                 REAL
 BEST       1217B                            REAL                   SEARCH   1233B                                 REAL        28
 BNEW       1265B                            REAL             2     SUM      1207B                                 REAL         2
 D          1231B                            REAL             2     TEST     1226B                                 REAL         2
 E          1230B                            REAL                   VALNEW   1225B                                 REAL
 I          1170B                            INTEGER                VALUE    1222B                                 REAL         3
 J          1173B                            INTEGER                W        1221B                                 REAL
 K          1171B                            INTEGER                WORST    1216B                                 REAL
 L          1172B                            INTEGER                X        117[illegible]B                       REAL         8
 N          1174B                            INTEGER

--PROCEDURES--(LO=A)
-NAME------TYPE--------ARGS------CLASS-----     -NAME------TYPE--------ARGS------CLASS-----

 C1         REAL          1      FUNCTION        C5        REAL           2      FUNCTION
 C2         REAL          1      FUNCTION        F         REAL           2      FUNCTION
 C3         REAL          2      FUNCTION        FACTOR    REAL           4      FUNCTION
 C4         REAL          2      FUNCTION

--STATEMENT LABELS--(LO=A)
-LABEL-ADDRESS-----PROPERTIES----DEF     -LABEL-ADDRESS-----PROPERTIES----DEF     -LABEL-ADDRESS-----PROPERTIES----DEF

    10  INACTIVE    DO-TERM      25         110   1015B     FORMAT       73         210    563B                   119
    20  INACTIVE    DO-TERM      35         120   1024B     FORMAT       77         220  INACTIVE    DO-TERM      127
    30  INACTIVE    DO-TERM      45         130 INACTIVE    DO-TERM      78         230  INACTIVE    DO-TERM      130
    40  INACTIVE    DO-TERM      51         140   1030B     FORMAT       80         240  INACTIVE    DO-TERM      134
    50  INACTIVE    DO-TERM      53         150    414B                  87         250    675B                   139
    60  INACTIVE    DO-TERM      56         160 INACTIVE    DO-TERM      89         260   1040B      FORMAT       142
    70    240B                   57         170 INACTIVE    DO-TERM      97         270  INACTIVE    DO-TERM      149
    80  INACTIVE    DO-TERM      63         180 INACTIVE    DO-TERM     108         280  INACTIVE    DO-TERM      157
    90  INACTIVE    DO-TERM      65         190 INACTIVE    DO-TERM     111         290   1045B      FORMAT       159
   100  INACTIVE    DO-TERM      68         200 INACTIVE    DO-TERM     115         300   1056B      FORMAT       163

--ENTRY POINTS--(LO=A)
-NAME---ADDRESS--ARGS---

 TEST3R     14B      0

--STATISTICS--

 PROGRAM-UNIT LENGTH        1313B =   715
 CM STORAGE USED           65300B =  27328
 COMPILE TIME              1.007 SECONDS
```

```
FUNCTION F          74/810  OPT=0,ROUND= A/ S/ M/-D,-DS      FTN 5.1+642         87/03/04. 15.00.15
DO=-LONG/-OT,ARG=-COMMON/-FIXED,CS= USER/-FIXED,DB=-TB/-SB/-SL/ ER/-ID/-PMD/-ST,-AL,PL=5000
FTN5,I=PROG4.

                    REAL FUNCTION F(X1,X2)
                    REAL X1, X2
                    F = -X1**2-X2**2
                    RETURN
                    END

--VARIABLE MAP--(LO=A)
 -NAME---ADDRESS --BLOCK-----PROPERTIES-------TYPE---------SIZE

 F          20B                               REAL
 X1          1    DUMMY-ARG                   REAL
 X2          2    DUMMY-ARG                   REAL

--ENTRY POINTS--(LO=A)
 -NAME---ADDRESS--ARGS---

 F          7B      2

--STATISTICS--

 PROGRAM-UNIT LENGTH              23B =   19
 CM STORAGE USED               63300B =  26304
 COMPILE TIME                  0.039 SECONDS
```

```
      FUNCTION C1          74/810  OPT=0,ROUND= A/ S/ M/-D,-DS      FTN 5.1+642          87/03/04. 15.00.15
     DO=-LONG/-OT,ARG=-COMMON/-FIXED,CS= USER/-FIXED,DB=-TB/-SB/-SL/ ER/-ID/-PMD/-ST,-AL,PL=5000
     FTNB,I=PROG4.

        1                     REAL FUNCTION C1(X1)
        2                     REAL X1
        3                     C1 = X1
        4                     RETURN
        5                     END

--VARIABLE MAP--(LO=A)
 -NAME---ADDRESS --BLOCK-----PROPERTIES-------TYPE---------SIZE

  C1         16B                              REAL
  X1           1   DUMMY-ARG                  REAL

--ENTRY POINTS--(LO=A)
 -NAME---ADDRESS--ARGS---

  C1         7B      1

--STATISTICS--

  PROGRAM-UNIT LENGTH            21B =   17
  CM STORAGE USED             63300B = 26304
  COMPILE TIME                 0.030 SECONDS
```

```
   FUNCTION C2          74/810  OPT=0,ROUND= A/ S/ M/-D,-DS      FTN 5.1+642         87/03/04. 15.00.18
  DO=-LONG/-OT,ARG=-COMMON/-FIXED,CS= USER/-FIXED,DB=-TB/-SB/-SL/ ER/-ID/-PMD/-ST,-AL,PL=5000
  FTN5,I=PROG4.

                      REAL FUNCTION C2(X2)
                      REAL X2
                      C2 = X2
                      RETURN
                      END

--VARIABLE MAP--(LO=A)
 -NAME---ADDRESS --BLOCK-----PROPERTIES-------TYPE---------SIZE

  C2          16B                              REAL
  X2           1    DUMMY-ARG                  REAL

--ENTRY POINTS--(LO=A)
 -NAME---ADDRESS--ARGS---

  C2          7B      1

--STATISTICS--

  PROGRAM-UNIT LENGTH           21B =    17
  CM STORAGE USED           63300B =  26304
  COMPILE TIME              0.031 SECONDS
```

```
FUNCTION C3          74/810  OPT=0,ROUND= A/ S/ Ṁ/-D,-DS     FTN 5.1+642          87/03/04. 15.00.15
DO=-LONG/-OT,ARG=-COMMON/-FIXED,CS= USER/-FIXED,DB=-TB/-SB/-SL/ ER/-ID/-PMD/-ST,-AL,PL=5000
FTN5,I=PROG4.

                REAL FUNCTION C3(X1,X2)
                REAL X1, X2
                C3 = -X1+X2+4
                RETURN
                END

--VARIABLE MAP--(LO=A)
-NAME---ADDRESS --BLOCK-----PROPERTIES-------TYPE---------SIZE

 C3          21B                             REAL
 X1           1    DUMMY-ARG                 REAL
 X2           2    DUMMY-ARG                 REAL

--ENTRY POINTS--(LO=A)
-NAME---ADDRESS--ARGS---

 C3          7B     2

--STATISTICS--

 PROGRAM-UNIT LENGTH           24B =  20
 CM STORAGE USED            633000 =  26304
 COMPILE TIME               0.035 SECONDS
```

```
   FUNCTION C4          74/810  OPT=0,ROUND= A/ S/ M/-D,-DS      FTN 5.1+642          87/03/04. 15.00.15
DO=-LONG/-OT,ARG=-COMMON/-FIXED,CS= USER/-FIXED,DB=-TB/-SB/-SL/ ER/-ID/-PMD/-ST,-AL,PL=5000
FTN5,I=PROG4.

                    REAL FUNCTION C4(X1,X2)
                    REAL X1, X2
                    C4 = X1/3-X2+4
                    RETURN
                    END

--VARIABLE MAP--(LO=A)
 -NAME---ADDRESS --BLOCK-----PROPERTIES--------TYPE----------SIZE

  C4          23B                              REAL
  X1            1    DUMMY-ARG                 REAL
  X2            2    DUMMY-ARG                 REAL

--ENTRY POINTS--(LO=A)
 -NAME---ADDRESS--ARGS---

  C4          7B      2

--STATISTICS--

  PROGRAM-UNIT LENGTH              26B =  22
  CM STORAGE USED              63300B =  26304
  COMPILE TIME                 0.036 SECONDS
```

```
   FUNCTION CS          74/810  OPT=0,ROUND= A/ S/ M/-D,-DS      FTN 5.1+642         87/03/04. 15.00.15
DO=-LONG/-OT,ARG=-COMMON/-FIXED,CS= USER/-FIXED,DB=-TB/-SB/-SL/ ER/-ID/-PMD/-ST,-AL,PL=5000
FTN5,I=PROG4.

                    REAL FUNCTION CS(X1,X2)
                    REAL X1, X2
                    CS = X1**2+X2**2-10*X1-10*X2+41
                    RETURN
                    END

--VARIABLE MAP--(LO=A)
 -NAME---ADDRESS --BLOCK-----PROPERTIES--------TYPE---------SIZE

  CS         25B                               REAL
  X1          1   DUMMY-ARG                    REAL
  X2          2   DUMMY-ARG                    REAL

--ENTRY POINTS--(LO=A)
 -NAME---ADDRESS--ARGS---

  CS         7B    2

--STATISTICS--

  PROGRAM-UNIT LENGTH            30B = 24
  CM STORAGE USED             63300B = 26304
  COMPILE TIME                0.045 SECONDS
```

```
   FUNCTION FACTOR      74/810  OPT=0,ROUND= A/ S/ M/-D,-DS      FTN 5.1+642         87/03/04. 15.00.15
DO=-LONG/-OT,ARG=-COMMON/-FIXED,CS= USER/-FIXED,DB=-TB/-SB/-SL/ ER/-ID/-PMD/-ST,-AL,PL=5000
FTN5,I=PROG4.

                   REAL FUNCTION FACTOR(AVE1,AVE2,D1,D2)
                   REAL AVE1, AVE2, D1, D2, S, C(5), R, F(2)
                   INTEGER I
                   R = 1
                   F(1) = -(AVE1+R*D1)**2-(AVE2+R*D2)**2
                   S = 0.1
                   DO 10 I=2, 60
                      R = 1+(I-1)*S
                      F(2) = -(AVE1+R*D1)**2-(AVE2+R*D2)**2
                      IF (F(2).GT.F(1)) THEN
 1                       R = R-S
 1                       GO TO 20
 1                    ELSE
 1                       F(1) = F(2)
 1                    ENDIF
              10   CONTINUE
              20   FACTOR = R
                   RETURN
                   END

--VARIABLE MAP--(LO=A)
 -NAME---ADDRESS --BLOCK-----PROPERTIES-------TYPE---------SIZE     -NAME---ADDRESS --BLOCK-----PROPERTIES-------TYPE-------SIZE

  AVE1         1   DUMMY-ARG                  REAL                   F          101B                             REAL         2
  AVE2         2   DUMMY-ARG                  REAL                   FACTOR      71B                             REAL
  C          73B            UND               RPAI           5       I          103B                             INTEGER
  D1           3   DUMMY-ARG                  REAL                   R          100B                             REAL
  D2           4   DUMMY-ARG                  REAL                   S           72B                             REAL

--STATEMENT LABELS--(LO=A)
 -LABEL-ADDRESS-----PROPERTIES----DEF

     10  INACTIVE   DO-TERM        16
     20     62B                    17

--ENTRY POINTS--(LO=A)
 -NAME---ADDRESS--ARGS---

  FACTOR      7B     4

--STATISTICS--

  PROGRAM-UNIT LENGTH                107B =  71
  CM STORAGE USED                  63300B =  26304
  COMPILE TIME                     0.126 SECONDS
```

2
6
6
0.000000001

X(K,1)	X(K,2)	F(X(K,1),X(K,2))
.0000000000	.0000000000	.0000000000
3.1250000000	.0000000000	-9.7656250000
1.9531250000	3.1250000000	-13.5803277856

NEWX(1)	NEWX(2)	VALNEW	REFLEXION FACTOR
3.634033203	2.236328125	-18.2073608041	0.9
2.757847786	2.000073157	-18.4746630603	0.9
5.330707223	1.702881740	-31.4121108277	0.9
5.237575541	1.530009359	-29.7455858000	0.9
5.464300146	1.853907236	-32.2733021448	0.9
5.894402120	1.783721220	-35.8078768900	0.9
5.831010415	1.861530576	-34.4689920577	0.9
6.004941307	2.013677850	-40.1142185232	0.9
6.038267078	2.103250000	-40.8843298700	0.9
6.162710083	2.178846717	-42.7180825730	0.9
6.141083344	2.184617203	-42.5387418028	0.9
6.250470337	2.250085775	-44.1822739393	0.9
6.220883880	2.221773102	-43.6318359965	0.9
6.251228107	2.252828338	-44.1530828724	0.9
6.270853324	2.283487309	-44.6143883895	0.9
6.280095436	2.277714838	-44.4895424868	0.9
6.277807595	2.285685333	-44.6362301011	0.9
6.278215543	2.286058863	-44.6545993320	0.9
6.278723880	2.286129704	-44.6487625828	0.9
6.288436312	2.288904270	-44.7583642574	0.9
6.286363843	2.288644772	-44.7582665198	0.9
6.289498480	2.289840834	-44.8016480841	0.9
6.290733351	2.290772532	-44.8209648917	0.9
6.290512623	2.290510850	-44.8170155811	0.9
6.290853296	2.290863395	-44.8194578807	0.9
6.291840160	2.292107028	-44.8422858042	0.9
6.291926240	2.292108412	-44.8421013809	0.9
6.292181817	2.292396228	-44.8467595862	0.9
6.292307192	2.292490950	-44.8486720610	0.9
6.292382981	2.292593024	-44.8500663528	0.9
6.292477195	2.292673292	-44.8516200708	0.9
6.292483087	2.292681898	-44.8517795357	0.9
6.292501091	2.292701909	-44.8520520205	0.9
6.292486300	2.292701895	-44.8519795401	0.9
6.292511227	2.292710623	-44.8522195418	0.9
6.292581083	2.292735738	-44.8532138517	0.9
6.292857098	2.292880954	-44.8573030860	0.9
6.292808732	2.292842817	-44.8565890050	0.9
6.292860066	2.292880282	-44.8573373535	0.9
6.292880080	2.292881262	-44.8576441838	0.9
6.292875868	2.292877565	-44.8575717063	0.9
6.292879794	2.292880506	-44.8576371182	0.9
6.292883819	2.292883746	-44.8577001158	0.9
6.292888941	2.292888398	-44.8577930089	0.9
6.292887617	2.292887742	-44.8577686378	0.9
6.292892297	2.292892707	-44.8578504338	0.9
6.292891914	2.292892491	-44.8578446135	0.9

6.292892191	2.292892685	-44.8578489929	6.9
6.292892529	2.292892873	-44.8578541118	6.9
6.292892797	2.292892971	-44.8578579303	6.9
6.292892821	2.292893015	-44.8578584306	6.9
6.292893050	2.292893096	-44.8578616897	6.9
6.292893055	2.292893129	-44.8578618972	6.9
6.292893152	2.292893155	-44.8578632437	6.9
6.292893150	2.292893181	-44.8578633300	6.9
6.292893172	2.292893176	-44.8578635865	6.9
6.292893164	2.292893188	-44.8578635508	6.9
6.292893178	2.292893183	-44.8578636716	6.9
6.292893176	2.292893177	-44.8578636459	6.9
6.292893205	2.292893210	-44.8578641571	6.9
6.292893197	2.292893204	-44.8578640301	6.9
6.292893206	2.292893212	-44.8578641846	6.9
6.292893209	2.292893214	-44.8578642316	6.9
6.292893208	2.292893214	-44.8578642191	6.9
6.292893241	2.292893215	-44.8578642605	6.9
6.292893217	2.292893217	-44.8578643389	6.9
6.292893216	2.292893217	-44.8578643290	6.9
6.292893217	2.292893217	-44.8578643419	6.9
6.292893217	2.292893217	-44.8578643410	6.9
6.292893218	2.292893218	-44.8578643591	6.9
6.292893218	2.292893218	-44.8578643587	6.9
6.292893218	2.292893218	-44.8578643662	6.9
6.292893218	2.292893218	-44.8578643661	6.9
6.292893218	2.292893218	-44.8578643669	6.9
6.292893218	2.292893219	-44.8578643667	6.9
6.292893218	2.292893219	-44.8578643674	6.9
6.292893218	2.292893219	-44.8578643697	6.9
6.292893219	2.292893219	-44.8578643714	6.9
6.292893219	2.292893219	-44.8578643719	6.9
6.292893219	2.292893219	-44.8578643725	6.9
6.292893219	2.292893219	-44.8578643724	6.9
6.292893219	2.292893219	-44.8578643733	6.9
6.292893219	2.292893219	-44.8578643747	6.9
6.292893219	2.292893219	-44.8578643753	6.9
6.292893219	2.292893219	-44.8578643757	6.9
6.292893219	2.292893219	-44.8578643759	6.9
6.292893219	2.292893219	-44.8578643759	6.9
6.292893219	2.292893219	-44.8578643760	6.9
6.292893219	2.292893219	-44.8578643760	6.9
6.292893219	2.292893219	-44.8578643760	6.9
6.292893219	2.292893219	-44.8578643760	6.9

MINIMUM = (6.292893219, 2.292893219) ; F(MINIMUM) = -44.8578643760

NUMBER OF ITERATIONS = 91

```
 PROGRAM TEST3R      74/810  OPT=0,ROUND= A/ S/ M/-D,-DS      FTN 5.1+642          87/03/04. 13.58.50
DO=-LONG/-OT,ARG=-COMMON/-FIXED,CS= USER/-FIXED,DB=-TB/-SB/-SL/ ER/-ID/-PMD/-ST,-AL,PL=5000
FTN5,I=PROG4.

                PROGRAM TEST3R (INPUT, OUTPUT)
          C
          C     ****************************************************************
          C     *  OBJECTIVE FUNCTION : F(X1,X2) = -X1**2-X2**2                *
          C     *  CONSTRAINTS        : X1.GE.0                                *
          C     *                       X2.GE.0                                *
          C     *                       -X1+X2+4.GE.0                          *
          C     *                       X1/3-X2+4.GE.0                         *
          C     *                       X1**2+X2**2-10*X1-10*X2+41.GE.0        *
          C     *  MINIMUM            : MIN = (12,8)    ;    F(12,8) = -208    *
          C     *  START              : X(0) = (0,0)    ;    F(0,0) = 0        *
          C     *                                                              *
          C     *  BESIDES THE ABOVE GLOBAL MINIMUM, THERE ARE TWO LOCAL MINIMA:*
          C     *           MIN1 = (2.018,4.673) ;   F(2.018,4.673) = -25.91   *
          C     *           MIN2 = (6.293,2.293) ;   F(6.293,2.293) = -44.86   *
          C     ****************************************************************
          C
                INTEGER I, K, L, J, N, A
                REAL X(0:3,2), S, SUM(2), AVE(2), NEWX(2), R, WORST, BEST, B, W,
               $     VALUE(3), VALNEW, TEST(2), E, D(2), SEARCH(26), BNEW(2)
                READ*, N
                DO 10 I=1, N
                      READ*, X(0,I)
                      X(1,I) = X(0,I)
          10    CONTINUE
                DO 100 K=2, N+1
                      S = 0.5
                      J = K-1
                      DO 20 I=1, N
                            IF ((J-I).EQ.0) THEN
                               X(K,I) = 0
                            ELSE
                               X(K,I) = X(J,I)
                            ENDIF
          20          CONTINUE
                      SEARCH(1) = F(X(K,1),X(K,2))
                      BEST = SEARCH(1)
                      DO 50 L=2, 26
                            DO 30 I=1, N
                                  IF ((J-I).EQ.0) THEN
                                     X(K,I) = X(K,I)+S
                                  ELSE
                                     X(K,I) = X(J,I)
                                  ENDIF
          30                CONTINUE
                            SEARCH(L) = F(X(K,1),X(K,2))
                            IF (SEARCH(L).LT.BEST) THEN
                               BEST = SEARCH(L)
                               DO 40 I=1, N
                                     BNEW(I) = X(K,I)
          40                   CONTINUE
                            ENDIF
          50          CONTINUE
                      DO 60 I=1, N
                            X(K,I) = BNEW(I)
```

```
60          CONTINUE
70          IF ((C3(X(K,1),X(K,2)).LT.0).OR.(C4(X(K,1),X(K,2)).LT.0)
     $         .OR.(C5(X(K,1),X(K,2)).LT.0)) THEN
               DO 90 I=1, N
                     SUM(I) = 0
                     DO 80 L=1, J
                           SUM(I) = SUM(I)+X(L,I)
80                   CONTINUE
                     X(K,I) = 0.5*(X(K,I)+SUM(I)/J)
90             CONTINUE
               GO TO 70
            ENDIF
100   CONTINUE
C
C     ***************************************************************
C
      PRINT 110
110   FORMAT (5(/), 10X, 'X(K,1)', 12X, 'X(K,2)', 8X,
     $        'F(X(K,1),X(K,2))')
      DO 130 K=1, N+1
            PRINT 120, X(K,1), X(K,2), F(X(K,1), X(K,2))
120         FORMAT (4X, F13.9, 5X, F13.9, 5X, F18.10)
130   CONTINUE
      PRINT 140
140   FORMAT (5(/), 9X, 'NEWX(1)', 10X, 'NEWX(2)', 12X, 'VALNEW',
     $        11X, 'REFLEXION FACTOR')
C
C     ***************************************************************
C
      A = 0
      READ*, E
150   DO 160 K=1, N+1
            VALUE(K) = F(X(K,1),X(K,2))
160   CONTINUE
      WORST = VALUE(1)
      W = 1
      DO 170 K=2, N+1
            IF (VALUE(K).GT.WORST) THEN
               WORST = VALUE(K)
               W = K
            ENDIF
170   CONTINUE
      TEST(1) = WORST
C
C     ****************************************************************
C
      DO 190 I=1, N
            SUM(I) = 0
            DO 180 K=1, N+1
                  IF (K.NE.W) THEN
                        SUM(I) = SUM(I)+X(K,I)
                  ENDIF
180         CONTINUE
            AVE(I) = SUM(I)/N
            D(I) = AVE(I)-X(W,I)
190   CONTINUE
      R = FACTOR(AVE(1),AVE(2),D(1),D(2))
```

```
      DO 200 I=1, N
            NEWX(I) = AVE(I)+R*D(I)
200   CONTINUE
C
C     ****************************************************************
C
210   VALNEW = F(NEWX(1),NEWX(2))
      IF ((C1(NEWX(1)).GE.0).AND.(C2(NEWX(2)).GE.0).AND.(C3(NEWX(1),
     $   NEWX(2)).GE.0).AND.(C4(NEWX(1),NEWX(2)).GE.0).AND.(C5(NEWX(1),
     $   NEWX(2)).GE.0)) THEN
         DO 230 K=1, N+1
               IF ((K.NE.W).AND.(VALNEW.LT.VALUE(K))) THEN
                  DO 220 I=1, N
                        X(W,I) = NEWX(I)
220               CONTINUE
                  GO TO 250
               ENDIF
230      CONTINUE
      ENDIF
      DO 240 I=1, N
            NEWX(I) = 0.5*(AVE(I)+NEWX(I))
240   CONTINUE
      GO TO 210
C
C     ****************************************************************
C
250   TEST(2) = VALNEW
      A = A+1
      PRINT 260, NEWX(1), NFWX(2), VALNFW, R
260   FORMAT (4X, F13.9, 5X, F13.9, 5X, F16.10, 11X, F3.1)
C
C     ****************************************************************
C
      IF ((TEST(1)-TEST(2)).LT.E) THEN
         DO 270 K=1, N+1
               VALUE(K) = F(X(K,1),X(K,2))
270      CONTINUE
         BEST = VALUE(1)
         B = 1
         DO 280 K=2, N+1
               IF (VALUE(K).LT.BEST) THEN
                  BEST = VALUE(K)
                  B = K
               ENDIF
280      CONTINUE
         PRINT 290, X(B,1), X(B,2), BEST
290      FORMAT (5(/), 8X, 'MINIMUM = ( ', F13.9, ', ', F13.9,
     $         ')  ;  ', 'F(MINIMUM) = ', F16.10)
         PRINT*, ' '
         PRINT 300, A
300      FORMAT (8X, 'NUMBER OF ITERATIONS = ', I3)
      ELSE
         GO TO 150
      ENDIF
      STOP
      END
```

--VARIABLE MAP--(LO=A)

-NAME-	--ADDRESS	--BLOCK-	-PROPERTIES-	-TYPE-	-SIZE
A	1175B			INTEGER	
AVE	1211B			REAL	2
B	1220B			REAL	
BEST	1217B			REAL	
BNEW	1285B			REAL	2
D	1231B			REAL	2
E	1230B			REAL	
I	1170B			INTEGER	
J	1173B			INTEGER	
K	1171B			INTEGER	
L	1172B			INTEGER	
M	1174B			INTEGER	
NEWX	1213B			REAL	2
R	1215B			REAL	
S	1206B			REAL	
SEARCH	1233B			REAL	26
SUM	1207B			REAL	2
TEST	1226B			REAL	2
VALNEW	1225B			REAL	
VALUE	1222B			REAL	3
W	1221B			REAL	
WORST	1216B			REAL	
X	1176B			REAL	8

--PROCEDURES--(LO=A)

-NAME-	-TYPE-	-ARGS-	-CLASS-
C1	REAL	1	FUNCTION
C2	REAL	1	FUNCTION
C3	REAL	2	FUNCTION
C4	REAL	2	FUNCTION
C5	REAL	2	FUNCTION
F	REAL	2	FUNCTION
FACTOR	REAL	4	FUNCTION

--STATEMENT LABELS--(LO=A)

-LABEL-	-ADDRESS-	-PROPERTIES-	-DEF
10	INACTIVE	DO-TERM	25
20	INACTIVE	DO-TERM	35
30	INACTIVE	DO-TERM	45
40	INACTIVE	DO-TERM	51
50	INACTIVE	DO-TERM	53
60	INACTIVE	DO-TERM	56
70	240B		57
80	INACTIVE	DO-TERM	63
90	INACTIVE	DO-TERM	65
100	INACTIVE	DO-TERM	68
110	1015B	FORMAT	73
120	1024B	FORMAT	77
130	INACTIVE	DO-TERM	78
140	1030B	FORMAT	80
150	414B		87
160	INACTIVE	DO-TERM	89
170	INACTIVE	DO-TERM	97
180	INACTIVE	DO-TERM	108
190	INACTIVE	DO-TERM	111
200	INACTIVE	DO-TERM	115
210	563B		119
220	INACTIVE	DO-TERM	127
230	INACTIVE	DO-TERM	130
240	INACTIVE	DO-TERM	134
250	675B		139
260	1040B	FORMAT	142
270	INACTIVE	DO-TERM	149
280	INACTIVE	DO-TERM	157
290	1045B	FORMAT	159
300	1056B	FORMAT	163

--ENTRY POINTS--(LO=A)

-NAME-	--ADDRESS-	-ARGS-
TEST3R	14B	0

--STATISTICS--

PROGRAM-UNIT LENGTH	1313B = 715
CM STORAGE USED	053000B = 27328
COMPILE TIME	1.028 SECONDS

```
   FUNCTION F             74/810  OPT=0,ROUND= A/ S/ M/-D,-DS      FTN 5.1+642           87/03/04. 13.58.50
  DO=-LONG/-OT,ARG=-COMMON/-FIXED,CS= USER/-FIXED,DB=-TB/-SB/-SL/ ER/-ID/-PMD/-ST,-AL,PL=5000
  FTN5,I=PROG4.

                     REAL FUNCTION F(X1,X2)
                     REAL X1, X2
                     F = -X1**2-X2**2
                     RETURN
                     END

--VARIABLE MAP--(LO=A)
 -NAME---ADDRESS --BLOCK-----PROPERTIES-------TYPE---------SIZE

  F           20B                             REAL
  X1            1   DUMMY-ARG                 REAL
  X2            2   DUMMY-ARG                 REAL

--ENTRY POINTS--(LO=A)
 -NAME---ADDRESS--ARGS---

  F           7B     2

--STATISTICS--

  PROGRAM-UNIT LENGTH            23B =   19
  CM STORAGE USED            033000B =  26304
  COMPILE TIME               0.039 SECONDS
```

```
     FUNCTION C1          74/810  OPT=0,ROUND= A/ S/ M/-D,-DS      FTN 5.1+642         87/03/04. 13.58.60
    DO=-LONG/-OT,ARG=-COMMON/-FIXED,GS= USER/-FIXED,DB=-TB/-SB/-SL/ ER/-ID/-PMD/-ST,-AL,PL=5000
    FTN5,I=PROG4.

       1                 REAL FUNCTION C1(X1)
       2                 REAL X1
       3                 C1 = X1
       4                 RETURN
       5                 END

--VARIABLE MAP--(LO=A)
 -NAME---ADDRESS +-BLOCK-----PROPERTIES-------TYPE---------SIZE

  C1         16B                             REAL
  X1          1   DUMMY-ARG                  REAL

--ENTRY POINTS--(LO=A)
 -NAME--ADDRESS--ARGS---

  C1         7B      1

--STATISTICS--

  PROGRAM-UNIT LENGTH              21B =  17
  CM STORAGE USED               63300B =  26304
  COMPILE TIME                  0.029 SECONDS
```

```
      FUNCTION C2          74/810  OPT=0,ROUND= A/ S/ M/-D,-DS      FTN 5.1+642          87/03/04. 13.58.50
    DO=-LONG/-OT,ARG=-COMMON/-FIXED,CS= USER/-FIXED,DB=-TB/-SB/-SL/ ER/-ID/-PMD/-ST,-AL,PL=5000
    FTN5,I=PROG4.

      1                    REAL FUNCTION C2(X2)
      2                    REAL X2
      3                    C2 = X2
      4                    RETURN
      5                    END

--VARIABLE MAP--(LO=A)
 -NAME---ADDRESS --BLOCK-----PROPERTIES-------TYPE---------SIZE

  C2         16B                              REAL
  X2          1   DUMMY-ARG                   REAL

--ENTRY POINTS--(LO=A)
 -NAME---ADDRESS--ARGS---

  C2          7B     1

--STATISTICS--

  PROGRAM-UNIT LENGTH          21B =  17
  CM STORAGE USED          633008 =  26304
  COMPILE TIME             0.031 SECONDS
```

```
 FUNCTION C3          74/810  OPT=0,ROUND= A/ S/ M/-D,-DS     FTN 5.1+642          87/03/04. 13.58.50
DO--LONG/-OT,ARG=-COMMON/-FIXED,CS= USER/-FIXED,DB=-TB/-SB/-SL/ ER/-ID/-PMD/-ST,-AL,PL=5000
FTN5,I=PROG4.

                  REAL FUNCTION C3(X1,X2)
                  REAL X1, X2
                  C3 = -X1+X2+4
                  RETURN
                  END

--VARIABLE MAP--(LO=A)
 -NAME---ADDRESS --BLOCK-----PROPERTIES-------TYPE---------SIZE

  C3         21B                                REAL
  X1           1   DUMMY-ARG                    REAL
  X2           2   DUMMY-ARG                    REAL

--ENTRY POINTS--(LO=A)
 -NAME---ADDRESS--ARGS---

  C3         7B      2

--STATISTICS--

  PROGRAM-UNIT LENGTH          24B =  20
  CM STORAGE USED      [illegible]3008 =  26304
  COMPILE TIME                 0.036 SECONDS
```

```
  FUNCTION C4         74/810  OPT=0,ROUND= A/ S/ M/-D,-DS      FTN 5.1+642         87/03/04. 13.58.50
DO=-LONG/-OT,ARG=-COMMON/-FIXED,CS= USER/-FIXED,DB=-TB/-SB/-SL/ ER/-ID/-PMD/-ST,-AL,PL=5000
FTN5,I=PROG4.

                  REAL FUNCTION C4(X1,X2)
                  REAL X1, X2
                  C4 = X1/3-X2+4
                  RETURN
                  END

--VARIABLE MAP--(LO=A)
 -NAME---ADDRESS --BLOCK-----PROPERTIES--------TYPE---------SIZE

 C4          23B                                REAL
 X1            1   DUMMY-ARG                    REAL
 X2            2   DUMMY-ARG                    REAL

--ENTRY POINTS--(LO=A)
 -NAME---ADDRESS--ARGS---

 C4          7B      2

--STATISTICS--

 PROGRAM-UNIT LENGTH            26B =  22
 CM.STORAGE USED             63300B =  26304
 COMPILE TIME                0.036 SECONDS
```

```
  FUNCTION C5            74/810  OPT=0,ROUND= A/ S/ M/-D,-DS     FTN 5.1+642        87/03/04. 13.58.50
 DO=-LONG/-OT,ARG=-COMMON/-FIXED,CS= USER/-FIXED,DB=-TB/-SB/-SL/ ER/-ID/-PMD/-ST,-AL,PL=5000
 FTNB,I=PROG4.

                     REAL FUNCTION C5(X1,X2)
                     REAL X1, X2
                     C5 = X1**2+X2**2-10*X1-10*X2+41-
                     RETURN
                     END

--VARIABLE MAP--(LO=A)
 -NAME---ADDRESS --BLOCK-----PROPERTIES-------TYPE---------SIZE

  C5          25B                             REAL
  X1            1   DUMMY-ARG                 REAL
  X2            2   DUMMY-ARG                 REAL

--ENTRY POINTS--(LO=A)
 -NAME--ADDRESS--ARGS---

  C5          7B      2

--STATISTICS--

  PROGRAM-UNIT LENGTH            30B =  24
  CM STORAGE USED             63300B =  26304
  COMPILE TIME                0.044 SECONDS
```

```
  FUNCTION FACTOR      74/810  OPT=0,ROUND= A/ S/ M/-D,-DS      FTN 5.1+642          07/03/84. 13.50.50
DO=-LONG/-OT,ARG=-COMMON/-FIXED,CS= USER/-FIXED,DB=-TB/-SB/-SL/ ER/-ID/-PMD/-ST,-AL,PL=5000
FTN5,I=PROG4.

                 REAL FUNCTION FACTOR(AVE1,AVE2,D1,D2)
                 REAL AVE1, AVE2, D1, D2, S, C(5), R, F(2)
                 INTEGER I
                 R = 1
                 F(1) = -(AVE1+R*D1)**2-(AVE2+R*D2)**2
                 S = 0.1
                 DO 10 I=2, 70
                     R = 1+(I-1)*S
                     F(2) = -(AVE1+R*D1)**2 (AVE2+R*D2)**2
                     IF (F(2).GT.F(1)) THEN
                        R = R-S
                        GO TO 20
                     ELSE
                        F(1) = F(2)
                     ENDIF
            10   CONTINUE
            20   FACTOR = R
                 RETURN
                 END

--VARIABLE MAP--(LO=A)
 -NAME---ADDRESS --BLOCK-----PROPERTIES-------TYPE---------SIZE     -NAME---ADDRESS --BLOCK-----PROPERTIES-------TYPE---------SIZE

  AVE1        1   DUMMY-ARG                   REAL                   F          101B                             REAL          2
  AVE2        2   DUMMY-ARG                   REAL                   FACTOR      71B                             REAL
  C         73B              UND              REAL             5     I          103B                             INTEGER
  D1          3   DUMMY-ARG                   REAL                   R          100B                             REAL
  D2          4   DUMMY-ARG                   REAL                   S           72B                             REAL

--STATEMENT LABELS--(LO=A)
 -LABEL-ADDRESS-----PROPERTIES----DEF

    10  INACTIVE   DO-TERM         16
    20    62B                      17

--ENTRY POINTS--(LO=A)
 -NAME---ADDRESS--ARGS---

  FACTOR      7B      4

--STATISTICS--

  PROGRAM-UNIT LENGTH           107B =  71
  CM STORAGE USED             63300B =  26304
  COMPILE TIME                0.123 SECONDS
```

2
0
0
0.0000000001

X(K,1)	X(K,2)	F(X(K,1),X(K,2))
.000000000	.000000000	.0000000000
3.125000000	.000000000	-9.7656250000
1.953125000	3.125000000	-13.5803222656

NEWX(1)	NEWX(2)	VALNEW	REFLEXION FACTOR
3.165893555	1.948242188	-13.8185296208	7.9
2.419903755	3.162849426	-15.8595506776	7.9
4.451451623	1.430873775	-21.8628213172	7.9
3.502280648	2.382927019	-17.9443109100	7.9
4.745616311	1.288775564	-24.1766655207	7.9
4.869171505	1.105999402	-24.9320658265	7.9
4.983140411	1.080609876	-25.9994060613	7.9
4.970726684	1.045541505	-25.8012808022	7.9
5.030141056	1.041882108	-26.3878373662	7.9
5.077570981	1.092262354	-26.9747641226	7.9
5.062584978	1.065401178	-26.7648463129	7.9
5.148953402	1.151807341	-27.8383812916	7.9
5.513612195	1.569440840	-32.8630639880	7.9
5.832363638	1.890638520	-37.5909796138	7.9
9.812860576	6.298075153	-135.9579833177	7.9
10.102749520	6.587711382	-145.4634891252	7.9
10.976273398	7.566731158	-177.7339980975	7.9
10.898295331	7.461824666	-174.4531608508	7.9
11.040297259	7.628707140	-180.0853361896	7.9
11.062592888	7.664767875	-181.1298249274	7.9
11.088561089	7.668240493	-182.0344799449	7.9
11.084288672	7.667067080	-181.9522575753	7.9
11.098190433	7.697452585	-182.4206071871	7.9
11.111326712	7.701305722	-182.7718911113	7.9
11.108757326	7.702622760	-182.7348867127	7.9
11.133448901	7.710874781	-183.4112740079	7.9
11.230068601	7.733483342	-185.8212053808	7.9
11.738171883	7.887078354	-199.9906836446	7.9
11.830408171	7.908444359	-202.5020496760	7.9
11.921113429	7.938317491	-205.1298299838	7.9
11.909728088	7.932343122	-204.7636900906	7.9
11.938408229	7.941867775	-205.5526935590	7.9
11.947555680	7.947844650	-205.9123213083	7.9
11.944557913	7.945719506	-205.8009222421	7.9
11.947247781	7.947378345	-205.8975205585	7.9
11.947782770	7.947843917	-205.9170191760	7.9
11.950008040	7.951590999	-206.0501798889	7.9
11.963158423	7.964296136	-206.5471007368	7.9
11.983587125	7.977000587	-207.0083148158	7.9
11.991558400	7.995630959	-207.7356070205	7.9
11.993885007	7.997615383	-207.8103340172	7.9
11.993843566	7.997810676	-207.8109664367	7.9
11.995164200	7.998307729	-207.8508921419	7.9
11.99[illegible]	7.998174862	-207.8413466223	7.9
11.999096796	7.999615122	-207.9081062847	7.9
11.999825129	7.999830909	-207.9828272811	7.9
11.999613884	7.999865397	-207.9828180031	7.9

11.999940400	7.999979798	-207.9984400873	7.9
11.999990000	7.999963341	-207.9984274084	7.9
11.999989878	7.999979591	-207.9988704002	7.9
11.999999996	7.999791091	-207.9984900313	7.9
11.999987030	7.999830009	-207.9981544884	7.9
11.999982078	7.999983067	-207.9986350278	7.9
11.999784787	7.999996308	-207.9912561130	7.9
11.999983407	7.999996430	-207.9999870525	7.9
11.999783129	7.999999434	-207.9912063001	7.9
11.999780064	7.999999185	-207.9934274318	7.9
11.999983308	7.999930874	-207.9999988010	7.9
11.999915882	7.999933960	-207.9969054805	7.9
11.999930478	7.999941306	-207.9976083318	7.9
11.999933182	7.999930061	-207.9974053802	7.9
11.999930952	7.999940309	-207.9975810780	7.9
11.999942182	7.999942274	-207.9976881934	7.9
11.999942803	7.999943134	-207.9977140093	7.9
11.999942800	7.999942843	-207.9977120898	7.9
11.999947183	7.999940983	-207.9979156427	7.9
11.999961671	7.999971417	-207.9986727947	7.9
11.999969996	7.999977063	-207.9988180900	7.9
11.999972064	7.999986711	-207.9991180210	7.9
11.999972063	7.999987056	-207.9991368148	7.9
11.999975508	7.999991733	-207.9992789187	7.9
11.999975084	7.999990719	-207.9992535185	7.9
11.999975621	7.999991741	-207.9992827514	7.9
11.999975824	7.999991862	-207.9992847677	7.9
11.999975848	7.999991937	-207.9992913492	7.9
11.999975743	7.999991910	-207.9992883860	7.9
11.999976134	7.999992045	-207.9992999400	7.9
11.999977952	7.999992834	-207.9993529873	7.9
11.999979404	7.999993133	-207.9993958148	7.9
11.999990771	7.999999508	-207.9999826434	7.9
11.999994586	7.999998141	-207.9998403108	7.9
11.999999811	7.999999527	-207.9999638948	7.9
11.999999829	7.999999858	-207.9999936244	7.9
11.999999862	7.999999874	-207.9999946744	7.9
11.999999861	7.999999871	-207.9999946163	7.9
11.999999878	7.999999880	-207.9999951494	7.9
11.999999886	7.999999889	-207.9999954959	7.9
11.999999887	7.999999887	-207.9999954827	7.9
11.999999895	7.999999896	-207.9999958249	7.9
11.999999921	7.999999935	-207.9999970642	7.9
11.999999952	7.999999967	-207.9999983182	7.9
11.999999957	7.999999970	-207.9999986127	7.9
11.999999962	7.999999982	-207.9999988114	7.9
11.999999963	7.999999986	-207.9999989065	7.9
11.999999966	7.999999987	-207.9999989805	7.9
11.999999965	7.999999988	-207.9999989781	7.9
11.999999968	7.999999989	-207.9999990492	7.9
11.999999980	7.999999988	-207.9999993210	7.9
11.999999989	7.999999991	-207.9999995893	7.9
11.999999988	7.999999990	-207.9999995554	7.9
11.999999991	7.999999991	-207.9999996344	7.9
11.999999993	7.999999993	-207.9999997233	7.9
11.999999992	7.999999992	-207.9999996899	7.9
11.999999996	7.999999998	-207.9999998493	7.9
11.999999997	7.999999997	-207.9999998815	7.9
11.999999998	7.999999999	-207.9999999356	7.9
11.999999998	7.999999999	-207.9999999378	7.9
11.999999999	8.000000000	-207.9999999639	7.9
11.999999999	7.999999999	-207.9999999584	7.9
11.999999999	8.000000000	-207.9999999689	7.9
11.999999999	8.000000000	-207.9999999724	7.9
11.999999999	8.000000000	-207.9999999700	7.9
11.999999999	8.000000000	-207.9999999754	7.9

12.000000000	8.000000000	-207.9999999881	7.9
12.000000000	8.000000000	-207.9999999920	7.9
12.000000000	8.000000000	-207.9999999981	7.9
12.000000000	8.000000000	-207.9999999980	7.9
12.000000000	8.000000000	-207.9999999988	7.9
12.000000000	8.000000000	-207.9999999987	7.9
12.000000000	8.000000000	-207.9999999988	7.9
12.000000000	8.000000000	-207.9999999998	7.9
12.000000000	8.000000000	-207.9999999996	7.9
12.000000000	8.000000000	-207.9999999999	7.9
12.000000000	8.000000000	-207.9999999999	7.9
12.000000000	8.000000000	-207.9999999999	7.9
12.000000000	8.000000000	-208.0000000000	7.9

MINIMUM = (12.000000000, 8.000000000) ; F(MINIMUM) = -208.0000000000

NUMBER OF ITERATIONS = 128

```
PROGRAM TEST30     74/810  OPT=0,ROUND= A/ S/ M/-D,-DS      FTN 5.1+642          87/03/04. 15.28.29
DO  LONG/-OT,ARG=-COMMON/-FIXED,CS= USER/-FIXED,DB=-IB/-SB/-SL/ ER/-ID/-PMD/-ST,-AL,PL=5000
FIMS,I=PROG4.

            PROGRAM TEST30 (INPUT, OUTPUT)
      C
      C     *****************************************************************
      C     *  THIS PROGRAM WILL SOLVE THE SAME PROBLEM AS IN PROGRAM TEST3R,  *
      C     *  AND IN ORDER TO CONSTRUCT THE INITIAL COMPLEX, WE NEED TO SET   *
      C     *  UP THE UPPER BOUND FOR X1, X2.  IN THIS CASE, WE WILL ASSUME    *
      C     *  THAT THE UPPER BOUND FOR BOTH VARIABLES IS 13 (U = 13).         *
      C     *****************************************************************
      C
            INTEGER I, K, L, J, N, A
            REAL X(0:3,2), SUM(2), AVE(2), NEWX(2), R, WORST, BEST, B, W,
           $     VALUE(3), VALNEW, TEST(2), E, D(2), S(2), U, M(2)
            READ*, N
            DO 10 I=1, N
                  READ*, X(0,I)
                  X(1,I) = X(0,I)
      10    CONTINUE
            DO 20 I=1, N
                  READ*, S(I), M(I)
      20    CONTINUE
            U = 13
            DO 30 I=1, N
                  X(2,I) = U*S(I)
                  X(3,I) = U*M(I)
      30    CONTINUE
            DO 70 K=2, N+1
                  J = K-1
      40          IF ((C3(X(K,1),X(K,2)).LT.0).OR.(C4(X(K,1),X(K,2)).LT.0)
           $         .OR.(C5(X(K,1),X(K,2)).LT.0)) THEN
                     DO 60 I=1, N
                           SUM(I) = 0
                           DO 50 L=1, J
                                 SUM(I) = SUM(I)+X(L,I)
      50                   CONTINUE
                           X(K,I) = 0.5*(X(K,I)+SUM(I)/J)
      60             CONTINUE
                     GO TO 40
                  ENDIF
      70    CONTINUE
      C
      C     *****************************************************************
      C
            PRINT 80
      80    FORMAT (5(/), 10X, 'X(K,1)', 12X, 'X(K,2)', 8X,
           $       'F(X(K,1),X(K,2))')
            DO 100 K=1, N+1
                  PRINT 90, X(K,1), X(K,2), F(X(K,1), X(K,2))
      90          FORMAT (4X, F13.9, 5X, F13.9, 5X, F16.10)
      100   CONTINUE
            PRINT 110
      110   FORMAT (5(/), 9X, 'NEWX(1)', 10X, 'NEWX(2)', 12X, 'VALNEW',
           $        11X, 'REFLEXION FACTOR')
      C
      C     *****************************************************************
      C
```

```
      A = 0
      READ*, R, E
120   DO 130 K=1, N+1
            VALUE(K) = F(X(K,1),X(K,2))
130   CONTINUE
      WORST = VALUE(1)
      W = 1
      DO 140 K=2, N+1
            IF (VALUE(K).GT.WORST) THEN
               WORST = VALUE(K)
               W = K
            ENDIF
140   CONTINUE
      TEST(1) = WORST
C
C     ****************************************************************
C
      DO 160 I=1, N
            SUM(I) = 0
            DO 150 K=1, N+1
                  IF (K.NE.W) THEN
                     SUM(I) = SUM(I)+X(K,I)
                  ENDIF
150         CONTINUE
            AVE(I) = SUM(I)/N
            D(I) = AVE(I)-X(W,I)
160   CONTINUE
      DO 170 I=1, N
            NEWX(I) = AVE(I)+R*D(I)
170   CONTINUE
C
C     ****************************************************************
C
180   VALNEW = F(NEWX(1),NEWX(2))
      IF ((C1(NEWX(1)).GE.0).AND.(C2(NEWX(2)).GE.0).AND.(C3(NEWX(1),
     $   NEWX(2)).GE.0).AND.(C4(NEWX(1),NEWX(2)).GE.0).AND.(C5(NEWX(1),
     $   NEWX(2)).GE.0)) THEN
         DO 200 K=1, N+1
               IF ((K.NE.W).AND.(VALNEW.LT.VALUE(K))) THEN
                  DO 190 I=1, N
                        X(W,I) = NEWX(I)
190               CONTINUE
                  GO TO 220
               ENDIF
200      CONTINUE
      ENDIF
      DO 210 I=1, N
            NEWX(I) = 0.5*(AVE(I)+NEWX(I))
210   CONTINUE
      GO TO 180
C
C     ****************************************************************
C
220   TEST(2) = VALNEW
      A = A+1
      PRINT 230, NEWX(1), NEWX(2), VALNEW, R
230   FORMAT (4X, F13.9, 5X, F13.9, 5X, F16.10, 11X, F3.1)
```

```
              C
              C     ***************************************************************
              C
                    IF ((TEST(1)-TEST(2)).LT.E) THEN
   1                   DO 240 K=1, N+1
   1                         VALUE(K) = F(X(K,1),X(K,2))
   1          240      CONTINUE
   1                   BEST = VALUE(1)
   1                   B = 1
   1                   DO 250 K=2, N+1
   1                         IF (VALUE(K).LT.BEST) THEN
   2                            BEST = VALUE(K)
   2                            B = K
   2                         ENDIF
   1          250      CONTINUE
   1                   PRINT 260, X(B,1), X(B,2), BEST
   1          260      FORMAT (5(/), 6X, 'MINIMUM = ( ', F13.9, ', ', F13.9,
   1               $         ') ', ', 'F(MINIMUM) = ', F16.10)
   1                   PRINT*, ' '
   1                   PRINT 270, A
   1          270      FORMAT (6X, 'NUMBER OF ITERATIONS = ', I3)
   1                ELSE
   1                   GO TO 120
   1                ENDIF
                    STOP
                    END
```

--VARIABLE MAP--(LO=A)

-NAME	---ADDRESS	--BLOCK	-----PROPERTIES	-------TYPE	---------SIZE
A	10558			INTEGER	
AVE	10708			REAL	2
B	10778			REAL	
BEST	10768			REAL	
D	11108			REAL	2
E	11078			REAL	
I	10508			INTEGER	
J	10538			INTEGER	
K	10518			INTEGER	
L	10528			INTEGER	
M	11158			REAL	2
N	10548			INTEGER	

-NAME	---ADDRESS	--BLOCK	-----PROPERTIES	-------TYPE	---------SIZE
NEWX	10728			REAL	2
R	10748			REAL	
S	11128			REAL	2
SUM	10668			REAL	2
TEST	11058			REAL	2
U	11148			REAL	
VALNEW	11048			REAL	
VALUE	11018			REAL	3
W	11008			REAL	
WORST	10758			REAL	
X	10568			REAL	8

--PROCEDURES--(LO=A)

-NAME	------TYPE	--------ARGS	------CLASS-----
C1	REAL	1	FUNCTION
C2	REAL	1	FUNCTION
C3	REAL	2	FUNCTION
C4	REAL	2	FUNCTION
C5	REAL	2	FUNCTION
F	REAL	2	FUNCTION

```
--STATEMENT LABELS--(LO=A)
 -LABEL-ADDRESS-----PROPERTIES----DEF        -LABEL-ADDRESS-----PROPERTIES----DEF        -LABEL-ADDRESS-----PROPERTIES----DEF

     10  INACTIVE    DO-TERM        17           100  INACTIVE   DO-TERM        49           190  INACTIVE   DO-TERM        97
     20  INACTIVE    DO-TERM        20           110    707B     FORMAT         51           200  INACTIVE   DO-TERM       100
     30  INACTIVE    DO-TERM        25           120    275B                    58           210  INACTIVE   DO-TERM       104
     40    121B                     28           130  INACTIVE   DO-TERM        60           220    553B                   108
     50  INACTIVE    DO-TERM        34           140  INACTIVE   DO-TERM        68           230    717B     FORMAT        112
     60  INACTIVE    DO-TERM        36           150  INACTIVE   DO-TERM        79           240  INACTIVE   DO-TERM       119
     70  INACTIVE    DO-TERM        39           160  INACTIVE   DO-TERM        82           250  INACTIVE   DO-TERM       127
     80    674B      FORMAT         44           170  INACTIVE   DO-TERM        85           260    724B     FORMAT        129
     90    703B      FORMAT         48           180    441B                    89           270    735B     FORMAT        133

- ENTRY POINTS--(LO=A)
 -NAME---ADDRESS--ARGS---

  TEST30     14B     0

--STATISTICS--

  PROGRAM-UNIT LENGTH              1140B =  608
  CM STORAGE USED                 65300B = 27328
  COMPILE TIME                    0.913 SECONDS
```

```
   FUNCTION F           74/810  OPT=0,ROUND= A/ S/ M/-D,-DS      FTN 5.1+642         87/03/04. 15.28.29
  DO=-LONG/-OT,ARG=-COMMON/-FIXED,CS= USER/-FIXED,DB=-TB/-SB/-SL/ ER/-ID/-PMD/-ST,-AL,PL=5000
  FTN5,I=PROG4.

                        REAL FUNCTION F(X1,X2)
                        REAL X1, X2
                        F = -X1**2-X2**2
                        RETURN
                        END

--VARIABLE MAP--(LO=A)
 -NAME---ADDRESS --BLOCK-----PROPERTIES-------TYPE---------SIZE

  F           20B                              REAL
  X1            1   DUMMY-ARG                  REAL
  X2            2   DUMMY-ARG                  REAL

--ENTRY POINTS--(LO=A)
 -NAME---ADDRESS--ARGS---

  F           7B      2

--STATISTICS--

  PROGRAM-UNIT LENGTH                23B =  19
  CM STORAGE USED                 63300B =  26304
  COMPILE TIME                    0.043 SECONDS
```

```
   FUNCTION C1          74/810  OPT=0,ROUND= A/ S/ M/-D,-DS      FTN 5.1+642          87/03/04. 15.28.29
 DO=-LONG/-OT,ARG=-COMMON/-FIXED,CS= USER/-FIXED,DB=-TB/-SB/-SL/ ER/-ID/-PMD/-ST,-AL,PL=5000
 FTN5,I=PROG4.

                    REAL FUNCTION C1(X1)
                    REAL X1
                    C1 = X1
                    RETURN
                    END

--VARIABLE MAP--(LO=A)
 -NAME---ADDRESS --BLOCK-----PROPERTIES-------TYPE---------SIZE

  C1          16B                            RFAI
  X1            1   DUMMY-ARG                REAL

--ENTRY POINTS--(LO=A)
 -NAME---ADDRESS--ARGS---

  C1          7B      1

--STATISTICS--

  PROGRAM-UNIT LENGTH          21B =   17
  CM STORAGE USED          63300B =  26304
  COMPILE TIME             0.031 SECONDS
```

```
      FUNCTION C2          74/810  OPT=0,ROUND= A/ S/ M/-D,-DS      FTN 5.1+642          87/03/04. 15.28.29
     DO=-LONG/-OT,ARG=-COMMON/-FIXED,CS= USER/-FIXED,DB=-TB/-SB/-SL/ ER/-ID/-PMD/-ST,-AL,PL=5000
     FTN5,I=PROG4.

                          REAL FUNCTION C2(X2)
                          REAL X2
                          C2 = X2
                          RETURN
                          END

--VARIABLE MAP--(LO=A)
 -NAME---ADDRESS --BLOCK-----PROPERTIES-------TYPE----------SIZE

 C2          16B                              REAL
 X2           1   DUMMY-ARG                   REAL

--ENTRY POINTS--(LO=A)
 -NAME---ADDRESS--ARGS---

 C2          7B     1

--STATISTICS--

 PROGRAM-UNIT LENGTH           21B =  17
 CM STORAGE USED            63300B =  26304
 COMPILE TIME               0.031 SECONDS
```

```
   FUNCTION C3          74/810  OPT=0,ROUND= A/ S/ M/-D,-DS      FTN 5.1+642          87/03/04. 15.28.20
DO=-LONG/-OT,ARG=-COMMON/-FIXED,CS= USER/-FIXED,DB=-TB/-SB/-SL/ ER/-ID/-PMD/-ST,-AL,PL=5000
FTN5,I=PROG4.

                    REAL FUNCTION C3(X1,X2)
                    REAL X1, X2
                    C3 = -X1+X2+4
                    RETURN
                    END

--VARIABLE MAP--(LO=A)
 -NAME---ADDRESS --BLOCK-----PROPERTIES-------TYPE---------SIZE

  C3         21B                              REAL
  X1           1   DUMMY-ARG                  REAL
  X2           2   DUMMY-ARG                  REAL

--ENTRY POINTS--(LO=A)
 -NAME---ADDRESS--ARGS---

  C3          7B      2

--STATISTICS--
  PROGRAM-UNIT LENGTH               24B =  20
  CM STORAGE USED               63300B =  26304
  COMPILE TIME                  0.035 SECONDS
```

```
   FUNCTION C4          74/810  OPT=0,ROUND= A/ S/ M/-D,-DS     FTN 5.1+642          87/03/04. 15.28.29
  DO=-LONG/-OT,ARG=-COMMON/-FIXED,CS= USER/-FIXED,DB=-TB/-SB/-SL/ ER/-ID/-PMD/-ST,-AL,PL=5000
  FTN5,I=PROG4.

                     REAL FUNCTION C4(X1,X2)
                     REAL X1, X2
                     C4 = X1/3-X2+4
                     RETURN
                     END

--VARIABLE MAP--(LO=A)
 -NAME---ADDRESS --BLOCK-----PROPERTIES-------TYPE----------SIZE

  C4          23B                                 REAL
  X1            1   DUMMY-ARG                     REAL
  X2            2   DUMMY-ARG                     REAL

--ENTRY POINTS--(LO=A)
 -NAME---ADDRESS--ARGS---

  C4          7B     2

--STATISTICS--

  PROGRAM-UNIT LENGTH             26B =   22
  CM STORAGE USED              83300B =   28304
  COMPILE TIME                 0.039 SECONDS
```

```
   FUNCTION C5          74/810  OPT=0,ROUND= A/ S/ M/ D,-DS     FTN 5.1+642         87/03/04  15.28.29
DO=-LONG/-OT,ARG=-COMMON/-FIXED,CS= USER/-FIXED,DB=-TB/-SB/-SL/ ER/-ID/-PMD/-ST,-AL,PL=5000
FTN5,I=PROG4.

                       REAL FUNCTION C5(X1,X2)
                       REAL X1, X2
                       C5 = X1**2+X2**2-10*X1-10*X2+41
                       RETURN
                       END

--VARIABLE MAP--(LO=A)
 -NAME---ADDRESS --BLOCK-----PROPERTIES-------TYPE---------SIZE

  C5          25B                                 REAL
  X1            1   DUMMY-ARG                     REAL
  X2            2   DUMMY-ARG                     REAL

--ENTRY POINTS--(LO=A)
 -NAME---ADDRESS--ARGS---

  C5          7B     2

--STATISTICS--

  PROGRAM-UNIT LENGTH              30B =   24
  CM STORAGE USED               63300B =  26304
  COMPILE TIME                  0.044 SECONDS
```

2
8
8
0.495 0.791
0.335 0.409
1.3
0.0000000001

X(K,1)	X(K,2)	F(X(K,1),X(K,2))
.000000000	.000000000	.0000000000
3.217500000	2.177500000	-15.0938125000
2.693031250	1.714781250	-10.1928920488

NEWX(1)	NEWX(2)	VALNEW	REFLEXION FACTOR
3.435496289	2.262388477	-16.9210363710	1.3
3.429436515	2.302033224	-17.0603917735	1.3
3.572194563	2.350272903	-18.2843567157	1.3
3.543273052	2.367600045	-18.1603138902	1.3
3.568157952	2.363559863	-18.3181664704	1.3
3.605150437	2.343027622	-18.4868881162	1.3
3.587241623	2.353416464	-18.4068715139	1.3
3.632645518	2.328282878	-18.6170148193	1.3
3.660051239	2.312565672	-18.7439350610	1.3
3.699905702	2.291039923	-18.9381661347	1.3
3.741511309	2.267378683	-19.1399130149	1.3
3.799562952	2.235846035	-19.4356861199	1.3
3.862357988	2.200356537	-19.7593781166	1.3
3.947244379	2.154040857	-20.2206293395	1.3
4.041610884	2.100956927	-20.7486385499	1.3
4.168118168	2.032783724	-21.4887502612	1.3
4.307470718	1.963548895	-22.3706572691	1.3
4.490533070	1.853038506	-23.5986389542	1.3
4.701750736	1.734958670	-25.1165346351	1.3
4.971414444	1.586580890	-27.2322004911	1.3
5.288446967	1.410818136	-29.9369293451	1.3
5.267764195	1.421915937	-29.7711845465	1.3
5.328780391	1.388707285	-30.3030972550	1.3
5.357118007	1.370983517	-30.5782543095	1.3
5.359987425	1.369768012	-30.6057241194	1.3
5.363715719	1.367384105	-30.6391858026	1.3
5.364928389	1.367022560	-30.6512072948	1.3
5.365730808	1.366370462	-30.6580353443	1.3
5.365854109	1.366473043	-30.6596388986	1.3
5.366073281	1.366226490	-30.6613172836	1.3
5.366266449	1.366322863	-30.6636537628	1.3
5.366221175	1.366242442	-30.6629481124	1.3
5.366299234	1.366300906	-30.6639456362	1.3
5.366363006	1.366402158	-30.6649067876	1.3
5.366352139	1.366360849	-30.6646772553	1.3
5.366433414	1.366486283	-30.6658923508	1.3
5.366458104	1.366552603	-30.6663385975	1.3
5.366553336	1.366671913	-30.6676868254	1.3
5.366599717	1.366770026	-30.6684692300	1.3
5.366730476	1.366946745	-30.6703394092	1.3
5.366810386	1.367107701	-30.6716371801	1.3
5.366992359	1.367353779	-30.6742633360	1.3
5.367123537	1.367599932	-30.6763446362	1.3
5.367379779	1.367958757	-30.6800713798	1.3

5.367588747	1.368330281	-30.6833367107	1.3
5.367953206	1.368850181	-30.6886724443	1.3
5.368279534	1.369413747	-30.6937191600	1.3
5.368802280	1.370174152	-30.7014191312	1.3
5.369304918	1.371020848	-30.7091334637	1.3
5.370059884	1.372136380	-30.7203014000	1.3
5.370826557	1.373404414	-30.7320175934	1.3
5.371923074	1.375044811	-30.7483051034	1.3
5.373084158	1.376939315	-30.7659952509	1.3
5.374683724	1.379356006	-30.7898481270	1.3
5.376433146	1.382181365	-30.8164587047	1.3
5.378774995	1.385748867	-30.8515148311	1.3
5.381400522	1.389954860	-30.8914455294	1.3
5.384838754	1.395220981	-30.9431299939	1.3
5.388767873	1.401481080	-31.0029861938	1.3
5.393826713	1.409288289	-31.0793980852	1.3
5.399893184	1.418572178	-31.1690332798	1.3
5.407149883	1.430088858	-31.2824240018	1.3
5.415894777	1.443914012	-31.4168039110	1.3
5.426800247	1.460959471	-31.5856488860	1.3
5.439819429	1.481488981	-31.7875330264	1.3
5.456179417	1.506727516	-32.0401216398	1.3
5.475643352	1.537201671	-32.3445639794	1.3
5.499585927	1.574582877	-32.7247586027	1.3
5.528365429	1.618806489	-33.1866972774	1.3
5.563937701	1.675185563	-33.7636494133	1.3
5.606486894	1.742293085	-34.4704882821	1.3
5.659343228	1.824340550	-35.3563842082	1.3
5.722815629	1.923875948	-36.4519173853	1.3
5.800789722	2.045480962	-37.8331537828	1.3
5.828401316	2.088898093	-38.3337571502	1.3
5.844423984	2.113768441	-38.6253002662	1.3
5.847980102	2.118483941	-38.6912004068	1.3
5.849100473	2.121130844	-38.7111724039	1.3
5.849884711	2.122433282	-38.7258738317	1.3
5.849980802	2.122628900	-38.7274054797	1.3
5.850203315	2.122919868	-38.7316875899	1.3
5.850228816	2.122913652	-38.7319173454	1.3
5.850253175	2.122979788	-38.7325053918	1.3
5.850287796	2.122981627	-38.7329182842	1.3
5.850284646	2.123002600	-38.7329700504	1.3
5.850329180	2.123008023	-38.7335145790	1.3
5.850319339	2.123020624	-38.7334529347	1.3
5.850375757	2.123029603	-38.7341515801	1.3
5.850395538	2.123016562	-38.7343272897	1.3
5.850459055	2.123042764	-38.7351817348	1.3
5.850494287	2.123028624	-38.7355383058	1.3
5.850582156	2.123061716	-38.7367026158	1.3
5.850641150	2.123049448	-38.7373408188	1.3
5.850764215	2.123089326	-38.7389501833	1.3
5.850858367	2.123079360	-38.7400212873	1.3
5.851033625	2.123129707	-38.7422742353	1.3
5.851183481	2.123124302	-38.7440046823	1.3
5.851432472	2.123188843	-38.7471932897	1.3
5.851684609	2.123181613	-38.7499213281	1.3
5.852023145	2.123276047	-38.7544760564	1.3
5.852378794	2.123292183	-38.7587061889	1.3
5.852880134	2.123404367	-38.7662826884	1.3
5.853438275	2.123442172	-38.7717462974	1.3
5.854184665	2.123583682	-38.7812440117	1.3
5.855010181	2.123686654	-38.7910896044	1.3
5.856115880	2.123873290	-38.8049287432	1.3
5.857341828	2.123987884	-38.8198283479	1.3
5.858862980	2.124288850	-38.8400481447	1.3
[illegible]	2.124401838	-38.8624437884	1.3
5.863183830	2.124888114	-38.8921072418	1.3

5.888828764	2.125728915	-38.9257864541	1.3
5.886438494	2.126863114	-38.9883471147	1.3
5.873634487	2.126315836	-39.0198254858	1.3
5.878711443	2.127143183	-39.0839880118	1.3
5.884812888	2.127823738	-39.1581223876	1.3
5.892480818	2.129127772	-39.2942408732	1.3
5.901536434	2.130334884	-39.3884581939	1.3
5.912837138	2.132867986	-39.5873548198	1.3
5.926334174	2.133898879	-39.6749530586	1.3
5.943848836	2.136474888	-39.8841488839	1.3
5.963183112	2.139181442	-40.1348858886	1.3
5.987841232	2.147888865	-40.4461811835	1.3
6.017821478	2.147819686	-40.8214816852	1.3
6.054248862	2.152449782	-41.2888596587	1.3
6.085456368	2.158644738	-41.8589178777	1.3
6.152782174	2.166833186	-42.5508438881	1.3
6.148781988	2.165858393	-42.4988141129	1.3
6.167721838	2.168777828	-42.7443848883	1.3
6.167884484	2.168845686	-42.7438521113	1.3
6.168885821	2.168888287	-42.7598867928	1.3
6.168742126	2.168886749	-42.7575358829	1.3
6.168913438	2.168861376	-42.7598863578	1.3
6.169018335	2.169019092	-42.7614310489	1.3
6.169025081	2.169058709	-42.7616862554	1.3
6.169040578	2.169047124	-42.7618270807	1.3
6.169051683	2.169086888	-42.7621789794	1.3
6.169073467	2.169089792	-42.7624158874	1.3
6.169091189	2.169152845	-42.7628101643	1.3
6.169122184	2.169152804	-42.7632811034	1.3
6.169148852	2.169235072	-42.7639906890	1.3
6.169194295	2.169247013	-42.7649908542	1.3
6.169230830	2.169356143	-42.7659903699	1.3
6.169301102	2.169388036	-42.7665205304	1.3
6.169368152	2.169534689	-42.7679594854	1.3
6.169459333	2.169598147	-42.7693845836	1.3
6.169557876	2.169798315	-42.7714881149	1.3
6.169883783	2.169910836	-42.7736345342	1.3
6.169847286	2.170187932	-42.7766894895	1.3
6.170041782	2.170375774	-42.7799400531	1.3
6.170264138	2.170764175	-42.7843768433	1.3
6.170556227	2.171066630	-42.7892844607	1.3
6.170889792	2.171616920	-42.7958008784	1.3
6.171319542	2.172092654	-42.8031713907	1.3
6.171817840	2.172879391	-42.8127378288	1.3
6.172451029	2.173615857	-42.8237576004	1.3
6.173193585	2.174749085	-42.8378523868	1.3
6.174120351	2.175876474	-42.8542893243	1.3
6.175233885	2.177510779	-42.8751013180	1.3
6.176614915	2.179230730	-42.8996183769	1.3
6.178290240	2.181622519	-42.9303840524	1.3
6.180301253	2.184206824	-42.9668830323	1.3
6.182745178	2.187703798	-43.0123858390	1.3
6.185786384	2.191587938	-43.0667634536	1.3
6.189398888	2.196718623	-43.1341950336	1.3
6.193868778	2.202535311	-43.2151722371	1.3
6.199258863	2.210075404	-43.3157449851	1.3
6.205881235	2.218770712	-43.4359053700	1.3
6.213881816	2.229877129	-43.5846792333	1.3
6.223898856	2.242846992	-43.7646904987	1.3
6.235562965	2.259230814	-43.9863893925	1.3
6.239861224	2.264794056	-44.0851602098	1.3
6.242268999	2.268241204	-44.1108404110	1.3
6.242853308	2.268885845	-44.1210604039	1.3
6.243438630	2.269788602	-44.1324862287	1.3
6.243430985	2.269893430	-44.1319387255	1.3
6.243482054	2.269810498	-44.1331078621	1.3

6.243469604	2.269834039	-44.1330627530	1.3
6.243500239	2.269844153	-44.1334877118	1.3
6.243518789	2.269818588	-44.1336033307	1.3
6.243545211	2.269858515	-44.1341144812	1.3
6.243573289	2.269831281	-44.1343414602	1.3
6.243611850	2.269879088	-44.1350400087	1.3
6.243664136	2.269850855	-44.1354398682	1.3
6.243710608	2.269908709	-44.1364079729	1.3
6.243774050	2.269880753	-44.1370730170	1.3
6.243858960	2.269951839	-44.1384313359	1.3
6.243951894	2.269976082	-44.1394998673	1.3
6.244073940	2.270014630	-44.1414257870	1.3
6.244215835	2.269994428	-44.1431035985	1.3
6.244395549	2.270106510	-44.1458593414	1.3
6.244606740	2.270097059	-44.1484539939	1.3
6.244872307	2.270241348	-44.1524289128	1.3
6.245186090	2.270250705	-44.1563950617	1.3
6.245579085	2.270439685	-44.1621544723	1.3
6.246046842	2.270480198	-44.1681789803	1.3
6.246626809	2.270731948	-44.1765710669	1.3
6.247321751	2.270822374	-44.1856633095	1.3
6.248180301	2.271163213	-44.1979394129	1.3
6.249212403	2.271331894	-44.2116042351	1.3
6.250463334	2.271800287	-44.2296184542	1.3
6.252015707	2.272089832	-44.2500926006	1.3
6.253897772	2.272742175	-44.2765943415	1.3
6.256172167	2.273216434	-44.3072031372	1.3
6.258960011	2.274135619	-44.3462732380	1.3
6.262334901	2.274890034	-44.3919630755	1.3
6.266465332	2.276198186	-44.4496657151	1.3
6.271472253	2.277375091	-44.5178015258	1.3
6.277592852	2.279262167	-44.6031624582	1.3
6.277154399	2.279001165	-44.5965136580	1.3
6.279291572	2.279695928	-44.6265161647	1.3
6.279279290	2.279781422	-44.6267517332	1.3
6.279636519	2.279898789	-44.6342635166	1.3
6.279902987	2.280025237	-44.6356966102	1.3
6.279986122	2.279990197	-44.6363172328	1.3
6.280062181	2.280151922	-44.6382734107	1.3
6.280089559	2.280100844	-44.6383344850	1.3
6.280215208	2.280303425	-44.6408867684	1.3
6.280266085	2.280287411	-44.6413490027	1.3
6.280441110	2.280525384	-44.6447364752	1.3
6.280442728	2.280456813	-44.6444441108	1.3
6.280671801	2.280781870	-44.6488042105	1.3
6.280704304	2.280909462	-44.6497945309	1.3
6.281009078	2.281282058	-44.6552316139	1.3
6.281097048	2.281480817	-44.6573348463	1.3
6.281506449	2.281972006	-44.6647195090	1.3
6.281882221	2.282330071	-44.6685620762	1.3
6.282240808	2.283022327	-44.6787405138	1.3
6.282583099	2.283591650	-44.6852642637	1.3
6.283326106	2.284576980	-44.6994782336	1.3
6.283840036	2.285464900	-44.7100959394	1.3
6.284931234	2.286879017	-44.7301782559	1.3
6.285772222	2.288245430	-44.7469995749	1.3
6.286329128	2.288925484	-44.7571137805	1.3
6.286778312	2.289684643	-44.7662831002	1.3
6.287061884	2.290002075	-44.7712542483	1.3
6.287016020	2.289990326	-44.7706627694	1.3
6.287123534	2.290099507	-44.7724780841	1.3
6.287192186	2.290118996	-44.7734305979	1.3
6.287220368	2.290178916	-44.7740594226	1.3
6.287313843	2.290213239	-44.7753920396	1.3
6.287291454	2.290221129	-44.7751466511	1.3
6.287409614	2.290208832	-44.7768422891	1.3

6.287453084	2.290284729	-44.7773788175	1.3
6.287584107	2.290334200	-44.7793446466	1.3
6.287680272	2.290341757	-44.7803370599	1.3
6.287842026	2.290433202	-44.7830415979	1.3
6.287986304	2.290458743	-44.7847374840	1.3
6.288223526	2.290579154	-44.7885079680	1.3
6.288425971	2.290628118	-44.7912783606	1.3
6.288788125	2.290794596	-44.7965959682	1.3
6.289105627	2.290883221	-44.8009955205	1.3
6.289624053	2.291112837	-44.8085692249	1.3
6.290114569	2.291282607	-44.8154256319	1.3
6.290862101	2.291583688	-44.8263017786	1.3
6.291611902	2.291878422	-44.8368486703	1.3
6.291986582	2.291992646	-44.8420737846	1.3
6.291939903	2.291962484	-44.8415997641	1.3
6.291980976	2.291989645	-44.8422420588	1.3
6.292017819	2.292028636	-44.8428835093	1.3
6.292010063	2.292014634	-44.8427217150	1.3
6.292056795	2.292062962	-44.8435313377	1.3
6.292072725	2.292086313	-44.8438388481	1.3
6.292125784	2.292134439	-44.8447271639	1.3
6.292154451	2.292172015	-44.8452601840	1.3
6.292227727	2.292240216	-44.8464949733	1.3
6.292275986	2.292289294	-44.8473731402	1.3
6.292378484	2.292396818	-44.8491101449	1.3
6.292458685	2.292488247	-44.8505123654	1.3
6.292601559	2.292628741	-44.8529809176	1.3
6.292724848	2.292768675	-44.8551742096	1.3
6.292797499	2.292835507	-44.8563950213	1.3
6.292787111	2.292830260	-44.8562402240	1.3
6.292814228	2.292853751	-44.8566892356	1.3
6.292818053	2.292853969	-44.8567383667	1.3
6.292819170	2.292858842	-44.8567656030	1.3
6.292824309	2.292857556	-44.8568335590	1.3
6.292824136	2.292859299	-44.8568393703	1.3
6.292830791	2.292860489	-44.8569285848	1.3
6.292831564	2.292862933	-44.8569495217	1.3
6.292840332	2.292864846	-44.8570686412	1.3
6.292842652	2.292868310	-44.8571137272	1.3
6.292854398	2.292871317	-44.8572753459	1.3
6.292859176	2.292876272	-44.8573582006	1.3
6.292875162	2.292880923	-44.8575807337	1.3
6.292883771	2.292888062	-44.8577218254	1.3
6.292886061	2.292887165	-44.8577465304	1.3
6.292891257	2.292891962	-44.8578339167	1.3
6.292889453	2.292889807	-44.8578013382	1.3
6.292891750	2.292892094	-44.8578407340	1.3
6.292891837	2.292892389	-44.8578431733	1.3
6.292892491	2.292892804	-44.8578524017	1.3
6.292892433	2.292892758	-44.8578523722	1.3
6.292892869	2.292892871	-44.8578583759	1.3
6.292892720	2.292892734	-44.8578558790	1.3
6.292892807	2.292892811	-44.8578573194	1.3
6.292892857	2.292892858	-44.8578581878	1.3
6.292892899	2.292892900	-44.8578588907	1.3
6.292892919	2.292892921	-44.8578592388	1.3
6.292892935	2.292892936	-44.8578595125	1.3
6.292892964	2.292892965	-44.8578600060	1.3
6.292892970	2.292892970	-44.8578600978	1.3
6.292893007	2.292893009	-44.8578607529	1.3
6.292893020	2.292893021	-44.8578609702	1.3
6.292893071	2.292893073	-44.8578618547	1.3
6.292893096	2.292893097	-44.8578622699	1.3
6.292893166	2.292893168	-44.8578634821	1.3
6.292893209	2.292893209	-44.8578642036	1.3
6.292893202	2.292893204	-44.8578640985	1.3

6.292893212	2.292893213	-44.8578642598	1.3
6.292893215	2.292893216	-44.8578643183	1.3
6.292893217	2.292893217	-44.8578643445	1.3
6.292893217	2.292893218	-44.8578643547	1.3
6.292893218	2.292893218	-44.8578643598	1.3
6.292893218	2.292893218	-44.8578643655	1.3
6.292893218	7.292893218	-44.8578643640	1.3
6.292893218	2.292893219	-44.8578643712	1.3
6.292893219	2.292893219	-44.8578643712	1.3
6.292893219	2.292893219	-44.8578643730	1.3
6.292893219	2.292893219	-44.8578643724	1.3
6.292893219	2.292893219	-44.8578643732	1.3
6.292893219	2.292893219	-44.8578643740	1.3
6.292893219	2.292893219	-44.8578643740	1.3
6.292893219	2.292893219	-44.8578643751	1.3
6.292893219	2.292893219	-44.8578643753	1.3
6.292893219	2.292893219	-44.8578643755	1.3
6.292893219	2.292893219	-44.8578643755	1.3
6.292893219	2.292893219	-44.8578643756	1.3
6.292893219	2.292893219	-44.8578643757	1.3
6.292893219	2.292893219	-44.8578643757	1.3
6.292893219	2.292893219	-44.8578643758	1.3
6.292893219	2.292893219	-44.8578643760	1.3
6.292893219	2.292893219	-44.8578643760	1.3
6.292893219	2.292893219	-44.8578643761	1.3
6.292893219	2.292893219	-44.8578643761	1.3
6.292893219	2.292893219	-44.8578643762	1.3
6.292893219	2.292893219	-44.8578643762	1.3
6.292893219	2.292893219	-44.8578643762	1.3

MINIMUM = (6.292893219, 2.292893219) ; F(MINIMUM) = -44.8578643762

NUMBER OF ITERATIONS = 337

```
PROGRAM TEST4R      74/810  OPT=0,ROUND= A/ S/ M/-D,-DS      FTN 5.1+642          87/03/01. 15.12.12
DO=-LONG/-OT,ARG=-COMMON/-FIXED,CS= USER/-FIXED,DB=-TB/-SB/-SL/ ER/-ID/-PMD/-ST,-AL,PL=5000
FTN5,I=PROG5.

      PROGRAM TEST4R (INPUT, OUTPUT)
C
C     *****************************************************************
C     *  OBJECTIVE FUNCTION : F(X1,X2) = 3*(X1**2)+X2**2-2*X1*X2-X2   *
C     *  CONSTRAINTS        : 0.GE.X1.LE.1                             *
C     *                       0.GE.X2.LE.1.                            *
C     *  MINIMUM            : MIN = (0.25,0.75)   : F(0.25,0.75) = -0.375 *
C     *  START              : X(0) = (0.5,0.5)    : F(0.5,0.5) = 0    *
C     *                                                                *
C     *  BESIDES THE ABOVE GLOBAL MINIMUM, THERE ARE THREE LOCAL MINIMA : *
C     *               MIN1 = (0,0)         :        F(0,0) = 0        *
C     *               MIN2 = (0,0.5)       :        F(0,0.5) = -0.25  *
C     *               MIN3 = (1/3,1)       :        F(1/3,1) = -1/3   *
C     *****************************************************************
C
      INTEGER I, K, L, J, N, A
      REAL X(0:3,2), S, SUM(2), AVE(2), NEWX(2), R, WORST, BEST, B, W,
     $     VALUE(3), VALNEW, TEST(2), D(2), SEARCH(10), BNEW(2), E
      READ*, N
      DO 10 I=1, N
            READ*, X(0,I)
            X(1,I) = X(0,I)
 10   CONTINUE
      DO 80 K=2, N+1
            S = 0.1
            J = K-1
            DO 20 I=1, N
                  IF ((J-I).EQ.0) THEN
                        X(K,I) = 0
                  ELSE
                        X(K,I) = X(J,I)
                  ENDIF
 20         CONTINUE
            SEARCH(1) = F(X(K,1),X(K,2))
            BEST = SEARCH(1)
            DO 30 I=1, N
                  BNEW(I) = X(K,I)
 30         CONTINUE
            DO 60 L=2, 10
                  DO 40 I=1, N
                        IF ((J-I).EQ.0) THEN
                              X(K,I) = X(K,I)+S
                        ELSE
                              X(K,I) = X(J,I)
                        ENDIF
 40               CONTINUE
                  SEARCH(L) = F(X(K,1),X(K,2))
                  IF (SEARCH(L).LT.BEST) THEN
                     BEST = SEARCH(L)
                     DO 50 I=1, N
                           BNEW(I) = X(K,I)
 50                  CONTINUE
                  ENDIF
 60         CONTINUE
            DO 70 I=1, N
```

```
              X(K,I) = BNEW(I)
70        CONTINUE
80    CONTINUE
C
C     ********************************************************************
C
      PRINT 90
90    FORMAT (5(/), 8X, 'X(K,1)', 10X, 'X(K,2)', 10X,
     $       'F(X(K,1),X(K,2))')
      DO 110 K=1, N+1
             PRINT 100, X(K,1), X(K,2), F(X(K,1),X(K,2))
100          FORMAT (4X, F11.9, 5X, F11.9, 8X, F14.10)
110   CONTINUE
      PRINT 120
120   FORMAT (5(/), 7X, 'NEWX(1)', 9X, 'NEWX(2)', 11X, 'VALNEW', 7X,
     $       'REFLEXION FACTOR')
C
C     ********************************************************************
C
      A = 0
      READ*, E
130   DO 140 K=1, N+1
             VALUE(K) = F(X(K,1),X(K,2))
140   CONTINUE
      WORST = VALUE(1)
      W = 1
      DO 150 K=2, N+1
             IF (VALUE(K).GT.WORST) THEN
                WORST = VALUE(K)
                W = K
             ENDIF
150   CONTINUE
      TEST(1) = WORST
C
C     ********************************************************************
C
      DO 170 I=1, N
             SUM(I) = 0
             DO 160 K=1, N+1
                    IF (K.NE.W) THEN
                       SUM(I) = SUM(I)+X(K,I)
                    ENDIF
160          CONTINUE
             AVE(I) = SUM(I)/N
             D(I) = AVE(I)-X(W,I)
170   CONTINUE
      R = FACTOR(AVE(1),AVE(2),D(1),D(2))
C
C     ********************************************************************
C
      DO 180 I=1, N
             NEWX(I) = AVE(I)+R*D(I)
180   CONTINUE
190   VALNEW = F(NEWX(1),NEWX(2))
      IF ((C1(NEWX(1)).EQ.1).AND.(C2(NEWX(2)).EQ.1)) THEN
         DO 210 K=1, N+1
                IF ((K.NE.W).AND.(VALNEW.LT.VALUE(K))) THEN
```

```
2                    DO 200 I=1, N
2                         X(W,I) = NEWX(I)
2        200         CONTINUE
2                    GO TO 230
2                 ENDIF
1        210      CONTINUE
1              ENDIF
               DO 220 I=1, N
                    NEWX(I) = 0.5*(AVE(I)+NEWX(I))
         220   CONTINUE
               GO TO 190
         C
         C     ****************************************************************
         C
         230   TEST(2) = VALNEW
               A = A+1
               PRINT 240, NEWX(1), NEWX(2), VALNEW, R
         240   FORMAT (4X, F11.9, 5X, F11.9, 5X, F14.10, 11X, F3.1)
         C
         C     ****************************************************************
         C
               IF ((TEST(1)-TEST(2)).LT.E) THEN
1                 DO 250 K=1, N+1
1                      VALUE(K) = F(X(K,1),X(K,2))
1        250      CONTINUE
1                 BEST = VALUE(1)
1                 B = 1
1                 DO 260 K=2, N+1
1                      IF (VALUE(K).LT.BEST) THEN
2                         BEST = VALUE(K)
2                         B = K
2                      ENDIF
1        260      CONTINUE
1                 PRINT 270, X(B,1), X(B,2), BEST
1        270      FORMAT (5(/), 6X, 'MINIMUM = ( ', F11.9, ', ', F11.9,
1             $        ')  ;  ', 'F(MINIMUM) = ', F14.10)
1                 PRINT*, ' '
1                 PRINT 280, A
1        280      FORMAT (6X, 'NUMBER OF ITERATIONS = ', I3)
1              ELSE
1                 GO TO 130
1              ENDIF
               STOP
               END
```

--VARIABLE MAP--(LO=A)

-NAME-	--ADDRESS	--BLOCK-	----PROPERTIES-	------TYPE-	-------SIZE
A	1074B			INTEGER	
AVE	1110B			REAL	2
B	1117B			REAL	
BEST	1116B			REAL	
BNEW	1143B			REAL	2
D	1127B			REAL	2
E	1145B			REAL	
I	1067B			INTEGER	
J	1072B			INTEGER	
K	1070B			INTEGER	
L	1071B			INTEGER	
N	1073B			INTEGER	
NEWX	1112B			REAL	2
R	1114B			REAL	
S	1105B			REAL	
SEARCH	1131B			REAL	10

```
-NAME---ADDRESS --BLOCK-----PROPERTIES-------TYPE---------SIZE     -NAME---ADDRESS --BLOCK-----PROPERTIES-------TYPE----------SIZE

 SUM       1106B                             REAL            2      W         1120B                             REAL
 TEST      1125B                             REAL            2      WORST     1115B                             REAL
 VALNEW    1124B                             REAL                   X         1075B                             REAL          8
 VALUE     1121B                             REAL            3

--PROCEDURES--(LO=A)
 -NAME------TYPE--------ARGS-----CLASS-----

 C1         REAL          1      FUNCTION
 C2         REAL          1      FUNCTION
 F          REAL          2      FUNCTION
 FACTOR     REAL          4      FUNCTION

--STATEMENT LABELS--(LO=A)
 -LABEL-ADDRESS-----PROPERTIES----DEF       -LABEL-ADDRESS-----PROPERTIES----DEF       -LABEL-ADDRESS-----PROPERTIES----DEF

     10  INACTIVE   DO-TERM        23           110  INACTIVE   DO-TERM        68           200  INACTIVE   DO-TERM        115
     20  INACTIVE   DO-TERM        33           120    732B     FORMAT         70           210  INACTIVE   DO-TERM        118
     30  INACTIVE   DO-TERM        38           130    330B                    77           220  INACTIVE   DO-TERM        122
     40  INACTIVE   DO-TERM        46           140  INACTIVE   DO-TERM        79           230    576B                    127
     50  INACTIVE   DO-TERM        52           150  INACTIVE   DO-TERM        87           240    742B     FORMAT         130
     60  INACTIVE   DO-TERM        54           160  INACTIVE   DO-TERM        98           250  INACTIVE   DO-TERM        137
     70  INACTIVE   DO-TERM        57           170  INACTIVE   DO-TERM       101           260  INACTIVE   DO-TERM        145
     80  INACTIVE   DO-TERM        58           180  INACTIVE   DO-TERM       108           270    747B     FORMAT         147
     90    717B     FORMAT         63           190    477B                   109           280    760B     FORMAT         151
    100    726B     FORMAT         67

--ENTRY POINTS--(LO=A)
 -NAME---ADDRESS--ARGS---

 TEST4R      14B      0

--STATISTICS--

 PROGRAM-UNIT LENGTH            1171B =  633
 CM STORAGE USED              65300B =  27328
 COMPILE TIME                 0.891 SECONDS
```

```
  FUNCTION F          74/810  OPT=0,ROUND= A/ S/ M/-D,-DS     FTN 5.1+642        87/03/01. 15.12.12
DO=-LONG/-OT,ARG=-COMMON/-FIXED,CS= USER/-FIXED,DB=-TB/-SB/-SL/ ER/-ID/-PMD/-ST,-AL,PL=5000
FTN5,I=PROGS.

                  REAL FUNCTION F(X1,X2)
                  REAL X1, X2
                  F = 3*(X1**2)+X2**2-2*X1*X2-X2
                  RETURN
                  [illegible]

--VARIABLE MAP--(LO=A)
 -NAME---ADDRESS --BLOCK-----PROPERTIES-------TYPE----------SIZE

  F           25B                                 REAL
  X1            1    DUMMY-ARG                    REAL
  X2            2    DUMMY-ARG                    REAL

--ENTRY POINTS--(LO=A)
 -NAME---ADDRESS--ARGS---

  F           7B      2

--STATISTICS--

  PROGRAM-UNIT LENGTH             30B =  24
  CM STORAGE USED              63300B = 26304
  COMPILE TIME                 0.048 SECONDS
```

```
FUNCTION C1          74/810  OPT=0,ROUND= A/ S/ M/-D,-DS      FTN 5.1+642        87/03/01. 15.12.12
DO=-LONG/-OT,ARG=-COMMON/-FIXED,CS= USER/-FIXED,DB=-TB/-SB/-SL/ ER/-ID/-PMD/-ST,-AL,PL=5000
FTN5,I=PROG5.

      1                 REAL FUNCTION C1(X1)
      2                 REAL X1
      3                 IF ((X1.GE.0).AND.(X1.LE.1)) THEN
 1    4                    C1 = 1
 1    5                 ELSE
 1    6                    C1 = 0
 1    7                 ENDIF
      8                 RETURN
      9                 END

--VARIABLE MAP--(LO=A)
 -NAME---ADDRESS --BLOCK-----PROPERTIES-------TYPE---------SIZE

  C1          27B                             REAL
  X1            1   DUMMY-ARG                 REAL

--ENTRY POINTS--(LO=A)
 -NAME---ADDRESS--ARGS---

  C1          7B      1

--STATISTICS--

  PROGRAM-UNIT LENGTH           32B =  26
  CM STORAGE USED           63300B =  26304
  COMPILE TIME              0.047 SECONDS
```

```
 FUNCTION C2        74/810  OPT=0,ROUND= A/ S/ M/-D,-DS     FTN 5.1+642        87/03/01. 18.12.12
DO=-LONG/-OT,ARG=-COMMON/-FIXED,CS= USER/-FIXED,DB=-TB/-SB/-SL/ ER/-ID/-PMD/-ST,-AL,PL=5000
FTN5,I=PROG6.

                    REAL FUNCTION C2(X2)
                    REAL X2
                    IF ((X2.GE.0).AND.(X2.LE.1)) THEN
                       C2 = 1
                    ELSE
                       C2 = 0.
                    ENDIF
                    RETURN
                    END

--VARIABLE MAP--(LO=A)
 -NAME---ADDRESS --BLOCK-----PROPERTIES-------TYPE---------SIZE

  C2           27B                            REAL
  X2             1   DUMMY-ARG                REAL

--ENTRY POINTS--(LO=A)
 -NAME---ADDRESS--ARGS---

  C2           7B      1

--STATISTICS--

  PROGRAM-UNIT LENGTH              32B =  26
  CM STORAGE USED               63300B =  26304
  COMPILE TIME                  0.047 SECONDS
```

```
  FUNCTION FACTOR      74/810  OPT=0,ROUND= A/ S/ M/-D,-DS      FTN 5.1+642          87/03/01. 15.12.12
DO=-LONG/-OT,ARG=-COMMON/-FIXED,CS= USER/-FIXED,DB=-TB/-SB/-SL/ ER/-ID/-PMD/-ST,-AL,PL=5000
FTN5,I=PROG5.

       1                REAL FUNCTION FACTOR(AVE1,AVE2,D1,D2)
       2                REAL AVE1, AVE2, D1, D2, S, R, F(2)
       3                INTEGER I
       4                R = 1
       5                F(1) = 3*((AVE1+R*D1)**2)+(AVE2+R*D2)**2-2*(AVE1+R*D1)*
       6               $       (AVE2+R*D2)-(AVE2+R*D2)
       7                S = 0
       8                DO 10 I=2, 20
       9                      R = 1+(I-1)*S
      10                      F(2) = 3*((AVE1+R*D1)**2)+(AVE2+R*D2)**2-2*(AVE1+R*D1)*
      11               $              (AVE2+R*D2)-(AVE2+R*D2)
      12                      IF (F(2).GT.F(1)) THEN
 1    13                         R = R-S
 1    14                         GO TO 20
 1    15                      ELSE
 1    16                         F(1) = F(2)
 1    17                      ENDIF
      18           10   CONTINUE
      19           20   FACTOR = R
      20                RETURN
      21                END

--VARIABLE MAP--(LO=A)
 -NAME---ADDRESS --BLOCK-----PROPERTIES-------TYPE----------SIZE     -NAME---ADDRESS --BLOCK-----PROPERTIES-------TYPE---------SIZE

  AVE1          1   DUMMY-ARG                  REAL                   FACTOR     100B                                REAL
  AVE2          2   DUMMY-ARG                  REAL                   I          105B                                INTEGER
  D1            3   DUMMY-ARG                  REAL                   R          102B                                REAL
  D2            4   DUMMY-ARG                  REAL                   S          101B                                REAL
  F          103B                              REAL             2

--STATEMENT LABELS--(LO=A)
 -LABEL-ADDRESS-----PROPERTIES----DEF

    10  INACTIVE   DO-TERM        18
    20     70B                    19

--ENTRY POINTS--(LO=A)
 -NAME---ADDRESS--ARGS---

  FACTOR      7B      4

--STATISTICS--

  PROGRAM-UNIT LENGTH            111B =  73
  CM STORAGE USED              63300B =  26304
  COMPILE TIME                 0.146 SECONDS
```

2
0.5
0.5
0.0000000001

X(K,1)	X(K,2)	F(X(K,1),X(K,2))
.500000000	.500000000	.0000000000
.300000000	.500000000	-.3300000000
.200000000	.700000000	-.3700000000

NEWX(1)	NEWX(2)	VALNEW	REFLEXION FACTOR
.125000000	.625000000	-.3437500000	1.0
.153125000	.703125000	-.3537304688	1.0
.228125000	.778125000	-.3718429688	1.0
.275000000	.775000000	-.3737500000	1.0
.277343750	.814843750	-.3720863887	1.0
.282177734	.787021484	-.3727088499	1.0
.279211426	.771594238	-.3732353637	1.0
.272033081	.749672754	-.3735245392	1.0
.267822266	.752978516	-.3741443968	1.0
.270788574	.778405762	-.3740778488	1.0
.263610840	.758384277	-.3745772668	1.0
.263180542	.742819214	-.3742379629	1.0
.258999116	.746224976	-.3748768968	1.0
.259399414	.759790039	-.3748231494	1.0
.254757080	.749630737	-.3749284431	1.0
.255187988	.763195801	-.3748820445	1.0
.250546265	.753036499	-.3749932019	1.0
.251383972	.745402527	-.3749603916	1.0
.247172546	.748808289	-.3749813354	1.0
.248543835	.751612377	-.3749863432	1.0
.250731301	.754082513	-.3749876998	1.0
.251162520	.754046288	-.3749889810	1.0
.250977483	.753000774	-.3749939974	1.0
.250361228	.751990485	-.3749970845	1.0
.250792447	.751954260	-.3749973942	1.0
.250178191	.750944471	-.3749993477	1.0
.250807410	.750908247	-.3749991716	1.0
.249991155	.749898458	-.3749999913	1.0
.249559936	.749934682	-.3749994723	1.0
.249575222	.749402619	-.3749996093	1.0
.249894615	.749508466	-.3749998286	1.0
.250126886	.749853883	-.3749998933	1.0
.250223206	.750243874	-.3749998999	1.0
.250087495	.750288449	-.3749999443	1.0
.249855443	.749943032	-.3749999505	1.0
.249882250	.749828819	-.3749999694	1.0
.249977332	.749823941	-.3749999754	1.0
.250009742	.749889295	-.3749999801	1.0
.250023584	.749943811	-.3749999925	1.0
.250004977	.749972974	-.3749999989	1.0
.250037387	.750018327	-.3749999968	1.0
.250018800	.750047490	-.3749999985	1.0
.249986391	.750002137	-.3749999994	1.0
.249989808	.749972572	-.3749999995	1.0
.249983940	.749990949	-.3749999994	1.0
.249987056	.749976666	-.3749999996	1.0
.249988764	.749973598	-.3749999995	1.0

```
MINIMUM = (  .249987056,  .749976666)  ;  F(MINIMUM) =   -.3749999996
NUMBER OF ITERATIONS =  47
```

```
      PROGRAM TEST40     74/810   OPT=0,ROUND= A/ S/ M/-D,-DS     FTN 5.1+642         87/03/04. 13.48.38
DO=-LONG/-OT,ARG=-COMMON/-FIXED,CS= USER/-FIXED,DB=-TB/-SB/-SL/ ER/-ID/-PMD/-ST,-AL,PL=5000
FTNB,I=PROGS.

      PROGRAM TEST40 (INPUT, OUTPUT)
C
C     ****************************************************************
C     *  OBJECTIVE FUNCTION  : F(X1,X2) = 3*(X1**2)+X2**2-2*X1*X2-X2   *
C     *  CONSTRAINTS         : 0.GE.X1.LE.1                          *
C     *                        0.GE.X2.LE.1                          *
C     *  MINIMUM             : MIN = (0.25,0.75)   : F(0.25,0.75) = -0.375 *
C     *  START               : X(0) = (0.5,0.5)    : F(0.5,0.5) = 0  *
C     *                                                              *
C     *  BESIDES THE ABOVE GLOBAL MINIMUM, THERE ARE THREE LOCAL MINIMA : *
C     *                  MIN1 = (0,0)          :      F(0,0) = 0     *
C     *                  MIN2 = (0,0.5)        :      F(0,0.5) = -0.25 *
C     *                  MIN3 = (1/3,1)        :      F(1/3,1) = -1/3 *
C     ****************************************************************
C
      INTEGER I, K, N, A
      REAL X(0:3,2), SUM(2), AVE(2), NEWX(2), R, WORST, BEST, B, W, E,
     $     VALUE(3), VALNEW, TEST(2), D(2)
      READ*, N
      DO 10 I=1, N
            READ*, X(0,I)
            X(1,I) = X(0,I)
   10 CONTINUE
      DO 20 K=2, N+1
            READ*, X(K,1), X(K,2)
   20 CONTINUE
C
C     ****************************************************************
C
      PRINT 30
   30 FORMAT (5(/), 8X, 'X(K,1)', 10X, 'X(K,2)', 10X,
     $        'F(X(K,1),X(K,2))')
      DO 50 K=1, N+1
            PRINT 40, X(K,1), X(K,2), F(X(K,1),X(K,2))
   40       FORMAT (4X, F11.9, 5X, F11.9, 8X, F14.10)
   50 CONTINUE
      PRINT 60
   60 FORMAT (5(/), 7X, 'NEWX(1)', 9X, 'NEWX(2)', 11X, 'VALNEW', 7X,
     $        'REFLEXION FACTOR')
C
C     ****************************************************************
C
      A = 0
      READ*,R, E
   70 DO 80 K=1, N+1
            VALUE(K) = F(X(K,1),X(K,2))
   80 CONTINUE
      WORST = VALUE(1)
      W = 1
      DO 90 K=2, N+1
            IF (VALUE(K).GT.WORST) THEN
               WORST = VALUE(K)
               W = K
            ENDIF
   90 CONTINUE
```

```
      TEST(1) = WORST
C
C     ****************************************************************
C
      DO 110 I=1, N
            SUM(I) = 0
            DO 100 K=1, N+1
                  IF (K.NE.W) THEN
                        SUM(I) = SUM(I)+X(K,I)
                  ENDIF
100         CONTINUE
            AVE(I) = SUM(I)/N
            D(I) = AVE(I)-X(W,I)
110   CONTINUE
C
C     ****************************************************************
C
      DO 120 I=1, N
            NEWX(I) = AVE(I)+R*D(I)
120   CONTINUE
130   VALNEW = F(NEWX(1),NEWX(2))
      IF ((C1(NEWX(1)).EQ.1).AND.(C2(NEWX(2)).EQ.1)) THEN
         DO 150 K=1, N+1
               IF ((K.NE.W).AND.(VALNEW.LT.VALUE(K))) THEN
                  DO 140 I=1, N
                        X(W,I) = NEWX(I)
140               CONTINUE
                  GO TO 170
               ENDIF
150      CONTINUE
      ENDIF
      DO 160 I=1, N
            NEWX(I) = 0.5*(AVE(I)+NEWX(I))
160   CONTINUE
      GO TO 130
C
C     ****************************************************************
C
170   TEST(2) = VALNEW
      A = A+1
      PRINT 180, NEWX(1), NEWX(2), VALNEW, R
180   FORMAT (4X, F11.9, 5X, F11.9, 5X, F14.10, 11X, F3.1)
C
C     ****************************************************************
C
      IF ((TEST(1)-TEST(2)).LT.E) THEN
         DO 190 K=1, N+1
               VALUE(K) = F(X(K,1),X(K,2))
190      CONTINUE
         BEST = VALUE(1)
         B = 1
         DO 200 K=2, N+1
               IF (VALUE(K).LT.BEST) THEN
                  BEST = VALUE(K)
                  B = K
               ENDIF
200      CONTINUE
```

```
  1    113                PRINT 210, X(B,1), X(B,2), BEST
  1    114       210      FORMAT (5(/), 6X, 'MINIMUM = ( ', F11.8, ', ', F11.8,
  1    115          $          ')  ;  ', 'F(MINIMUM) = ', F14.10)
  1    116                PRINT*, ' '
  1    117                PRINT 220, A
  1    118       220      FORMAT (6X, 'NUMBER OF ITERATIONS = ', I3)
  1    119             ELSE
  1    120                GO TO 70
  1    121             ENDIF
       122             STOP
       123             END

--VARIABLE MAP--(LO=A)
 -NAME---ADDRESS --BLOCK-----PROPERTIES-------TYPE---------SIZE     -NAME---ADDRESS --BLOCK-----PROPERTIES-------TYPE---------SIZE

  A          674B                              INTEGER                NEWX       711B                              REAL          2
  AVE        707B                              REAL           2       R          713B                              REAL
  B          716B                              REAL                   SUM        705B                              REAL          2
  BEST       715B                              REAL                   TEST       725B                              REAL          2
  D          727B                              REAL           2       VALNEW     724B                              REAL
  E          720B                              REAL                   VALUE      721B                              REAL          3
  I          671B                              INTEGER                W          717B                              REAL
  K          672B                              INTEGER                WORST      714B                              REAL
  N          673B                              INTEGER                X          675B                              REAL          8

--PROCEDURES--(LO=A)
 -NAME------TYPE--------ARGS------CLASS-----

  C1        REAL          1       FUNCTION
  C2        REAL          1       FUNCTION
  F         REAL          2       FUNCTION

--STATEMENT LABELS--(LO=A)
 -LABEL-ADDRESS-----PROPERTIES----DEF     -LABEL-ADDRESS-----PROPERTIES----DEF     -LABEL-ADDRESS-----PROPERTIES----DEF

     10  INACTIVE   DO-TERM      23         90  INACTIVE   DO-TERM      55        160  INACTIVE   DO-TERM      89
     20  INACTIVE   DO-TERM      26        100  INACTIVE   DO-TERM      66        170    401B                  94
     30    521B     FORMAT       31        110  INACTIVE   DO-TERM      69        180    544B     FORMAT       97
     40    530B     FORMAT       35        120  INACTIVE   DO-TERM      75        190  INACTIVE   DO-TERM     104
     50  INACTIVE   DO-TERM      36        130    302B                  76        200  INACTIVE   DO-TERM     112
     60    534B     FORMAT       38        140  INACTIVE   DO-TERM      82        210    551B     FORMAT      114
     70    136B                  45        150  INACTIVE   DO-TERM      85        220    562B     FORMAT      118
     80  INACTIVE   DO-TERM      47

--ENTRY POINTS--(LO=A)
 -NAME---ADDRESS--ARGS---

  TEST40      14B      0

--STATISTICS--

  PROGRAM-UNIT LENGTH          746B =   486
  CM STORAGE USED            65300B = 27328
  COMPILE TIME               0.680 SECONDS
```

```
FUNCTION F          74/810  OPT=0,ROUND= A/ S/ M/-D,-DS      FTN 5.1+642         87/03/04. 13.48.30
DO=-LONG/-OT,ARG=-COMMON/-FIXED,CS= USER/-FIXED,DB=-TB/-SB/-SL/ ER/-ID/-PMD/-ST,-AL,PL=5000
FTN5,I=PROG5.

                REAL FUNCTION F(X1,X2)
                REAL X1, X2
                F = 3*(X1**2)+X2**2-2*X1*X2-X2
                RETURN
                END

--VARIABLE MAP--(LO=A)
 -NAME---ADDRESS --BLOCK-----PROPERTIES-------TYPE--------SIZE

  F          25B                               REAL
  X1           1   DUMMY-ARG                   REAL
  X2           2   DUMMY-ARG                   REAL

--ENTRY POINTS--(LO=A)
 -NAME---ADDRESS--ARGS---

  F           7B     2

--STATISTICS--

  PROGRAM-UNIT LENGTH             30B =  24
  CM STORAGE USED             63300B =  26304
  COMPILE TIME                0.046 SECONDS
```

```
   FUNCTION C1          74/810  OPT=0,ROUND= A/ S/ M/-D,-DS     FTN 5.1+642          87/03/04. 13.48.30
DO=-LONG/-OT,ARG=-COMMON/-FIXED,CS= USER/-FIXED,DB=-TB/-SB/-SL/ ER/-ID/-PMD/-ST,-AL,PL=5000
FTNB,I=PROBB.

                      REAL FUNCTION C1(X1)
                      REAL X1
                      IF ((X1.GE.0).AND.(X1.LE.1)) THEN
                         C1 = 1
                      ELSE
                         C1 = 0
                      ENDIF
                      RETURN
                      END

--VARIABLE MAP--(LO=A)
 -NAME---ADDRESS --BLOCK-----PROPERTIES-------TYPE---------SIZE

  C1          27B                             REAL
  X1            1   DUMMY-ARG                 REAL

--ENTRY POINTS--(LO=A)
 -NAME---ADDRESS--ARGS---

  C1           7B      1

--STATISTICS--

  PROGRAM-UNIT LENGTH             32B =  26
  CM STORAGE USED              83300B =  26304
  COMPILE TIME                 0.046 SECONDS
```

```
FUNCTION C2         74/810  OPT=0,ROUND= A/ S/ M/-D,-DS     FTN 5.1+642        87/03/04. 13.48.30
DO=-LONG/-OT,ARG=-COMMON/-FIXED,CS= USER/-FIXED,DB=-TB/-SB/-SL/ ER/-ID/-PMD/-ST,-AL,PL=5000
FTN5,I=PROGS.

                  REAL FUNCTION C2(X2)
                  REAL X2
                  IF ((X2.GE.0).AND.(X2.LE.1)) THEN
                     C2 = 1
                  ELSE
                     C2 = 0
                  ENDIF
                  RETURN
                  END

--VARIABLE MAP--(LO=A)
 -NAME---ADDRESS --BLOCK-----PROPERTIES------ TYPE --------SIZE

  C2         27B                              REAL
  X2           1   DUMMY-ARG                  REAL

--ENTRY POINTS--(LO=A)
 -NAME---ADDRESS--ARGS---

  C2          7B      1

--STATISTICS--

  PROGRAM-UNIT LENGTH         32B =  26
  CM STORAGE USED          63300B =  26304
  COMPILE TIME             0.051 SECONDS
```

```
2
0.5
0.5
0.495   0.335
0.791   0.469
1.3
0.0000000001
```

X(K,1)	X(K,2)	F(X(K,1),X(K,2))
.500000000	.500000000	.0000000000
.495000000	.335000000	.1806500000
.791000000	.469000000	.8860460000

NEWX(1)	NEWX(2)	VALNEW	REFLEXION FACTOR
.115950000	.350550000	-.2686240350	1.3
.064842500	.542632500	-.3059401163	1.3
.057115945	.442251789	-.2873977085	1.3
.043113720	.538557092	-.2893753714	1.3
.051938517	.604517750	-.2937787222	1.3
.078250334	.619098569	-.3143355259	1.3
.097036686	.550117654	-.3260030538	1.3
.117284823	.639176406	-.3392941222	1.3
.144744302	.562860029	-.3461374420	1.3
.175185802	.667188951	-.3637417218	1.3
.215449350	.583626999	-.3552353731	1.3
.261062831	.706720304	-.3718021173	1.3
.219863045	.754117586	-.3720101543	1.3
.282893077	.771818441	-.3727067087	1.3
.245082960	.799281524	-.3720141610	1.3
.278328635	.795566762	-.3730978239	1.3
.283497497	.782263814	-.3727543134	1.3
.279626059	.800223238	-.3728203495	1.3
.277506298	.802975136	-.3728380273	1.3
.275698598	.798032972	-.3731803400	1.3
.278370528	.788772018	-.3734553938	1.3
.273052267	.790588948	-.3736298508	1.3
.273426039	.778822247	-.3738731565	1.3
.269170668	.779419251	-.3741599376	1.3
.269020585	.764212091	-.3742533148	1.3
.263463464	.762707122	-.3746368982	1.3
.262434767	.742712068	-.3743017683	1.3
.255056232	.737756351	-.3746495831	1.3
.255132454	.760007305	-.3749235517	1.3
.244214485	.730908946	-.3747560183	1.3
.248174674	.750464279	-.3749523330	1.3
.251699914	.759188905	-.3749381356	1.3
.248923867	.753142860	-.3749423989	1.3
.246130781	.751203511	-.3749443257	1.3
.245150248	.747832241	-.3749457667	1.3
.245053643	.748476433	-.3749490428	1.3
.248217245	.749299906	-.3749618787	1.3
.248938465	.752095771	-.3749546562	1.3
.247101990	.751001467	-.3749679972	1.3
.246297115	.747622076	-.3749708218	1.3
.247328553	.749327196	-.3749817028	1.3
.248434632	.745189756	-.3749730266	1.3
.247639113	.746785796	-.3749881243	1.3
.248845495	.751783258	-.3749887038	1.3

.249432779	.749229057	-.3749993150	1.3
.250114153	.752924392	-.3749920765	1.3
.250979829	.750158231	-.3749974046	1.3
.250221279	.749168647	-.3749987940	1.3
.249452369	.748887054	-.3749990806	1.3
.249379306	.749049070	-.3749991204	1.3
.249345817	.749468675	-.3749991295	1.3
.249402290	.749738301	-.3749991739	1.3
.249510768	.749503484	-.3749995213	1.3
.249516938	.749125750	-.3749993803	1.3
.249610249	.749425845	-.3749996727	1.3
.249627499	.749905253	-.3749996453	1.3
.249769762	.749876234	-.3749998826	1.3
.249781613	.749320562	-.3749996920	1.3
.249979058	.749822717	-.3749999747	1.3
.249934728	.750193270	-.3749999246	1.3
.250078528	.750093637	-.3749999874	1.3
.250044078	.749919974	-.3749999807	1.3
.250088033	.750066634	-.3749999840	1.3
.250086465	.750093149	-.3749999850	1.3
.250075300	.750128179	-.3749999859	1.3
.250084496	.750133994	-.3749999869	1.3
.250066588	.750095144	-.3749999903	1.3
.250077798	.750068647	-.3749999878	1.3
.250063958	.750066632	-.3749999918	1.3
.250048991	.750096800	-.3749999929	1.3
.250043327	.750064261	-.3749999958	1.3
.250023020	.750098598	-.3749999932	1.3
.250012811	.750061447	-.3749999973	1.3
.250034402	.750016386	-.3749999973	1.3
.249997740	.750005970	-.3749999999	1.3
.250018320	.749978503	-.3749999977	1.3
.249973748	.749960841	-.3749999985	1.3
.249975155	.749984999	-.3749999987	1.3
.249994703	.750018002	-.3749999994	1.3
.250009915	.750029527	-.3749999994	1.3
.250015688	.750017419	-.3749999995	1.3
.250002553	.749988511	-.3749999998	1.3
.249995095	.749990682	-.3749999999	1.3
.249995420	.749999921	-.3749999999	1.3
.249994855	.749993568	-.3749999999	1.3

MINIMUM = (.249994855, .749993568) ; F(MINIMUM) = -.3749999999

NUMBER OF ITERATIONS = 85