Institute of Socio-Economic Planning
Discussion Paper Series

No. 199 (83-22)

A FORTRAN program of the (3^n-1)-method for solving systems of equations

by

M.Kojima* and Y.Yamamoto**

Feb. 1984

* Department of Information Sciences, Tokyo Institute of Technology, Oh-Okayama, Meguro, Tokyo 152, Japan
** Institute of Socio-Economic Planning, University of Tsukuba, Sakura, Ibaraki 305, Japan

University of Tsukuba
Sakura, Ibaraki 305
Japan

1. Introduction

In this paper we describe the FORTRAN program VARDIM of a simplicial fixed point algorithm. The program is designed to solve the system of n equations in n variables:

$$f_i(x_1,x_2,\ldots,x_n) = 0 \quad (i=1,2,\ldots,n).$$

We shall simply write this system as

(1.1) $\quad f(x) = 0,\ x \in R^n.$

Here R^n denotes the n-dimensional Euclidean space, $x = (x_1,x_2,\ldots,x^n) \in R^n$ a variable vector and $f = (f_1,f_2,\ldots,f_n):R^n \to R^n$ a continuous mapping from R^n into itself. The method we implemented is the (3^n-1)-method, a simplicial variable dimension method proposed and tested by the authors [4,5].

As many other algorithms for solving systems of nonlinear equations, the program iterates a procedure that computes an approximate solution. We call it a major cycle. To start the kth major cycle, we need to give a grid size ρ_k and a starting point w^k. The user has to provide the initial grid size IGRID > 0 and the initial point XO $\in R^n$. Then the program sets ρ_1 = IGRID and w^1 = XO before starting the 1st major cycle. For k=2,3,..., the program automatically computes ρ_k and w^k from the outputs of the (k-1)st major cycle.

For simplicity of notations we define

$$g^k(x) = f(w^k+x) \quad \text{for every } x \in R^n.$$

and consider the system of equations

(1.2) $\quad g^k(x) = 0,\ x \in R^n$

instead of the system (1.1). The kth major cycle implicitly defines a PL (piecewise linear) approximation $G^k:R^n \to R^n$ of the mapping $g^k:R^n \to R^n$ on a simplicial subdivision (or a triangulation) of R^n with the grid size ρ_k. See Fig. 1.1 for an example of a PL approximation G^k of a continuous mapping $g^k:R^1 \to R^1$. Then we start from the initial point $x^0 = 0$ and generate a sequence $\{x^p : p=0,1,\ldots\} \subset R^n$ to attain a solution of the system of PL equations

(1.3) $\quad G^k(x) = 0,\ x \in R^n.$

Suppose that the sequence successfully attains a solution $\bar{x} = x^q$ of the system (1.3) for some $q \geq 1$. Since x is a solution of the system (1.2) if and only if w^k+x is a solution of the system (1.1) and the

system (1.3) is a PL approximation of the system (1.2), $x = w^k+\bar{x}$ is an approximate solution of the system (1.1). The accuracy depends on the grid size ρ_k of the simplicial subdivision of R^n on which the PL approximation $G^k:R^n\to R^n$ has been constructed, i.e.,

$$\| g^k(\bar{x}) \| \to 0 \quad \text{as} \quad \rho_k \to 0+$$

or

$$\| f(w^k+\bar{x}) \| \to 0 \quad \text{as} \quad \rho_k \to 0+.$$

If the accuracy is not satisfactory, we set

$$w^{k+1} = w^k+\bar{x},$$

$$\rho_{k+1} = \text{RATE} \times \rho_k$$

and then restart the (k+1)st major cycle, where RATE denotes the reduction factor of the grid size which takes a positive value less than 1.0 depending on an estimate of the distance from the initial point w^{k+1} to an unknown solution of the system (1.1).

When the mapping f is continuously differentiable, an approximation A^k of the inverse of the Jacobian matrix of the mapping f at the point $w^k+\bar{x}$ is obtained as an output of the kth major cycle. The program includes the quasi-Newton procedure and, at the option, we can apply it to the system (1.1) with the initial point $z^0 = w^k+\bar{x}$ and the initial approximation $D^0 = A^k$ of the inverse of the Jacobian matrix of the mapping f. The quasi-Newton iteration

$$z^{p+1} = z^p - D^p f(z^p) \quad (p=0,1,2,\dots)$$

is repeated until either we get an approximate solution z^p with satisfactory accuracy or we find out at the pth iteration that more quasi-Newton iterations may be of no use or harmful. In the former case we stop the program. In the latter case we restart the (k+1)st major cycle with the initial point $w^{k+1} = z^p$. The program employs Broyden's formula [2] for updating the matrix D^p.

The kth major cycle does not necessarily succeed in finding a solution of the system (1.3). The generated sequence $\{x^p : p=0,1,\dots\}$ may diverge to the infinity or the computation of an approximate solution with satisfactory accuracy may require a high cost which the user cannot afford to pay. To protect the user against such cases, the program stops when

(a) the generated point x^p does not lie in the bounded region

$$\{x \in R^n : | w_i^k+x_i | \leq BX \quad (i=1,2,\dots,n)\},$$

(b) the number of pivot operations (=the number of minor cycles) in the kth major cycle exceeds a bound BP,

or

(c) the number of the major cycles exceeds a bound BC.

In Sections 2 and 3 we illustrate the basic idea of the (3^n-1)-method in R^2. The readers who want to understand the technical detail of the method for higher dimensional cases can refer to [5]. In Section 4 we give a sufficient condition for the global convergence of the method. In Section 5 we explain several parameters, some of which the user must supply, and we also give a specification for the user-supplied subroutine FUNCT. VARDIM already has this subroutine which contains five test problems. Some computational results of these test problems are also shown. Finally the list of VARDIM is given in Appendix.

adjust the value of the vector y to $-tg(x)$. We will specify the local movement in the set Ω to construct a piecewise smooth path P which runs through the set Ω. For this purpose, we impose a certain locally linear (or piecewise linear) constraint on the variable vectors x and y so that their 'dimension of freedom' sum up to n; hence the variable vector $(x,y,t) \in R^{2n+1}$ will have $(n+1)$-dimensional freedom.

Suppose that R^n is subdivided by a family P of a finite number of n-dimensional polyhedral cones. Fig. 2.1 shows an example, which will be used in the (3^n-1)-method in R^2. We have two extreme cases of the locally linear constraint. One is fixing x to the origin (= the common vertex of all the cones of P) of R^n and letting y move freely in Y_0. This yields a constraint

$$x = 0 \quad (\text{0-dimensional freedom}),$$
$$y \in Y_0 \quad (\text{n-dimensional freedom}),$$

which is satisfied by the starting point s^0 of the path P. The other extreme case is obtained by

$$x \in X \quad (\text{n-dimensional freedom}),$$
$$y = y^* \quad (\text{0-dimensional freedom}),$$

where $X \in P$ and y^* is a vertex of Y_0. We have various cases of locally linear constraint between the above two extreme cases. That is, let X be a k-dimensional face of a cone of P and Y an $(n-k)$-dimensional face of Y_0, then the constraint $(x,y) \in X \times Y$ gives n-dimensional freedom. The whole constraint imposed on the system (2.3) is represented as the union of such linear constraints. The variable dimension algorithms make it the chief aim to approach a solution to the system (1.1) while keeping the variable vector x in lower dimensional faces of cones of P. Since y serves as a slack variable vector in the system (2.3), the work to trace the set of solutions is greatly reduced while x moves along lower dimensional faces. As the algorithms go, the allocation of 'freedom' to each of the variable vectors gradually changes from the first extreme case to the opposite extreme case.

In the case of the (3^n-1)-method, we construct a conical subdivision P having 3^n-1 rays extending in all directions. (The numbers $(n+1)$, $2n$ and 2^n attached to the variable dimension methods referred above are the numbers of rays of conical subdivisions used in the

methods, respectively.) For simplicity of discussion, we here restrict ourselves to the (3^n-1)-method in R^2. Let Y_0 be the octagon illustrated in Fig. 2.2, and $\mathcal{P}$ the conical subdivision of R^2 illustrated in Fig. 2.1. Furthermore we name faces of Y_0 and faces of cones in $\mathcal{P}$ as in Fig. 2.3 and 2.4, respectively. Here X_0 is the origin, $X_1, X_2, \ldots, X_8$ are 1-dimensional rays with a common vertex X_0, and $X_9, X_{10}, \ldots, X_{16}$ are 2-dimensional cones which partition R^2. Let

$$\begin{aligned}
\bar{\mathcal{P}} &= \{ \text{ all the faces of the pieces } X_9, \ldots, X_{16} \}, \\
&= \{ X_i : i=0,1,\ldots,16 \}, \\
\mathcal{D} &= \{ Y_0 \}, \\
\bar{\mathcal{D}} &= \{ \text{ all the faces of } Y_0 \} \\
&= \{ Y_i : i=0,1,\ldots,16 \}.
\end{aligned}$$

Let

$$L = \bigcup_{i=0}^{16} X_i \times Y_i .$$

We now add the constraint $(x,y) \in L$ to the system (2.3);

$$y + t\, g(x) = 0, \quad (x,y) \in L, \quad t \in R_+ . \tag{2.4}$$

Let Π denote the solution set of the system (2.4). We see from Fig. 2.3 and Fig. 2.4 that

$$\dim X_i + \dim Y_i = 2 \quad (i=0,1,\ldots,16).$$

Hence the constraint set L is the union of 2-dimensional polyhedral sets in R^4. Fig. 2.5 shows the adjacency relation among the polyhedral sets in the family $\{ X_i \times Y_i : i=0,1,\ldots,16 \}$. From this figure we see that L forms a 2-dimensional PL manifold. In fact, each point (x,y) in L has an open neighborhood which is homeomorphic to a 2-dimensional open ball in R^2. We furthermore see that if C is a facet (i.e. (n-1)-dimensional=1-dimensional face) of a polyhedral set $X_i \times Y_i$ then it is a facet of exactly one other polyhedral set $X_j \times Y_j$. For example, $X_0 \times Y_0$ and $X_2 \times Y_2$ share a common facet $X_0 \times Y_2$.

Since the variable vector $(x,y) \in R^4$ is restricted within the 2-dimensional PL manifold L and the scalar variable t can vary along the half-line R_+ in the system (2.4), we have 3-dimensional freedom in the domain of the variables x_1, x_2, y_1, y_2 and t. On the one hand, the system (2.4) consists of two scaler equations

$$y_1 + t\, g_1(x_1,x_2) = 0,$$

$$y_2 + t\, g_2(x_1,x_2) = 0.$$

Therefore, under a ceratin regularity assumption, the solution set Π of the system (2.4) is a disjoint union of 1-dimensional curves; each connected component of Π is either a path or a loop. Here we call a set which is homeomorphic to one of the intervals $[0, 1]$, $[0, 1)$ and $(0, 1)$ a path, and a set which is homeomorphic to the unit circle

$$\{(z_1,z_2) \in R^2 : z_1^2 + z_2^2 = 1\}$$

a loop. See Fig. 2.6.

By the construction, the system is subdivided into the family of the subsystems

(2.5-i) $\quad y + t\, g(x) = 0,\ x \in X_i,\ y \in Y_i,\ t \in R_+$

$(i=0,1,2,\ldots,16)$, and the solution set Π of the system (2.4) can be represented as the union of all the solution sets of the subsystems, i.e.,

$$\Pi = \bigcup_{i=0}^{16} \{(X_i \times Y_i \times R_+) \cap \Pi\}.$$

Obviously, the point $s^0 = (x^0,y^0,t^0) = (0,0,0)$ satisfies the subsystem (2.5-0). Hence $s^0 \in \Pi$. Let P be the connected component of the set Π which contains the point s^0. Then the set P forms a piecewise smooth path, which is traced by the smooth version of the (3^n-1)-method starting from the point s^0 until the variable t becomes sufficiently large. In general, the path runs through several pieces of $X_i \times Y_i \times R_+$ $(i=0,1,\ldots,16)$. If we focus our attention to the projection of the path P onto the x-space, the dimension of the X_i's through which the path runs varies.

In the beginning of the method, we deal with the subsystem

$$\begin{aligned} &y + t\, g(0) = 0, \\ &x \in X_0, \text{ i.e., } x = 0, \\ &y \in Y_0,\ t \geq 0, \end{aligned} \tag{2.5-0}$$

which has the solution set

$$\begin{aligned} &(X_0 \times Y_0 \times R_+) \cap P \\ &= \{(x,y,t) : x = (0,0),\ y = -t\, g(0),\ 0 \leq t \leq t^1\}, \end{aligned}$$

where

$$t^1 = \sup\{\, t \geq 0 : -t\, g(0) \in Y_0 \,\}.$$

We may assume that $g(0) \neq 0$ since otherwise w^0 is a trivial solution of the system (1.1). Then t^1 is finite and the set $(X_0 \times Y_0 \times R_+) \cap P$

forms the line segment with the endpoints $s^0 = (x^0,y^0,t^0) = (0,0,0)$ and

$$s^1 = (x^1,y^1,t^1) = (0,-t^1 g(0),t^1).$$

We see that the point y^1 lies on the boundary of the polyhedral set Y_0. More precisely, it lies on one of the facets Y_i $(i=1,2,\ldots,8)$ under the regularity assumption on the system (2.4). Hence the point (x^1,y^1) is on the common facet $X_0 \times Y_i$ of the pieces $X_0 \times Y_0$ and $X_i \times Y_i$ for some $i=1,2,\ldots,8$. This implies that the path P is now entering the new piece $X_i \times Y_i \times R_+$ for some $i=1,2,\ldots,8$. The variable vector x has not been changed so far. But now it will be able to vary along one of the $3^n-1 = 8$ rays X_i's $(i=1,2,\ldots,8)$.

Now we shall show, by an example, how to trace P. Consider the system of equations

$$\begin{aligned} f_1(x_1,x_2) &= x_1 + x_2 - x_2^2 - 1.4 = 0, \\ f_2(x_1,x_2) &= \quad x_2 \quad\quad - 1.2 = 0, \end{aligned} \tag{2.6}$$

which has a solution $x = (1.64, 1.2)$. Let $w^0 = (0,0)$. Then the family of the subsystems (2.5-i) $(i=0,1,\ldots,16)$ turns out to be

$$\begin{aligned} & y_1 + t \quad (x_1 + x_2 - x_2^2 - 1.4) = 0, \\ & y_2 + t \quad (\quad x_2 \quad\quad - 1.2) = 0, \end{aligned} \tag{2.7-i}$$

$$x = (x_1,x_2) \in X_i, \quad y = (y_1,y_2) \in Y_i, \quad t \in R_+,$$

$(i=0,1,\ldots,16)$. We set the parameter γ, which determines the shape of the polyhedral set Y_0, to be 0.2. See Fig. 2.2.

(i) We start with the subsystem

$$\begin{aligned} & y_1 + t\, g_1(0) = 0, \quad \text{i.e.,} \quad y_1 - 1.4\, t = 0, \\ & y_2 + t\, g_2(0) = 0, \quad \text{i.e.,} \quad y_2 - 1.2\, t = 0, \\ & (x_1,x_2) \in X_0, \ \text{i.e.,} \ x_1 = x_2 = 0, \\ & (y_1,y_2) \in Y_0, \\ & t \in R_+, \quad \text{i.e.,} \quad t \geq 0. \end{aligned} \tag{2.7-0}$$

The point

$$s^0 = (x^0,y^0,t^0) = (0,0,0,0,0)$$

is a known solution of the system (2.7-0). We can easily compute the solution set

$$\begin{aligned} & (X_0 \times Y_0 \times R_+) \cap P \\ & = \{ (x_1,x_2,y_1,y_2,t) : 0 \leq t \leq 5/13, \ x_1=0, \ x_2=0, \\ & \qquad\qquad y_1 = 1.4\, t, \quad y_2 = 1.2\, t \ \}, \end{aligned}$$

which forms a curve starting from s^0 and ending at the point

$$s^1 = (x^1,y^1,t^1) = (0,0,7/13,6/13,5/13).$$

Fig. 2.7 illustrates the projection of this curve onto the y-space.

(ii) The point (x^1,y^1) lies on the common facet $X_0 \times Y_2$ of the pieces $X_0 \times Y_0$ and $X_2 \times Y_2$. Now the path P will move into the piece $X_2 \times Y_2 \times R_+$, so that we need to solve the nonlinear subsystem

$$
\begin{aligned}
& y_1 + t\,(x_1 + x_2 - x_2^2 - 1.4) = 0, \\
& y_2 + t\,(\quad x_2 \quad - 1.2) = 0, \\
& (x_1,x_2) \in X_2 \text{ , i.e., } x_1 = x_2 \geq 0, \\
& (y_1,y_2) \in Y_2 \text{ , i.e., } y_1 + y_2 = 1,\ y_1 \geq 1/5,\ y_2 \geq 1/5.
\end{aligned} \tag{2.7-10}
$$

By a simple calculation, we see that the solution set of (2.7-10) is

$$
\begin{aligned}
& (X_2 \times Y_2 \times R_+) \cap P \\
& = \{(x_1,x_2,y_1,y_2,t) : x_1 = x_2 = 1.5 - 0.5\sqrt{4t - 1.4t^2}, \\
& \qquad\qquad\qquad\quad y_1 = 1 - y_2, \\
& \qquad\qquad\qquad\quad y_2 = 0.5\sqrt{4t - 1.4t^2} - 0.3t, \\
& \qquad\qquad\qquad\quad t^1 \leq t \leq t^2 \qquad\qquad\qquad\qquad\qquad\qquad \},
\end{aligned}
$$

and that the solution set forms a curve with the endpoints s^1 and $s^2 = (x^2,y^2,t^2)$, where

$$
\begin{aligned}
& x_1^2 = x_2^2 = \sqrt{4.4} - 1, \\
& y_1^2 = 0.8,\ y_2^2 = 0.2, \\
& t^2 = 1 + \sqrt{110}\ /\ 11.
\end{aligned}
$$

Fig. 2.8 illustrates the projections of this curve onto the x-space and y-space.

(iii) The point (x^2,y^2) lies on a common facet $X_2 \times Y_9$ of the pieces $X_2 \times Y_2$ and $X_9 \times Y_9$, and the path P will move into the piece $X_9 \times Y_9 \times R_+$. The subsystem to be considered is

$$
\begin{aligned}
& 0.8 + t\,(x_1 + x_2 - x_2^2 - 1.4) = 0, \\
& 0.2 + t\,(\quad x_2 \quad - 1.2) = 0, \\
& x \in X_9, \text{ i.e., } 0 \leq x_2 \leq x_1, \\
& y \in Y_9, \text{ i.e., } y_1 = 0.8,\ y_2 = 0.2, \\
& t \in R_+, \text{ i.e., } t \geq 0.
\end{aligned} \tag{2.7-1}
$$

We see that the solution set of (2.7-1)

$$
\begin{aligned}
& (X_9 \times Y_9 \times R_+) \cap P \\
& = \{(x_1,x_2,y_1,y_2,t) : x_1 = 1.64 - 1.06/t + 0.04/t^2, \\
& \qquad\qquad\qquad\quad x_2 = 1.2 - 0.2/t, \\
& \qquad\qquad\qquad\quad y_1 = 0.8,\ y_2 = 0.2,\ t^2 \leq t \ \}
\end{aligned}
$$

forms an unbounded curve with the endpoint $s^2 = (x^2,y^2,t^2)$, and converges $(1.64,1.2,0.8,0.2,+\infty)$ as $t \to +\infty$. Thus we obtain a solution $x =$

(1.64, 1.2). Fig. 2.9 illustrates the projection of this curve onto the x-space.

In the above example we have been able to compute the path P accurately. In nonlinear and higher dimensional cases, however, such an exact computation of the path P is generally impossible and we need a numerical method for approximating the path P. We may utilize various predictor-corrector techniques, which have been developed for smooth continuation methods (see [7]). Here we do not go into the detail.

If we define the dual operator " d " from $\bar{\mathcal{P}} \cup \bar{\mathcal{D}}$ onto itself by

$$Z^d = \begin{cases} X_i & \text{if } Z = Y_i \\ Y_i & \text{if } Z = X_i, \end{cases}$$

then the triplet $(\mathcal{P}, \mathcal{D}; d)$ satisfies the following conditions for $n = 2$:

(i) $\mathcal{P}$ is a subdivision of R^n.

(ii) $\mathcal{D}$ is a subdivision of a polyhedral subset of R^n such that each piece of $\mathcal{D}$ is bounded.

(iii) d is an operator from $\bar{\mathcal{P}} \cup \bar{\mathcal{D}}$ into itself such that $X^d \in \bar{\mathcal{D}}$ for every $X \in \bar{\mathcal{P}}$ and $Y^d \in \bar{\mathcal{P}}$ for every $Y \in \bar{\mathcal{D}}$.

(iv) If $Z \in \bar{\mathcal{P}} \cup \bar{\mathcal{D}}$ then $(Z^d)^d = Z$ and $\dim Z + \dim Z^d = n$.

(v) If $X_1, X_2 \in \bar{\mathcal{P}}$ and X_1 is a face of X_2, X_2^d is a face of X_1^d.

(v)' If $Y_1, Y_2 \in \bar{\mathcal{D}}$ and Y_1 is a face of Y_2, Y_2^d is a face of Y_1^d.

(vi) $\{ 0 \} \in \bar{\mathcal{P}}$, and $0 \in \text{int } Y_0$ (the interior of Y_0), where $Y_0 = \{ 0 \}^d$.

(vii) There exists a positive number α such that $x^t y \geq \alpha \| x \|$ whenever $x \in X$, $y \in X^d$ and $X \in \bar{\mathcal{P}}$.

In general we call a triplet $(\mathcal{P}, \mathcal{D}; d)$ satisfying the conditions (i) - (v)' a primal-dual pair of subdivided manifolds (abbreviated by PDM). A PDM has been introduced by the authors [4] to give a unified framework for various variable dimension algorithms. We call $\mathcal{P}$ (resp. $\mathcal{D}$) the primal (resp. dual) subdivided manifold, d the dual operator and Z^d the dual of Z for each $Z \in \bar{\mathcal{P}} \cup \bar{\mathcal{D}}$. Define

$$L = \bigcup \{ X \times X^d : X \in \bar{\mathcal{P}} \} \subset R^{2n}.$$

Then L forms an n-dimensional piecewise linear manifold. See [4,5] for the general definition of the PDM and its fundamental properties.

In a higher dimensional case of the (3^n-1)-method, using a PDM

(P, $\mathcal{D}$; d) with the additional conditions (vi), (vii) above and the primal subdivided manifold P having 3^n-1 rays, we construct the system of equations

(2.8) $\quad y + t\, g(x) = c, \quad (x,y) \in L, \quad t \in R_+,$

where $c \in \text{int } Y_0$ is a perturbation vector to avoid degeneracy. We have taken $c = 0$ in the two dimensional case above. Suppose that $\| c \|$ is sufficiently small. It follows from the condition (vi) that $(0,c) \in X_0 \times Y_0 \subset L$. Hence the point $s^0 = (x^0,y^0,t^0) = (0,c,0)$ is a solution of the system (2.8). Let P denote the connected component, containing the point s^0, of the solution set of the system (2.8). Under a ceratin regularity assumption the set P forms a piecewise smooth path with the endpoint s^0. Thus, if we trace the path P until it attains an $(\bar{x},\bar{y},\bar{t})$ with a sufficiently large $\bar{t}$ then we obtain an approximate solution $\bar{x}$. The condition (vii) is utilized to ensure the global convergence of the method for a wide class of systems of nonlinear equations (see Section 4).

We conclude this section by illustrating the PDM used in the (3^n-1)-method for the 3-dimensional case. Fig. 2.10 shows the intersections of some pieces of $\bar{P}$ with the octahedron having the origin as a center, and Fig. 2.11 the intersection of Y_0 with the set $\{(y_1,y_2,y_3) : y_1 \geq 0,\ y_3 \geq 0\}$.

3. The Simplicial (3^n-1)-Method in R^2.

Let

$$w^1 = XO \in R^2 \quad \text{and} \quad \rho_1 = IGRID > 0,$$

where XO (the initial point) and IGRID (the initial grid size) are user-supplied parameters. Suppose that $k = 1$ or that we have an approximate solution w^k of the system (1.1) and the grid size ρ_k as the outputs of the (k-1)st major cycle for some $k \geq 2$. We now start with the kth major cycle with the initial point w^k and the grid size ρ_k. Define

$$g^k(x) = f(x+w^k) \quad \text{for every} \quad x \in R^2.$$

In the kth major cycle we search a solution of the system of equations

$$g^k(x) = 0. \tag{3.1}$$

We use the same symbols P, $\mathcal{D}$ and L as those given in the previous section. Let T^k be the triangulation of R^2 with a grid size ρ_k illustrated in Fig. 3.1. Here the symbol T^k stands for the collection of all the 2-simplices (triangles) of the triangulation. We also use the symbol $\bar{T}^k$ for the collection of all the vertices, 1-simplices (edges) and 2-simplices of the triangulation. Let $G^k : R^2 \to R^2$ be a piecewise linear approximation of the mapping g^k on the triangulation T^k, i.e., for each simplex σ and each $x \in \sigma$ with barycentric coordinates $\lambda_0, \lambda_1, \lambda_2$, i.e.,

$$x = \sum_{i=0}^{2} \lambda_i v^i, \quad \sum_{i=0}^{2} \lambda_i = 1, \quad \lambda_i \geq 0 \quad (i=0,1,2), \tag{3.2}$$

define

$$G^k(x) = \sum_{i=0}^{2} \lambda_i g^k(v^i). \tag{3.3}$$

where v^0, v^1 and v^2 are the vertices of σ. The simplicial (3^n-1)-method can be considered as an application of the smooth (3^n-1)-method to the system of equations

$$G^k(x) = 0, \tag{3.4}$$

which is a PL approximation of the system (3.1). Let Π^k be the solution set of the system of equations

$$y + t\, G^k(x) = 0, \quad (x,y) \in L, \quad t \in R_+, \tag{3.5}$$

and P^k be the connected component of Π^k which contains $s^0 = (x^0,y^0,t^0) = (0,0,0)$. We shall make full use of the piecewise linearity of the mapping G^k to trace the path P^k and to compute an exact solution $\bar{x}$ of the system (3.4).

We represent the system (3.5) as the family of subsystems

(3.6-i) $\quad y + t\,G^k(x) = 0\,, \quad x \in X_i, \quad y \in Y_i, \quad t \in R_+$

(i=0,1,...,16). Then the intersection of the path P^k with the piece $X_i \times Y_i \times R_+$, if it is nonempty, is computed by solving the subsystem (3.6-i). As will be shown below we convert each subsystem (3.6-i) into a family of linear subsystems and generate the intersection of the path P^k with the piece $X_i \times Y_i \times R_+$ by solving some of the linear subsystems successively.

Suppose that the kth major cycle has successfully terminated and that we have gotten a solution $\bar{x}$ of the PL-system (3.4). Then $x = w^k + \bar{x}$ is an approximate solution of the system (1.1). If it satisfies the criteria

$$\| f(x) \| \leq \mathrm{ACC} \quad \text{or} \quad \| \bar{x} \| \leq \mathrm{FSTEP}$$

then the program stops, where ACC and FSTEP are user-supplied parameters. Otherwise the program calculates the RATE, which varies in the interval [RTMIN, RTMAX] $\subset (0,1)$, and the new grid size ρ_k by

$$\rho_k = \mathrm{RATE} \times \rho_{k-1}.$$

Here RTMAX and RTMIN are user-supplied parameters. Roughly speaking ρ_k is chosen to be proportional to the expected distance from the approximate solution $w^k + \bar{x}$ obtained to the target exact solution of the system (1.1). Letting

$$w^{k+1} = w^k + \bar{x},$$

we then restart the (k+1)st major cycle. As we have stated in the introduction, we may apply the quasi-Newton procedure before going into the (k+1)st major cycle when the mapping f is continuously differentiable.

We now describe some more detail of how to solve the subsystem (3.6-i) or how to compute the intersection of the path P^k with the piece $X_i \times Y_i \times R_+$ in the kth major cycle. If $i = 0$ then we can easily solve the system (3.6-i). In fact the system (3.6-0) turns out to be

$$y = -\,t\,G^k(0) \in Y_0, \quad x = 0, \quad t \in R_+.$$

Hence we have

$$(X_0 \times Y_0 \times R_+) \cap P^k = \{ \ (0,-tG^k(0),t) \ : \ 0 \leq t \leq t^1 \ \},$$

where

$$t^1 = \sup \{ \ t \in R_+ \ : \ - tG^k(0) \ \in Y_0 \ \}.$$

Therefore we have only to deal with the subsystem (3.6-i) for i=1,2,..., 16.

Comparing Fig. 3.1 with Fig. 2.2 we see that the triangualtion T^k subdivides or refines the collection $\bar{P}$ of polyhedral cones X_i's (i=0,1,...,16). Let

$$T^k|X_i = \{ \ \sigma \in \bar{T}^k \ : \ \sigma \subset X_i, \ \ \dim \sigma = \dim X_i \ \}.$$

(Recall that $\bar{T}^k$ denotes the collection of all the vertices, 1-simplices (edges) and 2-simplices of the triangulation T^k.) See Fig.3.2. Since each polyhedral cone $X_i \in \bar{P}$ is subdivided into the collection $T^k|X_i$ of simplices, each subsystem (3.6-i) is equivalent to the family of subsystems of equations

(3.6-i-σ) $\quad y + t\, G^k(x) = 0, \quad x \in \sigma, \quad y \in Y_i, \quad t \in R_+$

($\sigma \in T^k|X_i$). Therefore the original system (3.5), whose solution set is to be traced, can be represented as the family of subsystems { (3.6-i-σ) : $\sigma \in T^k|X_i$, i=0,1,2,...,16 }.

Suppose that $\sigma \in T^k|X_i$ for some i and that the path P^k intersects with the piece $\sigma \times Y_i \times R_+$. Let $m = \dim \sigma$, which is either 0, 1, or 2. Then every $x \in \sigma$ is uniquely represented by using barycentric coordinates on the simplex σ as follows

$$(3.7) \qquad x = \sum_{i=0}^{m} \lambda_i v^i, \quad \sum_{i=1}^{m} \lambda_i = 1, \quad \lambda_i \geq 0 \ \text{ for } \ i=0,...,m,$$

where $v^0,...,v^m$ are the vertices of σ. By the construction of the PL mapping G^k, we have

$$(3.8) \qquad G^k(x) = \sum_{i=0}^{m} \lambda_i g^k(v^i)$$

for every $x \in \sigma$ satisfying (3.7). Substituting (3.8) into (3.6-i-σ) and letting $\mu_i = t\,\lambda_i$ for i=0,..,m, we have the linear system

$$(3.9\text{-}i\text{-}\sigma) \qquad \begin{aligned} & y + \sum_{i=0}^{m} \mu_i g^k(v^i) = 0, \\ & \mu_i \geq 0 \ \text{ for } \ i=0,...,m, \end{aligned}$$

$$y \in Y_i,$$

which is equivalent to the system (3.6-i-σ) through the conversion

$$t = \sum_{i=0}^{m} \mu_i,$$

$$x = \begin{cases} 0 & \text{if } t = 0 \\ \sum_{i=0}^{m} \mu_i v^i / t & \text{if } t > 0. \end{cases} \tag{3.10}$$

(Note that if (x,y,t) is a solution of the system (3.6-i-σ) with $i \geq 1$ then $t>0$.) The linear system (3.9-i-σ) is the system dealt with by the simplicial (3^n-1)-method.

Now we shall show a numerical example. We consider the system (2.6) of equations, which we have solved by the smooth (3^n-1)-method in the previous section;

$$\begin{aligned} f_1(x_1,x_2) &= x_1 + x_2 - x_2^2 - 1.4 = 0, \\ f_2(x_1,x_2) &= \quad x_2 \quad - 1.2 = 0. \end{aligned} \tag{2.6}$$

This system has a solution $x = (1.64, 1.2)$. Let $k = 1$, $w^1 = (0,0)$ and $\rho_1 = 1$. We set the parameter γ, which determines the shape of the polyhedral set Y_0, to be 0.2. See Fig.2.2.

(i) We start with the linear subsystem

$$\begin{aligned} & y_1 + \mu_0 f_1(0,0) = 0, \text{ i.e., } y_1 - 1.4\,\mu_0 = 0, \\ & y_2 + \mu_0 f_2(0,0) = 0, \text{ i.e., } y_2 - 1.2\,\mu_0 = 0, \\ & (y_1,y_2) \in Y_0, \\ & \mu_0 \geq 0, \end{aligned} \tag{3.9-0-σ^0}$$

where $\sigma^0 = \{(0,0)\}$. The solution set is

$$\{ (\mu_0,y_1,y_2) : 0 \leq \mu_0 \leq 5/13, \quad y_1 = 1.4\,\mu_0, \quad y_2 = 1.2\,\mu_0 \},$$

which forms a line segment with the two endpoints

$$(\mu_0,y_1,y_2) = (0,0,0),$$
$$(\mu_0,y_1,y_2) = (5/13,\ 7/13,\ 6/13).$$

The line segment and its two endpoints above are transformed by (3.10) into the curve

$$(\sigma^0 \times Y^0 \times R_+) \cap P^1$$
$$= \{ (x_1,x_2,y_1,y_2,t) : 0 \leq t \leq 5/13,\ x_1 = 0,\ x_2 = 0, \quad y_1 = 1.4\,t,\ y_2 = 1.2\,t \}$$

and its endpoints

$$s^0 = (x^0,y^0,t^0) = (0,0,0,0,0),$$
$$s^1 = (x^1,y^1,t^1) = (0,0,7/13,6/13,5/13),$$

respectively. The point (x^1,y^1) lies on the common facet $X_0 \times Y_2$ of the pieces $X_0 \times Y_0$ and $X_2 \times Y_2$.

(ii) Now the path P^1 will move into the piece $\sigma^1 \times Y_2 \times R_+$, where σ^1 denotes the 1-dimensional simplex (edge) with the vertices (0,0) and (1,1) (see Fig. 3.3), so that we need to solve the linear subsystem

$$\begin{aligned} & y_1 + \mu_0 f_1(0,0) + \mu_1 f_1(1,1) = 0, \\ & y_2 + \mu_0 f_2(0,0) + \mu_1 f_2(1,1) = 0, \\ & \mu_0 \geq 0, \quad \mu^1 \geq 0, \\ & (y_1,y_2) \in Y_2, \text{ i.e., } y_1 + y_2 = 1,\ y_1 \geq 1/5,\ y_2 \geq 1/5. \end{aligned} \tag{3.9-2-σ^1}$$

We have already known from (i) that the point

$$(\mu_0,\mu_1,y_1,y_2) = (5/13,\ 0,\ 7/13,\ 6/13)$$

satisfies the system (3.9-2-σ^1). More precisely, it is a basic solution of the system. Hence, by performing a pivoting operation to the system, we obtain a new basic solution

$$(\mu_0,\mu_1,y_1,y_2) = (0,\ 5/3,\ 2/3,\ 1/3),$$

which is adjacent to the above basic solution. The solution set of the system (3.9-2-σ^1) can be represented as the convex combination of the two basic solutions above. Using the transformation (3.10), we obtain

$$\begin{aligned} & (\sigma^1 \times Y_2 \times R_+) \cap P^1 \\ & = \{ (x_1,x_2,y_1,y_2,t) : 5/13 \leq t \leq 5/3, \\ & \qquad x_1 = x_2 = 13/10 - 1/(2t), \\ & \qquad y_1 = 1/2 + (1/10)t, \\ & \qquad y_2 = 1/2 - (1/10)t \quad \}, \end{aligned}$$

which forms a curve with the endpoints s^1 and

$$s^2 = (x^2,y^2,t^2) = (1,\ 1,\ 2/3,\ 1/3,\ 5/3).$$

(iii) We see that the point (x^2,y^2) lies in the relative interior of the piece $X_2 \times Y_2$, and that the path P^1 will move into the piece $\sigma^2 \times Y_2 \times R_+$, where σ^2 is the 1-dimensional simplex with the vertices (1,1) and (2,2) (see Fig.3.3). Hence we deal with the linear subsystem

$$\begin{aligned} & y_1 + \mu_1 f_1(1,1) + \mu_2 f_1(2,2) = 0, \\ & y_2 + \mu_1 f_2(1,1) + \mu_2 f_2(2,2) = 0, \\ & (y_1,y_2) \in Y_2, \text{ i.e., } y_1 + y_2 = 1,\ y_1 \geq 1/5,\ y_2 \geq 1/5, \end{aligned} \tag{3.9-2-σ^2}$$

$$\mu_1 \geq 0, \quad \mu_2 \geq 0,$$

which has two distinct but adjacent basic solutions

$$(\mu_1,\mu_2,y_1,y_2) = (5/3,\ 0,\ 2/3,1/3) \quad \text{(from (ii))},$$
$$(\mu_1,\mu_2,y_1,y_2) = (23/15,2/15,4/5,1/5) \quad \text{(new)}.$$

The new basic solution corresponds to the point

$$s^3 = (x^3,y^3,t^3) = (27/25,27/25,4/5,1/5,5/3)$$

on the intersection of the path P^1 with the common facet $X_2 \times Y_9 \times R_+$ of the pieces $X_2 \times Y_2 \times R_+$ and $X_9 \times Y_9 \times R_+$.

(iv) Now the path will move into the piece $\sigma^3 \times Y_9 \times R_+$, where σ^3 is the 2-dimensional simplex with the vertices (1,1), (2,2), and (2,1) (see Fig 3.3). Hence we consider the linear subsystem

$$\begin{aligned} & y_1 + \mu_1 f_1(1,1) + \mu_2 f_1(2,2) + \mu_3 f_1(2,1) = 0, \\ & 1/5 + \mu_1 f_2(1,1) + \mu_2 f_2(2,2) + \mu_3 f_2(2,1) = 0, \\ & y_1 + 1/5 = 1, \\ & \mu_1 \geq 0,\ \mu_2 \geq 0,\ \mu_3 \geq 0, \end{aligned} \qquad (3.9\text{-}9\text{-}\sigma^3)$$

A basic solution

$$(\mu_1,\mu_2,\mu_3,y_1) = (23/15,2/15,0,4/5)$$

of the system is known from (iii). The solution set of the system above turns out to be an unbounded ray of the form

$$\{\ (23/15,2/15,0,4/5) + \theta\ (\nu_1,\nu_2,\nu_3,0)\ :\ \theta \geq 0\ \},$$

where

$$(\nu_1,\nu_2,\nu_3) = (1/5,1/5,3/5).$$

It is not necessary to transform the ray into $(\sigma^3 \times Y_9 \times R_+) \cap P^1$ and to take the limit along the path. In fact, we see that the direction vector (ν_1,ν_2,ν_3) satisfies

$$\begin{aligned} & \nu_1 f_1(1,1) + \nu_2 f_1(2,2) + \nu_3 f_1(2,1) = 0, \\ & \nu_1 f_2(1,1) + \nu_2 f_2(2,2) + \nu_3 f_2(2,1) = 0, \\ & \nu_1 + \nu_2 + \nu_3 = 1, \\ & \nu_1 \geq 0,\ \nu_2 \geq 0,\ \nu_3 \geq 0. \end{aligned}$$

This implies that the point

$$\bar{x} = \nu_1\ (1,1) + \nu_2\ (2,2) + \nu_3\ (2,1) = (9/5,6/5) = (1.8,\ 1.2)$$

is an exact solution of the system (3.4); hence an approximate solution of the system (3.1). This completes the 1st major cycle.

In the example above, we have computed the values of the endpoints of the intersection of the path P^1 with the pieces $\sigma \times Y_i \times R_+$ through which the path runs. Those values are redundant, however, if we only need an approximate solution of the system of equations to be solved. In fact, a new basic solution of the linear subsystem (3.9-i-σ) under consideration shows us the piece into which the path P^1 will move, and if the solution set of the linear subsystem forms an unbounded ray, then we can compute an approximate solution directly from the direction vector of the ray. In the program, we perform pivoting operations to the family of linear subsystems to generate a sequence of adjacent basic solutions and to find an unbounded ray.

4. Convergence Condition

One of the most important and useful properties of fixed point algorithms is that they converge under a mild condition on the function. For the simplicial subdivision T of R^n let

$$\delta = \sup_{\sigma \in T} \sup_{x,y \in \sigma} \| x - y \| ,$$

which is referred to as the mesh size of T. (The mesh size of the simplicial subdivision T^k in Fig. 3.1 is $\sqrt{2}\,\rho_k$.) VARDIM provides an exact zero of the piecewise linear approximation G of the function g on T, if g satisfies the following condition.

Condition (see Theorem 6.9, [5])

There exists a positive number μ such that for every $x \in R^n$ with $\| x \| \geq \mu$, there is an $x^* \in R^n$ such that $\| x^* \| \leq \mu$ and

$$(x - x^*)^t g(v) > 0$$

whenever $\| v - x \| \leq \delta$.

For example, let us consider the fixed point problem: find an x such that

$$k(x) = x, \quad x \in X,$$

where X is a compact subset of R^n and k is a continuous function from R^n into X. Let

$$g(x) = x - k(x) \qquad \text{for every } x \in R^n.$$

Then x is a fixed point of k if and only if $g(x) = 0$. Furthermore we see that g satisfies the above convergence condition. Thus VARDIM always provides an approximate fixed point of a continuous function from R^n into a compact subset of R^n.

5. Input Parameters and Subroutine FUNCT

We shall explain the parameters of VARDIM, some of which the user must supply.

PROB is the problem number. VARDIM already has five test problems. The user might offer several problems in subroutine FUNCT and solve one of them by setting PROB to an appropriate number. (integer)

NN is the dimension of the problem. NN must not exceed 50. The user must rewrite all PARAMETER statements to solve larger problems. (integer)

XO is an initial approximate solution. The default value is the origin. (double precision array of length NN)

IGRID is the initial grid size of the simplicial subdivision of R^n. (double precision real)

ACC is the desired accuracy. When $\| f(x) \| \leqq$ ACC holds at an approximate solution obtained, VARDIM successfully terminates. (double precision real)

FSTEP is the Euclidean distance of successive approximate solutions to conclude convergence. (double precision real)

BX is the maximum max. norm of the variable vector x to conclude divergence. (double precision real)

QNEWT is to be set to 1 when the user wishes to insert quasi-Newton procedures and to 0 otherwise. (integer)

The user has only to supply the above eight parameters to use VARDIM. VARDIM has other parameters which are set to their default values in subroutine PARAMT.

GAMMA is a parameter for defining the PDM used in VARDIM. It is a positive number less than 1/NN. The default value is 0.5/NN. (double precision real)

RTMAX is the maximum reduction factor of grid size. The default value is 0.8. (double precision real)

RTMIN is the minimum reduction factor of grid size. The default value is 0.4. (double precision real)

FGRID is the minimum grid size. If the current grid size becomes less than FGRID, it is reset to FGRID and the computation stops after one major cycle. The default value is 10^{-7}. (double precision real)

BC is the maximum number of cycles to conclude divergence. The default value is 100. (integer)

BP is the maximum number of pivot operations per cycle to conclude divergence. The default value is 400*NN. (integer)

EPS is the parameter to distinguish nonzero value from zero. The default value is 10^{-10}. (double precision real)

The only subroutine the user must supply is FUNCT. It has four arguments: PROB (integer), NN (integer), X (double precision array of length NN) and F (double precision array of length NN). When FUNCT is called with PROB, NN and X, it should return the function value at the point X of the PROBth problem with dimension NN. As will be shown below, the user might write several problems in this subroutine and choose one by setting PROB to an appropriate number.

```
      SUBROUTINE  FUNCT (PROB,NN,X,F)
      IMPLICIT DOUBLE PRECISION  (A-H,O-Z)
      INTEGER  DIM,PROB,NN
      PARAMETER  (DIM=53)
      DIMENSION  X(DIM),F(DIM)
```

```
C
      GOTO  (1000,2000,3000), PROB
C
 1000 CONTINUE

           Here compute the function value of the 1st problem.

      RETURN
C
 2000 CONTINUE

           Here compute the function value of the 2nd problem.

      RETURN
C
 3000 CONTINUE

           Here compute the function value of the 3rd problem.

      RETURN
      END
```

We shall show the results of solving five test problems in FUNCT for several dimensions. To solve, for example, the first problem of dimension 10, the user should input as follows.

```
For
     PROBLEM NUMBER  ( 0  TO STOP )
input  1.
For
     DIMENSION OF PROBLEM
input  10.
For
     INITIAL POINT  ( 0 FOR DEFAULT, 1 TO INPUT )
input  0  when the user adopts the origin as an initial point, or
input  1  when the user wants to input an initial point.  In this case
          input the coordinates of the initial point for
          INPUT COORDINATES OF INITIAL POINT.
For
     INITIAL GRID SIZE
input, for example, 0.5.
For
     DESIRED ACCURACY
input, for example, 1.0D-8.
```

For

LOWER BOUND OF STEP SIZE

input, for example, 1.0D-7.

For

UPPER BOUND OF ABS(X(I))

input, for example, 10.0.

For

QUASI-NEWTON PROCEDURE (1 TO INSERT, 0 OTHERWISE)

input, for example, 1.

Then VARDIM outputs the title, several parameters and the initial point.

```
(3**N-1)-METHOD WITH QUASI-NEWTON STEPS
        GAMMA= 0.50D+00 / N
        IGRID= 0.50D+00
        A C C= 0.10D-07
        FSTEP= 0.10D-06
        QNEWT=  1

PROB=  1    N=  50

INITIAL POINT
   0.0           0.0           0.0           0.0           0.0
   0.0           0.0           0.0           0.0           0.0
```

VARDIM starts computing and outputs as follows.

CYCLE	GRID	#PVT	#FFXP	#FNEW	EN(FUNCT)	EN(DIFF)
1	0.500D+00	13	13	3	0.7778D-08	0.1044D+01
1	0.500D+00	13	13	3	0.7778D-08	0.1044D+01

```
APPROXIMATE SOLUTION
   0.72344D-01   0.12234D+00   0.17234D+00   0.22234D+00   0.27234D+00
   0.32234D+00   0.37234D+00   0.42234D+00   0.47234D+00   0.52234D+00

FUNCTION VALUE
   0.24595D-08   0.24595D-08   0.24595D-08   0.24595D-08   0.24595D-08
   0.24595D-08   0.24595D-08   0.24595D-08   0.24595D-08   0.24595D-08
```

The column headed CYCLE gives the number of cycle. The column headed GRID gives the grid size used in each cycle. The columns headed #PVT and #FFXP give the numbers of pivot operations and function evaluations,

respectively, consumed by the fixed point algorithm in each cycle to obtain an approximate solution. The column headed #FNEW gives the number of function evaluations consumed by the quasi-Newton steps. The column headed EN(FUNCT) gives the Euclidean norm of $f(x)$ at the approximate solution and finally the column headed EN(DIFF) gives the Euclidean distance from the previous approximate solution to the current approximate solution. Either EN(FUNCT) $\leq$ ACC or EN(DIFF) $\leq$ FSTEP occurs (i.e. VARDIM terminates successfully), it outputs the number of cycles consumed, final grid size, total number of pivot operations and total numbers of function evaluations consumed by the fixed piont algorithm and quasi-Newton steps, and the Euclidean distance from the final approximate solution to the previous one. And finally VARDIM outputs the approximate solution and the function value at it.

The first three problems in subroutine FUNCT are the test problems by Watson which are found in Allgower and Georg [1].

The fourth problem is a combination of two-dimensional 'spiral functions' which are identical outside the unit circle and minus identical inside a smaller circle of radius $2^{-5} \simeq 0.03$. This problem was proposed by Todd. The solution to this problem is the origin, hence we report the result with the initial point X0 = (0.1,0.1,...,0.1).

The fifth one is the gradient mapping of the objective function of a nonlinear programming in Dixon [3].

(3**N-1)-METHOD WITH QUASI-NEWTON PROCEDURE
GAMMA= 0.50D+00 / N
IGRID= 0.40D+00
A C C= 0.10D-07
FSTEP= 0.10D-07
QNEWT= 1

PROB= 1 N= 30

INITIAL POINT

0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0

CYCLE	GRID	#PVT	#FFXP	#FNEW	EN(FUNCT)	EN(DIFF)
1	0.400D+00	39	39	4	0.4164D-13	0.1714D+01
1	0.400D+00	39	39	4	0.4164D-13	0.1714D+01

APPROXIMATE SOLUTION

0.36045D-01	0.52711D-01	0.69378D-01	0.86045D-01	0.10271D+00
0.11938D+00	0.13604D+00	0.15271D+00	0.16938D+00	0.18604D+00
0.20271D+00	0.21938D+00	0.23604D+00	0.25271D+00	0.26938D+00
0.28604D+00	0.30271D+00	0.31938D+00	0.33604D+00	0.35271D+00
0.36938D+00	0.38604D+00	0.40271D+00	0.41938D+00	0.43604D+00
0.45271D+00	0.46938D+00	0.48604D+00	0.50271D+00	0.51938D+00

FUNCTION VALUE

-0.76102D-14	-0.76102D-14	-0.76189D-14	-0.76189D-14	-0.76189D-14
-0.76050D-14	-0.76189D-14	-0.76189D-14	-0.76189D-14	-0.76050D-14
-0.76189D-14	-0.76189D-14	-0.76189D-14	-0.76189D-14	-0.75911D-14
-0.75911D-14	-0.75911D-14	-0.75911D-14	-0.75911D-14	-0.75911D-14
-0.75911D-14	-0.75911D-14	-0.75911D-14	-0.75911D-14	-0.75911D-14
-0.75911D-14	-0.75911D-14	-0.75911D-14	-0.75911D-14	-0.75911D-14

```
(3**N-1)-METHOD WITH QUASI-NEWTON PROCEDURE
        GAMMA= 0.50D+00 / N
        IGRID= 0.40D+00
        A C C= 0.10D-07
        FSTEP= 0.10D-07
        QNEWT=  1

PROB=  2   N=   6

INITIAL POINT
   0.0           0.0           0.0           0.0           0.0
   0.0
```

CYCLE	GRID	#PVT	#FFXP	#FNEW	EN(FUNCT)	EN(DIFF)
1	0.400D+00	146	128	2	0.1376D+01	0.3355D+01
2	0.320D+00	327	287	10	0.3084D-09	0.2466D+01
2	0.320D+00	473	415	12	0.3084D-09	0.2466D+01

```
APPROXIMATE SOLUTION
   0.24150D+01   0.17416D+01   0.11014D+01   0.68083D+00   0.46093D+00
   0.37481D+00

FUNCTION VALUE
  -0.11587D-10   0.43483D-10   0.10048D-09   0.10415D-09   0.13142D-09
   0.23420D-09
```

```
(3**N-1)-METHOD WITH QUASI-NEWTON PROCEDURE
        GAMMA= 0.50D+00 / N
        IGRID= 0.40D+00
        A C C= 0.10D-07
        FSTEP= 0.10D-07
        QNEWT=  1

PROB=  3   N=  30

INITIAL POINT
   0.0          0.0          0.0          0.0          0.0
   0.0          0.0          0.0          0.0          0.0
   0.0          0.0          0.0          0.0          0.0
   0.0          0.0          0.0          0.0          0.0
   0.0          0.0          0.0          0.0          0.0
   0.0          0.0          0.0          0.0          0.0
```

CYCLE	GRID	#PVT	#FFXP	#FNEW	EN(FUNCT)	EN(DIFF)
1	0.400D+00	6	6	0	0.9870D+00	0.5564D+01
2	0.160D+00	92	91	11	0.3457D-08	0.1011D+01
2	0.160D+00	98	97	11	0.3457D-08	0.1011D+01

```
APPROXIMATE SOLUTION
   0.10000D+01  0.10000D+01  0.10000D+01  0.10000D+01  0.10000D+01
   0.10000D+01  0.10000D+01  0.10000D+01  0.10000D+01  0.10000D+01
   0.10000D+01  0.10000D+01  0.10000D+01  0.10000D+01  0.10000D+01
   0.10000D+01  0.10000D+01  0.10000D+01  0.10000D+01  0.10000D+01
   0.10000D+01  0.10000D+01  0.10000D+01  0.10000D+01  0.10000D+01
   0.10000D+01  0.10000D+01  0.10000D+01  0.10000D+01  0.10000D+01

FUNCTION VALUE
  -0.34565D-08 -0.14211D-13 -0.14211D-13 -0.14211D-13 -0.14211D-13
  -0.14211D-13 -0.14211D-13 -0.14211D-13 -0.14211D-13 -0.14211D-13
  -0.14211D-13 -0.14211D-13 -0.14211D-13 -0.14211D-13 -0.14211D-13
  -0.14211D-13 -0.14211D-13 -0.14211D-13 -0.14211D-13 -0.14211D-13
  -0.14211D-13 -0.14211D-13 -0.14211D-13 -0.14211D-13 -0.14211D-13
  -0.14211D-13 -0.14211D-13 -0.14211D-13 -0.14211D-13 -0.14211D-13
```

(3**N-1)-METHOD WITH QUASI-NEWTON PROCEDURE
GAMMA= 0.50D+00 / N
IGRID= 0.40D+00
A C C= 0.10D-07
FSTEP= 0.10D-07
QNEWT= 1

PROB= 4 N= 5

INITIAL POINT
0.10000D+00 0.10000D+00 0.10000D+00 0.10000D+00 0.10000D+00

CYCLE	GRID	#PVT	#FFXP	#FNEW	EN(FUNCT)	EN(DIFF)
1	0.400D+00	9	8	0	0.9125D+00	0.7902D+00
2	0.320D+00	26	20	0	0.6030D+00	0.1085D+01
3	0.256D+00	42	32	0	0.5064D+00	0.9013D+00
4	0.205D+00	92	76	0	0.5389D+00	0.1043D+01
5	0.164D+00	165	148	0	0.3831D+00	0.6914D+00
6	0.131D+00	6	5	0	0.2240D+00	0.1663D+00
7	0.105D+00	10	8	0	0.1912D+00	0.1970D+00
8	0.839D-01	19	15	0	0.1872D+00	0.1784D+00
9	0.671D-01	33	27	0	0.1581D+00	0.2359D+00
10	0.537D-01	64	55	0	0.1275D+00	0.2292D+00
11	0.429D-01	82	75	0	0.8383D-01	0.1850D+00
12	0.344D-01	9	7	0	0.6294D-01	0.3775D-01
13	0.275D-01	27	24	0	0.3200D-01	0.6051D-01
14	0.220D-01	18	14	0	0.1562D-16	0.3200D-01
14	0.220D-01	602	514	0	0.1562D-16	0.3200D-01

APPROXIMATE SOLUTION
0.34694D-17 -0.59631D-18 -0.52042D-17 0.34694D-17 0.13878D-16

FUNCTION VALUE
-0.34694D-17 0.59631D-18 0.52042D-17 -0.34694D-17 0.13878D-16

(3**N-1)-METHOD WITH QUASI-NEWTON PROCEDURE
GAMMA= 0.50D+00 / N
IGRID= 0.40D+00
A C C= 0.10D-07
FSTEP= 0.10D-07
QNEWT= 1

PROB= 5 N= 10

INITIAL POINT

0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0

CYCLE	GRID	#PVT	#FFXP	#FNEW	EN(FUNCT)	EN(DIFF)
1	0.400D+00	54	49	1	0.3030D+00	0.1250D+01
2	0.160D+00	92	86	2	0.4803D-01	0.7635D+00
3	0.128D+00	51	43	2	0.1040D-01	0.1507D+00
4	0.512D-01	446	440	4	0.9389D-08	0.2047D+01
4	0.512D-01	643	618	9	0.9389D-08	0.2047D+01

APPROXIMATE SOLUTION

0.10000D+01	0.10000D+01	0.10000D+01	0.10000D+01	0.10000D+01
0.10000D+01	0.10000D+01	0.10000D+01	0.10000D+01	0.10000D+01

FUNCTION VALUE

0.48299D-09	0.12306D-08	0.74105D-09	-0.14080D-08	0.27723D-08
0.27130D-09	0.41427D-09	0.25434D-08	-0.80894D-08	0.20075D-08

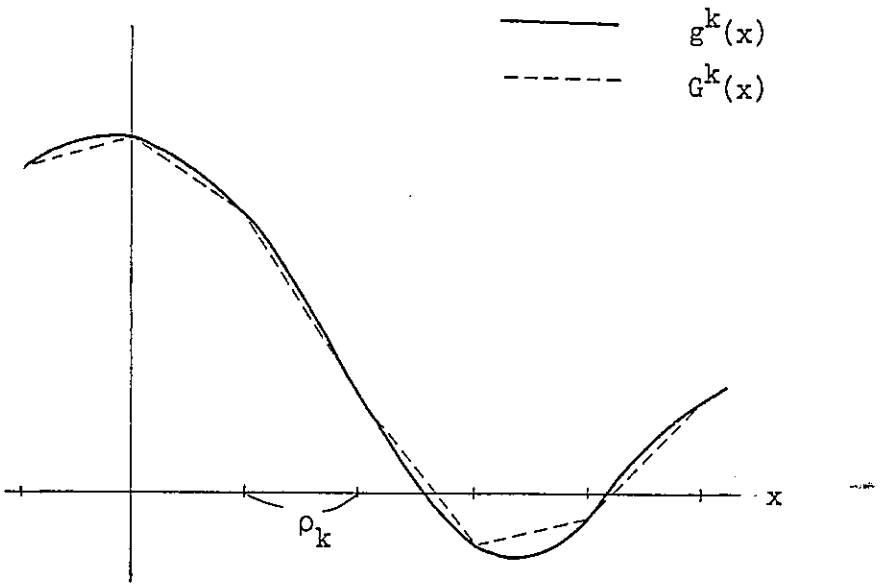

Fig. 1.1

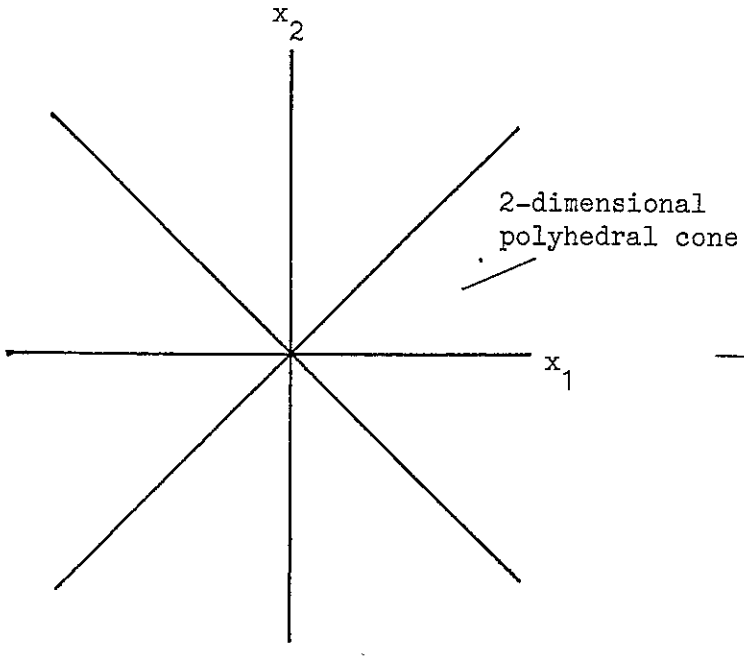

Fig. 2.1

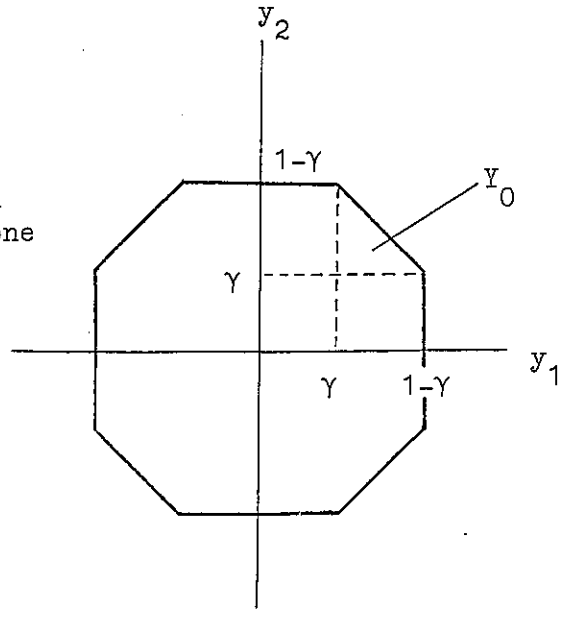

Fig. 2.2

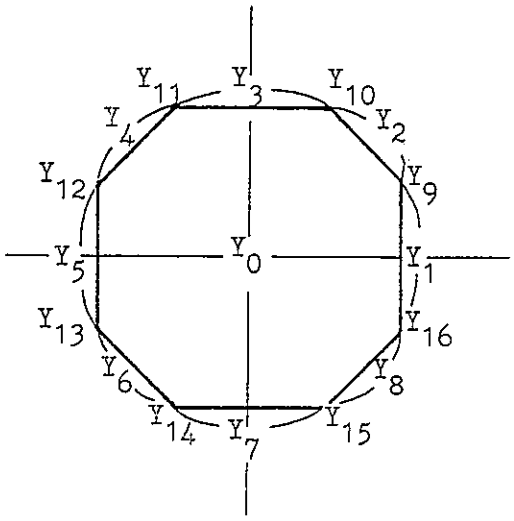

Fig. 2.3

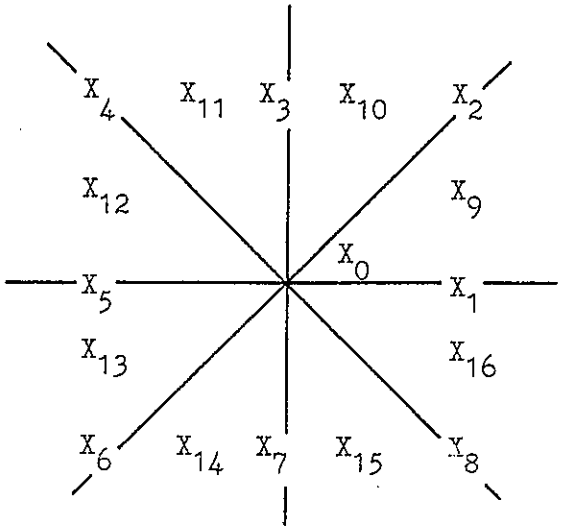

Fig. 2.4

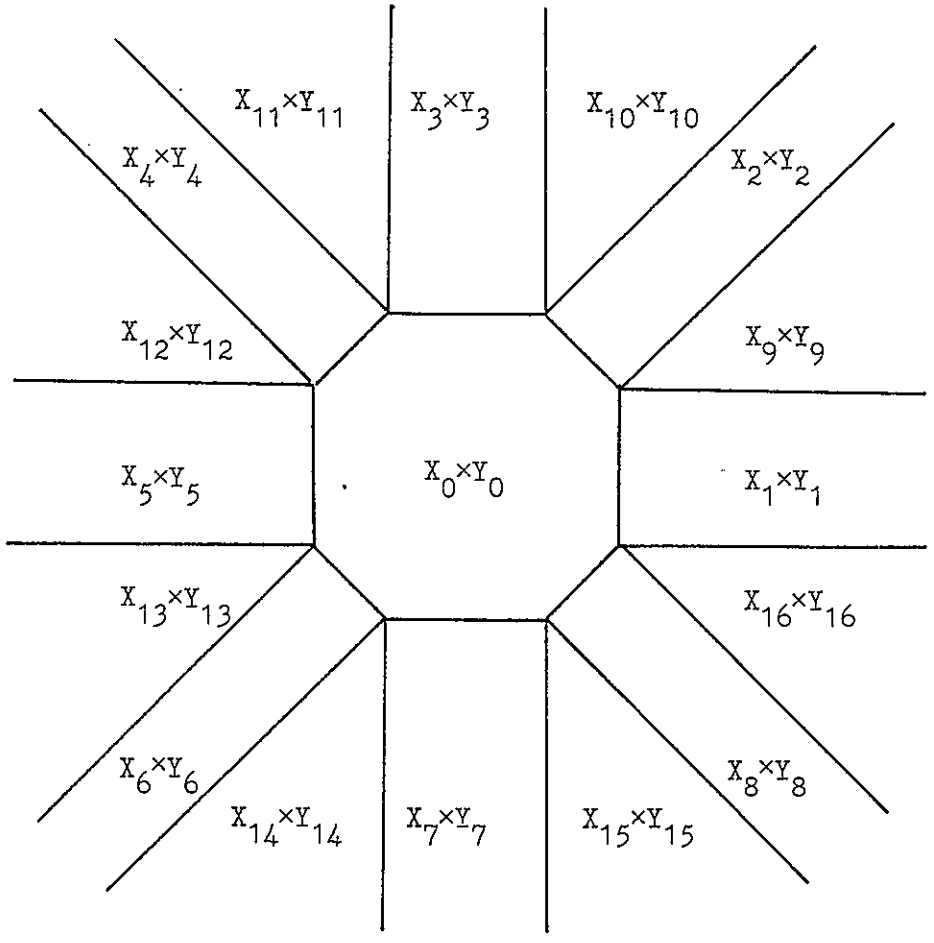

Fig. 2.5

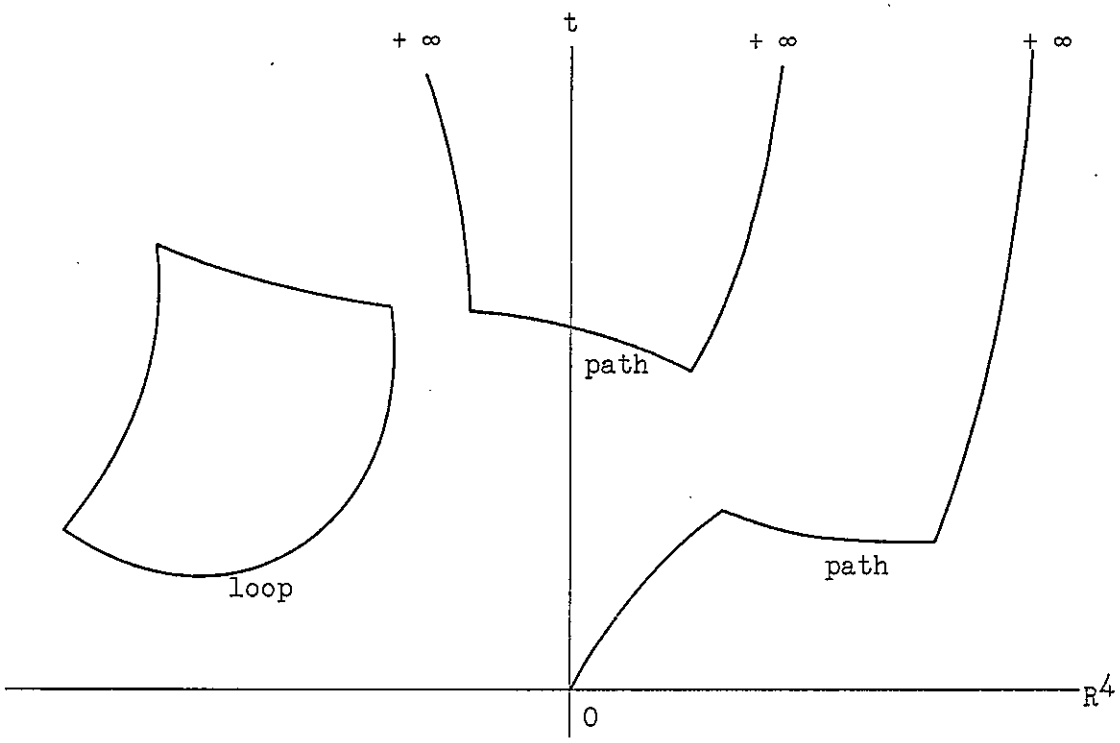

Fig. 2.6

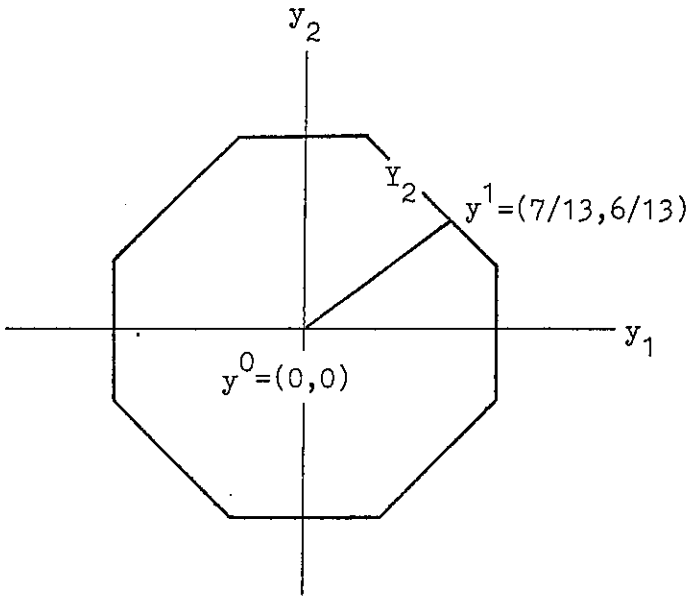

Fig. 2.7 the projection of $(X_0 \times Y_0 \times R_+) \cap P$

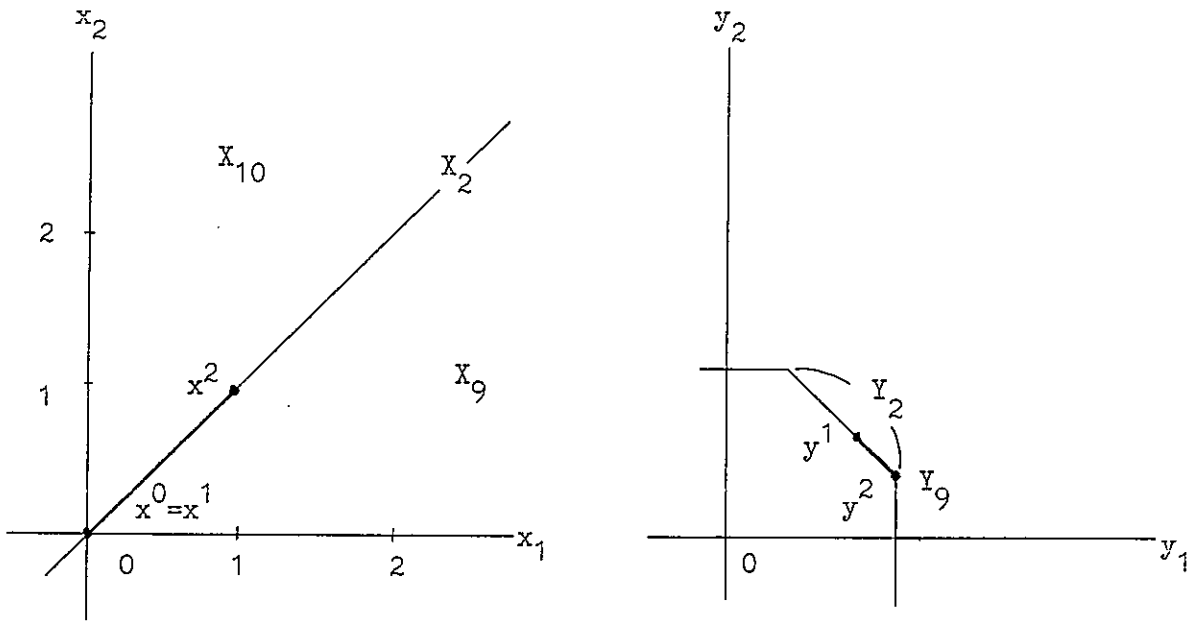

Fig. 2.8 the projections of $(X_2 \times Y_2 \times R_+) \cap P$

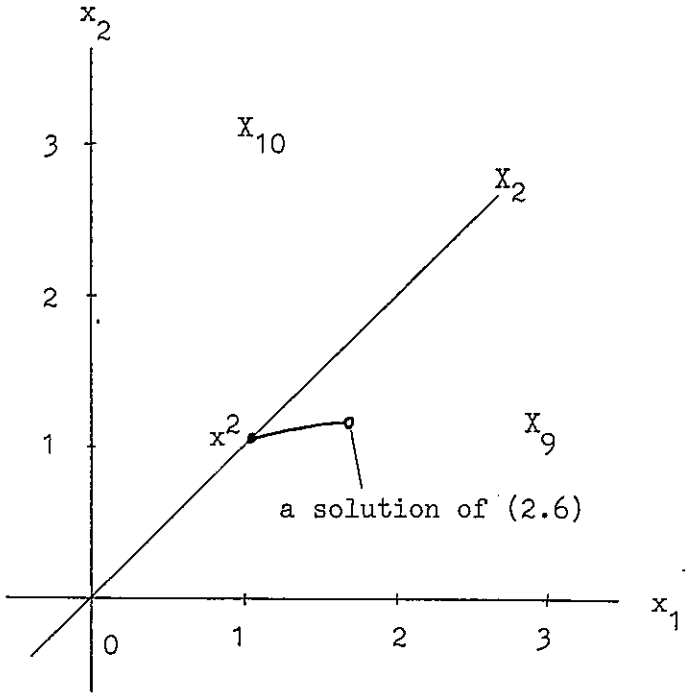

Fig. 2.9 the projection of $(X_9 \times Y_9 \times R_+) \cap P$

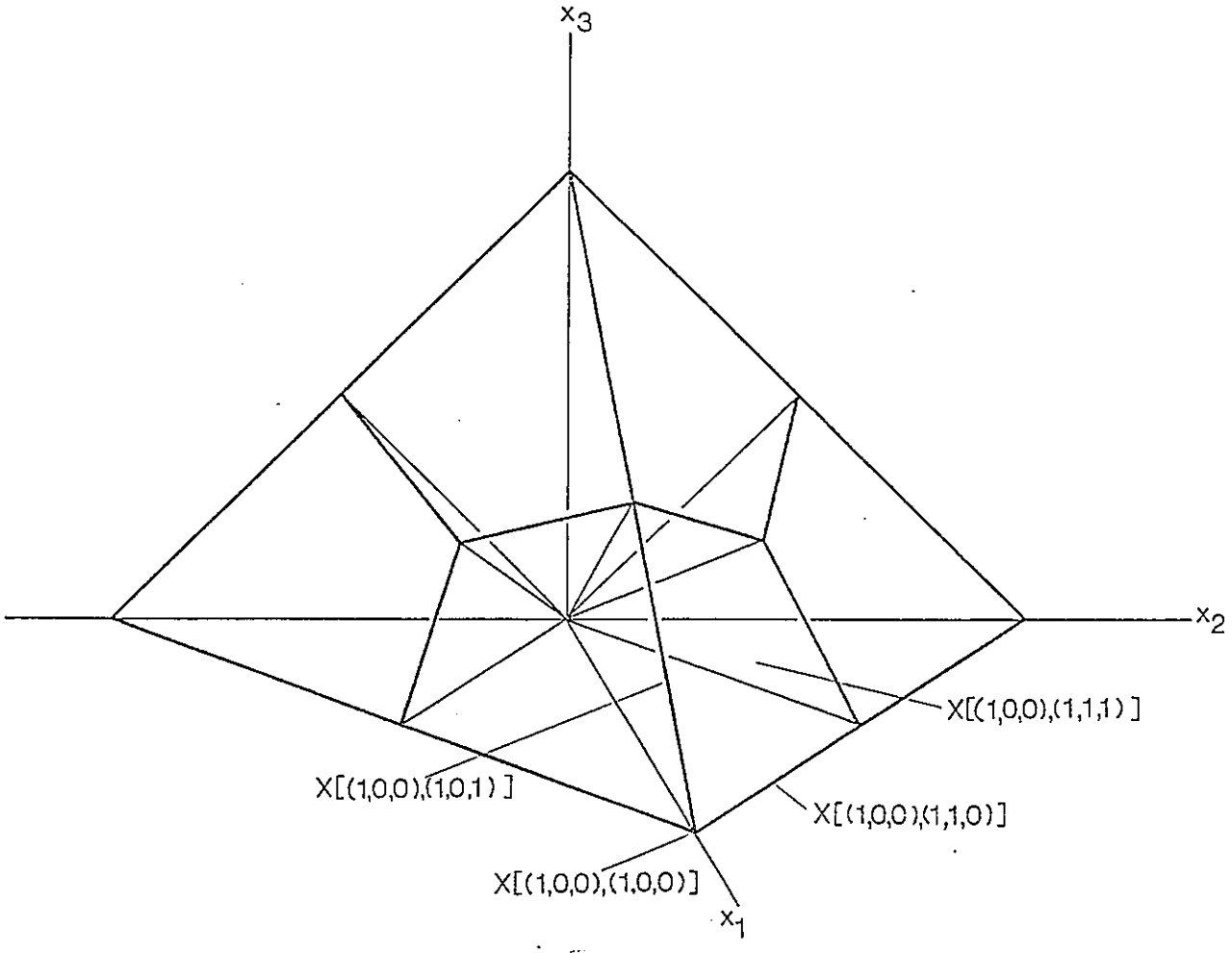

Fig. 2.10

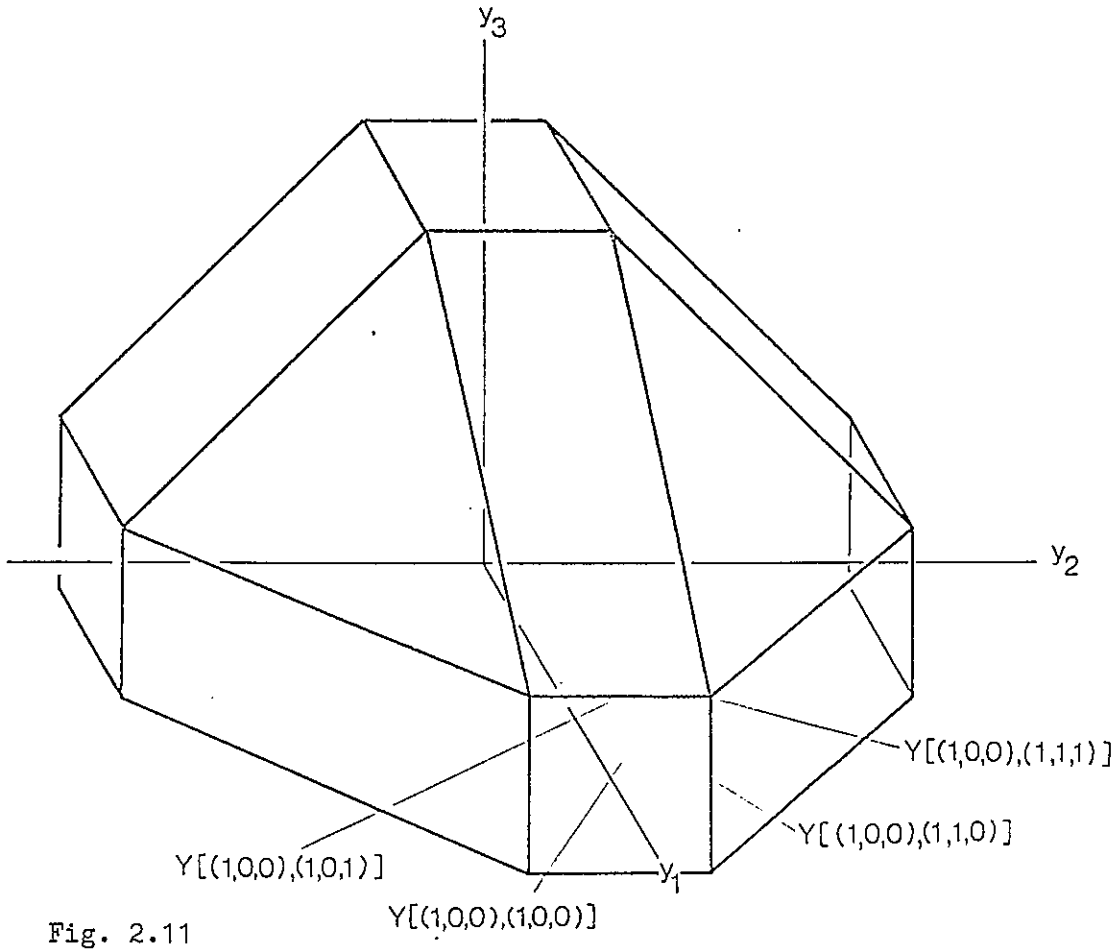

Fig. 2.11

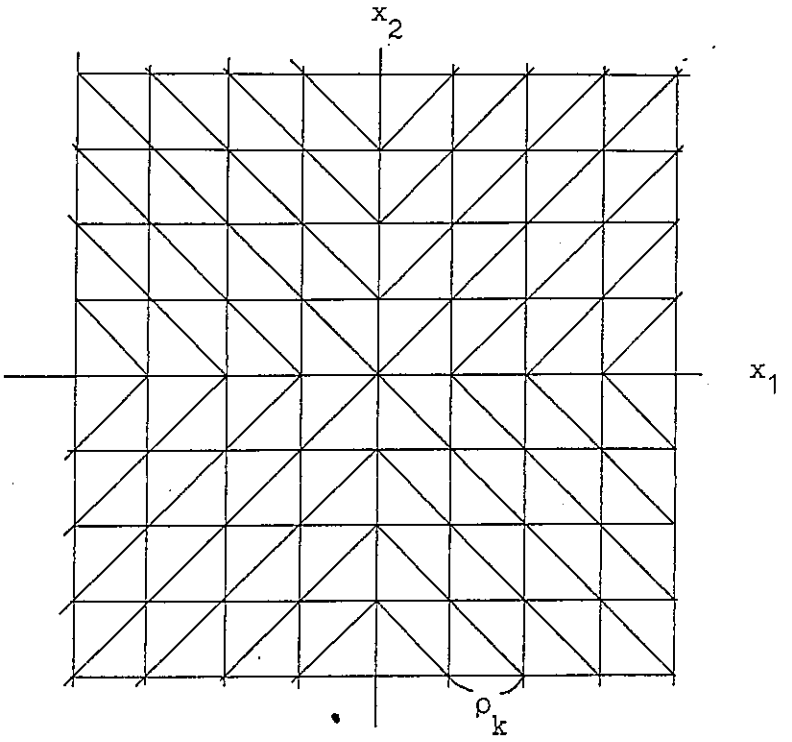

Fig. 3.1 the triangulation T^k

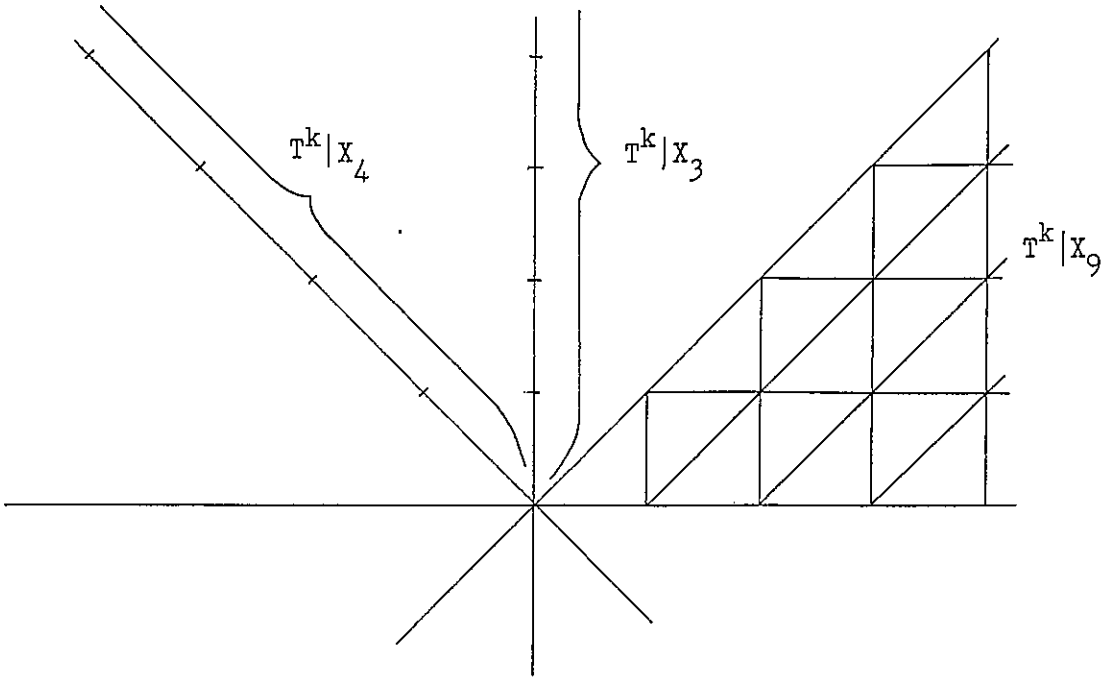

Fig. 3.2

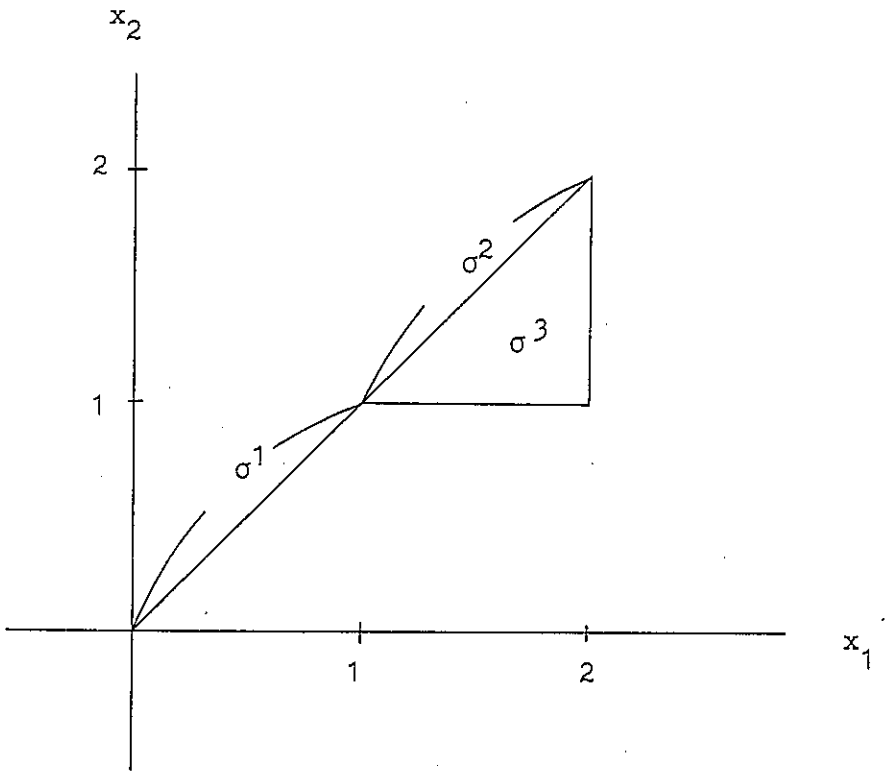

Fig. 3.3

References

[1] E. Allgower and K. Georg, "Simplicial and continuation methods for approximating fixed points and solutions to systems of equations", SIAM Review 22 (1980) 28-85.

[2] J.E. Dennis, Jr. and J.J. More, "Quasi-Newton methods, Motivation and theory", SIAM Review 19 (1977) 46-89.

[3] L.C.W. Dixon, "Conjugate directions without linear searches", Journal of the Institute of Mathematics and Its Applications 11 (1973) 317-328.

[4] M. Kojima and Y. Yamamoto, "Variable dimension algorithms: Basic theory, interpretations and extensions of some existing methods", Mathematical Programming 24 (1982) 177-215.

[5] M. Kojima and Y. Yamamoto, "A unified approach to the implementation of several restart fixed point algorithms and a new variable dimension algorithm", Mathematical Programming (to appear).

[6] G. van der Laan and A.J.J. Talman, "A class of simplicial restart fixed point algorithms without an extra dimension", Mathematical Programming 20 (1981) 33-48.

[7] T.Y. Li and J.A. Yorke, "A simple reliable numerical algorithm for following homotopy paths", in: S.M. Robinson, ed., Analysis and Computation of Fixed Points (Academic Press, New York, 1980).

[8] A.H. Wright, "The octahedral algorithm, a new simplicial fixed point algorithm", Mathematical Programming 22 (1981) 47-69.

```
C********** FILE NAME :  VARDIM.FORT ***************************************
C
C       VARDIM : A FORTRAN PROGRAM OF A VARIABLE DIMENSION FIXED POINT
C                ALGORITHM WITH QUASI-NEWTON PROCEDURE
C
C
C              -- (3**N-1)-METHOD ON TRIANGULATION K-PRIME --
C
C
C*******************************************************************************
C
C          VARDIM IS A FORTRAN 77 (JIS C 6201) PROGRAM FOR COMPUTING
C      AN APPROXIMATE SOLUTION OF A FUNCTION FROM N-DIMENSIONAL
C      EUCLIDEAN SPACE  R**N  INTO ITSELF OR OF AN CLOSED (OR
C      UPPER SEMI-CONTINUOUS) POINT-TO-SET MAPPING FROM  R**N
C      INTO THE SET OF NONEMPTY COMPACT CONVEX SUBSETS OF  R**N.
C      IT EMPLOYS A VARIABLE DIMENSION FIXED POINT ALGORITHM
C      (3**N-1)-METHOD WITH QUASI-MEWTON STEPS.  SEE [1] FOR THE
C      DETAILS OF TEH METHOD.
C
C           TO USE VARDIM THE USER MUST PROVIDE A SUBROUTINE
C
C                SUBROUTINE FUNCT(PROB,NN,X,F)
C                IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C                INTEGER DIM,PROB,NN
C                PARAMETER (DIM=53)
C                DIMENSION X(DIM),F(DIM)
C                          ....
C                RETURN
C                END
C
C      WHICH COMPUTES THE FUNCTION VALUE  F(1),...,F(NN)  AT
C      X = ( X(1),...,X(NN) ).  VARDIM ALREADY HAS A SUBROUTINE
C      FUNCT  WHICH HAS 5 TEST PROBLEMS:
C
C          PROB
C          1,2,3    :  TEST PROBLEMS BY L.T.WATSON (SEE [2])
C          4        :  SPIRAL PROBLEM BY M.J.TODD
C          5        :  NONLINEAR OPTIMIZATION PROBLEM BY L.C.W.DIXON [3]
C
C      THE RESULTS OF SOLVING THESE TEST PROBLEMS ARE FOUND IN [1].
C
C           FOR THE DETAILS OF VARDIM THE USER MIGHT SEE [4].
C
C
C      REFERENCES
C
C      [1]  M.KOJIMA AND Y.YAMAMOTO, "A UNIFIED APPROACH TO THE
C           INPLEMENTATION OF SEVERAL RESTART FIXED POINT ALGORITHMS
C           AND A NEW VARIABLE DIMENSION ALGORITHM" MATHEMATICAL
C           PROGRAMMING (TO APPEAR).
C      [2]  E.ALLGOWER AND K.GEORG, "SIMPLICIAL AND CONTINUATION
C           METHODS FOR APPROXIMATING FIXED POINTS AND SOLUTIONS
C           TO SYSTEMS OF EQUATIONS", SIAM REVIEW 22 (1980) 28-85.
C      [3]  L.C.W.DIXON, "CONJUGATE DIRECTIONS WITHOUT LINEAR SEARCH",
C           JOURNAL OF THE INSTITUTE OF MATHEMATICS AND ITS APPLICATIONS
C           11 (1973) 317-328.
C      [4]  M.KOJIMA AND Y.YAMAMOTO, "A FORTRAM PROGRAM OF THE (3**N-1)
C           METHOD FOR SOLVING SYSTEMS OF EQUATIONS", DISCUSSION
```

```
C          PAPER NO.199(83-22), INSTITUTE OF SOCIO-ECONOMIC PLANNING,
C          UNIVERSITY OF TSUKUBA (IBARAKI, 1984).
C
C
C*** VARIABLES, PARAMETERS, VECTORS  ***********************************
C
C       PROB     : PROBEM NUMBER
C       DIM      : MAX. DIMENSION OF THE SYSTEM OF EQUATIONS
C                  INITIALLY SET TO 50
C       NN       : DIMENXION OF THE SYSTEM OF EQUATIONS
C       XO       : INITIAL POINT
C       X        : VARIABL VECTOR WITH DIM. NN
C       F        : FUNCTION VALUE OF THE SYSTEM
C       AMT      : INVERSE OF ARTIFICIAL MATRIX IN RESTARTING
C       GAMMA    : PARAMETER WHICH DETERMINES DUAL MANIFOLD;
C                  0 < GAMMA < 1
C       RTMIN    : MINIMUM REDUCTION FACTOR OF GRID SIZE
C       RTMAX    : MAXIMUM REDUCTION FACTOR OF GRID SIZE
C       IGRID    : INITIAL GRID SIZE
C       FGRID    : LOWER BOUND OF GRID SIZE
C       CGRID    : CURRENT GRID SIZE
C       ACC      : ACCURACY TO BE ATTAINED
C       QNEWT    : =1 WHEN QUASI-NEWTON PROCEDURE IS INSERTED
C                  =0 OTHERWISE
C       FSTEP    : LOWER BOUND OF STEP SIZE
C       BC       : UPPER BOUND FOR NUMBER OF CYCLES
C       BP       : UPPER BOUND FOR NUMBER OF PIVOT OPERATIONS
C                  PER ONE MAJOR CYCLE
C       BX       : UPPER BOUND FOR ABS(X(I))
C       S , T    : PAIR OF SIGN VECTORS DEFINING PRIMAL CELL X(S,T)
C                  (  S  ALWAYS COMFORMS TO  T  )
C       PERM     : PERMUTATION FOR A PRIMAL SIMPLEX
C       FVECT    : 1-ST VECTOR OF A PRIMAL SIMPLEX
C       NORM     : MAX.ABS(X(I)): I=1,...,NN  OVER ALL VERTICES OF
C                  PRIMAL SIMPLEX
C       VMT      : MATRIX OF ALL VERTICES OF PRIMAL SIMPLEX
C       DM       : DIMENSION OF PRIMAL SIMPLEX
C       INDEX    : INDEX OF BASIC VARIABLES
C                  LET J = INDEX (I), THEN
C                  J > 0  IFF  VARIABLE LAMDA CORRESPONDING TO
C                              J-TH VERTEX OF PRIMAL SIMPLEX IS
C                              I-TH BASIC VARIABLE
C                  J < 0  IFF  (-J)-TH DUAL VARIABLE  Y(-J)  IS I-TH
C                              BASIC VARIABLE
C                  J = 0  IFF  SLACK VARIABLE IS I-TH BASIC VARIABLE
C       BASIS    : VALUE OF BASIC VARIABLES
C       BINV     : INVERSE OF BASIC MATRIX
C       H        : APPROXIMATE JACOBIAN INVVERSE
C       AF       : AMT*F
C       BAF      : BINV*AF
C       BOUND    : UPPER AND LOWER BOUNDS OF BAIC VARIABLES
C       PIVSGN   : +1 IF VALUE OF NEW BASIC VARIABLE WILL INCREASE
C                  -1 IF VALUE OF NEW BASIC VARIABLE WILL DECREASE
C       IDXIN    : INDEX OF NONBASIC VARIABLE TO BE PIVOTED IN
C       VALIN    : VALUE OF NONBASIC VARIABLE TO BE PIVOTED IN
C       LB       : LOWER BOUND OF NONBASIC VARIABLE TO BE PIVOTED IN
C       UB       : UPPER BOUND OF NONBASIC VARIABLE TO BE PIVOTED IN
C       PIVLEN   : STEP LENGTH OF NONBASIC VARIABLE
C       IDXOUT   : INDEX OF BASIC VARIABLE PIVOTED OUT
C       VALOUT   : VALUE OF BASIC VARIABLE PIVOTED OUT
```

```
C       NP       : NUMBER OF PIVOT OPERATIONS IN ONE MAJOR CYCLE
C       NF       : NUMBER OF FUNCTION EVALUATIONS CONSUMED BY FIXED POINT
C                  ALGORITHM IN ONE MAJOR CYCLE
C       NFNEW    : NUMBER OF FUNCTION EVALUATIONS CONSUMED BY QUASI-
C                  NEWTON STEPS IN ONE MAJOR CYCLE
C       TP       : TOTAL NUMBER OF PIVOT OPERATIONS
C       TF       : TOTAL NUMBER OF FUNCTION EVALUATIONS CONSUMED BY FIXED
C                  POINT ALGORITHM
C       TFNEW    : TOTAL NUMBER OF FUNCTION EVALUATIONS CONSUMED BY
C                  QUASI-NEWTON STEPS
C       CYCLE    : TOTAL NUMBER OF CYCLES
C       CTRL     : =1 IF ALGORITHM WILL RESTART AGAIN
C                  =2 IF ALGORITHM HAS SOLVED CURRENT PROBLEM
C
C*** SUBROUTINE AND FUNCTION SUBPROGRAMS ***********************************
C
C       MAIN ---+--  PARAMT
C               +--  INIT
C               +--  INIMAT  ---+---  IDENT
C               +               +
C               +--  PCOL    ---+
C               +--------------+--  FUNCT
C               +--  DCOL
C               +--  PIVOT
C               +--  PDM
C               +--  NEWGR   -----  DIST
C               +--  SOLUT   -----  ALLVTX
C               +--  NEWTON  ---+--  JINV
C               +               +--  BROYD
C               +
C               +--  PRINTA
C               +--  PRINTB
C               +--  PRINTP
C               +--  PRINTR
C               +--  PRINTT
C               +--  PRINTX
C
C                    ENORM
C                    MNORM
C                    SIGND
C                    ALPHA
C                    BETA
C                    DZETA
C
C
C   FUNCTION  ENORM
C                    PROVIDES THE EUCLIDEAN NORM OF N-DIMENSIONAL
C                    VECTOR.
C
C                    INPUT    : N, F
C                    OUTPUT   : ENORM
C
C   FUNCTION  MNORM
C                    PROVIDES THE MAXIMUM NORM OF N-DIMENSIONAL VECTOR.
C
C                    INPUT    : N, X
C                    OUTPUT   : MNORM
C   FUNCTION  SIGND
C                    PROVIDES THE SIGN OF A DOUBLE PRECISION VARIABLE.
C
```

```
C                       INPUT    : A
C                       OUTPUT   : SIGND
C
C    FUNCTION  ALPHA
C                       PROVIDES ALPHA=A**((P/N)**B), WHERE A=0.8, B=2.0
C                       AND P IS THE NUMBER OF ITERATES OF QUASI-NEWTON
C                       STEPS.
C
C                       INPUT    : P, N
C                       OUTPUT   : ALPHA
C
C    FUNCTION  BETA
C                       PROVIDES BETA=LOG2(NO/N1+1), WHERE NO AND N1
C                       ARE NORMS OF FUNCTION VALUES AT OLD AND NEW
C                       QUAIS-NEWTON ITERATES.
C
C                       INPUT    : NO, N1
C                       OUTPUT   : BETA
C
C    FUNCTION  DZETA
C                       PROVIDEDS DZETA.  D=(LOG10(FNORM)+10)/12.
C                                  0    IF     D< 0
C                       DZETA=     D    IF 0<=D<=1
C                                  1    IF     D> 1
C
C                       INPUT    : FNORM
C                       OUTPUT   : DZETA
C
C    SUBROUTINE  NEWGR
C                       PROVIDES A NEW GRID SIZE  CGRID.
C
C                       INPUT    : CGRID, FGRID, FNORM, NO, ACC,
C                                  RTMAX, RTMIN,RESET
C                       OUTPUT   : CGRID,RESET
C
C    SUBROUTINE  ALLVTX
C                       PROVIDES  DM+1 NN-DIMENSIONAL COLUMN VECTORS
C                       VMT(.,.), J-TH COLUMN OF WHICH IS THE J-TH VERTEX
C                       OF A COMPLETE SIMPLEX, WHERE  DM  IS THE DIMENSION
C                       OF THE SIMPLEX.
C
C                       INPUT    : NN, FVECT, PERM, DM, S, T
C                       OUTPUT   : VMT
C
C    SUBROUTINE  DCOL
C                       PROVIDES A (NN+1)-DIMENSIONAL COLUMN VECTOR  AF
C                       CORRESPONDING TO A DUAL VARIABLE, I.E.
C                                1              IF I=-IDXIN
C                       AF(I) = S(-IDXIN)       IF I=NN+1
C                                0              OTHERWISE,
C                       WHERE  -IDXIN  IS THE INDEX OF DUAL VARIABLE TO
C                       BE PIVOTED INTO THE BASIS, AND  S  IS THE NN
C                       DIMENSIONAL SIGN VECTOR DEFINING THE CURRENT
C                       DUAL CELL WITH A SIGN VECTOR T.
C
C                       INPUT    : MM (=NN+1), IDXIN, S
C                       OUTPUT   : AF
C
C    SUBROUTINE  PCOL
C                       PROVIDES A (NN+1)-DIMENSIONAL COLUMN VECTOR  AF
```

```
C                       CORRESPONDING TO A PRIMAL VARIABLE, I.E.
C                       NN-DIMENSIONAL VECTOR OF THE FIRST NN COMPONENTS
C                       OF  AF  IS
C                         AMT * FUNCTION VALUE AT THE VERTEX TO BE
C                               PIVOTED INTO,
C                       AND THE LAST ( =(NN+1)-TH ) COMPONENT IS ZERO,
C                       WHERE  AMT  IS THE MATRIX OF THE ARTIFICIAL
C                       FUNCTION.
C
C                       INPUT   : NN, XO, CGRID, FVECT, PERM, IDXIN
C                                 S, T, PROB, NF, TF, AMT
C                       OUTPUT  : AF
C
C     SUBROUTINE  INIMAT
C                       PROVIDES INITIAL BASIC REPRESENTATION OF THE
C                       SYSTEM
C                           Y + T*F(X) = C,   (X,Y) IN PDM, T >= 0.
C                       IT FINDS AN (NN-1)-FACET Y(S,S) OF THE DUAL
C                       SUBDIVIDED MANIFOLD WHICH THE VARIABLE
C                       Y = C - T*F(XO)  ENCOUNTERS AS T INCRESES,
C                       WHERE  XO  IS THE INITIAL POINT.  WE ADOPT
C                       A PERTURBATION VECTOR  PTB  AS THE RIGHT
C                       HAND CONSTANT  C.  INITIAL BASIC SOLUTIONS
C                       ARE GIVEN IN  BASIS.  INITIAL BASIS INVERSE
C                       IS GIVEN IN  BINV.  INDICES OF THE BASIC
C                       SOLUTIONS ARE GIVEN IN INDEX.  WHEN THE EUCLIDEAN
C                       NORM OF  F(XO) <= ACC, CTRL  IS SET TO 2.
C
C                       INPUT   : PROB, NN, X, NF, TF, GAMMA, ACC
C                       OUTPUT  : S, FVECT, PERM, BASIS, BINV, INDEX,
C                                 BOUND, DM, NORM, CTRL
C
C     SUBROUTINE  PDM
C                       PROVIDES THE INDEX OF
C                         - THE NEW VERTEX TO BE PIVOTED INTO WHEN THE
C                           PRIMAL VARIABLE REMAINS IN THE CURRENT PRIMAL
C                           CELL,
C                         - THE NEW DUAL VARIABLE TO BE PIVOTED INTO
C                           WHEN THE PRIMAL VARIABLE ENCOUNTERS A FACET
C                           OF THE CURRENT PRIMAL CELL.
C                       IT ALSO UPDATES
C                         - THE PAIR OF SIGN VECTORS  S  AND  T  WHICH
C                           DEFINE THE NEW PRIMAL AND DUAL CELLS,
C                         - REPRESENTATION OF SIMPLEX, FVECT, PERM, DM.
C
C                       INPUT   : IDXOUT, DM, PERM, FVECT, S, T, NORM,
C                                 GAMMA, INDEX
C                       OUTPUT  : DM, FVECT, PERM, NORM, INDEX, INDXIN,
C                                 VALIN, PIVSGN, UB, LB, BOUND, S, T
C
C     SUBROUTINE  PIVOT
C                       SETS THE CONTROL VARIABLE  CTRL  TO 2 WHEN AN
C                       UNBOUNDED RAY IS FOUND TO THE SYSTEM OF EQUATIONS.
C                       OTHERWISE, CTRL = 1  AND PIVOT OPERATIONS IS
C                       PERFORMED.
C
C                       INPUT   : AF, BINV, UB, LB, PIVSGN, PIVLEN, BOUND,
C                                 BASIS
C                       OUTPUT  : CTRL, BASIS, IDXOUT, VALOUT, BINV, INDEX,
C                                 BOUND
```

```
C
C    SUBROUTINE  SOLUT
C                     COMPUTES AN APPROXIMATE SOLUTION FROM THE
C                     UNBOUNDED RAY OBTAINED IN SUBROUTINE PIVOT.
C
C                     INPUT    : IDXIN, INDEX, BAF, VMT, XO, CGRID
C                     OUTPUT   : X
C
C    SUBROUTINE  PARAMT
C                     SETS UP PARAMETERS AND INITIAL APPROXIMATE
C                     SOLUTION.
C
C                     INPUT    : PROB, NN, XO, IGRID, ACC, FSTEP, BX,
C                                QNEWT
C                     OUTPUT   : GAMMA, IGRID, RTMAX, RTMIN, FGRID, ACC,
C                                BC, BP, PROB, NN, XO, RESET
C
C                    DEFAULT VALUES OF PARAMETERS ARE:
C                              GAMMA = 0.5 / NN
C                              FGRID = 1.0D-7
C                              BC    = 100
C                              BP    = 400 * NN
C                    DEFAULT INITIAL APPROXIMATE SOLUTION IS:
C                              XO(I) = 0.0  FOR EVERY I
C
C
C    SUBROUTINE  INIT
C                     INITIALIZES SEVERAL VARIABLES.
C
C                     INPUT    : NN, IGRID, XO
C                     OUTPUT   : MM, CYCLE, TP, TF, CGRID, AMT, X, CTRL,
C                                RESET
C
C    SUBROUTINE  IDENT
C                     PROVIDES  N X N IDENTITY MATRIX IN  MAT.
C
C                     INPUT    : N
C                     OUTPUT   : MAT
C
C
C    SUBROUTINE  NEWTON
C                     PERFORMS QUASI-NEWTON STEPS.
C
C                     INPUT    : ZMIN, FMIN, NMIN, SMIN, PROB, NN,
C                                CGRID, ACC, BINV, V, VMT, INDEX
C                     OUTPUT   : ZMIN, FMIN, NMIN, SMIN, CTRL
C
C    SUBROUTINE  BROYD
C                     UPDATES APPROXIMATE JACOBIAN INVERSE BY BROYDEN'S
C                     FORMULA.
C
C                     INPUT    : NN, H, ZF0, ZF1, S
C                     OUTPUT   : H
C
C    SUBROUTINE  JINV
C                     PROVIDES APPROXIMATE JACOBIAN INVERSE.
C
C                     INPUT    : NN, BINV, V, VMT, INDEX
C                     OUTPUT   : H
C    SUBROUTINE  PRINTT  AND  PRINTP
```

```
C                       PRINTS TITLE AND PARAMETERS.
C
C                       INPUT    : GAMMA, IGRID, ACC, FSTEP, PROB, NN
C                       OUTPUT   :
C
C    SUBROUTINE  PRINTA
C                       PRINTS SEVERAL STATISTICS.
C
C                       INPUT    : CYCLE, CGRID, NP, NF, NFNEW, FNORM,
C                                  DNORM
C                       OUTPUT   :
C
C    SUBROUTINE  PRINTX
C                       PRINTS COORDINATES OF VECTOR  X.
C
C                       INPUT    : X, NN, IR
C                       OUTPUT   :
C
C    SUBROUTINE  PRINTR
C                       PRINTS FINAL RESULTS.
C
C                       INPUT    : CYCLE, CGRID, TP, TF, TFNEW, FNORM,
C                                  DNORM, X, F, IR
C                       OUTPUT   :
C
C    SUBROUTINE  FUNCT
C                       IS THE USEROSUPPLIED SUBROUTINE WHICH PROVIDES
C                       FUNCTION VALUE.
C
C                       INPUT    : PROB, NN, X
C                       OUTPUT   : F
C
C
C
C*******************************************************************************
C     MAIN PROGRAM
C*******************************************************************************
C
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      INTEGER DIM
      PARAMETER (DIM=53)
      INTEGER BC,BP,CYCLE,IDXIN,IDXOUT,NF,NP,PROB,CTRL,TF,TP
      INTEGER PIVSGN,DM,S(DIM),T(DIM),PERM(DIM)
      INTEGER FVECT(DIM),VMT(DIM,DIM)
      INTEGER IR(25),NFNEW,TFNEW,QNEWT,RESET
      DOUBLE PRECISION IGRID,LB
      DOUBLE PRECISION AMT(DIM,DIM),F(DIM),X(DIM),XO(DIM),MNORM
      DOUBLE PRECISION AF(DIM),BAF(DIM),BINV(DIM,DIM),BOUND(DIM,2)
      DOUBLE PRECISION BASIS(DIM),DIFF(DIM),FO(DIM),NO,V(DIM)
      COMMON /GLOBAL/F,AMT,MM,NN,X,XO,EPS
      COMMON /CPDM1/GAMMA
      COMMON /CPDM2/DM,NORM,T,PERM,S,FVECT,VMT
      COMMON /CPIV1/AF,BAF,BINV,BOUND,BASIS
      COMMON /CPIV2/INDEX
      COMMON /STATIS/ NF,TF,NP,TP,NFNEW,TFNEW
C
C<<<  INITIALIZATION  >>>
C
   10 CONTINUE
```

```
      CALL PARAMT(PROB,NN,GAMMA,IGRID,FGRID,XO,BC,BP,BX,ACC,EPS,
     &            FSTEP,RTMAX,RTMIN,QNEWT)
      CALL PRINTP(ACC,GAMMA,IGRID,PROB,NN,FSTEP,QNEWT)
      CALL INIT(MM,NN,CYCLE,TP,TF,CGRID,IGRID,AMT,XO,X,IR,CRTL,
     &          NFNEW,TFNEW,RESET)
      WRITE(6,20)
   20 FORMAT(/1H ,'INITIAL POINT')
      CALL PRINTX (XO,IR)
      CALL PRINTT
      CALL PRINTB
C
C<<<  MAJOR CYCLE  >>>
C
  100 CYCLE=CYCLE+1
      IF (CYCLE.GE.BC) GOTO 700
  110 NP=1
      TP=TP+1
      NF=0
      NFNEW=0
      IF (CYCLE.EQ.1) GOTO 120
      NF=NF-1
      TF=TF-1
  120 CALL INIMAT(PROB,ACC,CTRL,FO,NO)
      IF (CTRL.EQ.2) GOTO 510
C
C<<<  1-ST COLUMN PIVOTED IN  >>>
C
      IDXIN=1
      VALIN=0.0D0
      LB=0.0D0
      UB=1.0D15
      PIVSGN=1
      NORM=0
      GOTO 400
C
C<<<  MINOR CYCLE  >>>
C
  200 CONTINUE
      IF (NP.GE.BP) GOTO 710
  210 IF (IDXIN) 220,230,300
C
C<<<  DUAL COLUMN WILL BE PIVOTED IN  >>>
C
  220 CALL DCOL(AF,IDXIN)
      GOTO 400
C
C<<<  IDXIN = 0  >>>
C
  230 STOP '** ERROR IN MAIN, IDXIN = 0'
C
C<<<  PRIMAL COLUMN WILL BE PIVOTED IN  >>>
C
  300 CALL PCOL(PROB,AF,CGRID,IDXIN)
      XMAX=MNORM(NN,X)
      IF (XMAX.GE.BX) GOTO 720
  310 FNORM=ENORM(NN,F)
      IF (FNORM.GT.ACC)
     &  THEN
          GOTO 400
        ELSE
```

```
            DO 320 I=1,NN
            DIFF(I)=X(I)-XO(I)
  320       CONTINUE
            DNORM=ENORM(NN,DIFF)
            CALL PRINTA(CYCLE,CGRID,NP,NF,NFNEW,FNORM,DNORM)
            CALL PRINTB
            CALL PRINTR(CYCLE,CGRID,TP,TF,TFNEW,FNORM,DNORM,X,F,IR)
            GOTO 10
          END IF
C
C<<<  PIVOT  >>>
C
  400 CONTINUE
          CALL PIVOT(CTRL,IDXIN,VALIN,PIVSGN,LB,UB,IDXOUT,VALOUT)
          NP=NP+1
          TP=TP+1
          GOTO (410,500),CTRL
  410 CALL PDM(IDXOUT,VALOUT,IDXIN,VALIN,LB,UB,PIVSGN)
          GOTO 200
C
C<<<  APPROXIMATE SOLUTION  >>>
C
  500 CALL SOLUT(IDXIN,CGRID,V)
          DO 505 I=1,NN
          DIFF(I)=X(I)-XO(I)
  505 CONTINUE
          CALL FUNCT(PROB,NN,X,F)
          NF=NF+1
          TF=TF+1
  510 FNORM=ENORM(NN,F)
  520 DNORM=ENORM(NN,DIFF)
          IF (FNORM.LE.ACC.OR.DNORM.LE.FSTEP)
     &   THEN
C
C<<<  APPROXIMATE SOLUTION WITH GIVEN ACCURACY HAS BEEN OBTAINED  >>>
C
            CALL PRINTA(CYCLE,CGRID,NP,NF,NFNEW,FNORM,DNORM)
            CALL PRINTB
            CALL PRINTR(CYCLE,CGRID,TP,TF,TFNEW,FNORM,DNORM,X,F,IR)
            GOTO 10
          ELSE
C
C<<<  GIVEN ACCURACY HAS NOT BEEN ATAINED YET  >>>
C
            IF (DM.LT.NN.OR.QNEWT.EQ.0)
     &        THEN
              SMIN=-1.0D0
            ELSE
  610         CONTINUE
              CALL NEWTON(X,F,FNORM,SMIN,PROB,NN,CGRID,ACC,BINV,V,VMT,
     &                    INDEX,CTRL)
              DO 600 I=1,NN
  600         DIFF(I)=X(I)-XO(I)
              DNORM=ENORM(NN,DIFF)
              IF (CTRL.EQ.2)
     &          THEN
                CALL PRINTA(CYCLE,CGRID,NP,NF,NFNEW,FNORM,DNORM)
                CALL PRINTB
                CALL PRINTR(CYCLE,CGRID,TP,TF,TFNEW,FNORM,DNORM,X,F,
     &                      IR)
```

```
                  GOTO 10
                ELSE
              END IF
            END IF
            CALL PRINTA(CYCLE,CGRID,NP,NF,NFNEW,FNORM,DNORM)
            IF (RESET.EQ.0) GOTO 625
            DO 620 I=1,NN
            XO(I)=X(I)
  620       FO(I)=F(I)
            NO=FNORM
            GOTO 730
  625       CALL NEWGR(CGRID,FGRID,FNORM,NO,SMIN,DNORM,ACC,RTMAX,
     &                  RTMIN,RESET)
            DO 630 I=1,NN
            XO(I)=X(I)
            FO(I)=F(I)
  630       CONTINUE
            NO=FNORM
            GOTO 100
          END IF
C
C<<<  STOP  >>>
C
  700 WRITE(6,800) CYCLE,BC
  800 FORMAT(1H ,I4,'   STOP BECAUSE NUMBER OF CYCLES => ',I5)
      GOTO 900
  710 WRITE(6,810) CYCLE,BP
  810 FORMAT(1H ,I4,'   STOP BECAUSE NUMBER OF PIVOTS => ',I5)
      GOTO 900
  720 WRITE(6,820) CYCLE,BX
  820 FORMAT(1H ,I4,'   STOP BECAUSE MAX. NORM OF X => ',D12.4)
      GOTO 900
  730 WRITE(6,830) CYCLE,FGRID
  830 FORMAT(1H ,I4,'   STOP BECAUSE GRID SIZE <= ',D12.4)
  900 CONTINUE
      CALL PRINTB
      CALL PRINTR(CYCLE,CGRID,TP,TF,TFNEW,NO,DNORM,XO,FO,IR)
      GOTO 10
      END
C
C
C
C*****************************************************************************
C     FUNCTION ENORM
C*****************************************************************************
      FUNCTION ENORM(N,F)
      INTEGER DIM
      PARAMETER (DIM=53)
      INTEGER I,N
      DOUBLE PRECISION F(DIM),ENORM
C
      ENORM=0.0D0
      DO 10 I=1,N
   10 ENORM=ENORM+F(I)*F(I)
      ENORM=DSQRT(ENORM)
      RETURN
      END
C
C
C
```

```
C*****************************************************************************
C     FUNCTION MNORM
C*****************************************************************************
      FUNCTION MNORM(N,X)
      INTEGER DIM
      PARAMETER (DIM=53)
      INTEGER I,N
      DOUBLE PRECISION A,X(DIM),MNORM
C
      MNORM=0.0D0
      DO 10 I=1,N
      A=DABS(X(I))
   10 IF (A.GT.MNORM) MNORM=A
      RETURN
      END
C
C
C
C*****************************************************************************
C     FUNCTION SIGND
C*****************************************************************************
      FUNCTION SIGND(A)
      INTEGER SIGND
      DOUBLE PRECISION A
C
      SIGND=0
      IF (A.GT.0.0D0) SIGND=+1
      IF (A.LT.0.0D0) SIGND=-1
      RETURN
      END
C
C
C
C*****************************************************************************
C     SUBROUTINE NEWGR
C*****************************************************************************
      SUBROUTINE NEWGR(CGRID,FGRID,FNORM,NO,SMIN,DNORM,ACC,
     &                 RTMAX,RTMIN,RESET)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      INTEGER DIM
      PARAMETER (DIM=53)
      INTEGER MM,NN,RESET
      DOUBLE PRECISION F(DIM),AMT(DIM,DIM),X(DIM)
      DOUBLE PRECISION XO(DIM),NO
      COMMON /GLOBAL/F,AMT,MM,NN,X,XO,EPS
C
      GRMAX=RTMAX*CGRID
      GRMIN=RTMIN*CGRID
      IF (FNORM.GE.NO)
     &  THEN
C
C<<<  FNORM INCREASES  >>>
C
          CGRID=GRMAX
          GOTO 100
        ELSE
C
C<<<  FNORM DECREASES  >>>
C
          CALL DIST(ACC,NO,FNORM,DNORM,D1)
```

```
            D1=DABS(D1)
            IF (SMIN.LT.0.0D0)
     &        THEN
                CGRID=D1
                GOTO 10
              ELSE
                D2=2.0D0*SMIN
                DZET=DZETA(FNORM)
                CGRID=DZET*D1+(1.0D0-DZET)*D2
            END IF
   10       CONTINUE
            IF (CGRID.LT.GRMIN) CGRID=GRMIN
            IF (CGRID.GT.GRMAX) CGRID=GRMAX
         END IF
  100  IF (CGRID.GE.FGRID) RETURN
       CGRID=FGRID
       RESET=1
       RETURN
       END
C
C
C
C*************************************************************************
C      SUBROUTINE DIST
C*************************************************************************
       SUBROUTINE DIST(ACC,Y0,Y1,D,DEST)
       IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
C<<< QUADRATIC ESTIMATION >>>
C
       SUM=Y0+Y1
       DEST=(Y1-ACC)*SUM*D/(Y0**2-Y1**2)
       RETURN
       END
C
C
C
C*************************************************************************
C      FUNCTION DZETA
C*************************************************************************
       FUNCTION DZETA(FNORM)
       IMPLICIT DOUBLE PRECISION (A-H,O-Z)
C
       DZETA=(DLOG10(FNORM)+10.0D0)/12.0D0
       IF (DZETA.GT.1.0D0) DZETA=1.0D0
       IF (DZETA.LT.0.0D0) DZETA=0.0D0
       RETURN
       END
C
C
C
C*************************************************************************
C     SUBROUTINE ALLVTX
C*************************************************************************
       SUBROUTINE ALLVTX
       INTEGER DIM
       PARAMETER (DIM=53)
       INTEGER MM,NN
       INTEGER DM,NORM,T(DIM),PERM(DIM),S(DIM),FVECT(DIM)
       INTEGER VMT(DIM,DIM)
```

```
      DOUBLE PRECISION AMT(DIM,DIM),F(DIM),GAMMA,X(DIM),XO(DIM)
      COMMON /GLOBAL/F,AMT,MM,NN,X,XO,EPS
      COMMON /CPDM1/GAMMA
      COMMON /CPDM2/DM,NORM,T,PERM,S,FVECT,VMT
C
      DO 100 I=1,NN
      VMT(I,1)=FVECT(I)
  100 CONTINUE
      DO 200 J=1,DM
      K=PERM(J)
      IF (K.EQ.MM)
     &  THEN
          DO  110 I=1,NN
          VMT(I,J+1)=VMT(I,J)+S(I)
  110     CONTINUE
        ELSE
          DO  120 I=1,NN
          VMT(I,J+1)=VMT(I,J)
  120     CONTINUE
          VMT(K,J+1)=VMT(K,J)+T(K)
      END IF
  200 CONTINUE
      RETURN
      END
C
C
C
C*********************************************************************************
C     SUBROUTINE DCOL
C*********************************************************************************
      SUBROUTINE DCOL(AF,IDXIN)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      INTEGER DIM
      PARAMETER (DIM=53)
      INTEGER IDXIN
      INTEGER MM,NN
      INTEGER DM,NORM,T(DIM),PERM(DIM),S(DIM),FVECT(DIM)
      INTEGER VMT(DIM,DIM)
      DOUBLE PRECISION AF(DIM),AMT(DIM,DIM),F(DIM),GAMMA,X(DIM),XO(DIM)
      COMMON /GLOBAL/F,AMT,MM,NN,X,XO,EPS
      COMMON /CPDM1/GAMMA
      COMMON /CPDM2/DM,NORM,T,PERM,S,FVECT,VMT
C
      DO 100 I=1,MM
  100 AF(I)=0.0D0
      AF(-IDXIN)=1.0D0
      AF(MM)=DFLOAT(S(-IDXIN))
      RETURN
      END
C
C
C
C*********************************************************************************
C     SUBROUTINE PCOL
C*********************************************************************************
      SUBROUTINE PCOL(PROB,AF,CGRID,IDXIN)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      INTEGER DIM
      PARAMETER (DIM=53)
      INTEGER IDXIN,MM,NN,PROB,NF,TF,NP,TP,TFNEW
```

```
      INTEGER DM,NORM,T(DIM),PERM(DIM),S(DIM),FVECT(DIM)
      INTEGER VMT(DIM,DIM)
      DOUBLE PRECISION AF(DIM),AMT(DIM,DIM),F(DIM),GAMMA,X(DIM),XO(DIM)
      COMMON /GLOBAL/F,AMT,MM,NN,X,XO,EPS
      COMMON /CPDM1/GAMMA
      COMMON /CPDM2/DM,NORM,T,PERM,S,FVECT,VMT
      COMMON /STATIS/ NF,TF,NP,TP,NFNEW,TFNEW
C
C<<< COORDINATES OF THE VERTEX TO BE PIVOTED INTO >>>
C
      DO 100 I=1,NN
  100 X(I)=XO(I)+CGRID*DFLOAT(FVECT(I))
      DO 130 I=1,IDXIN-1
      K=PERM(I)
      IF (K.EQ.MM)
     &  THEN
          DO 120 J=1,NN
  120     X(J)=X(J)+CGRID*DFLOAT(S(J))
        ELSE
          X(K)=X(K)+CGRID*DFLOAT(T(K))
      END IF
C
C<<< FUNCTION VALUE AT THE VERTEX AND PIVOT-IN COLUMN >>>
C
  130 CONTINUE
      CALL FUNCT(PROB,NN,X,F)
      NF=NF+1
      TF=TF+1
      DO 150 I=1,NN
      AF(I)=0.0D0
      DO 140 J=1,NN
  140 AF(I)=AF(I)+AMT(I,J)*F(J)
  150 CONTINUE
      AF(MM)=0.0D0
      RETURN
      END
C
C
C
C*********************************************************************
C     SUBROUTINE INIMAT
C*********************************************************************
      SUBROUTINE INIMAT(PROB,ACC,CTRL,FO,NO)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      INTEGER DIM
      PARAMETER (DIM=53)
      INTEGER PROB,NF,TF,NP,TP,TFNEW
      INTEGER INDEX(DIM),MM,NN,SIGND
      INTEGER DM,NORM,T(DIM),PERM(DIM),S(DIM),FVECT(DIM)
      INTEGER VMT(DIM,DIM),CTRL
      DOUBLE PRECISION AMT(DIM,DIM),F(DIM),GAMMA,X(DIM),XO(DIM)
      DOUBLE PRECISION AF(DIM),BAF(DIM),BINV(DIM,DIM),BOUND(DIM,2)
      DOUBLE PRECISION BASIS(DIM),V(DIM),MAX,MIN
      DOUBLE PRECISION PO,PTB(DIM),RH,SUM1,SUM2,RATIO,FO(DIM),NO
      COMMON /GLOBAL/F,AMT,MM,NN,X,XO,EPS
      COMMON /CPDM1/GAMMA
      COMMON /CPDM2/DM,NORM,T,PERM,S,FVECT,VMT
      COMMON /CPIV1/AF,BAF,BINV,BOUND,BASIS
      COMMON /CPIV2/INDEX
      COMMON /STATIS/ NF,TF,NP,TP,NFNEW,TFNEW
```

```
C
C<<< FUNCTION VALUE AT THE INITIAL POINT >>>
C
      CALL FUNCT(PROB,NN,XO,AF)
      DO 1 I=1,NN
      FO(I)=AF(I)
    1 CONTINUE
      FNEW=ENORM(NN,AF)
      NO=FNEW
      IF (FNEW.LE.ACC)
     &  THEN
          CTRL=2
          RETURN
        ELSE
          CTRL=1
      END IF
      AF(MM)=0.0D0
      NF=NF+1
      TF=TF+1
C
      CALL IDENT(AMT,NN)
      DO 10 I=1,NN
      V(I)=DABS(AF(I))
      IF (AF(I).EQ.0.0)
     &  THEN
          S(I)=-1
        ELSE
          S(I)=-SIGND(AF(I))
      END IF
   10 CONTINUE
C
C<<< PERMUTATION SUCH THAT ABS(AF(PERM(K))) >= ABS(AF(PERM(K+1))) >>>
C
      DO 20 K=1,NN
      MAX=0.0
      II=0
      DO 30 I=1,NN
      IF (MAX.GT.V(I)) GOTO 30
      II=I
      MAX=V(I)
   30 CONTINUE
      PERM(K)=II
      V(II)=-1.0
   20 CONTINUE
C
C<<< PERTURBATION VECTOR PTB >>>
C
      PO=GAMMA/10.0D0
      DO 40 K=1,NN
      I=PERM(K)
      PTB(I)=(PO**DLOG(DFLOAT(K+2)))*DFLOAT(S(I))
   40 CONTINUE
C
C<<< (NN-1)-FACET OF DUAL SUBDIVIDED MANIFOLD >>>
C
      MIN=1.0D15
      SUM1=0.0
      SUM2=0.0
      DO 50 K=1,NN
      RH=1.0-DFLOAT(NN-K)*GAMMA
```

```
      I=PERM(K)
      SUM1=SUM1+DFLOAT(S(I))*PTB(I)
      SUM2=SUM2+DFLOAT(S(I))*AF(I)
      RATIO=(SUM1-RH)/SUM2
      IF (RATIO.LT.MIN)
     &  THEN
          MIN=RATIO
          KK=K
          BASIS(MM)=RH-SUM1
        ELSE
      END IF
   50 CONTINUE
C
C<<< INITIALIZATION OF BINV, INDEX, BASIS, BOUND, FVECT, DM, NORM,
C     S, T >>>
C
      CALL IDENT(BINV,MM)
      DO 100 K=1,NN
      I=PERM(K)
      IF (K.GT.KK) S(I)=0
     - T(I)=S(I)
      BINV(MM,I)=-DFLOAT(S(I))
      PERM(K)=0
  100 CONTINUE
      DO 110 I=1,NN
      FVECT(I)=0
      INDEX(I)=-I
      BOUND(I,1)=-1.0D15
      BOUND(I,2)= 1.0D15
      BASIS(I)=PTB(I)
  110 CONTINUE
      INDEX(MM)=0
      BOUND(MM,1)=0.0
      BOUND(MM,2)=1.0D15
      DM=0
      NORM=0
      RETURN
      END
C
C
C
C*************************************************************************
C     SUBROUTINE PDM
C     PRIMAL-DUAL MANIFOLD WITH  K  PRIME
C*************************************************************************
      SUBROUTINE PDM(IDXOUT,VALOUT,IDXIN,VALIN,LB,UB,PIVSGN)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      INTEGER DIM
      PARAMETER (DIM=53)
      INTEGER INDEX(DIM),MM,NN,SIGND,IDXIN,IDXOUT,LAB,PIVSGN
      INTEGER DM,NORM,T(DIM),PERM(DIM),S(DIM),FVECT(DIM)
      INTEGER VMT(DIM,DIM)
      DOUBLE PRECISION AMT(DIM,DIM),F(DIM),GAMMA,X(DIM),XO(DIM)
      DOUBLE PRECISION VALOUT,VALIN,LB,UB
      DOUBLE PRECISION AF(DIM),BAF(DIM),BINV(DIM,DIM),BOUND(DIM,2)
      DOUBLE PRECISION BASIS(DIM)
      COMMON /GLOBAL/F,AMT,MM,NN,X,XO,EPS
      COMMON /CPDM1/GAMMA
      COMMON /CPDM2/DM,NORM,T,PERM,S,FVECT,VMT
      COMMON /CPIV1/AF,BAF,BINV,BOUND,BASIS
```

```
      COMMON /CPIV2/INDEX
C
      IF (IDXOUT.LT.0)                                  LAB=5
      IF (IDXOUT.EQ.0)                                  LAB=1
      IF (IDXOUT.EQ.1)                                  LAB=2
      IF (IDXOUT.GT.1.AND.IDXOUT.LT.DM+1)               LAB=3
      IF (IDXOUT.EQ.DM+1)                               LAB=4
      IF (IDXOUT.GT.DM+1)                               STOP
     &'** ERROR IN SUBROUTINE PDM, IDXOUT > DM + 1'
     1 GOTO (10,100,200,300,500),LAB
C
C<<<  IDXOUT=0 ; SLACK VARIABLE IS PIVOTED OUT AT 1-ST PIVOT  >>>
C
   10 DM=1
      NORM=1
      PERM(1)=NN+1
      IDXIN=2
      VALIN=0.0D0
      PIVSGN=1
      LB=0.0D0
      UB=1.0D15
      DO 50 I=1,NN
        IF (T(I).NE.0) GOTO 20
        BOUND(I,1)=-GAMMA
        BOUND(I,2)= GAMMA
        GOTO 50
   20   IF (S(I).EQ.1) GOTO 30
        BOUND(I,1)=-1.0D15
        BOUND(I,2)=-GAMMA
        GOTO 50
   30   BOUND(I,1)=GAMMA
        BOUND(I,2)=1.0D15
   50  CONTINUE
      RETURN
C
C<<<  IDXOUT = 1  >>>
C
  100 CONTINUE
      K=PERM(1)
      IF (K.EQ.NN+1)
     &  THEN
          DO  140 I=1,NN
          FVECT(I)=FVECT(I)+S(I)
  140     CONTINUE
        ELSE
          FVECT(K)=FVECT(K)+T(K)
      END IF
      DO  150 I=1,DM-1
      PERM(I)=PERM(I+1)
  150 CONTINUE
      PERM(DM)=K
      IF (K.EQ.NN+1) NORM=NORM+1
      DO  160 I=1,NN+1
      IF (INDEX(I).GT.0) INDEX(I)=INDEX(I)-1
  160 CONTINUE
      IDXIN=DM+1
      VALIN=0.0D0
      PIVSGN=1
      LB=0.0D0
      UB=1.0D15
```

```
      RETURN
C
C<<<  1 < IDXOUT < DM + 1  >>>
C
  200 CONTINUE
      VALIN=0.0D0
      PIVSGN=1
      LB=0.0D0
      UB=1.0D15
      K=PERM(IDXOUT)
      K1=PERM(IDXOUT-1)
      JJ=T(K)
      IA=IABS(FVECT(K))
      IF ((K1.EQ.NN+1).AND.(IA.GE.NORM-1))
     &  THEN
           S(K)=JJ
           T(K)=JJ
           DM=DM-1
           IDXIN=-K
           VALIN=GAMMA*DFLOAT(JJ)
           DO 220 I=IDXOUT,DM+1
           PERM(I)=PERM(I+1)
  220      CONTINUE
           PERM(DM+1)=0
           DO  230 I=1,NN+1
           IF (INDEX(I).GT.IDXOUT) INDEX(I)=INDEX(I)-1
  230      CONTINUE
           IF (JJ.GE.0)
     &       THEN
               PIVSGN=1
               LB=GAMMA
               UB=1.0D15
               RETURN
             ELSE
  240          PIVSGN=-1
               LB=-1.0D15
               UB=-GAMMA
               RETURN
           END IF
         ELSE
           PERM(IDXOUT)=K1
           PERM(IDXOUT-1)=K
           IDXIN=IDXOUT
           RETURN
      END IF
C
C<<<  IDXOUT = DM + 1  >>>
C
  300 CONTINUE
      IDXIN=1
      VALIN=0.0D0
      PIVSGN=1
      LB=0.0D0
      UB=1.0D15
      K=PERM(DM)
      IF (K.LT.NN+1)
     &  THEN
           IF (FVECT(K).NE.0) GOTO  320
           VALIN=GAMMA*DFLOAT(T(K))
           PERM(DM)=0
```

```
            DM=DM-1
            IDXIN=-K
            PIVSGN=-T(K)
            S(K)=0
            T(K)=0
            LB=-GAMMA
            UB= GAMMA
            RETURN
          ELSE
            NORM=NORM-1
        END IF
  320 CONTINUE
        IF (DM.LT.2) GOTO  340
        DO 330 I=DM,2,-1
        PERM(I)=PERM(I-1)
  330 CONTINUE
  340 PERM(1)=K
        IF (K.EQ.NN+1)
     &  THEN
            DO  370 I=1,NN
            FVECT(I)=FVECT(I)-S(I)
  370       CONTINUE
          ELSE
            FVECT(K)=FVECT(K)-T(K)
        END IF
  360 CONTINUE
        DO  380 I=1,NN+1
        IF (INDEX(I).GT.0) INDEX(I)=INDEX(I)+1
  380 CONTINUE
        RETURN
C
C<<<  INCREASING DIMENSION OF PRIMAL SIMPLEX  >>>
C
  500 IADD=-IDXOUT
        DM=DM+1
        VALIN=0.0D0
        PIVSGN=1
        LB=0.0D0
        UB=1.0D15
        K=S(IADD)+T(IADD)
        IF ((K.EQ.2).OR.(K.EQ.-2))
     &  THEN
            I=0
  520       I=I+1
            IF (PERM(I).NE.NN+1) GOTO  520
            DO  530 J=DM+1,I+1,-1
            PERM(J)=PERM(J-1)
  530       CONTINUE
            PERM(I+1)=IADD
            IF (K.GT.0) T(IADD)=1
            IF (K.EQ.0) T(IADD)=0
            IF (K.LT.0) T(IADD)=-1
            S(IADD)=0
            IDXIN=I+1
            DO  540 I=1,NN+1
            IF (INDEX(I).GE.IDXIN)  INDEX(I)=INDEX(I)+1
  540       CONTINUE
          ELSE
            T(IADD)=SIGND(VALOUT)
            S(IADD)=0
```

```
          PERM(DM)=IADD
          IDXIN=DM+1
        END IF
        RETURN
        END
C
C
C
C*************************************************************************
C     SUBROUTINE PIVOT
C*************************************************************************
        SUBROUTINE PIVOT(CTRL,IDXIN,VALIN,PIVSGN,LB,UB,IDXOUT,VALOUT)
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
        INTEGER DIM
        PARAMETER (DIM=53)
        INTEGER CTRL,IDXIN,IDXOUT,PIVSGN,BSSGN
        INTEGER INDEX(DIM),MM,NN
        DOUBLE PRECISION VALIN,LB,UB,VALOUT
        DOUBLE PRECISION AMT(DIM,DIM),F(DIM),X(DIM),XO(DIM)
        DOUBLE PRECISION AF(DIM),BAF(DIM),BINV(DIM,DIM),BOUND(DIM,2)
        DOUBLE PRECISION BASIS(DIM)
        COMMON /GLOBAL/F,AMT,MM,NN,X,XO,EPS
        COMMON /CPIV1/AF,BAF,BINV,BOUND,BASIS
        COMMON /CPIV2/INDEX
C
        CTRL=1
C
C<<<  BAF = BINV * AF  >>>
C
        DO 20 I=1,MM
        BAF(I)=0.0D0
        DO 10 J=1,MM
   10 BAF(I)=BAF(I)+BINV(I,J)*AF(J)
   20 CONTINUE
C
C<<<  FINDING PIVOT ROW  >>>
C
        PIVLEN=UB-LB
        IJ=9999
        DO 100 I=1,MM
        DA=DABS(BAF(I))
        IF (DA.LT.EPS) GOTO 100
   30 CONTINUE
        IF (BAF(I).GE.0)
      &  THEN
            BSSGN=PIVSGN
          ELSE
            BSSGN=-PIVSGN
        END IF
        IF (BSSGN) 40,35,60
   35 STOP '** ERROR IN SUBROUTINE PIVOT, PIVSGN = 0 **'
C
C<<<  BSSGN < 0  >>>
C
   40 IF (BOUND(I,2).GT.1.0D14) GOTO 100
        A=BOUND(I,2)-BASIS(I)-PIVLEN*DA
        IF (A.GT.0.0D0) GOTO 100
        IF (A.LT.0.0D0) GOTO 50
        STOP '****** TIE *******'
   50 IJ=I
```

```
      PIVLEN=(BOUND(I,2)-BASIS(I))/DA
      GOTO 100
C
C<<<  BSSGN > 0  >>>
C
   60 IF (BOUND(I,1).LT.-1.0D14) GOTO 100
      A=BASIS(I)-PIVLEN*DA-BOUND(I,1)
      IF (A.GT.0.0D0) GOTO 100
      IF (A.LT.0.0D0) GOTO 70
      STOP '****** TIE *******'
   70 IJ=I
      PIVLEN=(BASIS(I)-BOUND(I,1))/DA
  100 CONTINUE
C
C<<<  UNBOUNDED RAY IF IJ = 9999 AND IDXIN > 0  >>>
C
      IF (IJ.LT.9999.OR.IDXIN.LT.0) GOTO 200
      IF (IDXIN.EQ.0)
     &  THEN
          STOP '** ERROR IN SUBROUTINE PIVOT, IDXIN=0 **'
        ELSE
          CTRL=2
          RETURN
      END IF
  200 CONTINUE
      IF (PIVSGN.GT.0)
     &  THEN
          GOTO 210
        ELSE
          PIVLEN=-PIVLEN
      END IF
  210 DO 220 I=1,MM
      BASIS(I)=BASIS(I)-PIVLEN*BAF(I)
  220 CONTINUE
      IF (IJ.LT.9999)
     &  THEN
          GOTO 300
        ELSE
C
C<<<  IDXOUT = IDXIN IF IJ = 9999  >>>
C
          IDXOUT=IDXIN
          VALOUT=VALIN+PIVLEN
          RETURN
      END IF
C
C<<<  PIVOT OPERATION  >>>
C
  300 CONTINUE
      DO 310 J=1,MM
  310 BINV(IJ,J)=BINV(IJ,J)/BAF(IJ)
      DO 330 I=1,MM
      IF (I.EQ.IJ)
     &  THEN
          GOTO 330
        ELSE
          DO 320 J=1,MM
  320     BINV(I,J)=BINV(I,J)-BAF(I)*BINV(IJ,J)
      END IF
  330 CONTINUE
```

```
      IDXOUT=INDEX(IJ)
      VALOUT=BASIS(IJ)
      INDEX(IJ)=IDXIN
      BASIS(IJ)=VALIN+PIVLEN
      BOUND(IJ,1)=LB
      BOUND(IJ,2)=UB
      RETURN
      END
C
C
C
C*************************************************************************
C     SUBROUTINE SOLUT
C*************************************************************************
      SUBROUTINE SOLUT(IDXIN,CGRID,V)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      INTEGER DIM
      PARAMETER (DIM=53)
      INTEGER IDXIN
      INTEGER INDEX(DIM),MM,NN
      INTEGER DM,NORM,T(DIM),PERM(DIM),S(DIM),FVECT(DIM)
      INTEGER VMT(DIM,DIM)
      DOUBLE PRECISION V(DIM),SUM
      DOUBLE PRECISION CGRID,AMT(DIM,DIM),F(DIM),GAMMA,X(DIM),XO(DIM)
      DOUBLE PRECISION AF(DIM),BAF(DIM),BINV(DIM,DIM),BOUND(DIM,2)
      DOUBLE PRECISION BASIS(DIM)
      COMMON /GLOBAL/F,AMT,MM,NN,X,XO,EPS
      COMMON /CPDM1/GAMMA
      COMMON /CPDM2/DM,NORM,T,PERM,S,FVECT,VMT
      COMMON /CPIV1/AF,BAF,BINV,BOUND,BASIS
      COMMON /CPIV2/INDEX
C
      IF (IDXIN.LE.O) STOP '** ERROR IN SUBROUTINE SOLUT, IDXIN < O **'
C
C<<<  COMPUTATION OF AN APPROXIMATE SOLUTION  >>>
C
      CALL ALLVTX
      SUM=1.0DO
      DO 10 I=1,MM
      IF (INDEX(I).GT.O) SUM=SUM-BAF(I)
   10 CONTINUE
      DO 20 I=1,NN
   20 V(I)=DFLOAT(VMT(I,IDXIN))/SUM
      DO 40 J=1,MM
      K=INDEX(J)
      IF (K.GT.O)
     &  THEN
          DO 30 I=1,NN
          V(I)=V(I)-DFLOAT(VMT(I,K))*BAF(J)/SUM
   30     CONTINUE
          GOTO 40
        ELSE
      END IF
   40 CONTINUE
      DO 50 I=1,NN
      X(I)=XO(I)+CGRID*V(I)
   50 CONTINUE
      RETURN
      END
C
```

```
C
C
C*************************************************************************
C     SUBROUTINE PARAMT
C*************************************************************************
      SUBROUTINE PARAMT(PROB,NN,GAMMA,IGRID,FGRID,XO,BC,BP,BX,ACC,
     &                  EPS,FSTEP,RTMAX,RTMIN,QNEWT)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      INTEGER DIM
      PARAMETER (DIM=53)
      INTEGER BC,BP,NN,PROB,QNEWT
      DOUBLE PRECISION IGRID,XO(DIM)
C
      WRITE(6,50)
      WRITE(6,100) 'PROBLEM NUMBER  ( 0  TO STOP )'
        READ(5,*) PROB
        IF (PROB.LE.0) STOP
      WRITE(6,100) 'DIMENSION OF PROBLEM'
        READ(5,*) NN
        IF (NN.LE.0)   STOP
      WRITE(6,100) 'INITIAL POINT ( 0 FOR DEFAULT, 1 TO INPUT )'
        READ(5,*) IP
        IF (IP.LE.0)
     &    THEN
            DO 10 I=1,NN
            XO(I)=0.0D0
 10         CONTINUE
          ELSE
            WRITE(6,100) 'COORDINATES OF INITIAL POINT'
            READ(5,*) (XO(I),I=1,NN)
        END IF
      WRITE(6,100) 'INITIAL GRID SIZE'
        READ(5,*) IGRID
      WRITE(6,100) 'DESIRED ACCURACY'
        READ(5,*) ACC
      WRITE(6,100) 'LOWER BOUND OF STEP SIZE'
        READ(5,*) FSTEP
      WRITE(6,100) 'UPPER BOUND OF ABS(X(I))'
        READ(5,*) BX
      WRITE(6,100) 'QUASI-NEWTON PROCEDURE ( 1 TO INSERT, 0 OTHERWISE )'
        READ(5,*) QNEWT
 50   FORMAT(/)
100   FORMAT(1H ,A)
      GAMMA=0.5D0/DFLOAT(NN)
      FGRID=1.0D-7
      BC=100
      BP=400*NN
      EPS=1.0D-10
      RTMAX=0.8D0
      RTMIN=0.4D0
      RETURN
      END
C
C
C
C*************************************************************************
C     SUBROUTINE INIT
C*************************************************************************
      SUBROUTINE INIT(MM,NN,CYCLE,TP,TF,CGRID,IGRID,AMT,XO,X,IR,CTRL,
     &                NFNEW,TFNEW,RESET)
```

```
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      INTEGER DIM
      PARAMETER (DIM=53)
      INTEGER CYCLE,TP,TF,IR(25),CTRL,NFNEW,TFNEW,RESET
      DOUBLE PRECISION IGRID,AMT(DIM,DIM),XO(DIM),X(DIM)
C
      CTRL=1
      RESET=0
      MM=NN+1
      CYCLE=0
      TP=0
      TF=0
      TFNEW=0
      NFNEW=0
      CGRID=IGRID
      CALL IDENT(AMT,NN)
      DO 10 I=1,NN
      X(I)=XO(I)
   10 CONTINUE
      DO 20 K=1,25
   20 IR(K)=0
      DO 30 K=2,25
      IF (NN.LE.5*(K-1))
     &  THEN
          IR(K)=NN
          GOTO 100
        ELSE
          IR(K)=5*(K-1)
      END IF
   30 CONTINUE
  100 CONTINUE
      RETURN
      END
C
C
C
C*************************************************************************
C     SUBROUTINE IDENT
C*************************************************************************
      SUBROUTINE IDENT(MAT,N)
      INTEGER DIM
      PARAMETER (DIM=53)
      DOUBLE PRECISION MAT(DIM,DIM)
C
      DO 10 I=1,N
      DO 20 J=1,N
      MAT(I,J)=0.0D0
   20 CONTINUE
      MAT(I,I)=1.0D0
   10 CONTINUE
      RETURN
      END
C
C
C
C*************************************************************************
C     SUBROUTINE NEWTON
C*************************************************************************
      SUBROUTINE NEWTON(ZO,FO,NO,SMIN,PROB,NN,CGRID,ACC,
     &                  BINV,V,VMT,INDEX,CTRL)
```

```
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      INTEGER DIM,P,CTRL,TF,TP,TFNEW,PROB
      PARAMETER (DIM=53)
      DIMENSION H(DIM,DIM)
      DIMENSION ZO(DIM),FO(DIM)
      DIMENSION Z1(DIM),F1(DIM),S(DIM),BINV(DIM,DIM),V(DIM)
      INTEGER VMT(DIM,DIM),INDEX(DIM)
      DOUBLE PRECISION N1,NO
      COMMON /STATIS/ NF,TF,NP,TP,NFNEW,TFNEW
C
      CTRL=1
      ALPH=1.0DO
      SMIN=-1.0DO
      FNORM=NO
C
C<<<  APPROXIMATE JACOBIAN INVERSE  >>>
C
      CALL JINV(NN,BINV,V,VMT,INDEX,CGRID,H)
      P=1
  200 CONTINUE
C
C<<<  QUASI-NEWTON STEP  >>>
C
      DO 300  I=1,NN
      S(I)=0.0DO
      DO 400 J=1,NN
  400 S(I)=S(I)-H(I,J)*FO(J)
      Z1(I)=ZO(I)+S(I)
  300 CONTINUE
      STEP=ENORM(NN,S)
      CALL FUNCT(PROB,NN,Z1,F1)
      NFNEW=NFNEW+1
      TFNEW=TFNEW+1
      N1=ENORM(NN,F1)
  600 IF (N1.LE.ACC)
     &  THEN
          CTRL=2
          DO 700 I=1,NN
          ZO(I)=Z1(I)
  700     FO(I)=F1(I)
          NO=N1
          RETURN
        ELSE
          CTRL=1
      END IF
  710 CONTINUE
      BET=BETA(NO,N1)
      IF (N1.LT.ALPH*FNORM.AND.STEP.LE.BET*CGRID)
     &  THEN
C
C<<<  UPDATE OF ALPH AND APPROXIMATE JACOBIAN INVERSE  >>>
C
          ALPH=ALPHA(P,NN)
          CALL BROYD(NN,H,FO,F1,S)
          DO 800 I=1,NN
          ZO(I)=Z1(I)
          FO(I)=F1(I)
  800     CONTINUE
          NO=N1
          P=P+1
```

```
            GOTO 200
          ELSE
            IF (NO.LE.N1) RETURN
            DO 820 I=1,NN
            ZO(I)=Z1(I)
  820       FO(I)=F1(I)
            NO=N1
            SMIN=STEP
            RETURN
        END IF
        END
C
C
C
C*************************************************************************
C       SUBROUTINE BROYD
C*************************************************************************
        SUBROUTINE BROYD(NN,H,ZFO,ZF1,S)
        IMPLICIT DOUBLE PRECISION (A-H,O-Z)
        INTEGER DIM
        PARAMETER (DIM=53)
        DIMENSION H(DIM,DIM),ZFO(DIM),ZF1(DIM),S(DIM),SH(DIM),H1(DIM,DIM)
        DIMENSION H2(DIM,DIM),H3(DIM,DIM),H4(DIM,DIM)
C
        DO 110 J=1,NN
          SH(J)=0.0DO
          DO 120 I=1,NN
  120     SH(J)=SH(J)+S(I)*H(I,J)
  110   CONTINUE
        DO 130 I=1,NN
        DO 130 J=1,NN
  130   H1(I,J)=S(I)*SH(J)
        DO 150 I=1,NN
          SUM=0.0DO
          DO 160 K=1,NN
  160     SUM=SUM+H(I,K)*ZF1(K)
          DO 170 J=1,NN
  170     H2(I,J)=SUM*SH(J)
  150   CONTINUE
        DO 180 I=1,NN
          SUM=0.0DO
          DO 190 K=1,NN
  190     SUM=SUM+H(I,K)*ZFO(K)
          DO 200 J=1,NN
  200     H3(I,J)=SUM*SH(J)
  180   CONTINUE
        DO 210 I=1,NN
        DO 210 J=1,NN
  210   H4(I,J)=H1(I,J)-H2(I,J)+H3(I,J)
        H5=0.0DO
        H6=0.0DO
        DO 220 I=1,NN
          H5=H5+SH(I)*ZF1(I)
          H6=H6+SH(I)*ZFO(I)
  220   CONTINUE
        H7=H5-H6
        DO 230 I=1,NN
        DO 230 J=1,NN
  230   H(I,J)=H(I,J)+H4(I,J)/H7
        RETURN
```

```
      END
C
C
C
C*************************************************************************
C       FUNCTION ALPHA
C*************************************************************************
       FUNCTION ALPHA(P,NN)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      INTEGER P
C
      A=0.8D0
      B=2.0D0
      RN=DFLOAT(NN)
      ALPHA=A**((DFLOAT(P)/RN)**B)
      RETURN
      END
C
C
C
C*************************************************************************
C     FUNCTION BETA
C*************************************************************************
      FUNCTION BETA(NO,N1)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      DOUBLE PRECISION NO,N1
C
      T=NO/N1+1.0D0
      BETA=DLOG(T)/DLOG(2.0D0)
      RETURN
      END
C
C
C
C*************************************************************************
C       SUBROUTINE JINV
C*************************************************************************
      SUBROUTINE JINV(NN,BINV,V,VMT,INDEX,CGRID,H)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      INTEGER DIM
      PARAMETER (DIM=53)
      INTEGER VMT(DIM,DIM),INDEX(DIM),PVMT(DIM,DIM)
      DIMENSION BINV(DIM,DIM),V(DIM)
      DIMENSION H(DIM,DIM),B(DIM,DIM)
C
      I=0
      DO 100 K=1,NN+1
      IF (INDEX(K).LE.0) GOTO 100
      I=I+1
      DO 110 J=1,NN
      B(I,J)=BINV(K,J)
  110 CONTINUE
      DO 120 L=1,NN
      PVMT(L,I)=VMT(L,INDEX(K))
  120 CONTINUE
  100 CONTINUE
      DO 200 I=1,NN
      DO 200 J=1,NN
      SUM1=0.0D0
      SUM2=0.0D0
```

```
      DO 210 K=1,NN
      SUM1=SUM1+DFLOAT(PVMT(I,K))*B(K,J)*CGRID
      SUM2=SUM2+V(I)*B(K,J)*CGRID
  210 CONTINUE
      H(I,J)=SUM1-SUM2
  200 CONTINUE
      RETURN
      END
C
C
C
C*************************************************************************
C     SUBROUTINE PRINTP
C*************************************************************************
      SUBROUTINE PRINTP(ACC,GAMMA,IGRID,PROB,NN,FSTEP,QNEWT)
      DOUBLE PRECISION ACC,GAMMA,G,IGRID,FSTEP
      INTEGER PROB,NN,QNEWT
C
      G=GAMMA*DFLOAT(NN)
      WRITE(6,100) G,IGRID,ACC,FSTEP,QNEWT,PROB,NN
  100 FORMAT(/' (3**N-1)-METHOD WITH QUASI-NEWTON PROCEDURE'/
     &      9X,'GAMMA=',D9.2,' / N'/
     &      9X,'IGRID=',D9.2/
     &      9X,'A C C=',D9.2/
     &      9X,'FSTEP=',D9.2/
     &      9X,'QNEWT=',I3//
     &      1X,'PROB=',I3,'    N=',I4)
      RETURN
      END
C
C
C
C*************************************************************************
C     SUBROUTINE PRINTT
C*************************************************************************
      SUBROUTINE PRINTT
      WRITE(6,100)
  100 FORMAT(/1H ,1X,'CYCLE',4X,'GRID',8X,'#PVT',3X,'#FFXP',
     &       3X,'#FNEW',5X,'EN(FUNCT)',6X,'EN(DIFF)')
      RETURN
      END
C
C
C
C*************************************************************************
C     SUBROUTINE PRINTA
C*************************************************************************
      SUBROUTINE PRINTA(CYCLE,CGRID,NP,NF,NFNEW,FNORM,DNORM)
      DOUBLE PRECISION CGRID,FNORM,DNORM
      INTEGER CYCLE
C
      WRITE(6,100) CYCLE,CGRID,NP,NF,NFNEW,FNORM,DNORM
  100 FORMAT(1H ,I4,3X,D10.3,3I8,3X,D12.4,3X,D12.4)
      RETURN
      END
C
C
C
C*************************************************************************
C     SUBROUTINE PRINTR
```

```
C*****************************************************************************
      SUBROUTINE PRINTR(CYCLE,CGRID,TP,TF,TFNEW,FNORM,DNORM,
     &                  X,F,IR)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      INTEGER DIM
      PARAMETER (DIM=53)
      INTEGER CYCLE,TF,TP,IR(25)
      DIMENSION X(DIM),F(DIM)
C
      CALL PRINTA(CYCLE,CGRID,TP,TF,TFNEW,FNORM,DNORM)
      WRITE(6,200) ' APPROXIMATE SOLUTION'
      CALL PRINTX(X,IR)
      WRITE(6,200) ' FUNCTION VALUE'
      CALL PRINTX(F,IR)
  200 FORMAT(/A)
      RETURN
      END
C
C
C
C*****************************************************************************
C   SUBROUTINE PRINTX
C*****************************************************************************
      SUBROUTINE PRINTX(X,IR)
      INTEGER DIM
      PARAMETER (DIM=53)
      DOUBLE PRECISION X(DIM)
      INTEGER IR(25)
C
      DO 10 K=2,25
      IF (IR(K).EQ.0) RETURN
      WRITE(6,100) (X(I), I=IR(K-1)+1,IR(K))
   10 CONTINUE
  100 FORMAT(1H ,5D14.5)
      RETURN
      END
C
C
C
C*****************************************************************************
C     PRINTB
C*****************************************************************************
      SUBROUTINE PRINTB
C
      WRITE(6,100)
  100 FORMAT(1H ,'-----------------------------------------------',
     &           '-----------------------------------------------')
      RETURN
      END
C
C
C
C*****************************************************************************
C   SUBROUTINE FUNCT
C     PROBLEMS  1,2,3     :     WATSON'S PROBLEMS
C     PROBLEM   4         :     TODD'S SPIRAL PROBLEM
C     PROBLEM   5         :     DIXON'S OPTIMIZATION PROBLEM
C*****************************************************************************
      SUBROUTINE FUNCT(PROB,NN,X,F)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
```

```
      INTEGER DIM,PROB,NN
      PARAMETER (DIM=53,PI=3.14159265359)
      DIMENSION X(DIM),F(DIM)
C
      GOTO (1000,2000,3000,4000,5000)
     &     ,PROB
C
 1000 SUM=0
      DO 1010 I=1,NN
 1010 SUM=SUM+X(I)*X(I)*X(I)
      DO 1020 I=1,NN
 1020       F(I)=X(I)-(SUM+DFLOAT(I))/(DFLOAT(2*NN))
      RETURN
C
 2000 SUM=0.0D0
      DO 2010 I=1,NN
 2010 SUM=SUM+X(I)
      DO 2020 I=1,NN
      Y=DFLOAT(I)*SUM
      Y=COS(Y)
      F(I)=X(I)-EXP(Y)
 2020 CONTINUE
      RETURN
C
 3000 TSUM=1.0D0
      SUM=0.0D0
      DO 3010 I=1,NN
      SUM=SUM+X(I)
 3010 TSUM=TSUM*X(I)
      F(1)=TSUM-1.0D0
      DO 3020 I=2,NN
 3020 F(I)=SUM+X(I)-DFLOAT(NN+1)
      RETURN
C
 4000 CONTINUE
      RAD=5.0D0
      K=INT(NN/2)*2
      DO 4010 I=1,K-1,2
      Y1=X(I)
      Y2=X(I+1)
      YN=DSQRT(Y1**2+Y2**2)
      IF(YN.GE.1.0D0) GO TO 4020
      IF(YN.LE.1.0D0/2.0D0**RAD) GO TO 4030
      ETA=-DLOG(YN)/DLOG(2.0D0)
      ETA=ETA*PI
      F(I)=DCOS(ETA)*Y1-DSIN(ETA)*Y2
      F(I+1)=DSIN(ETA)*Y1+DCOS(ETA)*Y2
      GO TO 4010
 4020 F(I)=Y1
      F(I+1)=Y2
      GO TO 4010
 4030 F(I)=-Y1
      F(I+1)=-Y2
 4010 CONTINUE
      IF(K.EQ.NN) RETURN
      F(NN)=X(NN)
      RETURN
C
 5000 CONTINUE
      F(1)=-2.0D0*(1.0D0-X(1))+4.0D0*(X(1)*X(1)-X(2))
```

```
      DO 5010 I=2,NN-1
      F(I)=-2.0D0*(X(I-1)*X(I-1)-X(I))
     &     +4.0D0*X(I)*(X(I)*X(I)-X(I+1))
5010  CONTINUE
      F(NN)=-2.0D0*(X(NN-1)*X(NN-1)-X(NN))
     &      -2.0D0*(1.0D0-X(NN))
      RETURN
      END
```