ALGORITHMS FOR THE DESIGN OF SYSTEMS WITH TOLERANCE ERRORS

by

Alexander Voreadis

A Thesis submitted for the Degree

of Doctor of Philosophy

July 1981

Department of Electrical Engineering

Imperial College of Science and Technology,

University of London.

ALGORITHMS FOR THE DESIGN OF SYSTEMS WITH TOLERANCE ERRORS

by

Alexander Voreadis

ABSTRACT

In this thesis the problem of synthesizing a system from components whose values are known only to certain tolerances is considered.

In the pure tolerance case the design objective is the determination of a set of nominal system parameters so that all the specifications are met whatever the actual system parameters are, as long as they fall within a tolerance region centered on the nominal values. Quite commonly, very tight tolerances are necessary for a solution to the pure tolerance problem to exist. The introduction of post-manufacture tuning allows the adoption of higher parameter tolerances and may be desirable in order to reduce manufacturing costs. The tolerance-tuning problem is the determination of a set of nominal system parameters so that whatever the actual parameters are - within a tolerance region - the specifications can be met by tuning.

Conceptual and implementable algorithms for the solution of the pure tolerance and the tolerance-tuning problems are proposed. The algorithms solve the general non-convex problems and belong to the class of cut map algorithms of Eaves and Zangwill. Convergence is established, numerical examples are presented and comparisons with other methods in the literature are made.

To my Family

ACKNOWLEDGEMENTS

I would like to express my most sincere thanks to my supervisor, Professor D.Q. Mayne, for his invaluable help and advice during the course of the research that led to this thesis.

The encouragement and help from my fellow research students will be always remembered with gratitude.

I would also like to thank Mrs. Moriarty for typing this thesis.

A large part of the work that led to this thesis was financially supported by the Bodossaki Foundation, Athens, Greece.

C O N T E N T S

GENERAL NOTATION

1. R^n denotes the Euclidean space of ordered n-tuples of real numbers. Supercripts are used to denote the components of a vector in R^n.

2. f denotes a function. If A is the domain of f and B its range, we then write $f: A \to B$.

3. Given a function $f: R^n \to R$ we denote by $\nabla f(x)$ its gradient at x.

4. We denote by $B_2(x,\varepsilon)$ the following set in R^n: $\{x' \in R^n \mid \|x-x'\|_2 < \varepsilon\}$, (open ball with the Euclidean norm).

5. The max norm in R^n is defined as : $\|x\|_\infty \triangleq \max_i \{|x^i| \mid i=1,\ldots,n\}$.

6. We denote by $B_\infty(x,\varepsilon)$ the following set in R^n: $\{x' \in R^n \mid \|x'-x\|_\infty < \varepsilon\}$, (open ball with the max. norm).

7. $\bar{F}$ denotes the closure of the set F.

8. F^o denotes the interior of the set F.

9. F^c denotes the complement of the set F.

10. Z^+ denotes the set of positive integers including zero, i.e. $Z^+ = \{0,1,2,3,\ldots.\}$.

11. $A \cup B$ denotes the union of the sets A and B.

12. $A \cap B$ denotes the intersection of the sets A and B.

13. $A \subseteq B$ denotes that A is a subset of B.

14. $A \setminus B$ denotes the set that contains all the elements of the set A not belonging to the set B.

15. 2^{R^n} denotes the set of all subsets of R^n.

CHAPTER I

I N T R O D U C T I O N

I.1 Computer-aided design

In recent years the role of the digital computer in solving engineering design problems has been invaluable. Design, as opposed to synthesis, is iterative in nature since the often imprecise objectives and constraints require constant interaction with a decision maker - the designer. However, synthesis techniques which solve precisely specified problems are useful tools for solving subproblems which may recur in the design process.

Many engineering problems can be formulated as constrained optimization problems or inequality solving problems, in which the inequality constraints correspond to the design specifications. The solution of such precisely formulated problems is only one stage of the design process. The designer has to interact at each stage modifying the constraints and thus trading off one desirable quality for another. Hence constrained optimization or inequality solving algorithms play an important role in computer-aided design.

Many of the design specifications can be transcribed into standard inequality constraints, so that standard algorithms may be employed. However, surprisingly often, [13] , design specification lead to functional (or infinite dimensional) constraints which cannot be treated with standard methods. The pure tolerance and the tolerance-tuning problems that are examined in this thesis belong to this class of problems. Hence the development of algorithms suitable for infinitely constrained problems was stimulated and infinite dimensional analogues of finite dimensional algorithms have been derived. These algorithms are conceptual

since they require the solution of infinite dimensional linear programmes.

It seems that the only implementable algorithms (requiring a finite number of operations at each iteration) suitable for infinitely constrained problems are the feasible directions algorithms of [2] and [3] and the outer approximations algorithm of [4] . A summary of these new algorithms and a description of their use can be found in [1] .

I.2 Statement of the problems.

Consider the problem of synthesizing a system from components whose values are known only to certain tolerances. Uncertainties on the values of the components arise from fluctuations inherent in the production processes or identification errors. Such deviations from the nominal values may cause failure to meet imposed specifications and hence may lead to low production yields. It is thus necessary to take into account these possible parameter deviations in the design stage. The design objective is to choose a set of nominal parameter values so that certain specifications will be met whatever the actual values are, as long as they fall in a certain tolerance region. Such problems frequently occur in circuit design where they are known as centering problems [6] , [7] , [8] , [14] , but also arise in control system design since the properties of transducers, actuators etc., are known only to a certain degree of accuracy.

Let $x \in R^n$ denote the nominal value of the parameter vector and t the manufacturing error so that the actual value of the parameter vector is x+t; t lies in a known tolerance region T, a subset of R^n. Suppose that the system must satisfy certain specifications no matter what the tolerance error is. This problem can be expressed as :

P_T : Find a point (nominal design) x such that :

$$f^j(x+t) \leq 0 \ , \ j=1,\ldots,m \ , \text{ for all } t \in T. \tag{1}$$

The functions $f^j : R^n \to R$ define the design constraints. The problem can be normalized so that T can be defined by :

$$T \triangleq \{t \in R^n \mid |t^i| \leq 1 \ , \ i = 1,\ldots,n\} \ . \tag{2}$$

Note that the constraint (1) is <u>infinite dimensional</u>. An equivalent form for this design problem is :

P_T : Find a point a satisfying :

$$\theta_T(x) \leq 0 \tag{3}$$

where $\theta_T : R^n \to R$ is defined by:

$$\theta_T(x) = \max\{\psi(x+t) \mid t \in T\} \tag{4}$$

and $\psi : R^n \to R$ is defined by:

$$\psi(x) \triangleq \max\{f^j(x) \mid j=1,\ldots,m\}. \tag{5}$$

Let G denote the feasible set for the problem P_T. The determination of $\theta_T(x)$, given x, is a <u>global</u> optimization problem and is sometimes referred to as the worst case problem (WCP) in the literature [9] , [10]. Bandler, [7], [8], refers to a design satisfying (3) as a worst case design. Let V denote the set of vertices of T and F the set of feasible nominal designs i.e.:

$$V \triangleq \{t \in R^n \mid |t^i| = 1, \ i=1,\ldots,n\} \tag{6}$$

and

$$F \triangleq \{x \in R^n \mid \psi(x) \leq 0\} . \qquad (7)$$

Bandler [6] has shown that if F is one-dimensionally convex the (infinite dimensional) problem P_T is equivalent to the (finite dimensional) feasibility problem P_V defined by:

P_V : Find a point x such that:

$\psi(x+t) \leq 0$ for all $t \in V$

(i.e. satisfying $\theta_V(x) \triangleq \max\{\psi(x+t) \mid t \in V\} \leq 0$).

In principle a standard algorithm can be employed for P_V although the cardinality of V can be very large. In [8] Bandler considers the problem of minimizing a cost function subject to $\theta_T(x) \leq 0$ for the case when F is one-dimensionally convex (so that $\theta_T(x) \leq 0$ can be replaced by $\theta_V(x) \leq 0$). A procedure for replacing V by a relatively small subset of V is presented. This (heuristic) procedure determines which vertex is likely to violate which constraint by sensitivity (gradient) analysis, so that only the "worst case" vertices are utilized. In [9] Schjaer-Jacobson and Madsen present an interval arithmetic algorithm for solving (WCP) in the general case and a more efficient algorithm for the one-dimensionally convex case. They also present algorithms for the fixed (FTP) and variable (VTP) tolerance problems ($\min\{\theta_T(x) \mid x \in R^n\}$ and $\max\{w \mid \theta_{\tilde{T}}(x) \leq 0\}$, $\tilde{T} \triangleq \{t \mid |t^i| \leq w\}$ respectively). These algorithms require at each iteration solution of the WCP and seem to assume incorrectly the differentiability of the function $\phi^j(x) \triangleq \max\{f^j(x+t) \mid t \in T\}$. More sophisticated algorithms for the fixed and variable tolerance problems (FTP and VTP) are presented by Brayton et al, [10] . One-dimensional convexity is assumed. The non-differentiability of the functions $\phi^j(x)$, $j=1,\ldots,m$ is coped with by "function splitting"; if $f^j(x+t)$ achieves its maximum at t_1 and t_2 (i.e. $\phi^j(x) = f^j(x+t_1) = f^j(x+t_2)$) both

$\nabla f^j(x+t_1)$ and $\nabla f^j(x+t_2)$ are employed in the optimization algorithms. The function $\phi^j(x)$ is non-differentiable at x but possesses a generalized gradient [18] which is the convex hull of $\nabla f^j(x+t_1)$ and $\nabla f^j(x+t_2)$. A vertex list updated at each iteration defines the function and gradient information applied to a quadratic program to determine a search direction; this list is a subset of the set $\{f^j(x+v) \mid j=1,\ldots,m,\ v\in V\}$. Because of the very complex nature of the rules of updating the vertex list, the algorithms in [10] are not explicitly stated but are only vaguely described.

All these algorithms are, in the main, restricted to the case where F is one-dimensionally convex. They possess heuristic features in order to improve efficiency and, hence, do not have guaranteed convergence. It seems that the only algorithm suitable for the infinite dimensional non-convex problem P_T which has guaranteed convergence properties is the outer approximation algorithm of [4] (the algorithms of [2] and [3] are restricted to the case in which T is a subset of R). This replaces the infinite dimensional problem P_T by an infinite sequence of conventional (finite dimensional) feasibility problems $\{P_{T_i}\}$ where T_i is a finite subset of T and P_{T_i} is defined by :

P_{T_i}: Determine an x such that :

$$\psi(x+t) \leq 0 \quad \text{for all } t\in T_i .$$

Since T_i is a subset of T, the feasible set for the problem P_{T_i} is an outer approximation to the feasible set G for the problem P_T. At iteration i the algorithm solves P_{T_i}, a conventional feasibility problem, using a standard algorithm yielding x_i. It then solves (approximately) WCP $(\max\{\psi(x_i+t) \mid t\in T\})$ yielding t_i; T_{i+1} is then formed by adding t_i to T_i and discarding other points which are judged to be

irrelevant. Precise rules for increasing the accuracy of the solution of WCP and for updating T_i are given in [4], which ensure convergence for non-convex problems. The set T_i may be regarded as an extension to non-convex problems of the "vertex list" concept of Brayton et al [10]; of course the elements of T_i are not necessarily vertices.

In the cut map algorithms proposed in this thesis the complement of the feasible set G for the problem P_T is approximated at each iteration by the union of a finite number of very simply described regions. Typically at iteration i, G^c (the complement of G) is approximated by $W_i = \cup \{B(x_j,\delta_j) \mid j<i\}$ (or a subset of this set) where $B(x_j,\delta_j)$ denotes an open ball with centre x_j and radius $\delta_j > 0$ such that $G \cap B(x_j,\delta_j) = \phi$. Clearly $G \subset W_i^c$ so that W_i^c is an outer approximation to G of a particularly simple kind. The class of cut map algorithms of Eaves and Zangwill are discussed in more detail in section 4.

All the methods described so far are deterministic in nature. There also exist some methods for the tolerance problem which utilize a statistical approach. These are now briefly described. In [11] the tolerance problem is tackled by approximating the assumed convex feasible set F by a polyhedron. Then a maximal hyperellipsoid is inscribed in the polyhedron to provide a design center and a set of parameter tolerances. All the computations required for the above operations are performed by linear programmes. In [12] this method is extended to the case of arbitrary distributions (in [11] the case of joint Gaussian distributions is only considered). In [13] the tolerance problem is formulated as a minimization problem of the variance of some performance index which relates to the design specifications. Finally in [15] a deterministic algorithm for yield maximization is proposed.

At each iteration the search direction is computed to be the direction of the line joining the centres of gravity of the feasible and non-feasible parts of the tolerance region (estimated by Monte Carlo analysis). The step length rule is heuristic.

Quite commonly very tight tolerances are required for a solution to the tolerance problem to exist. Tight tolerances make manufacture costly or even impossible. To overcome this difficulty post-manufacture tuning or trimming of certain parameters is usually introduced [16] , [17] . The problem now becomes to determine a set of nominal parameter values so that whatever the actual values are - within a tolerance region - the specifications can be met by tuning.

Suppose that the first ℓ parameters can be tuned or trimmed and let the map $r: R \overset{\ell}{\rightarrow} R^n$ be defined by:

$$r^i(q)=\begin{cases} q^i & \text{if } i \leq \ell \\ 0 & \text{if } \ell < i \leq n \end{cases} \quad , \; q \in R^{\ell} . \tag{8}$$

Also let the tuning region Q be defined as :

$$Q \overset{\Delta}{=} \{ \; q \in R^{\ell} \; \big| \; -\alpha \leq q \leq \beta \; \} \tag{9}$$

where $\alpha \geq 0$, $\beta \geq 0$, $\alpha , \beta \in R^{\ell}$. The tolerance-tuning problem can be expressed as:

$P_{T,Q}$: Find a point x such that for each $t \in T$ there exists some $q \in Q$ satisfying:

$$f^j(x+t+r(q)) \leq 0, \quad j=1,\ldots,m.$$

An alternative form for the problem $P_{T,Q}$ is the following:

$P_{T,Q}$: Find a point x such that:

$$\theta_{T,Q}(x) \stackrel{\Delta}{=} \max_{t\in T} \min_{q\in Q} \psi(x+t+r(q)) \leq 0. \qquad (10)$$

It is clear that the determination of $\theta_{T,Q}(x)$ is considerably more difficult than (the already prohibitively difficult) WCP of determining θ_T. As shown in [18] $\theta_{T,Q}$ is Lipschitz continuous but <u>not differentiable</u>; in fact it may even fail to have directional derivatives. Polak and Sangiovanni-Vincetelli,[18],formulate the general engineering problem when tuning is permitted and propose an algorithm for its solution. This algorithm consists of two parts : an outer approximation algorithm which replaces T by an infinite sequence$\{T_i\}$ of discrete subsets of T and an inner subalgorithm which solves the resulting simpler subproblems. Because the inner subalgorithm utilizes non-differentiable optimization ideas, it has two major disadvantages. The first is that it is computationally expensive and the second that it is applicable to the case where there is <u>only one</u> constraint function. Recently, Polak has shown that by employing certain simple transformations these inner subproblems are equivalent to ordinary <u>differentiable</u> optimization problems so that they can be solved by standard algorithms. Hence in [19] an outer approximations algorithm that does not possess the computational disadvantages caused by the need for a subalgorithm suitable for non-differentiable problems is presented.

Bandler [8], tackles the tolerance-tuning problem by distinguishing between effectively toleranced and effectively tuned parameters. A (finite dimensional) reduced problem can be obtained in this way, which under assumptions of one-dimensional convexity can be

shown to be equivalent to the original problem. For non-convex problems the reduced problem is more restrictive (i.e. it may fail to have a solution, although a solution to the original problem exists). Hence Bandler's approach is strictly applicable only when one-dimensional convexity is present. To further simplify the reduced problem, heuristic procedures are employed in [8]. Note that it is precisely in cases in which the set F is strongly non-convex (e.g. when it possesses "black holes"), that one hopes to obtain large increases in the component tolerances by the introduction of tuning.

In the second part of this thesis a cut map algorithm for the non-convex problem $P_{T,Q}$ is proposed, which has the same general features as the cut map algorithm proposed in the first part for the pure tolerance problem P_T.

I.3 Outline and contributions of the thesis.

In Chapter II the complete theoretical development of specialized cut map algorithms for the problem P_T is presented. Conceptual and implementable algorithms are proposed and convergence is established. In Chapter III the implementation details of these algorithms are discussed. Numerical examples are presented and conclusions about the performance of the algorithms are drawn.

In Chapter IV the ideas of Chapter II are extended to the case when tuning is also permitted. Hence, specialized conceptual and implementable algorithms for the problem $P_{T,Q}$ are proposed and convergence is established. Chapter V is concerned with the implementation details of the algorithms of Chapter IV. Numerical examples are presented and the properties of the algorithms are discussed. Finally, Chapter VI contains concluding remarks and comparisons with the rest of the methods

in the literature for the problems P_T and $P_{T,Q}$.

Each of the specialized algorithms presented in this thesis is believed to be completely original. More specifically it seems that algorithms for the problems P_T and $P_{T,Q}$ utilizing the concepts of cut maps and separators, [20], have not been proposed before. In Chapters III and V the procedures that (approximately) solve the specialized global optimization problems for the computation of the separator estimates were especially developed for the algorithms proposed in Chapter II and IV respectively.

I.4 Cut map algorithms.

The class of cut map algorithms has been examined in a general way by Eaves and Zangwill [20] and by Hogan [21] . The key concepts employed are those of separators and cut maps. A function $\delta : G^c \to R$ where G^c is the complement of a closed set G in R^n is a separator, [20], if :

$$\text{(i)} \quad \delta(x) > 0 \quad \text{for all } x \in G^c \tag{11}$$

$$\text{(ii)} \quad x_i \to x^* \quad \text{and } \delta(x_i) \to 0 \quad \text{as} \quad i \to \infty \quad \text{imply that } x^* \in G. \tag{12}$$

It is shown in [20] that any positive lower semi-continuous function mapping G^c into R, is a separator. The significance of a separator δ is that if an infinite sequence $\{x_i\}$ satisfies :

$$x_{i+1} \notin B(x_j, \delta(x_j)) \ , \ j=1,\ldots,i. \tag{13}$$

for i = 1,2,..., then any accumulation point x^* of the sequence $\{x_i\}$ lies in G. The reason for this is simple. Suppose there is a subsequence of $\{x_i\}$ indexed by J such that $x_i \overset{J}{\to} x^*$ as $i \to \infty$. Hence the

distance between successive points in this subsequence converges to zero. This can only happen if $\delta(x_i) \to 0$ as $i \to \infty$, $i \in J$. It follows from the definition of a separator that $x^* \in G$. The balls in (13) can be open or closed and can be defined using any norm. We now define cut maps as follows :

Let L be the class of closed sets defined by :

$$L \triangleq \{Z \subset R^n \mid Z \supset G, \quad Z \text{ closed}\} . \tag{14}$$

A point to set function $w: G^c \to 2^L$ (the set of all subsets of L) is a <u>cut map</u> if there exists a separator $\delta: G^c \to R$ such that :

$$Z \cap B(x,\delta(x)) = \phi \tag{15}$$

for each $x \in G^c$ and each $Z \in w(x)$. A set $Z \in L$ is a <u>cut</u> if $Z \in w(x)$ for some $x \in G^c$. It is shown in [20] that G is the intersection of all cuts in L. Note that if for all $x \in G^c$, $0 < \delta(x) \leq d(x,G) \triangleq \min \{||y-x|| \mid y \in G\}$, then $x \to B(x,\delta(x))^c$ is a simple example of a cut map. The reason for this is that $G \not\subset B(x,\delta(x))^c$ for all $x \in G^c$. We can now state the following model cut map algorithm.

<u>Algorithm 1</u>

<u>Step 0</u> : Set $i=0$, $W_0 = \phi$.

<u>Step 1</u> : Compute any $x_i \in W_i^c$.

If $x_i \in G$ stop; else proceed to step 2.

<u>Step 2</u> : Set $W_{i+1} = W_i \cup Z_i^c$

where $Z_i \in w(x)$ and w is a cut map.

Set $i = i+1$ and go to step 1.

□

Theorem 1, [20]

(i) If the algorithm stops at x_i, then $x_i \in G$.

(ii) If x^* is an accumulation point of a sequence $\{x_i\}$ generated by algorithm 1, then $x^* \in G$.

□

Hence a cut map algorithm generates approximations to the set G by forming intersections of finite numbers of cuts. At each iteration a point in the current approximation of the set G in computed and then the approximation is updated by the introduction of a new cut involving the latest point. The importance of utilizing cut maps is that outer approximations to the set G are only generated.

The above algorithm is due to Eaves and Zangwill and contains the essential features of cut map algorithms. The other algorithms presented in their papers incorporate schemes for dropping cuts (in algorithm 1 a new cut is introduced at each iteration). Since these schemes are not appropriate in our case, different cut dropping schemes will be employed.

CHAPTER II

CUT MAP ALGORITHMS FOR THE TOLERANCE PROBLEM

II.1 Introduction

In this Chapter the complete theoretical development of conceptual and implementable algorithms for the tolerance problem is presented. The algorithms belong to the class of cut map algorithms of Eaves and Zangwill [20] and the concepts of separator functions and cut maps are utilized for establishing their convergence properties.

Suppose that $x \in R^n$ is the vector of nominal values of the design parameters of a certain system. The object of a possible design procedure is to choose x so that all the specifications will be met, whatever the actual parameter values are, as long as they fall in a certain given tolerance region. Hence one way of formulating the tolerance problem in a normalized form is the following :

P_T : Find a point in the set G defined by :

$$G \triangleq \{x \in R^n \mid f^j(x+t) \leq 0,\ j=1,\ldots,m,\ \text{for all } t \in T\} \tag{1}$$

where

$$T \triangleq \{t \in R^n \mid |t^i| \leq 1,\ i = 1,\ldots, n\} . \tag{2}$$

The functions $f^j: R^n \to R$, $j=1,\ldots,m$ specify the inequality constraints that represent the specifications of the design. Also T is a compact subset of R^n (hypercube) that specifies the tolerance region. Hence for $i=1,\ldots,n$, the maximum possible deviation of the ith parameter from its nominal value is unity. The tolerance problem belongs to the

class of problems with infinite dimensional constraints (t can take infinitely many values) and therefore is computationally complex. It is clear that it is not even possible to test if a point belongs to the set G or not, since an infinite number of constraints has to be evaluated.

Let V denote the set of vertices of the set T so that :

$$V \triangleq \{ t \in R^n \mid |t^i| = 1, \quad i = 1,\ldots,n \} . \tag{3}$$

Note that the cardinality of V is 2^n, i.e. it increases very rapidly with the problem dimension. Let

$$F \triangleq \{ x \in R^n \mid f^j(x) \leq 0 , \; j=1,\ldots,m \} . \tag{4}$$

Definition

A set $A \subset R^n$ is said to be one-dimensionally convex if for any $x \in A$, any $y \in A$ such that $y = x + ae_j$ for some $a \in R$ and some $j \in \{0,1,2,\ldots,n\}$ all the points $z = x + \lambda(y-x)$, $\lambda \in [0,1]$ also belong to A.

Note that e_j above denotes the jth unit vector. Now consider the following (finite dimensional) problem P_V:

P_V : Find a point in the set G_V defined by :

$$G_V \triangleq \{ x \in R^n \mid f^j(x+t) \leq 0, \quad j=1,\ldots,m, \text{ for all } t \in V \} . \tag{5}$$

Theorem (Bandler [6]).

A solution to the reduced problem P_V is a solution to the original problem P_T and vice versa, if the set of feasible nominal designs F is one-dimensionally convex.

A two-dimensional example illustrating the above theorem is shown in Figure I. Its validity is based on the fact that "worst case" points of a certain design x, (i.e. the points in x+T that mostly violate the specifications) are always vertices, if the set F is one-dimensionally convex. One-dimensional convexity is the fundamental property required by most existing methods for tackling the tolerance problem, as discussed in Chapter I, since it allows the adoption of finite dimensional non-linear programming algorithms. As the number of vertices increases very rapidly with the problem dimension all these methods incorporate procedures for selecting only a small number of vertices at each iteration. Because of the heuristic nature of these procedures, none of these methods has established convergence properties. The only algorithm with established convergence properties which is suitable for general non-convex infinite dimensional problems seems to be the outer approximations algorithm of [4]. In this Chapter specialized cut map algorithms for the tolerance problem will be proposed that possess the following general features :

(i) They construct sequences of points that converge (if convergence occurs) to solutions of the non-convex problem P_T.

(ii) They have established convergence properties.

(iii) They are directly implementable, i.e. truncation rules are given for every infinite computation, so that each iteration requires a finite number of operations on a digital computer.

(iv) They are particularly suitable for interactive computer-aided design.

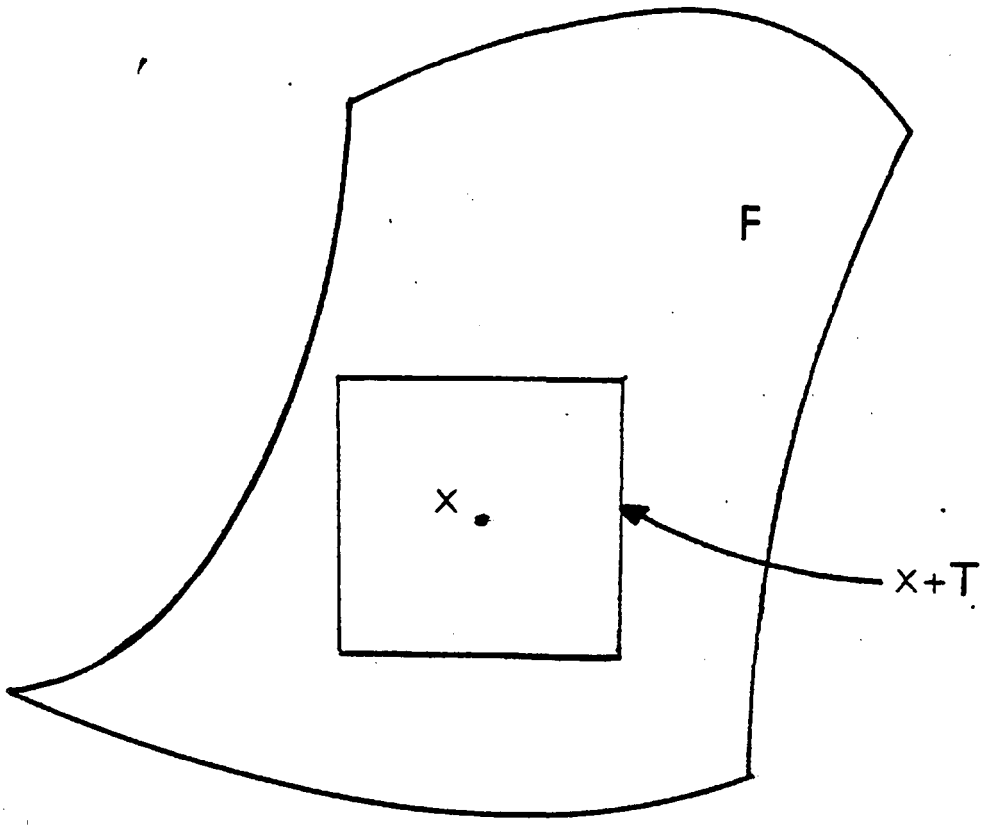


FIGURE I

F is one-dimensionally convex.

In section 2 assumptions are made and results are proven that lead to the definition of a separator function for the tolerance problem and to the statement of a conceptual algorithm. In section 3 implementable algorithms are proposed and discussed. Implementation details and numerical examples are presented in Chapter III.

II.2 A conceptual algorithm for the tolerance problem.

We firstly make some definitions and assumptions necessary for the analysis that follows. Let :

$$\psi(x) \triangleq \max \{ f^j(x) \mid j = 1,\ldots,m \} \tag{6}$$

$$U \triangleq \{ x \in R^n \mid \psi(x) \geq 0 \}. \tag{7}$$

Figure II illustrates the sets F,G and U. The following assumptions are made :

A1: The functions $f^j : R^n \to R$, $j=1,\ldots,m$ are continuous.

A2: The sets G and F are not empty and the set F is equal to the closure of its interior.

A3: F^o (the interior of F) satisfies :

$$F^o = \{ x \in R^n \mid \psi(x) < 0 \} .$$

All these assumptions are mild. A1 is a standard assumption. A2 excludes isolated points, whiskers etc. from the set F and together with A3 is required to prove results that lead to the definition of a cut map for the tolerance problem.

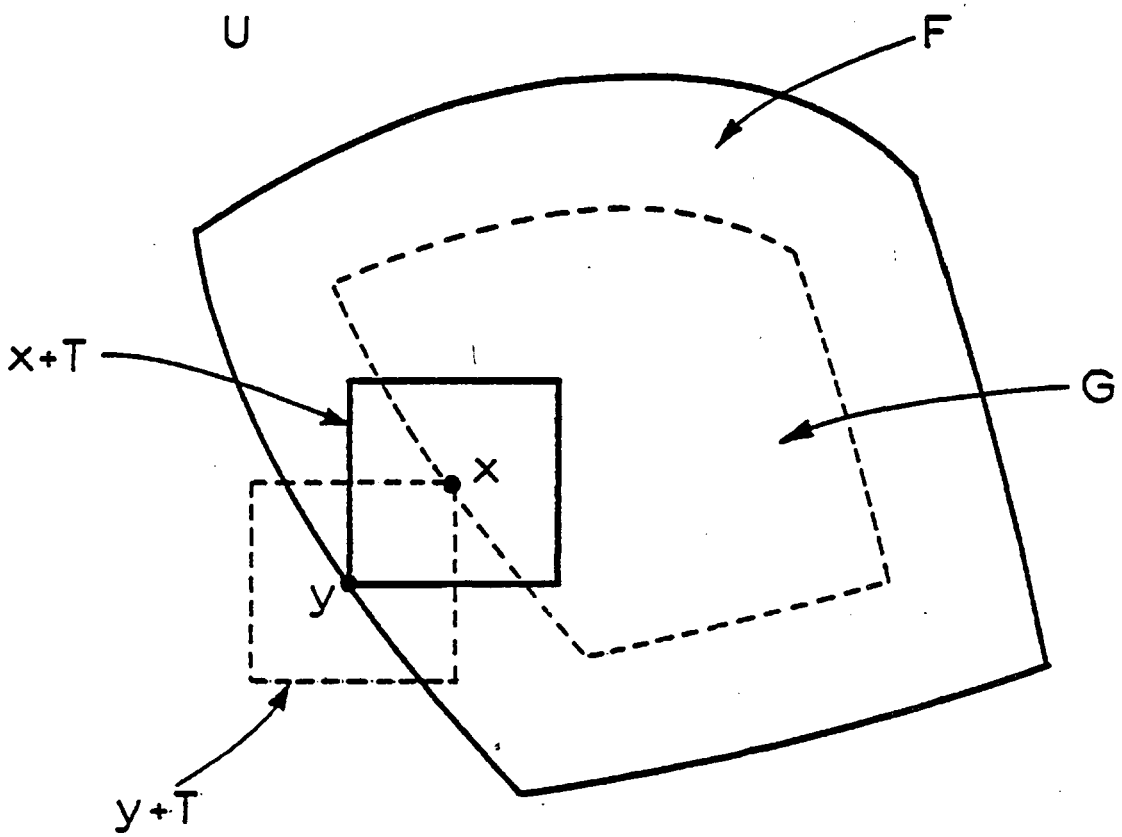


FIGURE II

The sets F, U and G.

Consider the function $\eta: R^n \to R$ defined by:

$$\eta(x) \triangleq \min \{ \| y-x \|_\infty \mid y \in U \} \tag{8}$$

Because ψ is continuous (i.e. the set U is closed), $\| y-x \|_\infty$ is bounded from below for fixed x and $\| y-x \|_\infty \to \infty$ as $\| y \|_\infty \to \infty$, the minimum exists. We are now in a position to define a separator for the tolerance problem. Let $\delta: R^n \to R$ be defined by :

$$\delta(x) \triangleq 1 - \eta(x). \tag{9}$$

Proposition 1

The function $\delta(x)$ is a separator for the problem P_T.

Proof

(i) $\delta(x) > 0$ for all $x \in G^c$.

Choose any $x \in G^c$. By the definition of G there exists some $y \in (x+T)$ such that $\psi(y) > 0$. Since x+T is equal to the closure of its interior and ψ is continuous, there exists some $y' \in (x+T)^o$ such that $\psi(y') > 0$. Let $\hat{y}$ be a minimizer associated with x, then :

$$\eta(x) = \| x-\hat{y} \|_\infty \leq \| x-y' \|_\infty < 1.$$

Hence $\eta(x) \in [0,1)$ for all $x \in G^c$ and the result follows from (9).

(ii) $\delta(x)$ is continuous.

We prove that $\eta(x)$ is a continuous function and use (9). Suppose that y is a minimizer associated with x (y' with x'), then we have:

$$\eta(x) = \| x-y \|_\infty \leq \| x-y' \|_\infty = \| x-x'+x'-y' \|_\infty \leq$$

$$\leq \| x-x' \|_\infty + \| x'-y' \|_\infty = \eta(x') + \| x-x' \|_\infty .$$

Similarly :

$$\eta(x') \leq \eta(x) + \| x-x' \|_\infty .$$

Combining the two results we have :

$$| \eta(x) - \eta(x') | \leq \| x-x' \|_\infty$$

which establishes continuity.

□

Note that $\delta(x) = 1$ for all $x \in U$. We next prove some results that follow directly from our assumptions and will lead to the definition of a cut map for the tolerance problem P_T.

Proposition 2

(i) The set U is equal to the closure of its interior (i.e. $U = \overline{(U^o)}$, where the overbar denotes closure).

(ii) The interior of U satisfies :

$$U^o = \{ x \in R^n \mid \psi(x) > 0 \} .$$

Proof

We firstly note that by the continuity of ψ:

$$F^c = \{x \mid \psi(x) > 0\} \subset U^o. \tag{10}$$

(i) We have that $\overline{(U^o)} \subset U$ since U is a closed set. Let

$$\Delta(F) \triangleq \{x \in R^n \mid \psi(x) = 0\}.$$

Choose any $x \in \Delta(F)$, then :

$$x \in \Delta(F) \Rightarrow \psi(x) = 0 \Rightarrow x \notin F^o$$

by assumption A3. Hence for any $\varepsilon > 0$:

$$B_\infty(x,\varepsilon) \cap F^c \neq \phi.$$

Using (10), for any $\varepsilon > 0$:

$$B_\infty(x,\varepsilon) \cap U^o \neq \phi \Rightarrow x \in \overline{(U^o)}.$$

Hence $\Delta(F) \subset \overline{(U^o)}$. Now :

$$U = [F^c \cup \Delta(F)] \subset [U^o \cup \Delta(F)] \subset$$

$$[U^o \cup \overline{(U^o)}] = \overline{(U^o)}.$$

(ii) Suppose, contrary to what is to be proven that $x \in U^o$ and $\psi(x) = 0$. Then $x \in F$ and since $F = \overline{(F^o)}$ by assumption A2, there exists an infinite sequence $\{x_j\}$ in F^o converging to x. By assumption A3, $\psi(x_j) < 0$ for all j and thus $x_j \in U^c$ for all j, which contradicts the fact that $x \in U^o$.

□

Proposition 3

Suppose that :

$$W \triangleq \cup \{B_\infty(y,1) \mid y \in U\}$$

$$W' = \cup \{B_\infty(y,1) \mid y \in U^o\} .$$

Then $W=W'$.

Proof

(i) Obviously $W' \subset W$.

(ii) Note that both W and W' are open sets since they are equal to the union of open sets. Let $x \in W$ so that $x \in B_\infty(y,1)$ for some $y \in U$. If $y \in U^o$ we also have $x \in W'$. Thus consider the case where $\psi(y) = 0$. Since $U = \overline{(U^o)}$ by proposition 2, there exists a sequence $\{y_i\}$ in U^o converging to y so that :

$$\|x-y_i\|_\infty \leq \|x-y\|_\infty + \|y-y_i\|_\infty < 1 \quad \text{for all } i \geq i_0 ,$$

$$\text{for some } i_0 > 0.$$

Thus $x \in B_\infty(y_{i_0},1)$ and since $y_{i_0} \in U^o$ we have that $x \in W'$ so that $W \subset W'$.

□

Proposition 4

$$G = \cap\{B_\infty(y,1)^c \mid y \in U\} = W^c. \tag{11}$$

Proof

We prove that $G = (W')^c$ and use proposition 3.

(i) $G \subset (W')^c$

If this is not true, there exists some $x \in (G \cap W')$. Hence there exists some $y \in U^o$ such that :

(a) $\psi(x+t) \leq 0$ for all $t \in T$ (since $x \in G$).

(b) $\|x-y\|_\infty < 1$ (since $x \in W'$).

(c) $\psi(y) > 0$ (by proposition 2(ii)).

Let $t' \triangleq y-x$. Using (b) we have that $t' \in T$. Also $\psi(y) = \psi(y-x+x) = \psi(x+t') \leq 0$ using (a). But this contradicts (c) so that (i) is true.

(ii) $(W')^c \subset G$

Suppose this is false. Then there exists some $x \in (W')^c \cap G^c$. Now $x \in G^c$ implies that there exists a $t \in T$ such that $\psi(x+t) > 0$. Hence by the continuity of ψ, there exists some $\gamma \in (0,1)$ such that $\psi(x+\gamma t) > 0$. Also $\|x+\gamma t - x\|_\infty = \gamma \|t\|_\infty < 1$. Thus $x \in B_\infty(x+\gamma t, 1)$ and $\psi(x+\gamma t) > 0$ so that $x \in W'$, a contradiction.

□

Let:

$$w(x) \triangleq \{y \in R^n \mid \psi(y) \geq 0, \|x-y\|_\infty = \eta(x)\}. \tag{12}$$

$w(x)$ is the set of minimizers of (8).

Proposition 5

The maps :

(i) $x \to B_\infty (y,1)^c$, $y \in w(x)$

(ii) $x \to B_\infty (x,\delta(x))^c$

are cut maps.

Proof

It is sufficient to prove that for any $x \in G^c$, $y \in w(x)$:

$$B_\infty (x, \delta(x)) \subset B_\infty(y,1) \subset G^c . \tag{13}$$

Let $x \in G^c$, $y \in w(x)$. Suppose that $z \in B_\infty (x,\delta(x))$, then :

$$\| z-y\|_\infty \leq \|z-x\|_\infty + \|x-y\|_\infty < \delta(x) + \eta(x) = 1.$$

Hence $z \in B_\infty(y,1)$ and the left hand side of (13) is true. The right hand side follows from the fact that $y \in U$ and proposition 4.

□

Figure III illustrates the definitions of $\delta(x)$, $\eta(x)$ and the cut maps $x \to B_\infty(x,\delta(x))^c$ and $x \to B_\infty(y,1)^c$, $y \in w(x)$. The validity of proposition 5 is based on the following intuitive result. If a point $x+t$ in $x+T$, $(x+T = \bar{B}_\infty(x,1))$, does not satisfy the specifications, then x should be moved through a distance of at least $1-\|t\|_\infty$ for the whole tolerance region $x+T$ to lie in F and thus for x to lie in G. Hence if $y \in w(x)$, x should be moved by at least $1 - \|x-y\|_\infty = 1 - \eta(x) = \delta(x)$ for $x+T$ to lie in F, which implies that $B_\infty(x, \delta(x)) \subset G^c$. Note that if $x \in U$ we have that :

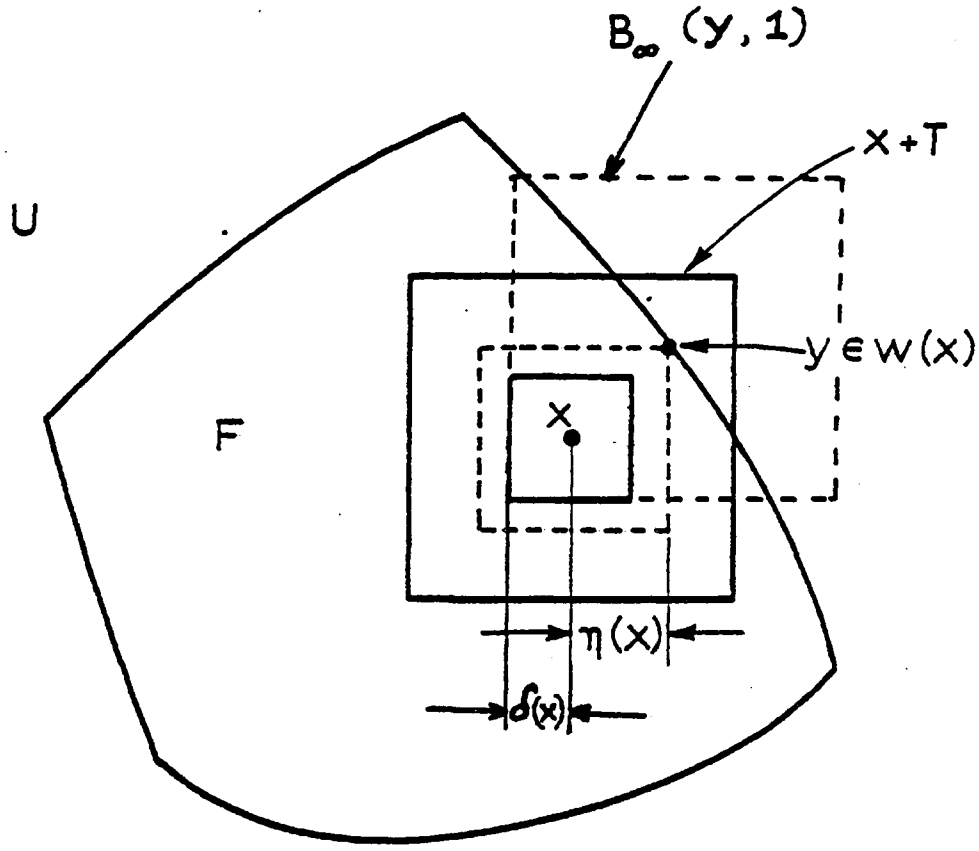


FIGURE III

The definitions of the separator and the cut maps.

$$B_\infty(x,\delta(x)) = B_\infty(y,1) = B_\infty(x,1)$$

since $w(x) = \{x\}$.

We are now in a position to state a conceptual algorithm for the problem P_T.

Algorithm 1

Step 0: Set k=0, $W_0 = \phi$.

Step 1: Compute any $x_k \in W_k^c$.

If $x_k \in G$ stop; else proceed to step 2.

Step 2: Compute $y_k \in w(x_k)$, $\delta(x_k) = 1 - \|x_k - y_k\|_\infty$.

Step 3: Set $W_{k+1} = W_k \cup B_\infty(y_k,1)$

[or $W_{k+1} = W_k \cup B_\infty(x_k,\delta(x_k))$.] .

Set k=k+1 and go to step 1.

□

Theorem 1

(i) If the algorithm stops at x_k, then $x_k \in G$.

(ii) If x^* is an accumulation point of an infinite sequence $\{x_k\}$ generated by the algorithm, then $x^* \in G$.

Proof

(i) The result is obvious.

(ii) The result follows from theorem 1 of Chapter I and the fact that, as shown, the maps $x \to B_\infty(y,1)^c$, $y \in w(x)$ and $x \to B_\infty(x,\delta(x))^c$ are cut maps.

□

Note that the utilization of cut maps ensures that each W_k^c is an outer approximation to the set G. Algorithm 1 possesses the following practical disadvantages.

(i) A feasibility subalgorithm is needed in step 1 to solve a non-differentiable problem. Hence unless such an algorithm is available, the computation is step 1 is not possible.

(ii) The test in Step 1 requires an infinite number of operations.

(iii) An exact global minimization is required in step 2 to compute y_k.

(iv) The subproblems in step 1 increase in complexity with k, since a new constraint is introduced at each iteration.

In the next section we proceed to obtain implementable algorithms, i.e. algorithms which do not have the disadvantages (i) - (iv) listed above.

II.3 Implementable algorithms for the tolerance problem.

To employ a standard finite dimensional feasibility subalgorithm in step 1 of algorithm 1, we need to have continuously differentiable constraint functions. We can modify the cut maps defined in the previous section by observing that :

$$B_2(x,\delta(x)) \subset B_\infty(x,\delta(x)) \quad \text{and} \quad B_2(y,1) \subset B_\infty(y,1).$$

Hence the maps $x \to B_2(x,\delta(x))^c$ and $x \to [B_2(x,\delta(x)) \cup B_2(y,1)]^c$, $y \in w(x)$ are cut maps. Note that $B_2(x, \delta(x))$ is not a subset of $B_2(y,1)$. The relation $x' \notin B_2(x,\delta(x))$ can be expressed as the following continuously differentiable inequality:

$$g(x') \triangleq \delta(x)^2 - \| x - x' \|_2^2 \leq 0. \tag{14}$$

Further details about the implementation of step 1 can be found in Chapter III.

Our next task is to replace the exact global minimization involved in Step 2 by a finite procedure.

Proposition 6

For any $x \in G^c$, let $y' \in x+T$ be such that $\psi(y') \geq 0$ and let $\bar{\delta} = 1 - \| x-y' \|_\infty$. Then :

$$B_2(x,\bar{\delta}) \subset G^c.$$

Proof

We have :

$$\bar{\delta} = 1 - \| y'-x \|_\infty \leq 1 - \eta(x) = \delta(x).$$

Hence:

$$B_2(x,\bar{\delta}) \subset B_2(x, \delta(x)) \subset G^c.$$

□

The important consequence of proposition 6 is that if we replace $y \in w(x)$ by some approximate minimizer $y' \in x+T$ satisfying $\psi(y') \geq 0$, the map $x \to B_2(x,\bar{\delta})^c$, where $\bar{\delta} = 1 - \| x-y' \|_\infty$, possesses the same basic property of cut maps (see proposition 5).

Let a map S: $R^n \times Z^+ \to R^n$ be such that for any $x \in R^n$ and $j \in Z^+$, S(x,j) is the result of applying $\tau(j)$ iterations of a certain algorithm to the problem:

$$\min_y \{ \|y-x\|_\infty \mid \psi(y) \geq 0 \}$$

where τ: $Z^+ \to Z^+$ is a monotonically increasing truncation function ($\tau(j) \to \infty$ as $j \to \infty$). We now impose the following conditions on S:

A4: (i) For all $x \in G^c$, there exists an integer I(x) such that :

$$\| S(x,j) - x \|_\infty < 1, \quad \psi(S(x,j)) \geq 0 \quad \text{for all } j \geq I(x).$$

(ii) For any compact subset X of G^c

$$| \eta(x) - \|S(x,j) - x\| |_\infty \to 0 \quad \text{as} \quad j \to \infty \text{ uniformly in } x \text{ for } x \in X.$$

The purpose of (i) is to ensure that at each $x \in G^c$ an approximation to the separator $\delta(x)$ having the property of proposition 6 can be generated in a finite number of iterations. Hence if we define

$$\bar{\delta}(x,j) \triangleq 1 - \bar{\eta}(x,j) \triangleq 1 - \| S(x,j) - x \|_\infty \qquad (15)$$

we observe that for any $x \in G^c$, we have :

$$0 < \bar{\delta}(x,j) \leq \delta(x) \quad \text{for all } j \geq I(x).$$

Condition (ii) ensures that the approximation of $\delta(x)$ is of increasing accuracy as j increases and is required for establishing convergence. It is clear that if $\{x_j\}$ is a convergent sequence in G^c, then :

$$| \bar{\delta}(x_j,j) - \delta(x_j) | \to 0 \quad \text{as} \quad j \to \infty .$$

Procedures that generate $S(x,j)$ and satisfy A4 are presented in Chapter III.

To avoid having to solve an increasingly complex subproblem in Step 1, a procedure for dropping cuts should be incorporated in our algorithm. It is possible to utilize the scheme proposed in [4] . It has the property of accumulating constraints and dropping them en masse when certain conditions are satisfied. However it is preferable to adopt the scheme proposed by Gonzaga and Polak in [5] . It requires more storage, but does not have an oscillatory behaviour, resulting in better computational properties.

Let $\{\varepsilon_j^k\}$ be a double indexed sequence satisfying:

(i) $\varepsilon_j^k > 0$ if $j < k$, $\varepsilon_j^k = 0$ otherwise.

(ii) $\varepsilon_j^k \nearrow \bar{\varepsilon}_j$ as $k \to \infty$, uniformly in j. (16)

(iii) $\bar{\varepsilon}_j \searrow 0$ as $j \to \infty$.

Examples of such sequences are :

$$\varepsilon_j^k = \gamma \max \{0, \delta^j - \delta^k\}, \ \gamma > 0, \ \delta \in (0,1) \tag{17}$$

and

$$\varepsilon_j^k = \gamma \max \{0, \frac{1}{(1+j)^{1/\delta}} - \frac{1}{(1+k)^{1/\delta}}\}, \ \gamma > 0, \ \delta > 0. \tag{18}$$

Let the set $J(k)$ be defined by:

$$J(k) \overset{\Delta}{=} \{ j \in Z^+ \mid j \leq k, \ \bar{\delta}(x_j, j) \geq \varepsilon_j^k \}. \tag{19}$$

The set $J(k)$ is, in a sense, the set of "most important" constraints at iteration k. The criteria according to which an index j associated with the point x_j is contained in $J(k)$, $k \geq j$, are the magnitude of $\bar{\delta}(x_j, j)$ and the value of j relative to k. Hence $J(k)$ contains the indeces of the most recent points whose separator estimates are large enough. In the implementable algorithm that will be presented, it specifies the balls utilized to form the next approximation (W_{k+1}^c) to the set G.

We are now in a position to state an implementable algorithm for the problem P_T.

Algorithm 2

Step 0: Set $k=0$, $W_0 = \phi$.

Step 1: Compute any $x_k \in W_k^c$.

Step 2: Compute $y_k = S(x_k,k)$.

Step 3: If $\psi(y_k) < 0$ or $\| y_k - x_k \|_\infty \geq 1$
set $x_{k+1} = x_k$, $W_{k+1} = W_k$, $k = k+1$
and go to step 2.

Else proceed to step 4.

Step 4: Compute $\bar{\delta}(x_k,k) = 1 - \| y_k - x_k \|_\infty$.

Set $W_{k+1} = \cup\{B_2(x_j, \delta(x_j,j)) \mid j \in J(k)\}$

Set $k=k+1$ and go to step 1.

□

Notes

(i) The algorithm is directly implementable since each step requires a finite number of operations.

(ii) If during the operation of the algorithm a point x_k in G is generated the algorithm will jam up, i.e. will start cycling between steps 2 and 3 increasing the accuracy of the estimation of $\delta(x_k)$ and establishing assymptotically that $\delta(x_k)$ is not positive.

(iii) The cut dropping scheme works as follows. A ball involving the point x_j is kept for all subsequent

iterations if $\bar{\delta}(x_j,j) \geq \bar{\varepsilon}_j$. If not, it is kept until k becomes large enough for $\bar{\delta}(x_j,j) \leq \varepsilon_j^k$ to be satisfied. In (17), δ determines how fast ε_j^k tends to $\bar{\varepsilon}_j$ and γ should be chosen to provide the right scaling. J(k) may be regarded as that subset of $\{1,\ldots,k\}$ which (approximately) defines the most important constraints. Roughly speaking these correspond (at iteration k) to those balls $B_\infty(x_j, \delta(x_j))$ for which $\delta(x_j)$ is largest <u>and</u> j is close to k. Once a constraint is dropped it does not reappear.

(iv) As already discussed each set W_k^c can be represented by a set of continuously differentiable inequalities as follows :

$$W_k^c = \{x \mid g^j(x) \leq 0, \quad \text{for all } j \in J(k-1)\} \tag{20}$$

where

$$g^j(x) \triangleq \bar{\delta}(x_j,j)^2 - \|x_j - x\|_2^2 . \tag{21}$$

Let

$$\chi^k(x) \triangleq \max \{g^j(x) \mid j \in J(k-1)\} . \tag{22}$$

Then $\chi^k(x_k) \leq 0$ implies that $x_k \in W_k^c$.

(v) It is possible to improve the convergence properties of algorithm 2 by modifying Step 1 to Step 1' as follows :

<u>Step 1'</u>: Compute any x_k such that :

$$x_k \in (W_k^c \cap G_k)$$

where

$$G_k \triangleq \{x \in R^n \mid f^j(x+t) \leq 0,\ j=1,\ldots,m,\ \text{for all } t \in T_k\} \tag{23}$$

and T_k is a finite subset of T. Any heuristic rule for specifying T_k can be employed. For example T_k can be some subset of the set V of vertices of T. The designer can use any knowledge about the problem to include the "worst case" vertices only. Since $G \subset G_k$, the maps $x \to [B_2(x,\delta(x))^c \cap G_k]$ are cut maps. The reason of this modification is that $W_k^c \cap G_k$ is a much better outer approximation to the set G than W_k^c. The introduction of these conventional constraints assumes that the functions f^j, $j=1,\ldots,m$ are continuously differentiable. Further implementation details are discussed in Chapter III.

(vi) Let :

$$H \triangleq \{x \in R^n \mid h^\ell(x) \leq 0,\ \ell = 1,\ldots,p\} \tag{24}$$

where $h^\ell : R^n \to R$, $\ell = 1,\ldots,p$ are continuously differentiable. Suppose we want to find a point in $G \cap H$. In other words suppose that we also have a set of conventional constraints to satisfy. Then Step 1 can be modified to Step 1" as follows:

Step 1": Compute any x_k such that :

$$x_k \in (W_k^c \cap G_k \cap H).$$

(vii) Finally note that since the map $x \to B_\infty(y,1)^c$, $y \in W(x)$ is also a cut map, it is possible to employ it in our

algorithms if a subalgorithm suitable for the non-differentiable feasibility subproblem $x \in [\ \cup \{ B_\infty(y_j, 1) \mid j \in J(k) \}\ ]^c$ is available.

Theorem 2

Any accumulation point x^* of a sequence $\{x_k\}$ constructed by algorithm 2 lies in G.

Proof

Case 1: The algorithm starts cycling between steps 2 and 3 so that $x^* = x_k$ for all $k \geq k^*$, for some $k^* > 0$. This implies that at x^* the subalgorithm utilized in the definition of the map S cannot find a point $y \in (x^* + T)^o$ satisfying $\psi(y) \geq 0$ in a finite number of iterations. By assumption A4(i) this implies that $x^* \notin G^c$, so that $x^* \in G$.

Case 2: Suppose that step 4 is entered infinitely many times and let K be the set of indeces k of a sequence converging to x^* so that $x_k \overset{K}{\to} x^*$. Take any $j, k \in K$ with $j < k$. By construction of the algorithm we have :

$$x_k \notin B_2(x_j,\ \bar{\delta}(x_j, j)) \quad \text{if} \quad j \in J(k-1)$$

$$0 < \bar{\delta}(x_j, j) \leq \varepsilon_j^{k-1} < \bar{\varepsilon}_j \quad \text{if} \quad j \notin J(k-1).$$

Hence :

$$0 < \bar{\delta}(x_j, j) \leq \| x_k - x_j \|_2 \quad \text{if} \quad j \in J(k-1)$$

$$0 < \bar{\delta}(x_j, j) \leq \bar{\varepsilon}_j \quad \text{if} \quad j \notin J(k-1)$$

for $j, k \in K$, $j < k$. Combining the above :

$$0 < \bar{\delta}(x_j,j) \leq \max\{\bar{\varepsilon}_j, \| x_k - x_j \|_2\} \leq \bar{\varepsilon}_j + \| x_k - x_j \|_2 .$$

Hence, using (16):

$$\bar{\delta}(x_j,j) \to 0 \quad \text{as} \quad j \to \infty . \tag{25}$$

Suppose, contrary to what is to be proven, that $x^* \in G^c$. Then, since G^c is open by (11) there exists a compact neighbourhood $N(x^*) \subset G^c$. By convergence, there exists some $j^* > 0$ such that $x_j \in N(x^*)$ for all $j \geq j^*$. Hence by A4(ii), $| \delta(x_j) - \bar{\delta}(x_j,j)| \to 0$ as $j \to \infty$ and using (25) we see that $\delta(x_j) \to 0$ as $j \to \infty$. By the continuity of δ, the above implies that $\delta(x^*) = 0$, a contradiction to $x^* \in G^c$. Hence $x^* \in G$.

□

The computation of $y_k = S(x_k,k)$ in step 2 combined with the tests of step 3 has the following two purposes:

(i) It estimates using a finite number of operations if x_k lies in G^c (assumption A4(i)), replacing the conceptual test in step 1 of algorithm 1.

(ii) It generates a suitable approximation to the separator $\delta(x_k)$ that allows convergence results to be established (assumption A4(ii)).

However, a procedure that only performs (i) above may be computationally cheaper than one that performs both (i) and (ii). Consider the following assumption:

A5: Suppose that for all $x \notin G^c$ the set

$$\Delta(x) \triangleq \{y \mid \|x-y\|_\infty = 1\} \tag{26}$$

satisfies

$$\Delta(x) \cap U^o \neq \phi. \tag{27}$$

Note that :

$$\Delta(x) = \{(x+T) \setminus (x+T)^o\}. \tag{28}$$

This assumption should be frequently satisfied in practice. In simple words it requires that the set F does not possess "black holes" that can be totally contained in x+T for some $x \in R^n$. The following proposition defines a class of special but common cases for which A5 is true.

Proposition 7

Suppose that :

(i) For all $x \in G^c$

$$(x+T)^c \cap U \neq \phi$$

(ii) The interior of U is connected.

Then A5 is satisfied.

Proof

Choose any $x \in G^c$. Since $(x+T)^c \cap U \neq \phi$ and $U = \overline{(U^o)}$ by proposition 1 we have that :

$$(x+T)^c \cap U^o \neq \phi. \tag{29}$$

Also since $x \in G^c$:

$$(x+T)^o \cap U^o \neq \phi. \tag{30}$$

Now suppose, contrary to what is to be proven that :

$$\Delta(x) \cap U^o = \phi . \tag{31}$$

We then have using (28) and (31):

$$U^o = [(x+T)^o \cap U^o] \cup [(x+T)^c \cap U^o] \cup [\Delta(x) \cap U^o] = A \cup B$$

$$\stackrel{\Delta}{=} [(x+T)^o \cap U^o] \cup [(x+T)^c \cap U^o]$$

where A and B are <u>non-empty</u> by (29) and (30). Because $(x+T)^o$ and $(x+T)^c$ are open and disjoint, also A and B are open and disjoint and hence they are separated (i.e. $\bar{A} \cap B = \phi$ and $A \cap \bar{B} = \phi$). But then U^o is the union of two separated sets and thus is not connected, which is a contradiction.

□

Note that (i) in the statement of proposition 7 will be satisfied for any practical problem. Suppose that A5 is true, then :

$$G = \{ x \mid \Delta(x) \cap U^o = \phi \} = \{ x \mid \psi(x) \leq 0 \quad \text{for all } x \in \Delta(x) \}. \tag{32}$$

Relation (32) can be utilized to obtain procedures that satisfy A4(i) (but not A4 (ii)) and are computationally cheaper than those that generate S. Suppose that the map S': $R^n \times Z^+ \to R^n$ is defined by such a procedure (see Chapter III for implementation details). Then :

$$\delta(x) \geq 1 - \|x-S'(x,j)\|_\infty \quad \text{for all } j \geq I(x)$$

for some $I(x) > 0$, where $x \in G^c$.

To present our final algorithm for the problem P_T we first state a slightly stronger version of assumption A4.

A4': For any compact set X of G^c:

(i) There exists an integer I_X such that

$$\psi(S(x,j)) \geq 0 \quad , \|S(x,j) - x\|_\infty < 1 \quad \text{for all } j \geq I_X,$$

for all $x \in X$.

(ii) $|\eta(x) - \|x-S(x,j)\|_\infty| \to 0$ as $j \to \infty$ uniformly in x for $x \in X$.

We are now in a position to state our last implementable algorithm for the problem P_T.

Algorithm 3.

Step 0: Set k=0, $W_0 = \phi$.

Step 1: Compute any $x_k \in W_k^c$.

Step 2: Compute

$$y_k = \begin{cases} S(x_k,k) & \text{if } x_k \neq x_{k-1}. \\ S'(x_k,k) & \text{if } x_k = x_{k-1}. \end{cases}$$

Step 3: If $\psi(y_k) < 0$ or $\|y_k - x_k\|_\infty \geq 1$ set

$x_{k+1} = x_k$, $W_{k+1} = W_k$, k= k+1 and go to step 2.

Else proceed to step 4.

Step 4: Compute $\bar{\delta}(x_k,k) = 1 - \|x_k - y_k\|_\infty$.

Set $W_{k+1} = \cup \{B_2(x_j, \bar{\delta}(x_j,j)) \mid j \in J(k)\}$.

Set k=k+1 and go to step 1.

□

Theorem 3

Suppose that the map S satisfies assumption A4' and the map S' assumption A4(i). Then any accumulation point x* of an infinite sequence generated by algorithm 3 lies in G.

Proof

Case 1: Suppose that the algorithm starts cycling between steps 2 and 3 so that $x^* = x_k$, $y_k = S'(x_k,k)$ for all $k \geq k^*$ for some $k^* > 0$. Then $x^* \in G$ by exactly the same arguments as in the proof of theorem 2 since S' satisfies A4(i).

Case 2: Suppose that step 4 is entered infinitely many times and that $x_k \xrightarrow{K} x^*$. Also suppose that $x^* \in G^c$. There there exists a compact neighbourhood $N(x^*) \subset G^c$ and some $k^* > 0$ such that $x_k \in N(x^*)$ for all $k \geq k^*$. Let I_N be an integer associated with the map S and the compact set $N(x^*)$ in the sense of A4'(i). We observe that for $k > \max\{ I_N, k^* \}$, step 2 of algorithm 3 will always generate $y_k = S(x_k,k)$ as the conditions of step 3 will always be satisfied (assumption A4'(i)). Hence by exactly the same arguments as in the proof of theorem 2 we can conclude that $\delta(x^*) = 0$ which contradicts $x^* \in G^c$, so that $x^* \in G$.

□

All the discussion concerning algorithm 2 also applies to algorithm 3. The difference is that in algorithm 3 a computationally cheaper subprocedure is utilized in order to estimate if a certain point x_k lies in G^c or not. Hence algorithm 3 should be computationally more efficient than algorithm 2, especially in cases in which an indefinite cycling between steps 2 and 3 eventually occurs (i.e. a point in G is generated).

In the next Chapter implementation details of the algorithms proposed are presented, numerical examples are given and conclusions about the properties of the algorithms are drawn.

CHAPTER III

IMPLEMENTATION OF THE ALGORITHMS FOR THE TOLERANCE PROBLEM

III.1 Introduction.

This chapter is dedicated to the implementation details of the algorithms in Chapter II. Computational efficiency and overall practicability are very much dependent on the way the algorithms are implemented. In Section 2 ways of solving the subproblem of step 1 are discussed. Procedures for the global optimization problem of step 2 that satisfy assumptions A4 and A4' are proposed in section 3. Numerical examples are presented in section 4. Finally, in section 5 conclusions are drawn about the properties and the performance of the algorithms.

III.2 On the solution of the feasibility subproblem.

We recall that each set W_k^c which is the intersection of the complements of a finite number of balls, can be represented by a set of continuously differentiable inequalities as follows :

$$W_k^c = \{ x \mid g^j(x) \leq 0 \quad \text{for all } j \in J(k-1) \} \tag{1}$$

where

$$g^j(x) = \bar{\delta}(x_j, j)^2 - \| x_j - x \|_2^2 \tag{2}$$

and

$$\nabla_x g^j(x) = 2(x_j - x). \tag{3}$$

The first important consideration is that the sets W_k are bounded so that a linear search in any direction will yield a point in W_k^c. A suitable search

direction is $(x_{k-1}-y_{k-1})$ since, usually, moving in this direction will yield points that lie nearer the set G. Hence starting from x_{k-1} and taking steps in the above direction will produce some $x_k \in W_k^c$. A suitable step length for these steps is $a\bar{\delta}(x_{k-1},k-1)$ where $a<1$. Note that solving the subproblem $x_k \in W_k^c$ in this way is computationally very cheap since the evaluation of constraints of the form of (2) is only required. More sophisticated methods that take into account all the active constraints in (1) for the determination of the search direction can be employed.

The simple approach described is crude and has the disadvantage that other conventional constraints cannot be included in the feasibility subproblem of step 1. A much better method is to employ a standard inequality solving algorithm. The algorithm in [22] which is summarized in the appendix is very suitable for this purpose. It combines the quadratic rate of convergence of Newton's method with the finite convergence of first order algorithms. At iteration k it generates x_k in the <u>interior</u> of each constraint set, thus increasing the probability of x_k lying in G.

The general subproblem in step 1 of the algorithms of Chapter II has the following form:

P_{W_k}: Find a point x_k in the set $W_k^c \cap H \cap G_k$ defined by the constraints :

$$g^j(x) \leq 0 \ , \quad j \in J(k-1)$$

and

$$h^{\ell}(x) \leq 0, \quad \ell \in L_k \qquad (4)$$

where each $g^j(x)$ has the form of (2) and L_k is some finite set (for

notational simplicity we assume that the constraints $h^{\ell}(x) \leq 0$, $\ell \in L_k$ correspond to both the set H and the set G_k). Let :

$$\chi_k(x) \triangleq \max \{ g^j(x) \mid j \in J(k-1) \} \tag{5}$$

$$\phi_k(x) \triangleq \max \{ h^{\ell}(x) \mid \ell \in L_k \} \tag{6}$$

$$\bar{\psi}_k(x) \triangleq \max \{ \chi_k(x), \ \phi_k(x) \} \ . \tag{7}$$

Also let:

$$\bar{J}_k(x) \triangleq \{ j \in J(k-1) \mid g^j(x) = \bar{\psi}_k(x) \} \tag{8}$$

$$\bar{L}_k(x) \triangleq \{ \ell \in L_k \mid h^{\ell}(x) = \bar{\psi}_k(x) \} \ . \tag{9}$$

The standard assumption employed to prove convergence results for the modified Newton algorithm of [22] is the following :

A1: The set $\{ \nabla_x g^j(x) \mid j \in \bar{J}_k(x); \ \nabla_x h^{\ell}(x) \mid \ell \in \bar{L}_k(x) \}$ is positive linearly independent for all x such that $\bar{\psi}_k(x) \geq 0$.

This assumption ensures that all x such that $\bar{\psi}_k(x) \geq 0$, a descent direction for $\bar{\psi}_k(x)$ exists and hence the algorithm cannot jam up at a non-feasible point. Now clearly in the case of the problem P_{W_k} there are some non-feasible points that do not satisfy A1. For example at the centre of the $(k-1)^{th}$ ball:

$$\nabla_x g^{k-1}(x_{k-1}) = 2(x_{k-1} - x_{k-1}) = 0$$

and hence if $(k-1) \in \bar{J}_k(x_{k-1})$ (usually true), A1 is violated at x_{k-1}.

Also if two balls intersect, there is usually some point on the line segment connecting their centers which does not satisfy A1. Because all these points are either local maxima or "saddle points" of the function $\bar{\psi}_k(x)$ (and all algorithms always reduce $\bar{\psi}_k$ at each iteration), they do not usually create any practical problems since it is very improbable for an algorithm to jam up at them, unless they are used as initial points. From the above we conclude that x_{k-1} is not a suitable initial point for the problem P_{W_k}. A suitable initial point for this problem is :

$$\hat{x}_k = x_{k-1} + \lambda(x_{k-1} - y_{k-1})$$

where

$$\lambda = a\bar{\delta}(x_{k-1},k-1)\Big/ \|x_{k-1}-y_{k-1}\|_2, \quad a<1$$

and $y_{k-1} = S(x_{k-1},k-1)$. The above initial point has the highly desirable property that it usually lies nearer the set G than x_{k-1}. Hence the probability of x_k lying nearer G than x_{k-1} is large. If $x_{k-1} = y_{k-1}$ we can replace $x_{k-1} - y_{k-1}$ by $x_{k-1} - x_{k-2}$ in the expressions above.

From the above discussion we conclude that for each subproblem P_{W_k} there exists a set of known "singular points" that do not satisfy A1. Almost always these points do not create practical problems so that a solution to the problem P_{W_k} can be efficiently obtained by direct application of the algorithm in [22] employing the initial point rule proposed above. However, it is not theoretically guaranteed that jamming at these points will not occur. Let the set X_{W_k} contain all these known "probable" singular points, i.e. those which are either centres of balls or lie on the line segment connecting the centers of two intersecting balls. More explicitly, X_{W_k} will contain the ball centers $x_j, j\in J(k-1)$.

Also considering the balls corresponding to the constraints $g^{j_1}(x)$ and $g^{j_2}(x)$, $j_1, j_2 \in J(k-1)$, the gradients $\nabla_x g^{j_1}(x)$ and $\nabla_x g^{j_2}(x)$ are linearly dependent on the line segment connecting their centers. This segment can be described by the equation :

$$y = \alpha\, x_{j_1} + (1-\alpha)\, x_{j_2} \quad , \quad 0 \leq \alpha \leq 1. \tag{10}$$

Substituting and solving for α in the equation :

$$g^{j_1}(y) = g^{j_2}(y)$$

we obtain :

$$\alpha_{j_1,j_2} = \frac{\bar{\delta}(x_{j_1},j_1)^2 - \bar{\delta}(x_{j_2},j_2)^2 + \|x_{j_1} - x_{j_2}\|_2^2}{2\|x_{j_1} - x_{j_2}\|_2^2}. \tag{11}$$

Suppose that y_{j_1,j_2} is the point obtained if α_{j_1,j_2} from (11) is substituted in (10). Then y_{j_1,j_2} will be included in X_{W_k} if the following conditions are satisfied :

$$0 \leq \alpha_{j_1,j_2} \leq 1$$

$$g^{j_1}(y_{j_1,j_2}) = g^{j_2}(y_{j_1,j_2}) = \chi_k(y_{j_1,j_2}) \geq 0.$$

The satisfaction of these conditions implies that A1 is probably violated at y_{j_1,j_2}. From all the above it is clear that X_{W_k} is a set of known probable singular points that can be analytically computed for each k. Let :

$$d_k \triangleq \min\{\chi_k(x) \mid x \in X_{W_k}\}. \tag{12}$$

By the definition of X_{W_k} we have that :

$$d_k \geq 0. \tag{13}$$

We make the following assumption :

A2: $d_k > 0.$

Note that this assumption is mild, since $d_k=0$ will only happen if two balls are tangent to each other. Now suppose that we have a point $\hat{x}$ satisfying :

$$0 < \chi_k(\hat{x}) < d_k. \tag{14}$$

It will be shown later that such a point can be computed by a finite procedure. Choose $c_k \geq 1$ such that :

$$c_k \chi_k(\hat{x}) > \phi_k(\hat{x}). \tag{15}$$

Note that when no conventional constraints are present (L_k empty) c_k can be taken equal to unity. Let :

$$\bar{\psi}_k(x,c) \triangleq \max \{ c\chi_k(x), \phi_k(x) \}. \tag{16}$$

Then using all the above relations :

$$\bar{\psi}_k(\hat{x},c_k) = c_k\chi_k(\hat{x}) < c_k d_k \leq c_k\chi_k(x) \leq \bar{\psi}_k(x,c_k)$$

$$\text{for all } x \in X_{W_k}. \tag{17}$$

Now consider the problem:

$P_{W_k'}$: Find a point in the set $(W_k')^c$ defined by :

$$g'^j(x) \lesssim 0 \quad , \quad j \in J(k-1) \tag{18}$$

$$h^\ell(x) \lesssim 0 \quad , \quad \ell \in L_k$$

where

$$g'^j(x) \triangleq c_k\, g^j(x). \tag{19}$$

It is clear that $W_k = W_k'$ since $c_k > 0$. Also by the definition of X_{W_k} we see that:

$$X_{W_k} = X_{W_k'} \tag{20}$$

where $X_{W_k'}$ is the set of probable singular points of the sort considered and is defined for $P_{W_k'}$ in the same way that X_{W_k} is defined for P_{W_k}.
Let :

$$\psi_k'(x) = \max\{\chi_k'(x) \, , \, \phi_k(x)\} \tag{21}$$

where

$$\chi_k'(x) = \max \{ g'^j(x) \,\big|\, j \in J(k-1)\} \quad . \tag{22}$$

Clearly :

$$\psi_k'(x) = \bar{\psi}_k(x, c_k) \, . \tag{23}$$

Let the set C_k be defined by :

$$C_k \triangleq \{ x \,\big|\, \psi_k'(x) \lesssim \psi_k'(\hat{x})\} \, . \tag{24}$$

Using (14), (17), (20), (23) and (24) we deduce that :

$$W_k^c = (W_k')^c \subset C_k$$

$$C_k \cap X_{W_k'} = \phi \tag{25}$$

$$\hat{x} \in C_k .$$

In words, we have a point $\hat{x}$ in a set C_k that contains the set $(W_k')^c$ and does not contain any singular points of the sort considered for the problem $P_{W_k'}$. Let $J_k'(x)$ and $L_k'(x)$ be defined for $P_{W_k'}$ in the same way that $\bar{J}_k(x)$ and $\bar{L}_k(x)$ are defined for P_{W_k}. Then we make the following assumption in the place of A1:

A1': The set $\{\nabla_x g^j(x) \mid j \in J_k'(x);\ \nabla_x h^\ell(x) \mid \ell \in L_k'(x)\}$ is positive linearly independent for all $x \in C_k$ such that $\psi_k'(x) \geq 0$.

We can now directly apply the algorithm of [22] to the problem $P_{W_k'}$ starting from the initial point $\hat{x} \in C_k$. Because the algorithm always reduces $\psi_k'(x)$ at each iteration we see that (by (24)):

$$x \in C_k \Longrightarrow A(x) \in C_k$$

where $A(x)$ is the algorithm model map (see definition in [22]). Hence A1' ensures that the convergence theory of [22] is valid and any accumulation point generated should lie in the interior of $W_k'^c$ (and hence of W_k^c). Note that if no conventional constraints are present, the algorithm should generate a point in the interior of W_k^c in a finite number of iterations even if the sequence produced is not convergent, since W_k is compact.

To summarize, it is possible by choosing $\hat{x}$ and c_k to ensure that at all the "probable" singular points of the sort considered the value of the maximum of the constraints is greater than its value at the initial point $\hat{x}$. Hence since the algorithm in [22] always reduces the maximum of the constraints at each iteration, it is not possible for it to jam at any of these singular points.

It is possible to generate some $\hat{\hat{x}}$ that satisfies (14) with a finite procedure because of the special form that the constraints $g^j(x)$ have. Note that $\chi_k(x) \to -\infty$ as $\| x \|_\infty \to \infty$. For practical reasons it is advantageous to generate $\hat{x}$ satisfying the following relation instead of (14), to avoid the parameter c_k being chosen too large unnecessarily.

$$0 < \frac{d_k}{2} < \chi_k(\hat{x}) < d_k \quad . \tag{26}$$

The following algorithm will compute some $\hat{x}$ satisfying (26) in a finite number of iterations. The first three steps perform a linear search to generate points z_1^h and z_1^ℓ satisfying :

$$\chi_k(z_1^h) \geq d_k$$
$$\chi_k(z_1^\ell) \leq \frac{d_k}{2} .$$

The rest of the steps implement a bisection procedure to generate $\hat{x}$.

Algorithm 1

Data: (i) $\alpha > 0$, $d_k > 0$

(A suitable value for α is $\alpha = \dfrac{a\,\bar{\delta}(x_{k-1},k-1)}{\|x_{k-1}-y_{k-1}\|_2}$, $a<1$).

(ii) $\hat{y}_1 \in R^n$ such that $\chi_k(\hat{y}_1) \geq d_k$ (i.e. $\hat{y}_1 \in X_{W_k}$).

(A suitable value is $\hat{y}_1 = x_{k-1}$).

(iii) $\hat{y}_2 \in R^n$.

(For reasons discussed in the definition of the initial point rule it is advantageous to use $\hat{y}_2 = y_{k-1}$).

Step 0: Set i=1, j=1.

Step 1: Compute $z_j = \hat{y}_1 + j\alpha(\hat{y}_2-\hat{y}_1)$.

Step 2: If $\chi_k(z_j) \geq d_k$ set j=j+1 and go to step 1.
Else proceed to step 3.

Step 3: If $\chi_k(z_j) > \dfrac{d_k}{2}$, stop and set $\hat{x} = z_j$.
Else set $z_i^{\ell} = z_j$, $z_i^h = z_{j-1}$ and proceed to step 4.

Step 4: Set $x_i = \dfrac{z_i^h + z_i^{\ell}}{2}$.

Step 5: If $\dfrac{d_k}{2} < \chi_k(x_i) < d_k$ stop and set $\hat{x}=x_i$.

Else proceed to step 6.

Step 6: If $\chi_k(x_i) \geq d_k$, set $z_{i+1}^h = x_i$, $z_{i+1}^{\ell} = z_i^{\ell}$.

If $\chi_k(x_i) \leq \dfrac{d_k}{2}$, set $z_{i+1}^h = z_i^h$, $z_{i+1}^{\ell} = x_i$.

Set i=i+1 and go to step 4. □

Theorem 1

The algorithm generates some $\hat{x}$ satisfying (26) in a finite number of iterations (i.e. one of the "stop" commands is executed after a finite number of iterations).

Proof

By construction of the algorithm once step 3 has been entered steps 1 and 2 are not used again. We have that:

$$\chi_k(z_j) \to -\infty \quad \text{as} \quad \|z_j\| \to \infty$$

$$\|z_j\|_\infty \to \infty \quad \text{as} \quad j \to \infty.$$

Hence there exists some finite $\hat{j}>0$ such that $\chi_k(z_{\hat{j}}) < d_k$ and step 3 will be entered after a finite number of cyclings between steps 1 and 2.

Now suppose that none of the stopping commands (steps 3 and 5) is executed so that the algorithm cycles indefinitely often between steps 4 and 6. Then two infinite sequences $\{z_i^h\}$ and $\{z_i^\ell\}$ are generated that have the following properties :

$$\|z_{i+1}^h - z_{i+1}^\ell\|_2 = \frac{1}{2}\|z_i^h - z_i^\ell\|_2$$

$$\chi_k(z_i^h) \geq d_k \tag{27}$$

$$\chi_k(z_i^\ell) \leq \frac{d_k}{2} .$$

Hence :

$$\|z_i^h - z_i^\ell\|_2 \to 0 \quad \text{as} \quad i \to \infty. \tag{28}$$

Since $\{z_i^{\ell}\}$ and $\{z_i^{h}\}$ lie in a compact set (the line segment joining z_1^{h} and z_1^{ℓ}), they have accumulation points and by (28) they have common accumulation points. Suppose z^* is such an accumulation point so that $z_i^{\ell} \overset{I_{\ell}}{\to} z^*$ and $z_i^{h} \overset{I_h}{\to} z^*$. By the continuity of χ_k and (27) we have :

$$\chi_k(z^*) = \lim_{\substack{i \to \infty \\ i \in I_h}} \chi_k(z_i^h) \geq d_k > \frac{d_k}{2}$$

$$\chi_k(z^*) = \lim_{\substack{i \to \infty \\ i \in I_{\ell}}} \chi_k(z_i^{\ell}) \leq \frac{d_k}{2} .$$

This is a contradiction and hence the theorem is valid.

□

To summarize, it is possible by employing the techniques described to readily solve the subproblem in step 1 of the algorithms of Chapter II. If no constraints on the vertices of the tolerance region are present ($G_k = R^n$), the solution is computationally very cheap since evaluation of the design constraints and their derivatives is not required. The advantage of employing "vertex constraints" is that the set $W_k^{c} \cap H \cap G_k$ is a much better approximation of G than W_k^{c} or $W_k^{c} \cap H$. Other conventional constraints (e.g. "box" constraints on the parameter values) that define the set H improve the convergence properties of the algorithm without seriously affecting computational efficiency. Note that compactness of H guarantees the generation of accumulation points. The initial point rule proposed causes the very desirable "centering

effect" described which greatly speeds up convergence. The introduction of conventional constraints has the sole purpose of improving the convergence properties of the algorithms and does not affect the convergence proofs presented in Chapter II. Hence these constraints may be relaxed or removed if any difficulties occur during the computation of x_k in Step 1 (e.g. jamming at a non-feasible point) due to assumptions A1 or A1' being violated (see appendix for a jamming criterion). An interactive program allowing the specification of the conventional constraints (i.e. the sets G_k and H) according to the progress of the algorithm, is thus very suitable for the implementation of the algorithm.

III.3 On the computation of the separator estimates.

We recall from Chapter II that S(x,j) is the result of applying $\tau(j)$ iterations of a certain algorithm to the global optimization problem:

$$\eta(x) = \min_y \{ \| y-x \|_\infty \mid \psi(y) \geq 0 \} \tag{29}$$

where τ is a monotonically increasing truncation function. The estimates of $\eta(x)$ and $\delta(x)$ are defined as :

$$\bar{\eta}(x,j) \triangleq \| x - S(x,j) \|_\infty \tag{30}$$

and

$$\bar{\delta}(x,j) \triangleq 1 - \bar{\eta}(x,j).$$

This is where the inherent complexity of the non-convex problem P_T occurs. Problem (29) is equivalent to the determination of the "worst case" points of the other methods in the literature. It is only in

the one-dimensionally convex case that this computation can be replaced by a finite dimensional problem involving the vertices only. The general methods for global optimization that exist in the literature [23] , [24], [25] are yet at an early stage. They utilize heuristic principles and convergence proofs are not available. However the problem (29) is considerably simpler than the general global optimization problem. Note that (29) is equivalent to :

$$\min_{y,w} \{ w \mid |y^j - x^j| \leq w, \quad j=1,\ldots,n, \quad \psi(y) \geq 0 \}. \tag{31}$$

The procedures proposed below for computing $S(x,j)$ will be shown to satisfy assumptions A4 and A4' of Chapter II. They utilize sets of "mesh points" in $(x+T)$. In this way uniform convergence to the global minimum can be established since certain continuity properties are present.

Procedure 1

Compute $\tau(j)$ "uniformly spaced" points $Z_j(x) = \{z_1,\ldots,z_{\tau(j)}\}$ in $(x+T)^o$ where $\tau(j) \to \infty$ as $j \to \infty$. Order these points so that $\|z_i - x\|_\infty$ increases with i. Set $y_j = z_i^*$ where z_i^* is the first point in the ordered sequence such that $\psi(z_i^*) \geq 0$. Set $S(x,j) = y_j$ and $\bar{\eta}(x,j) = \|x - S(x,j)\|_\infty$. If no such point exists set $\bar{\eta}(x,j) = 1$.

Procedure 2

Compute $\tau(j)$ points $Z_j(x) = \{z_1,\ldots,z_{\tau(j)}\}$ in $(x+T)^o$ spaced so that ε_j tends to zero as $j \to \infty$, where :

$$\varepsilon_j \overset{\Delta}{=} \max_z \min_r \{ \|z - z_r\|_\infty \mid z \in x+T, \ r = 1,\ldots,\tau(j) \}. \tag{32}$$

Order these points and compute $S(x,j)$ as in Procedure 1 above.

Procedure 3

Compute a set $\tilde{Z}_j(x)$ of points as in procedure 2. Apply an optimization algorithm for problem (29) using each of these points as an initial point, stopping the algorithm when some optimality condition is approximately satisfied. This yields an improved set of points $Z_j(x)$. Order these and compute $S(x,j)$ as in procedure 1.

□

Notes

(i) Most pseudo-random number generators should satisfy (32).

(ii) By the definition of the procedures the number of system analyses - evaluations of ψ - required for the computation of $S(x,j)$ is $N(j)$ where :

$$N(j) = i^* \doteq \tau(j) \quad \bar{\eta}(x,j)^n. \tag{33}$$

Initially when the yield is small ($\eta(x)$ small), $N(j)$ will be small.

(iii) Any heuristic procedure that improves on the estimate of $\bar{\eta}(x,j)$ generated by the above procedures can be employed without affecting the validity of assumptions A4 and A4' of Chapter II. A simple technique is to employ a linear search starting from $S(x,j)$ and moving in steps along the line segment joining $S(x,j)$ to x. The step length should be chosen according to the density of the mesh points in $Z_j(x)$. Figure I illustrates the operation of this technique ($\|x-\hat{S}(x,j)\|_\infty$ is the improved estimate of $\eta(x)$).

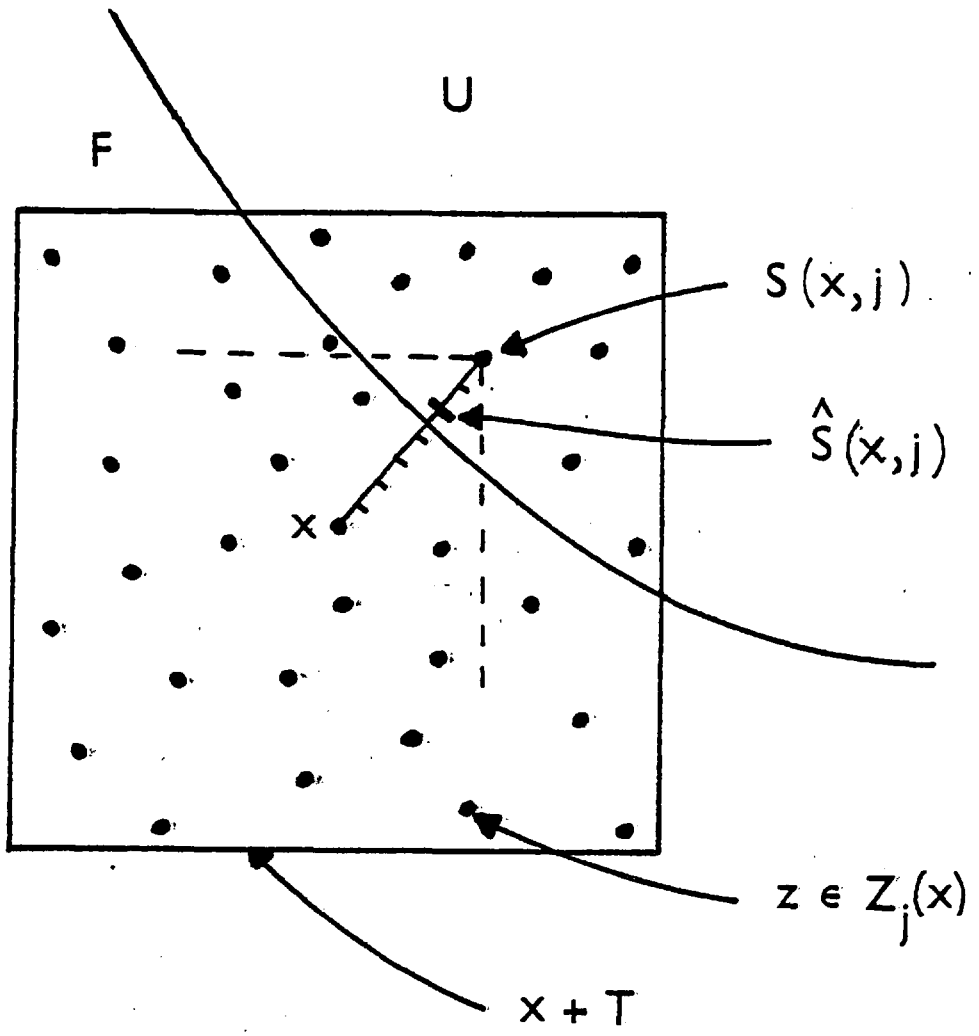


FIGURE I

The linear searching technique.

$1-\left\| x-\hat{S}(x,j) \right\|_{\infty}$ is the improved separator estimate.

(iv) The procedures proposed generate $S(x,j)$ given x and j. If algorithm 2 of Chapter II cycles between steps 2 and 3, j is increased while x remains constant (this only happens if all points in $Z_j(x)$ satisfy $\psi(z) < 0$). The mechanism for generating $Z_{j+1}(x)$ should be such that when j is increased to $j+1$, $\tau(j+1)-\tau(j)$ extra points are generated, i.e. $Z_{j+1}(x) = Z_j(x) \cup Z^e_{j+1}(x)$. The extra points are ordered and $S(x,j+1)$ set equal to the first element z_i satisfying $\psi(z_i) \geq 0$ in the ordered extra set. This procedure is repeated when $j+1$ is increased to $j+2$ etc.

(v) As discussed in note (ii) the proportion of points in $Z_j(x)$ that has to be tested for $S(x,j)$ to be computed is small when x is far from G. However as x approaches G this proportion becomes larger and larger. It is possible to utilize the following artifice (proposed by Soin and Spence [15]) to save a considerable amount of computation. Suppose that x_i is close to G so that $\delta(x_i)$ is small, then x_{i+1} will be close to x_i and $(x_{i+1}+T)$ will overlap a lot with x_i+T. Therefore many points in $Z_i(x_i)$ can be employed in $Z_{i+1}(x_{i+1})$ saving re-evaluation of ψ for these points. (This artifice was not employed in the examples presented in the next section).

(vi) The choice of the truncation function $\tau(j)$ is important for computational efficiency. It is desirable for $\tau(j)$ to be small and increase slowly as long as this permits the computation of $S(x,j)$ satisfying $\psi(S(x,j)) \geq 0$. However if algorithm 2 cycles between steps 2 and 3, $\tau(j)$ should be increased considerably to avoid a large number of such cyclings before a

suitable S(x,j) is generated. One possible definition employed in the filter examples of the next section is the following :

$$\tau(j+1) = \tau(j) + 100 \quad \text{if} \quad x_{j+1} \neq x_j$$

$$\tau(j+1) = 2\tau(j) \quad \text{if} \quad x_{j+1} = x_j$$

where $\tau(0)$ is specified a priori (e.g. $\tau(0) = 100$). Ideally, an interactive program allowing the specification of the truncation function as the algorithm progresses should be utilized.

Theorem 2

Procedures 1-3 that generate S(x,j) satisfy assumptions A4 and A4' of Chapter II.

Proof

Firstly note that procedure 1 also satisfies $\varepsilon_j \to 0$ as $j \to \infty$, where ε_j is defined by (32). Also $\bar{n}(x,j) \geq n(x)$, by the definition of the procedures, for all $x \in G^c$.

(i) Choose any $x \in G^c$. Since U is equal to the closure of its interior the open set $(x+T)^o \cap U^o$ is not empty. Hence there exists some $y \in R^n$ and some $\varepsilon_x > 0$ such that :

$$B_\infty(y, \varepsilon_x) \subset [(x+T)^o \cap U^o] .$$

Now choose I(x) large enough so that $\varepsilon_j < \frac{\varepsilon_x}{2}$ for all $j \geq I(x)$. Then, by construction of the set $Z_j(x)$, at least one point

$z_j \overset{\Delta}{=} (x+\sigma_j) \in Z_j(x)$ lies in $B_\infty(y,\frac{\varepsilon_x}{2})$ for all $j \geq I(x)$.
Hence $z'_j \overset{\Delta}{=} (x'+\sigma_j) \in Z_j(x')$ lies in $B_\infty(y,\varepsilon_x)$ for all $x' \in B_\infty(x, \frac{\varepsilon_x}{2})$ and for all $j \geq I(x)$. We then have :

$$z'_j \in (x'+T)^o,\ \psi(z'_j) \geq 0 \quad \text{for all } x' \in B_\infty(x,\tfrac{\varepsilon_x}{2}),\ \text{for all } j \geq I(x)$$

where $z'_j \in Z_j(x')$. By the procedure definitions, all the above imply that :

$$\|x' - S(x',j)\|_\infty < 1,\ \psi(S(x',j)) \geq 0 \quad \text{for all } x' \in B_\infty(x,\tfrac{\varepsilon_x}{2}),\ \text{for all } j \geq I(x).$$

Hence A4(i) is satisfied. Now the family of balls $B_\infty(x,\frac{\varepsilon_x}{2})$, $x \in X \subset G^c$ forms an open cover for the compact set X and there exists a finite subcover so that :

$$X \subset \cup \{B_\infty(x, \tfrac{\varepsilon_x}{2}) \mid x \in \hat{X}\}$$

where $\hat{X}$ is a <u>finite</u> set. Taking $I_X = \max\{I(x) \mid x \in \hat{X}\}$ we see that :

$$\|x-S(x,j)\|_\infty < 1,\ \psi(S(x,j)) \geq 0 \quad \text{for all } j \geq I_X,\ \text{for all } x \in X$$

so that assumptions A4'(i) is also satisfied.

(ii) Choose any $x \in G^c$ and let $y \in w(x)$ (recall that $w(x)$ is the set of minimizers of (29)). Choose any $\varepsilon > 0$. As in the proof of (i) above the open set

$$Y_\varepsilon(y) \triangleq B_\infty(y, \tfrac{\varepsilon}{2}) \cap (x+T)^o \cap U^o$$

is not empty. Choose $y' \in R^n$ and $\varepsilon' > 0$ such that :

$$B_\infty(y', \varepsilon') \subset Y_\varepsilon(y) .$$

By arguments similar to those in the proof of (i), there exists some integer $N(x) > 0$ such that at least one point $y_j \triangleq (x+\sigma_j) \in Z_j(x)$ lies in $B_\infty(y', \frac{\varepsilon'}{2})$ for all $j \geq N(x)$. Hence $y'_j \triangleq (x'+\sigma_j) \in Z_j(x')$ lies in $B_\infty(y', \varepsilon')$ (and in U^o) for all $x' \in B_\infty(x, \frac{\varepsilon'}{2})$ and for all $j \geq N(x)$. We then have :

$$0 \leq \eta(x') \leq \bar{\eta}(x', j) \leq \|x' - y'_j\|_\infty = \|q_j\|_\infty = \|x - y_j\|$$

$$\leq \|x-y\|_\infty + \|y-y_j\|_\infty \leq \eta(x) + \frac{\varepsilon}{2} \quad \text{for all } x' \in B_\infty(x, \tfrac{\varepsilon'}{2})$$
$$\text{for all } j \geq N(x).$$

By the continuity of $\eta(x)$ there exists some $\delta' > 0$ such that :

$$|\eta(x) - \eta(x')| \leq \frac{\varepsilon}{2} \quad \text{for all } x' \in B_\infty(x, \delta').$$

Taking $\delta_x = \min\{\delta', \frac{\varepsilon'}{2}\} > 0$, we have :

$$|\eta(x') - \bar{\eta}(x', j)| \leq \varepsilon \text{ for all } x' \in B_\infty(x, \delta_x)$$
$$\text{for all } j \geq N(x).$$

The family of balls $B_\infty(x, \delta_x)$, $x \in X$ forms an open cover for the compact set $X \subset G^c$ and there exists a finite subcover, so that :

$$X \subset \cup \{B_\infty(x, \delta_x) \mid x \in \hat{X}\}$$

where $\hat{X}$ is a finite set. Setting $N_X = \max\{N(x) \mid x \in \hat{X}\}$ we have that :

$$|\eta(x) - \bar{\eta}(x,j)| \leq \varepsilon \quad \text{for all } j \geq N_X, \text{ for all } x \in X$$

so that A4 (ii) and A4' (ii) are satisfied.

□

We finally turn out attention to defining procedures that satisfy A4(i) only and hence generate S'(x,j) for algorithm 3 of Chapter II. Recall that :

$$\Delta(x) \triangleq \{y \mid \|x-y\|_\infty = 1\}. \tag{34}$$

Procedure 4

Compute $\tau(j)$ "uniformly spaced" points $Z_j'(x) = \{z_1, \ldots, z_{\tau(j)}\}$ in $\Delta(x)$. Set z_j' equal to the first point $z_i \in Z_j'(x)$ that satisfies $\psi(z_i) > 0$. Then, by performing a linear search along the line segment joining z_j' to x and using a bisection procedure if necessary, compute a point $\hat{z}$ such that $\psi(\hat{z}) > 0$, $\|x-\hat{z}\|_\infty < 1$ (this is always possible by the continuity of ψ). Set $S'(x,j) = \hat{z}$. If no point z_i as above exists, set $S'(x,j) = z_{\tau(j)}$.

Procedure 5

Compute $\tau(j)$ points $Z_j'(x) = \{z_1,\ldots,z_{\tau(j)}\}$ in $\Delta(x)$ spaced so that $\varepsilon_j' \to 0$ as $j \to \infty$ where :

$$\varepsilon_j' \triangleq \max_z \min_r \{ \| z-z_r \|_\infty \mid z \in \Delta(x), \quad r=1,\ldots,\tau(j) \}. \tag{35}$$

Then proceed as in procedure 4 above.

□

Notes

(i) Procedures 4 and 5 that generate $S'(x,j)$ are cheaper computationally than those that generate $S(x,j)$ since no ordering of points is necessary. Also since $\Delta(x)$ is of dimension n-1 (where as x+T is of dimension n), for a given j the value of $\tau(j)$ may be chosen to be smaller (less points are necessary to form a mesh of a certain density in $\Delta(x)$ than in x+T).

(ii) All the points in $Z_j'(x)$ satisfy $\| z-x \|_\infty = 1$ (i.e. lie on the boundary of x+T) and are more likely to violate the specifications than points in the interior of x+T. Hence to estimate if a point lies in G or not, it is advantageous to compute $S'(x,j)$ instead of $S(x,j)$ (given that assumption A5 of Chapter II is satisfied).

Theorem 3

Suppose that assumption A5 of Chapter II is satisfied. Then procedures 4 and 5 that generate $S'(x,j)$ satisfy assumption A4(i) of Chapter II.

Proof

Choose any $x \in \Gamma^c$. Since assumption A5 is satisfied (see Chapter II, relations (27) and (32)), there exists some $y \in \Delta(x)$ such that $\psi(y) > 0$. By the continuity of ψ there exists some $\varepsilon > 0$ such that :

$$\psi(y') > 0 \quad \text{for all } y' \in [B_\infty (y,\varepsilon) \cap \Delta(x)] .$$

By the definition of the sets $Z_j'(x)$, we can choose an integer $I(x)$ large enough so that at least one point $z_j \in Z_j'(x)$ lies in $B_\infty (y,\varepsilon) \cap \Delta(x)$ and hence satisfies $\psi(z_j) > 0$, for all $j \geq I(x)$. The desired result now easily follows from the procedure definitions.

□

III.4 Examples.

Algorithm 2 of Chapter II was programmed in FORTRAN IV and several examples were studied. Six of these examples are presented. The first three are two-dimensional and are intended to highlight the properties of the algorithm. The last two are practical filter design problems and were kindly provided by R. Soin [15].

Example 1

The set $F \subset R^2$ is defined by the constraints :

$$f^1(x) = -x_1^2 + x_2 - 1.5 \leq 0$$

$$f^2(x) = x_1 - 0.5\,(x_2-1)^2 - 1.5 \leq 0$$

$$f^3(x) = -0.2\,x_1^2 - x_2 - 1 \leq 0$$

$$f^4(x) = -x_1 - (2x_2 - 1)^2 - 1 \leq 0$$

$$f^5(x) = x_1^2 + x_2^2 - 13 \leq 0.$$

Note that in the above equations the notation has been modified for

convenience, so that x_1 is the first component of the vector x and so on. Also :

$$T \triangleq \{ t \in R^2 \mid |t^i| \leq 1, \quad i = 1,2 \} .$$

The values of the parameters utilized in the algorithm are $a = \frac{1}{3}$, $\dot{\gamma} = 6$, $\delta = 0.5$ (see (17) of Chapter II). Procedure 1 of section 3 that generates the separator estimates was implemented by simply taking $Z_k(x)$ to be the set of uniformly spaced mesh points in $(x+T)^o$ such that the distance between adjacent points is $\frac{1}{\sigma(k)}$, where :

$$\sigma(k+1) = \sigma(k) + 1 \quad \text{if} \quad x_{k+1} \neq x_k$$

$$\sigma(k+1) = 2\sigma(k) \quad \text{if} \quad x_{k+1} = x_k$$

and $\sigma(0) = 4$. All the vertices and the nominal point were utilized in the definition of the set G_k at each iteration, except when difficulties were encountered (e.g. jamming of the feasibility subalgorithm) in step 1. In these cases all the vertices were removed from T_k and the computation in step 1 was allowed to proceed with no conventional constraints present. Tables 1, 2 and 3 summarize the results of three runs from different initial points ((4,4) for run 1, (-5,-2) for run 2 and (-4,4) for run 3). The progress of the algorithm for these runs is illustrated in figures 1, 2 and 3 respectively. The operation of the algorithm was stopped when $\sigma(k)$ reached the value of 100 and $S(x_k,k)$ did not satisfy the conditions in Step 3. ISUB denotes the number of iterations that the feasibility subalgorithm performed in step 1. Starred points are points at which the "vertex constraints" had to be removed. Note that in run 3, when $k \neq 0$, the feasibility subalgorithm failed to find a point in G_0 (i.e. a

point x such that the vertices of x+T satisfy the specifications) and jammed up at a point $\hat{x}$. Then x_0 was taken to be equal to $\hat{x}$ and the algorithm was allowed to proceed from then on in the normal way. Several trial runs with less "vertex constraints"were performed. As expected, these required more iterations for convergence to occur. If the initial point lies reasonably close to the set G convergence can be obtained with no conventional constraints at all. However, it is recommended to include at least the nominal point in each set T_k, so that the initial point rule operates in the desired manner.

k	σ_k	ISUB	x_k		y_k		$\bar{\delta}(x_k,k)$
			x^1	x^2	y^1	y^2	
0	4	4	0.700	0.461			
1	8		"	"	1.575	0.711	0.125
2	9	1	0.515	0.305			
3	18		"	"			
4	36		"	"			
5	72		"	"	1.502	0.944	0.039
6	100	1	0.488	0.279			

TABLE 1 (Initial point (4,4))

k	σ_k	ISUB	x_k		y_k		$\bar{\delta}(x_k,k)$
			x^1	x^2	y^1	y^2	
0	4	4	-1.553	0.125	-1.803	0.125	0.750
1	5	1	-0.370	0.061	-1.170	0.461	0.200
2	6	1	-0.094	-0.154			
3	12		"	"	-1.010	0.513	0.083
4	13	1	0.075	-0.166	-0.232	-0.012	0.154
5	14	1	0.263	0.052			
6	28		"	"			
7	56		"	"			
8	100		0.263	0.052			

TABLE 2 (Initial point (-5,-2))

k	$\sigma(k)$	ISUB	x_k		y_k		$\bar{\delta}(x_k,k)$
			x^1	x^2	y^1	y^2	
0*	4	6	-1.481	1.815	-0.981	2.565	0.250
1*	5	1	-1.778	1.500	-2.378	0.900	0.400
2	6	2	-0.524	0.669	-1.024	0.502	0.500
3	7	1	0.426	0.226			
4	14		"	"			
5	28		"	"			
6	56		"	"			
7	100		0.426	0.226			

TABLE 3 (Initial point (-4,4))

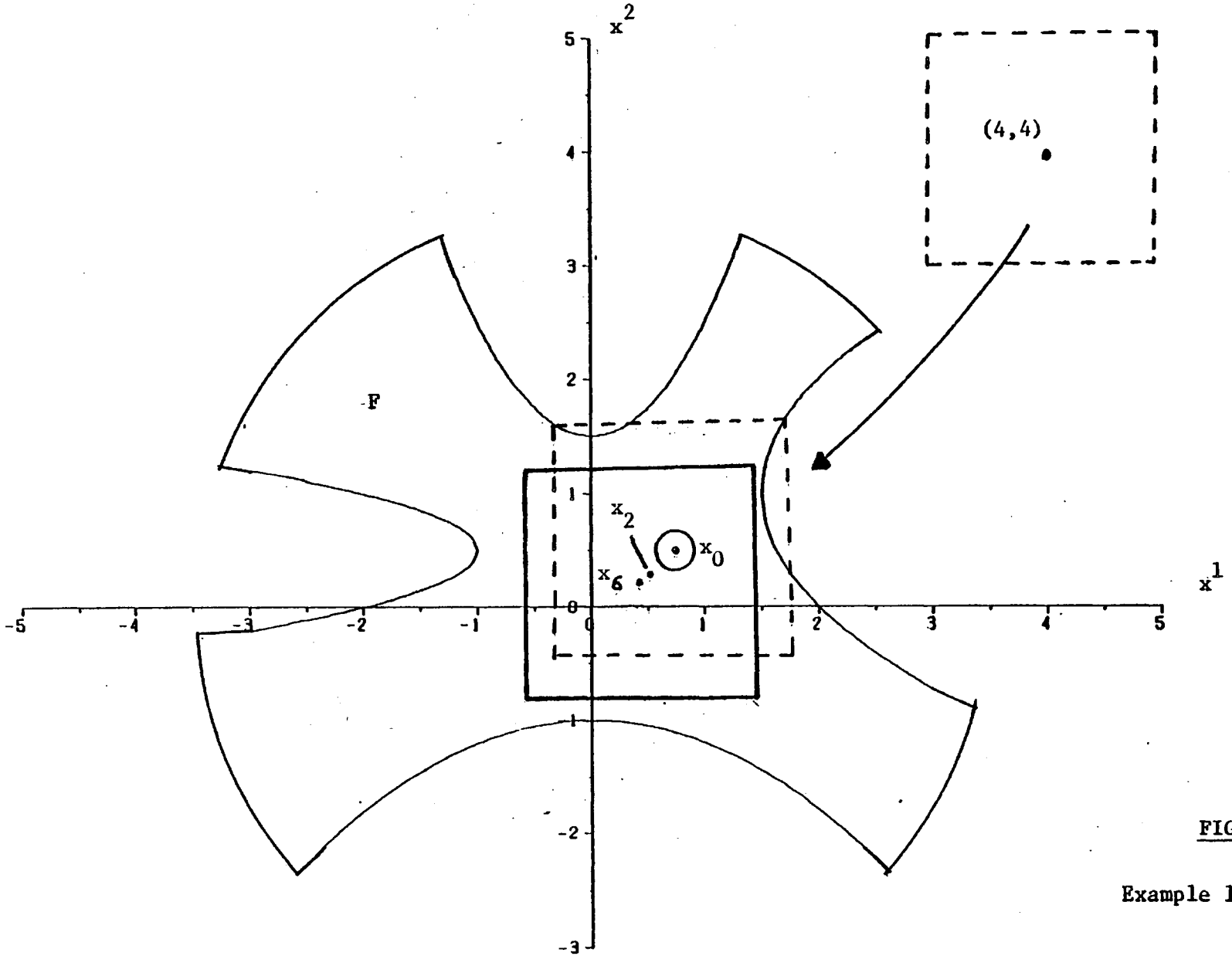
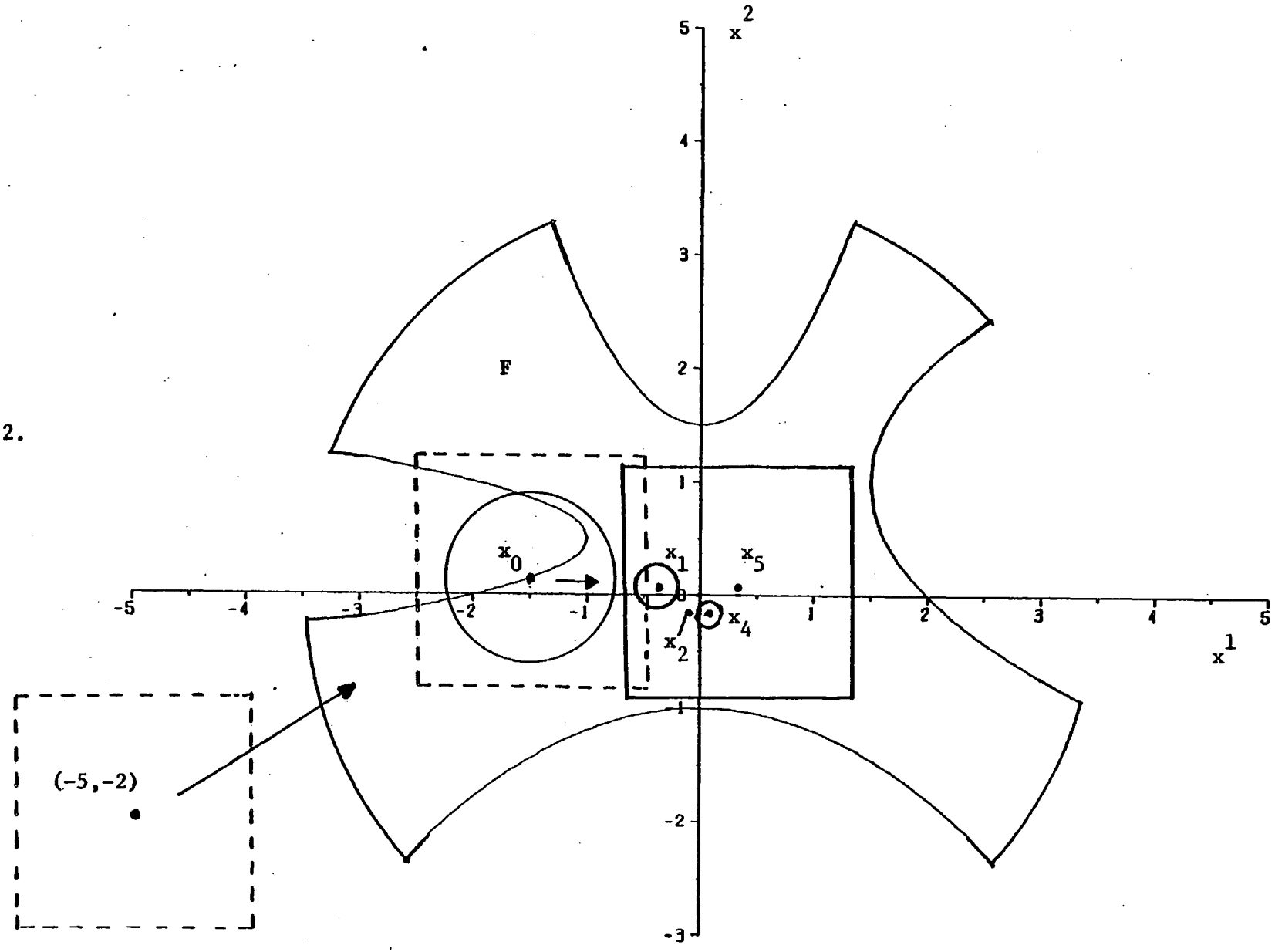


FIGURE 1

Example 1, run 1.

FIGURE 2

Example 1, run 2.



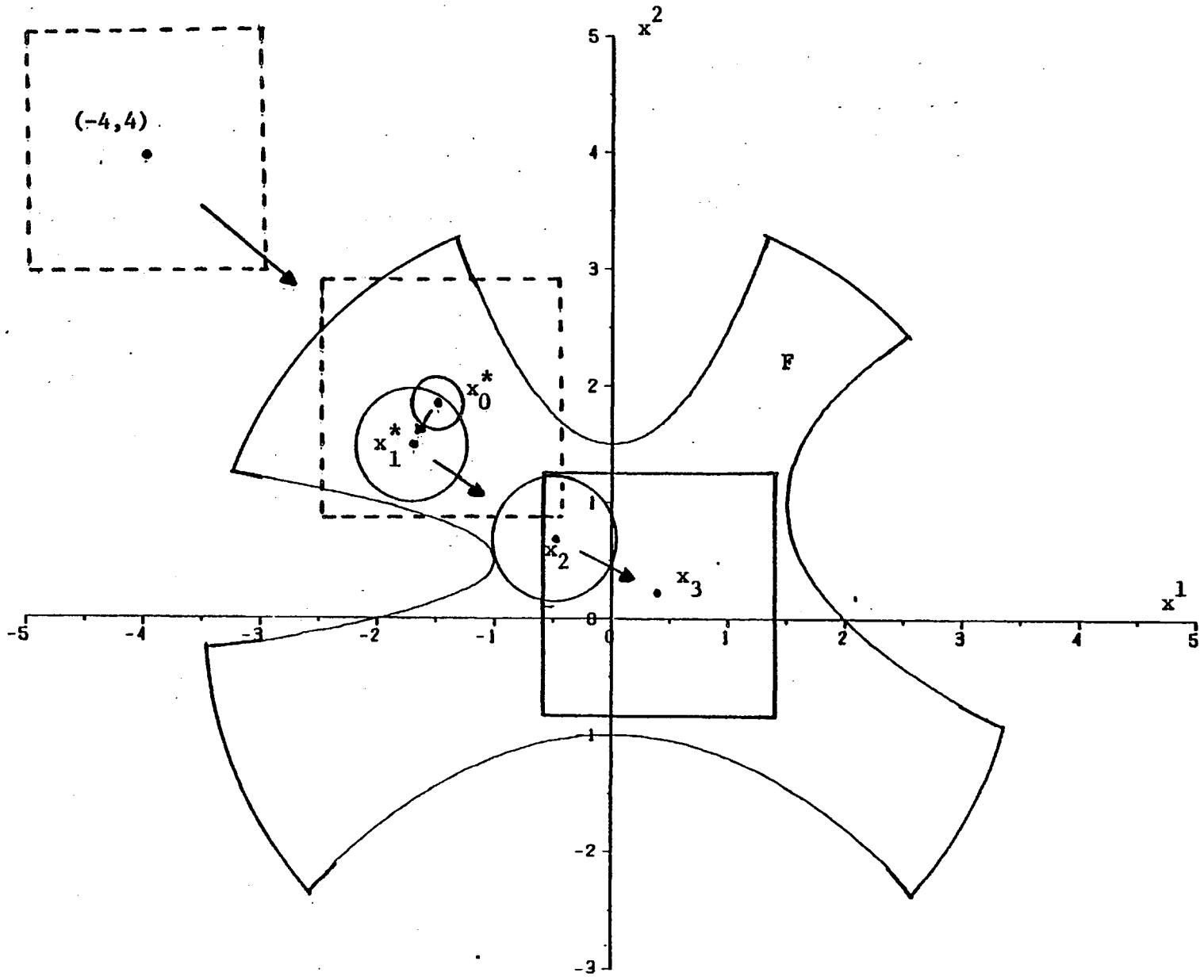


FIGURE 3

Example 1, run 3.

Example 2

The feasible set $F \subset R^2$ is defined by :

$$f^1(x) = x_2 - 2 \leq 0$$

$$f^2(x) = -x_2 - 3 \leq 0$$

$$f^3(x) = -x_1 - 2 \sin 2x_2 \leq 0$$

$$f^4(x) = x_1 - 2\cos 2x_2 - .5 \leq 0.$$

All the rest of the data and conventions are as in Example 1. Tables 4,5 and 6 summarize the results of three runs from different initial points. These runs are illustrated in figures 4,5 and 6. As before, starred points are points that have been computed after the vertex constrains have been removed because of difficulties (e.g. jamming) in step 1. Note that in this example, because T_k contains all the vertices of T each G_k consists of three disjoint subsets one of which contains G. Hence, if the point $x_0 \epsilon G_0$ (computed in the first iteration) lies in one of those subsets that do not contain G, it is probable that jamming will occur in step 1 at a certain iteration and that the "vertex constraints" will have to be removed at some stage. (this happened in all three runs presented).

k	σ(k)	ISUB	x_k		y_k		$\bar{\delta}(x_k,k)$
			x^1	x^2	y^1	y^2	
0	4	8	2.116	-1.992	1.866	-2.242	0.750
1	5	2	3.234	-1.835	3.434	-1.835	0.800
2	6	1	1.890	-1.916	1.557	-2.250	0.667
3*	7	1	2.428	-0.874	3.143	-1.588	0.286
4	8	1	1.837	-0.785			
5	16		"	"			
6	32		"	"			
7	64		"	"			
8	100		1.837	-0.785			

TABLE 4 (Initial point (-3,-4))

k	σ(k)	ISUB	x_k		y_k		$\bar{\delta}(x_k,k)$
			x^1	x^2	y^1	y^2	
0	4	6	2.575	1.000	3.075	1.500	0.500
1	5	1	2.025	0.700	1.825	0.700	0.800
2*	6	1	3.368	0.700	3.535	1.198	0.500
3	7	2	2.700	0.081	1.984	0.800	0.286
4	8	2	3.158	0.000			
5	16		"	"			
6	32		"	"			
7	64		"	"			
8	100		3.158	0.000			

TABLE 5 (Initial point (6,3))

k	$\alpha(k)$	ISUB	x_k		y_k		$\bar{\delta}(x_k,k)$
			x^1	x^2	y^1	y^2	
0	4	5	2.286	-1.994	1.786	-2.494	0.500
1	5	1	3.345	-1.870	3.545	-1.870	0.800
2	6	1	2.002	-1.881	1.668	-2.214	0.667
3*	7	1	2.592	-0.890	3.164	-1.462	0.429
4	8	2	1.842	-0.785			
5	16		"				
6	32		"				
7	64		"				
8	100		1.842	-0.785			

TABLE 6 (Initial point (5,-4))

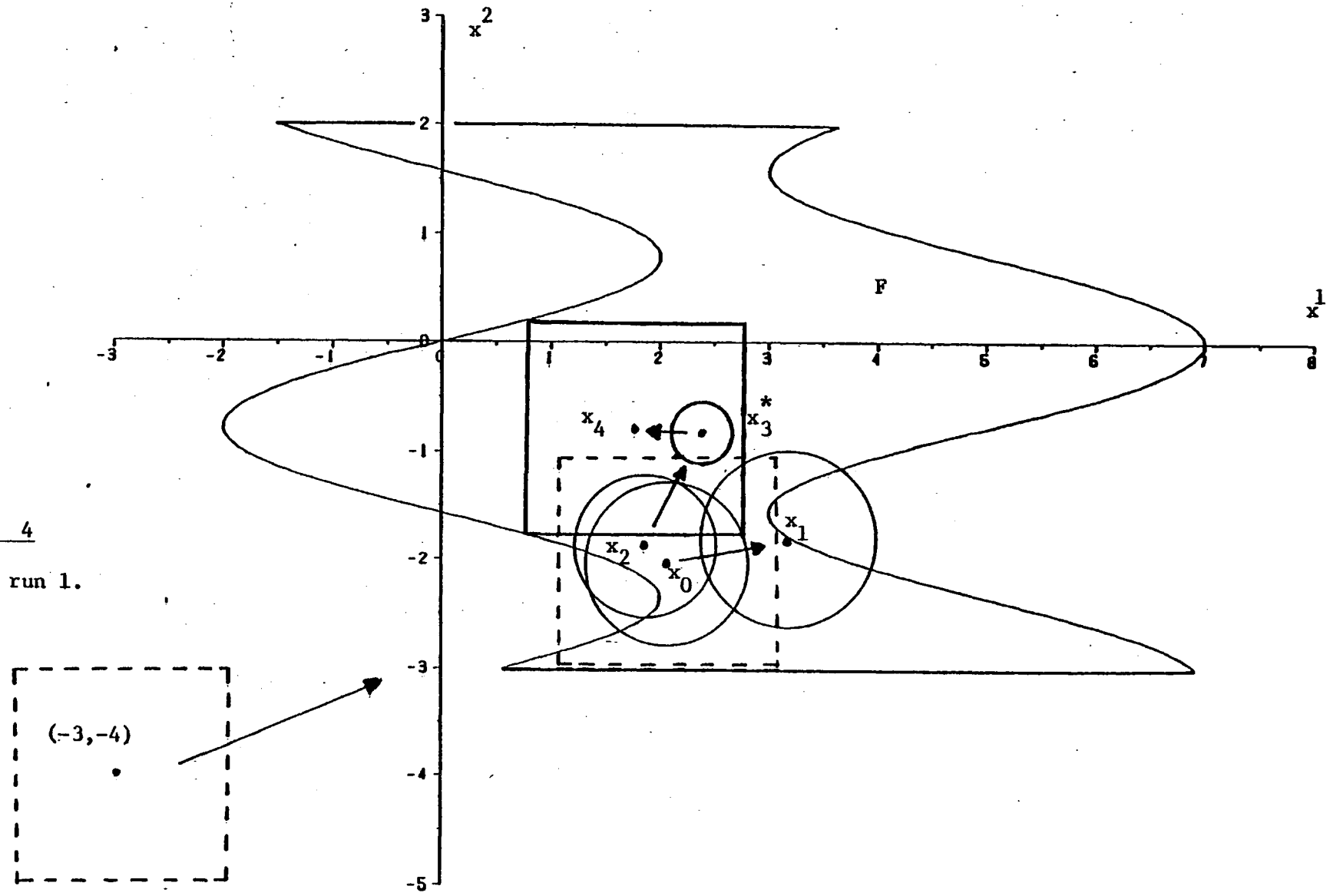


FIGURE 4

Example 2, run 1.

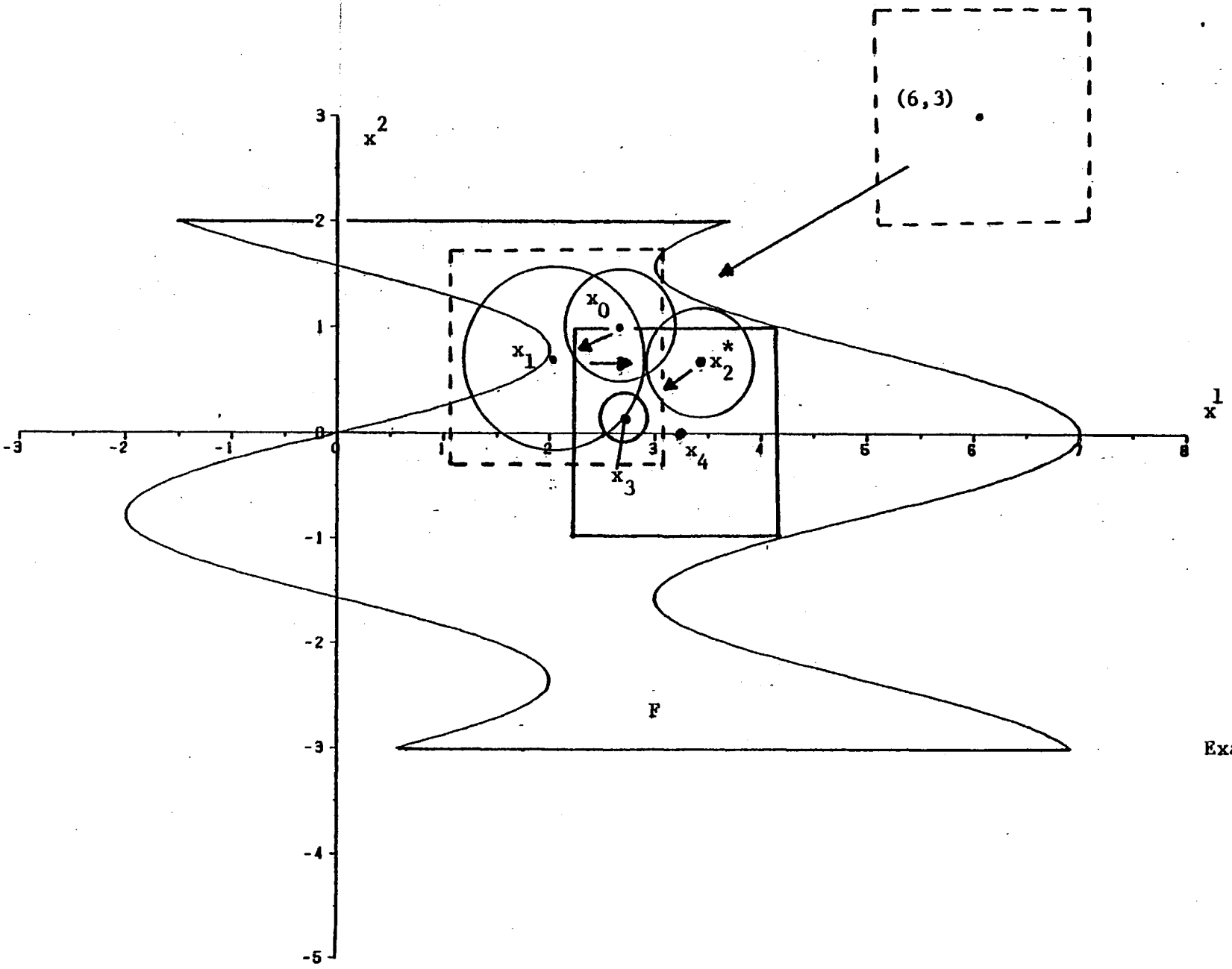


FIGURE 5

Example 2, run 2.

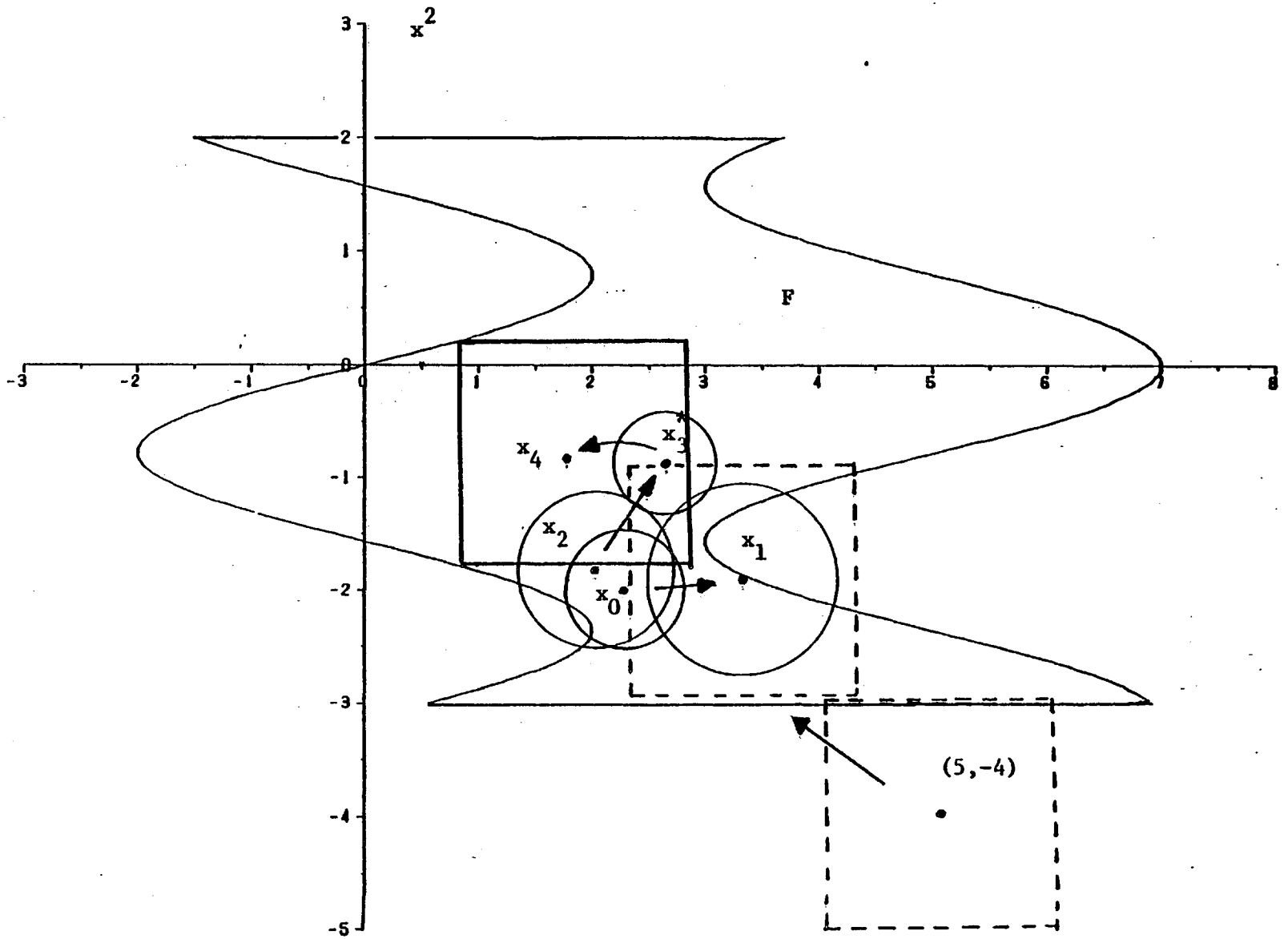


FIGURE 6

Example 2, run. 3.

Example 3

The set $E \subseteq R^2$ is defined by the constraints :

$$f^1(x) = x_1 - 4 \leq 0$$

$$f^2(x) = -x_1 - 2 \leq 0$$

$$f^3(x) = -0.5\, x_1 \sin 2x_1 + x_2 - 3.9 \leq 0 \qquad (3.9 \doteq \frac{5\pi}{4})$$

$$f^4(x) = 1.5 \cos 2x_1 - x_2 \leq 0$$

$$f^5(x) \quad -x_1 - (x_2-2)^2 - 0.5 \leq 0.$$

Tables 7,8 and 9 summarize the results of three runs which are illustrated in Figures 7,8 and 9 respectively. Note that removal of the "vertex constraints" was necessary at some stage in all three runs.

k	$\sigma(k)$	ISUB	x_k		y_k		$\bar{\delta}(x_k,k)$
			x^1	x^2	y^1	y^2	
0*	4	6	-2.749	-0.225	-2.749	-0.225	1.000
1*	5	1	-1.590	1.015	-1.590	1.015	1.000
2	6	2	-0.154	2.170	-0.655	1.670	0.500
3	7	1	0.539	2.669			
4	14		"	"			
5	28		"	"			
6	56		"	"			
7	100		0.539	2.669			

TABLE 7 (Initial point (-4,-2))

k	σ(k)	ISUB	x_k		y_k		$\bar{\delta}(x_k,k)$
			x^1	x^2	y^1	y^2	
0*	4	8	-2.147	4.178	-2.147	4.178	1.000
1*	5	1	-0.501	4.500	-0.501	4.500	1.000
2	6	13	0.737	2.589			
3	12		"	"			
4	24		"	"			
5	48		"	"			
6	96		"	"			
7	100		0.737	2.589			

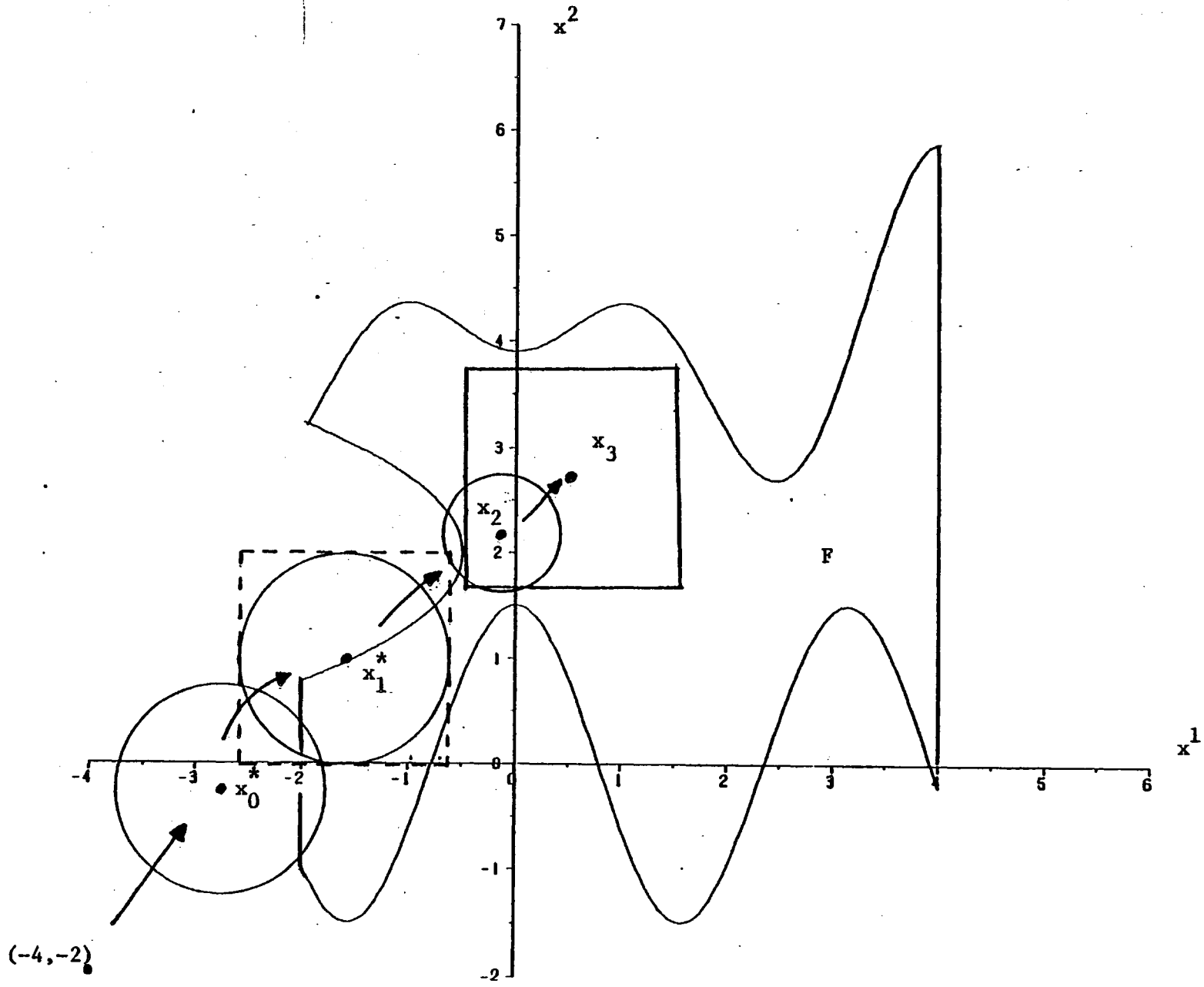
TABLE 8 (Initial point (-4,4))

k	σ(k)	ISUB	x_k		y_k		$\bar{\delta}(x_k,k)$
			x^1	x^2	y^1	y^2	
0	4	10	2.994	1.639	2.994	1.389	0.750
1	5	2	2.670	2.708	2.470	2.708	0.800
2	6	2	2.993	1.889	2.993	1.389	0.500
3*	7	1	4.522	2.726	4.522	2.726	1.000
4	8	8	2.815	1.371	2.940	1.246	0.875
5	9	1	2.271	2.578	2.271	2.800	0.778
6*	10	1	0.849	1.277	0.449	0.877	0.600
7	11	2	0.820	2.449			
8	22		"	"	0.002	1.495	0.045
9	23	2	0.718	2.623			
10	46		"	"			
11	92		"	"			
12	100		0.718	2.623			

TABLE 9 (Initial point (6,7))

FIGURE 7

Example 3, run 1.



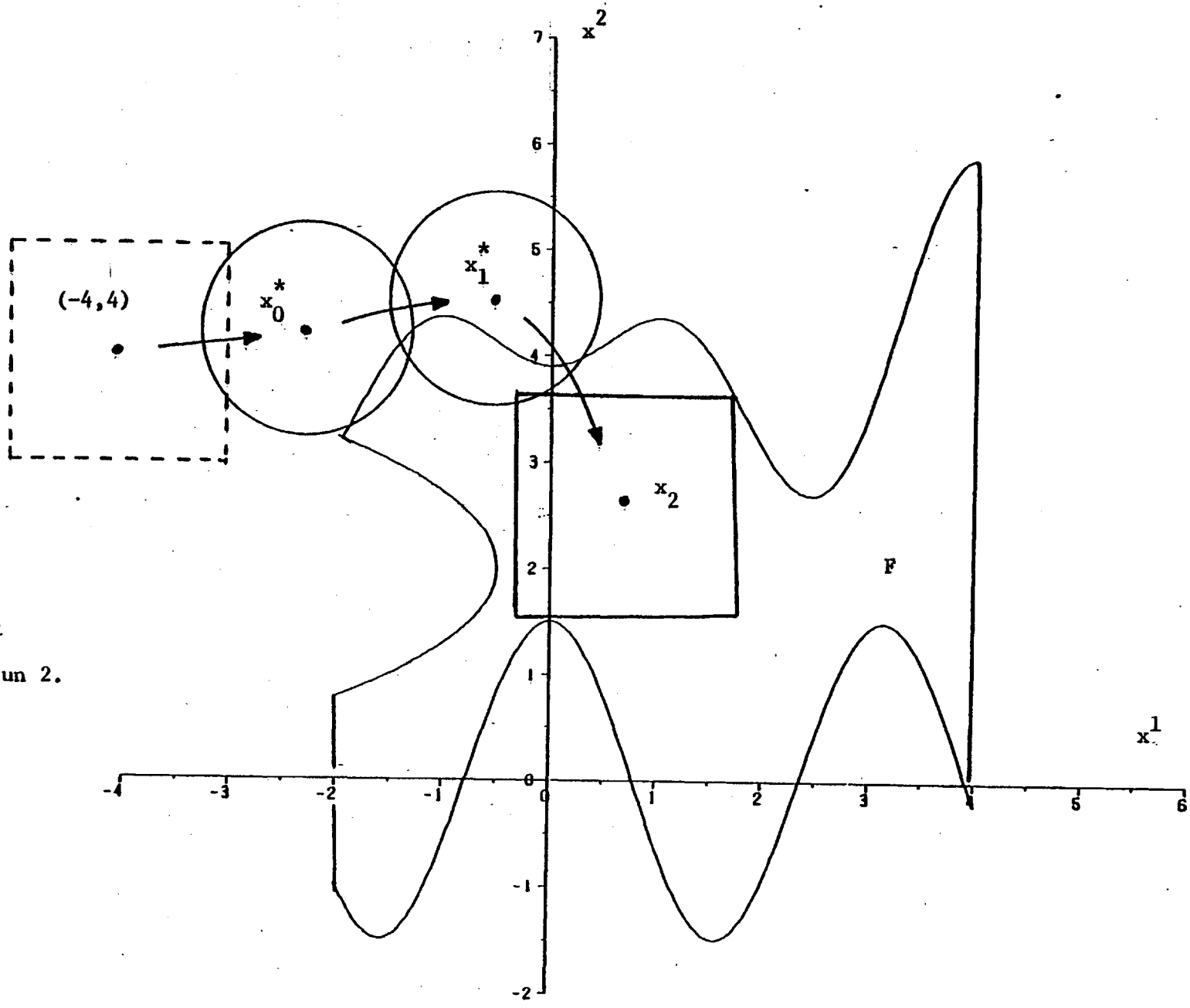


FIGURE 8

Example 3, run 2.

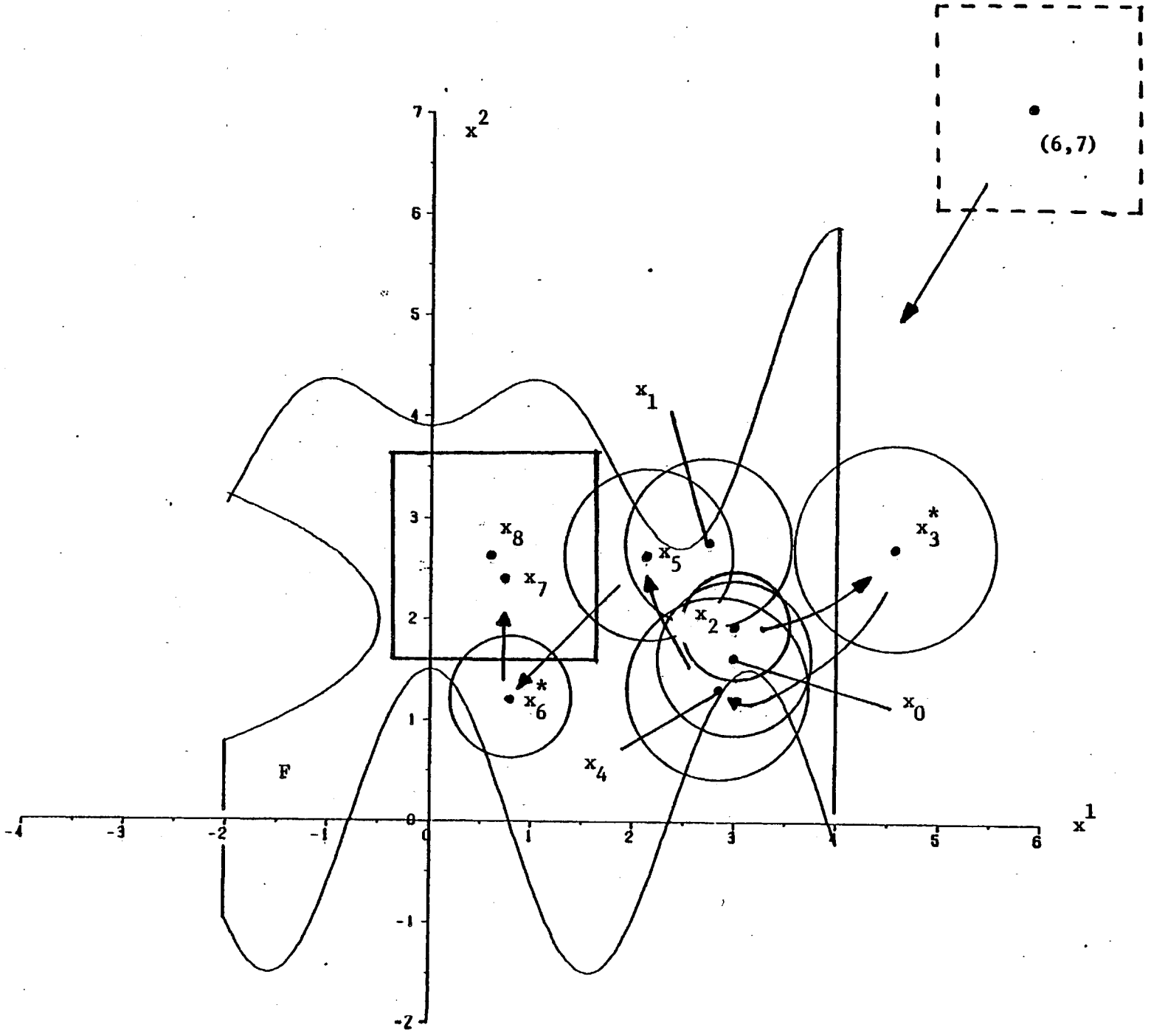


FIGURE 9

Example 3, run 3.

Example 4.

The set $F \subset R^3$ is defined by the constraints :

$$f^1(x) = x_1 - x_2^2 - 1.2 \leq 0$$

$$f^2(x) = -2x_1^2 + x_2 \leq 0$$

$$f^3(x) = -x_1 - 0.5\,(x_3-1)^2 - 1 \leq 0$$

$$f^4(x) = x_1^2 + x_2^2 + x_3^2 - 8 \leq 0.$$

Tables 10 and 11 summarize the results of two runs.

k	σ(k)	ISUB	x_k			y_k			$\bar{\delta}(x_k, k)$
			x^1	x^2	x^3	y^1	y^2	y^3	
0	4	12	-0.310	-0.699	0.835	-1.060	-1.499	0.835	0.250
1	5	1	-0.006	-0.394	0.835	-0.006	-0.006	-0.435	0.600
2	6	5	0.000	-1.085	0.606				
3	12		"	"	"				
4	24		"	"	"				
5	48		"	"	"				
6	96		"	"	"				
7	100		0.000	-1.085	0.606				

TABLE 10 (Initial point (-3,3,3))

k	σ(k)	ISUB	x_k			y_k			$\bar{\delta}(x_k, k)$
			x^1	x^2	x^3	y^1	y^2	y^3	
0*	4	6	1.511	-0.057	0.000	1.511	-0.057	0.000	1.000
1	5	9	0.274	-0.585	0.943	0.074	0.015	0.343	0.400
2	6	4	0.000	-1.022	0.684				
3	12		"	"	"				
4	24		"	"	"				
5	48		"	"	"				
6	96		"	"	"				
7	100		0.000	-1.022	0.684				

TABLE 11 (Initial point (2,0,-2))

Example 5

This is a three-dimensional low-pass filter design example normalized in frequency [8]. The circuit details and specifications are shown in the appendix. The parameter values shown on the circuit diagram are given typical values and were used as the initial point. Procedure 2 of section 3 was employed to implement step 2. The points constituting each set $Z_j(x)$ were generated by a pseudo-random number generator. The truncation function defined in section 3 was utilized and the linear searching technique described was employed to improve the separator estimates. All the nominal points generated by the algorithm were required to lie in F, but no other conventional constraints were introduced. The values of the parameters used are $\gamma = 1$, $\phi = 0.8$, $a = \frac{1}{3}$. Table 12 summarizes the results of four runs with different parameter tolerances. The initial yields were estimated by Monte Carlo analysis involving 100 points, where as the final yields by Monte Carlo analysis involving at least 1000 points. The percentage tolerances shown are with respect to the final point. IT denotes the number of iterations and N the total number of circuit analyses performed in step 2 of the algorithm. Tables 13, 14 and 15 show in detail the progress of the algorithm for runs 2, 3 and 4 respectively. Almost all the subproblems in step 1 were solved in one iteration. N(k) denotes the number of circuit analyses performed in step 2 of the algorithm at iteration k. The yields at intermediate iterations were estimated by Monte Carlo analysis involving $\tau(k)$ points. It can be observed that very few circuit analyses are necessary for the separator estimations at low yield points. The number of circuit analyses increases as x approaches G. Hence at high yield points it is necessary to employ the artifice that takes advantage of the overlap of successive tolerance regions (see section 3) to ensure an efficient implementation. (This artifice was not employed in the examples presented). An interactive

program allowing the specification of the truncation function $\tau(k)$ as the algorithm progresses would also be desirable. Note that in run 4 the tolerances of the parameters are near their maximum values (see Bandler [8]).

Run	TOLERANCES			Initial yield	Final yield	IT	N
	L_1(H)	L_2(H)	C(F)				
1	0.085 5.1%	0.085 5.1%	0.055 5.0%	64%	100.0%	1	9
2	0.100 6.0%	0.100 6.0%	0.070 6.3%	62%	99.8%	2	188
3	0.120 7.1%	0.120 7.0%	0.075 6.9%	61%	99.4%	3	217
4	0.200 10.5%	0.200 10.6%	0.075 7.7%	59%	98.7%	4	162

TABLE 12

k	L_1(H)	L_2(H)	C(F)	$\bar{\delta}(x_k,k)$	Yield	$\tau(k)$	N(k)
0	1.600	1.600	1.050	0.700	64.0%	100	12
1	1.671	1.688	1.118	0.138	98.5%	200	176
2	1.658	1.674	1.106		99.8%		

TABLE 13 (RUN 2)

k	L_1(H)	L_2(H)	C(F)	$\bar{\delta}(x_k,k)$	Yield	τ(k)	N(k)
0	1.600	1.600	1.050	0.737	61.0%	100	14
1	1.689	1.710	1.135	0.411	88.5%	200	53
2	1.639	1.663	1.103	0.247	97.7%	300	150
3	1.684	1.707	1.086		99.4%		

TABLE 14 (Run 3)

k	L_1(H)	L_2(H)	C(F)	$\bar{\delta}(x_k,k)$	Yield	τ(k)	N(k)
0	1.600	1.600	1.050	0.831	59.0%	100	19
1	1.735	1.775	1.174	0.979	43.0%	200	14
2	1.872	1.854	1.041	0.449	90.7%	300	47
3	1.789	1.778	0.979	0.430	93.2%	400	82
4	1.891	1.878	0.987		98.7%		

TABLE 15 (Run 4)

Example 6

This is a seven-dimensional high-pass filter design example. The circuit details and specifications are shown in the appendix. Table 16 summarizes the results of four runs with different parameter tolerances and Table 17, 18 and 19 the progress of the algorithm for runs 1,2 and 4 respectively. All the conventions and algorithm parameters are as in example 5 except that no conventional constraints at all were employed in this example.

Run	TOLERANCES x_1 (nF)	x_2 (nF)	x_3 (H)	x_4 (nF)	x_5 (nF)	x_6 (H)	x_7 (nF)	Initial Yield	Final Yield	IT	N
1	0.33	1.11	0.12	0.33	2.70	0.09	0.45	86%	99.7%	2	64
	3.0%	3.0%	3.1%	3.0%	3.0%	3.0%	3.0%				
2	0.47	1.44	0.16	0.41	3.40	0.13	0.60	75%	98.9%	4	198
	4.1%	4.0%	4.1%	3.9%	3.8%	4.2%	4.0%				
3	0.72	1.85	0.20	0.50	4.50	0.15	0.70	62%	98.7%	4	164
	5.8%	5.1%	5.1%	4.9%	5.0%	5.3%	4.9%				
4	0.76	2.10	0.22	0.61	5.20	0.16	0.81	57%	98.6%	4	166
	6.0%	5.8%	5.7%	6.0%	5.8%	5.7%	5.7%				

TABLE 16

k	x_1 (nF)	x_2 (nF)	x_3 (H)	x_4 (nF)	x_5 (nF)	x_6 (H)	x_7 (nF)	$\bar{\delta}(x_k,k)$	Yield	$\tau(k)$	N(k)
0	11.00	37.00	4.00	11.00	90.00	3.00	15.00	0.585	86.0%	100	29
1	10.67	36.54	3.95	10.66	89.48	2.96	14.65	0.202	96.0%	200	35
2	10.82	36.70	3.94	10.51	89.79	2.97	14.80		99.7%		

TABLE 17 (RUN 1)

k	x_1(nF)	x_2(nF)	x_3(H)	x_4(nF)	x_5(nF)	x_6(H)	x_7(nF)	$\bar{\delta}(x_k,k)$	Yield	$\tau(k)$	N(k)
0	11.00	37.00	4.00	11.00	90.00	3.00	15.00	0.322	75.0%	100	21
1	11.21	37.23	4.02	10.79	89.55	3.03	15.21	0.664	64.0%	200	9
2	11.70	36.50	3.97	10.30	88.68	3.08	14.71	0.279	95.3%	300	67
3	11.43	36.15	3.93	10.57	88.24	3.10	14.99	0.179	97.5%	400	101
4	11.58	35.96	3.91	10.42	88.51	3.09	14.84		98.9%		

TABLE 18 (RUN 2)

k	x_1(nF)	x_2(nF)	x_3(H)	x_4(nF)	x_5(nF)	x_6(H)	x_7(nF)	$\bar{\delta}(x_k,k)$	Yield	$\tau(k)$	N(k)
0	11.00	37.00	4.00	11.00	90.00	3.00	15.00	0.459	57.0%	100	8
1	11.61	37.62	3.93	10.39	91.28	2.93	14.39	0.449	82.5%	200	3
2	12.00	37.14	3.87	10.77	90.73	2.89	14.00	0.358	89.0%	300	53
3	12.40	36.67	3.92	10.37	89.90	2.84	14.40	0.220	96.2%	400	102
4	12.67	36.34	3.89	10.11	90.38	2.82	14.14		98.6%		

TABLE 19 (Run 4)

III.5 Discussion.

As the numerical examples reveal the cut map algorithm (algorithm 2 of Chapter II)performs well, mainly because of the following reasons. Firstly, the feasibility subalgorithm, [22] , works well finding points in the interior of each constraint set very efficiently. Secondly, because the constraints that define the cuts are very simple (much simpler than the circuit constraints), the subproblem in step 1, is usually solved in one or two iterations. The initial point rule adopted causes the very desirable "centering effect" (i.e. encourages the generation of new points that lie nearer the set G and have smaller separator values and substantially reduces the total number of iterations), In addition, the cut dropping scheme ensures that the complexity of the subproblems remains low throughout the operation of the algorithm. The inclusion of conventional constraints either involving the vertices or relating to the magnitudes of the parameters results in better convergence properties since it ensures that each cut is a good approximation to the set G. These constraints can be removed if problems are created in step 1 since they are not necessary for the convergence proofs in Chapter II. When a good initial point lying reasonably close to the set G is used, it is not necessary to employ any conventional constraints at all, because the "centering effect" of the initial point-rule usually causes the algorithm to converge rapidly. In such cases the derivatives of the constraints are not utilized and the algorithm works by performing "pass-fail" tests only in step 2. Because the feasibility subproblems are easily and efficiently solved, most of the computational effort is dedicated to performing "pass-fail" tests. Hence, skilful programming dramatically increases the efficiency of the implementation. By adopting the techniques discussed in Section 3 in an interactive program it should be possible to obtain further

improvements on the results presented. The points that constitute the sets $Z_j(x)$ should be generated either so that they form a uniform mesh in x+T (procedure 1 of section 3) or by employing a pseudo-random number generator (Procedure 2 of section 3). Some loss of accuracy in the estimation of the separators was (as expected) detected when the second method was employed. However, since the programming complexity of creating uniform meshes of points is great for high dimensional problems, procedure 2 of section 3 is more appropriate for practical applications.

Because the separator estimates are always smaller than their exact values and the feasibility subalgorithm finds points in the interior of each set, the algorithm is not sensitive to the accuracy of the separator estimates. Combining the above with the fact that at low yield points few pass-fail tests (circuit analyses) are required for the generation of these estimates, we conclude that large increases in yield can be efficiently obtained. In Chapter VI the cut map algorithm is compared with the rest of the methods in the literature.

CHAPTER IV

CUT MAP ALGORITHMS FOR THE TOLERANCE-TUNING PROBLEM

IV.1 Introduction.

In Chapter II cut map algorithms for the pure tolerance problem have been proposed. Their purpose is to generate nominal values for the components of a certain system so that the design specifications are met whatever the actual values are, as long as they fall in a certain given tolerance region. However, in practice it sometimes turns out that very tight tolerances are necessary for a solution to the pure tolerance problem to exist. Very tight tolerances may be very costly or even impossible to produce. Hence it is standard practice to permit post-manufacture tuning or trimming of certain components, so that acceptable component tolerances can be adopted. The tolerance-tuning problem is to find a set of nominal values for the components so that whatever the actual values are - as long as they fall in a certain tolerance region - the specifications can be met by tuning. In this chapter the ideas of chapter II will be extended to problems in which tuning is also present and algorithms for the tolerance-tuning problem will be proposed.

Suppose, as before, that $x \in R^n$ is the vector of nominal values of the components. Also suppose that the first ℓ parameters can be tuned or trimmed. We now define the continuous map $r : R^{\ell} \to R^n$ by:

$$r^i(q) \triangleq \begin{cases} q^i & \text{if } i \leq \ell \\ 0 & \text{if } \ell < i \leq n \end{cases}, \quad q \in R^{\ell}. \tag{1}$$

The tolerance region is defined as before (after normalization) by :

$$T \triangleq \{ t \in R^n \mid |t^i| \leq 1, \quad i = 1,\ldots,n \}. \tag{2}$$

Let the tuning region be defined by:

$$Q \triangleq \{ q \in R^{\ell} \mid -\alpha_i \leq q^i \leq \beta_i \} = \{ q \mid -\alpha \leq q \leq \beta \}. \tag{3}$$

Hence, as before, the maximum deviation of the ith parameter (i=1,...,n) from its nominal value is unity. Also the ith parameter (i=1,...,ℓ) can be tuned by an amount specified by α_i and β_i. Usually $\alpha_i = \beta_i > 0$. If either α_i or β_i is zero, we have one way or irreversible tuning or trimming; an example is laser beam resistor trimming in the manufacture of high quality integrated circuits. The tolerance-tuning problem can now be stated as:

$P_{T,Q}$: Find a point in the set G defined by:

$$G \triangleq \{ x \in R^n \mid \text{for all } t \in T \text{ there exists some } q \in Q \text{ such that } f^j(x+t+r(q)) \leq 0, \quad j=1,\ldots,m \}. \tag{4}$$

The functions $f^j: R^n \to R$, j=1,...,m specify as before the inequality constraints that correspond to the design specifications. Let :

$$\psi(x) \triangleq \max \{ f^j(x) \mid j=1,\ldots,m \} \tag{5}$$

$$\hat{\psi}(x,q) \triangleq \psi(x+r(q)). \tag{6}$$

Note that $\hat{\psi}: R^n \times R^{\ell} \to R$ is a continuous function by the continuity of ψ and r. Also note that G can be expressed in the following way :

$$G = \{x \in R^n \mid \text{for all } t \in T \text{ there exists some } q \in Q \text{ such that } \hat{\psi}(x+t,q) \leq 0\}. \quad (7)$$

Bandler [8] tackles the tolerance-tuning problem by distinguishing between effectively toleranced and effectively tuned components, as discussed in Chapter I. All his results are based on the assumption of one-dimensional convexity. Note that it is precisely in strongly non-convex cases that one hopes to obtain large increases in the component tolerances by the introduction of tuning. Polak and Sangiovanni-Vincetelli discuss the tolerance-tuning problem in [18] and present algorithms that employ non-differentiable optimization techniques. Finally, Polak in [19] presents an implementable outer approximations algorithm for the general engineering problem when tuning is permitted.

It is the purpose of this Chapter to present specialized cut map algorithms for the infinite dimensional problem $P_{T,Q}$ which possess the following features :

(i) They construct sequences of points whose accumulation points (if they exist) are solutions of the non-convex problem $P_{T,Q}$.

(ii) They have established convergence properties.

(iii) They are directly implementable and are suitable for interactive computer-aided design.

In section 2 theoretical results that lead to the statement of a conceptual algorithm for the problem $P_{T,Q}$ are presented. An

implementable algorithm is proposed in section 3. Implementation details, numerical examples and conclusions can be found in Chapter V.

II.2 A conceptual algorithm for the tolerance-tuning problem.

Let the sets F (the set of feasible nominal designs) and U be defined in the same way as in Chapter II :

$$F \triangleq \{x \in R^n \mid \psi(x) \leq 0\} \tag{8}$$

$$U \triangleq \{x \in R^n \mid \psi(x) \geq 0\} . \tag{9}$$

Also let :

$$Y \triangleq \{ x \in R^n \mid \hat{\psi}(x,q) \geq 0 \quad \text{for all } q \in Q\} . \tag{10}$$

As it will be shown, the interior of Y consists of all those nominal points which cannot be tuned to satisfy the specifications (i.e. to lie in F). Figure I illustrates the definitions of the sets F,U,G and Y. The following assumptions are made:

A1: The functions f^j, $j=1,\ldots,m$ are continuous.

A2: The sets F and G are non-empty and F is equal to the closure of its interior.

A3: The interior of F satisfies:

$$F^o = \{ x \in R^n \mid \psi(x) < 0\} . \tag{11}$$

A4: The set Y is equal to the closure of its interior.

The first three assumptions are the same as those of Chapter II. Note that A2 and A3 do not imply A4. (Consider $F = \{ x \in R^2 \mid -|x^1| + 1 \leq 0\}$ and $Q = \{q \in R \mid -1 \leq q \leq 1\}$ so that $U = \{ x \in R^2 \mid |x^1| \leq 1\}$ and $Y = \{ x \in R^2 \mid x^1 = 0\}$; clearly Y has no interior).

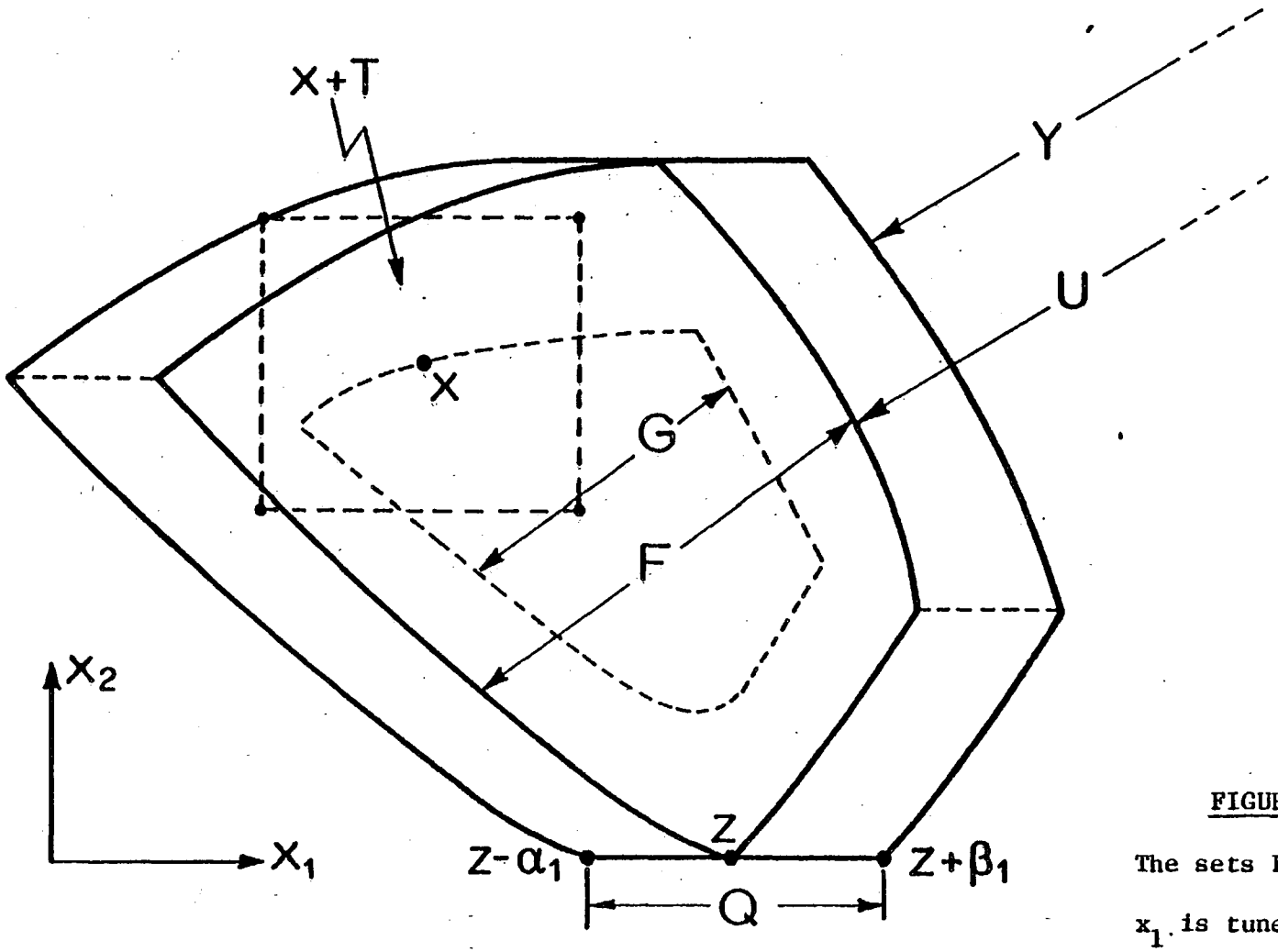


FIGURE I

The sets F, U, Y and G when x_1 is tuned. The arrows point to the set boundaries.

Now consider the function $\hat{\eta}: R^n \to R$ defined by:

$$\hat{\eta}(x) \triangleq \min \{ \|y-x\|_\infty \mid y \in Y \} \tag{12}$$

Since Y is a closed set, $\| y-x \|_\infty$ is bounded from below for fixed x and $\| x-y \|_\infty \to \infty$ as $\| y \|_\infty \to \infty$, the minimum exists. Note that $\hat{\eta}(x)$ is defined analogously to $\eta(x)$ in Chapter II except that the set U is replaced by the set Y. Let :

$$\hat{\delta}(x) \triangleq 1 - \hat{\eta}(x). \tag{13}$$

Proposition 1

The function $\hat{\delta}(x)$ is a separator for the problem $P_{T,Q}$.

Proof

(i) $\hat{\delta}(x) > 0$ for all $x \in G^c$.

Choose any $x \in G^c$. Then there exists some $y \in (x+T)$ such that :

$$\hat{\Psi}(y,q) > 0 \quad \text{for all } q \in Q. \tag{14}$$

Using the continuity of $\hat{\Psi}$, the fact that x+T is equal to the closure of its interior and the compactness of Q, there exists some $y' \in (x+T)^o$ that satisfies (14) (i.e. lies in Y). Hence :

$$\hat{\eta}(x) \leq \| x-y' \| < 1 \Rightarrow \hat{\delta}(x) > 0.$$

(ii) $\hat{\delta}(x)$ is continuous.

This follows by exactly the same arguments as in the proof of (ii) in proposition 1 of Chapter II.

□

We have thus obtained a separator for the problem $P_{T,Q}$ by simply extending the ideas that led to the definition of the separator for the problem P_T. The next step is to define suitable cut maps.

Proposition 2

The set U satisfies :

(i) $U = \overline{(U^o)}$

(ii) $U^o = \{x \in R^n \mid \psi(x) > 0\}$.

Proof

See proposition 2 of Chapter II.

□

Proposition 3

Let :

$$W = \cup \{B_\infty (y,1) \mid y \in Y\}$$

$$W' = \cup \{B_\infty (y,1) \mid y \in Y^o\}$$

Then W=W'.

Proof

The proof is exactly the same as that of proposition 3 of Chapter II except that Y replaces U. ($Y = \overline{(Y^o)}$ by assumption A4).

□

Proposition 4

The interior of Y satisfies :

$$Y^o = \hat{Y} \overset{\Delta}{=} \{ y \in R^n \mid \hat{\psi}(y,q) > 0 \text{ for all } q \in Q \} .$$

Proof

(i) $\hat{Y} \subset Y^o$

Suppose that $y \in \hat{Y}$, then $\hat{\psi}(y,q) > 0$ for all $q \in Q$. By the continuity of $\hat{\psi}$ and the compactness of Q, there exists some $\varepsilon > 0$ such that :

$$\hat{\psi}(y',q) \geq 0 \text{ for all } y' \in B_\infty(y,\varepsilon) \text{ and all } q \in Q.$$

Hence $y' \in Y$ for all $y' \in B_\infty(y,\varepsilon)$, so that $y \in Y^o$.

(ii) $Y^o \subset \hat{Y}$

Suppose that $y \in Y^o$ and that (contrary to what is to be proven) $\hat{\psi}(y,\hat{q}) = 0$ for some $\hat{q} \in Q$. By proposition 2 (ii) and (6) we have :

$$\psi(y+r(\hat{q})) = 0 \Rightarrow y + r(\hat{q}) \notin U^o. \tag{15}$$

Since $y \in Y^o$ there exists some $\varepsilon > 0$ such that :

$$B_\infty (y,\varepsilon) \subset Y.$$

Hence:

$$\hat{\psi}(y',q) \geq 0 \text{ for all } y' \in B_\infty(y,\varepsilon) \text{ and all } q \in Q$$

so that :

$$\hat{\psi}(y',\hat{q}) = \psi(y'+r(\hat{q})) \geq 0 \quad \text{for all } y' \in B_\infty(y,\varepsilon) .$$

Hence:

$$\psi(z) \geq 0 \quad \text{for all } z \in B_\infty(y+r(\hat{q}),\varepsilon)$$

which implies that $y+r(\hat{q}) \in U^o$, a contradiction to (15). Hence the proposition is true.

□

Proposition 5

$$G = \cap\{B_\infty(y,1)^c \mid y \in Y\} = W^c. \tag{16}$$

Proof

We prove that $G = (W')^c$ and use proposition 3.

(i) $G \subset (W')^c$

If this is not true there exists some $x \in (G \cap W')$ so that there exists some $y \in Y^o$ satisfying :

(a) For all $t \in T$ there exists some $q \in Q$ such that

$$\hat{\psi}(x+t,q) \leq 0 \qquad (\text{as } x \in G)$$

(b) $$\|x-y\|_\infty < 1 \qquad (\text{as } x \in W')$$

(c) $\hat{\psi}(y,q) > 0$ for all $q \in Q$ (by proposition 4).

Let $\hat{t} = y - x$, so that $\hat{t} \in T$ by (b). Then :

$$\hat{\psi}(y,\hat{q}) = \hat{\psi}(y+x-x,\hat{q}) = \hat{\psi}(x+\hat{t},\hat{q}) \leq 0 \text{ for some } \hat{q}\in Q. \text{ by (a).}$$

But this is a contradiction to (c) so that (i) is true.

(ii) $\underline{(W')^c \subset G}$

If this is not true, there exists some $x\in (W')^c \cap G^c$. Hence there exists some $\hat{t}\in T$ such that $\hat{\psi}(x+\hat{t},q) > 0$ for all $q\in Q$. By the continuity of $\hat{\psi}$ and the compactness of Q, there exists some $\gamma\in (0,1)$ such that :

$$\hat{\psi}(x+\gamma\hat{t},q) > 0 \quad \text{for all } q\in Q.$$

Hence $(x+\gamma\hat{t}) \in Y^o$ by proposition 4. Also $\|x+\gamma\hat{t} - x\|_\infty = \gamma\|\hat{t}\|_\infty < 1$. Therefore $x \in B_\infty(x+\gamma\hat{t},1)$ and $(x+\gamma\hat{t})\in Y^o$, which is a contradiction to $x\in (W')^c$.

□

Let :

$$\hat{w}(x) \triangleq \{y\in Y \mid \|x-y\|_\infty = \hat{n}(x)\} \tag{17}$$

$\hat{w}(x)$ is the set of minimizers of (12).

Proposition 6

The maps :

$$x \to B_\infty(x, \hat{\delta}(x))^c$$

$$x \to B_\infty(y,1)^c, \quad y\in\hat{w}(x)$$

are cut maps.

Proof

The proof is exactly the same as the proof of proposition 5 of Chapter II.

□

It is now possible to state a conceptual algorithm for the problem $P_{T,Q}$, analogous to algorithm 1 of Chapter II.

Algorithm 1

Step 0: Set $k=0$, $W_0 = \phi$

Step 1: Compute any $x_k \in W_k^c$.

If $x_k \in G$ stop; else proceed to step 2.

Step 2: Compute $y_k \in \hat{w}(x_k)$ and $\hat{\delta}(x_k) = 1 - \|x_k - y_k\|_\infty$.

Step 3: Set $W_{k+1} = W_k \cup B_\infty(y_k, 1)$.

[or $W_{k+1} = W_k \cup B_\infty(x_k, \hat{\delta}(x_k))$].

Set $k=k+1$ and go to step 1.

□

Theorem 1

(i) If the algorithm stops at x_k, then $x_k \in G$.

(ii) Any accumulation point of an infinite sequence generated by the algorithm lies in G.

Proof

The proof is exactly the same as the proof of theorem 1 of Chapter II since $\hat{\delta}(x)$ is a separator for the problem $P_{T,Q}$.

□

Algorithm 1 above possesses the same practical disadvantages as algorithm 1 of Chapter II, which for completeness are repeated

below :

(i) The computation in step 1 cannot be performed by a standard algorithm, since the constraints are non-differentiable.

(ii) The test in step 1 is conceptual.

(iii) An exact global minimization is required in step 2 to compute y_k.

(iv) The subproblem in step 1 increases in complexity with k since a new constraint is introduced at each iteration.

In the next section we proceed to obtain an implementable algorithm that does not possess the disadvantages (i) to (iv) listed above.

IV.3 An implementable algorithm for the tolerance-tuning problem.

Difficulties (i) and (iv) listed above can be overcome in exactly the same way as in Chapter II. The cut maps $x \to B_\infty(x, \hat{\delta}(x))^c$ and $x \to B_\infty(y,1)^c$, $y \in \hat{w}(x)$ can be replaced by $x \to B_2(x, \hat{\delta}(x))^c$ and $x \to [B_2(x, \hat{\delta}(x)) \cup B_2(y,1)]^c$, $y \in \hat{w}(x)$ so that the subproblem in step 1 can be solved by a standard inequality solving algorithm. The same cut dropping scheme can also be employed to keep the number of constraints low.

To avoid the need of an exact mimization in step 2, the separator values have to be approximated in some suitable way. In

Chapter II the separator estimates are required to be smaller than or equal to the exact values so that only outer approximations to the set G of the problem P_T are generated. In the case of the pure tolerance problem it is possible to generate such estimates with a finite number of operations because the set U is defined by a finite number of constraints. Another consequence of employing such estimates is that the algorithms of Chapter II jam up (i.e. start cycling between steps 2 and 3) once a point $x_k \in G$ is generated since a positive estimate of the separator at x_k can never be computed. In the case of the tolerance-tuning problem $P_{T,Q}$ the global optimization problem for the determination of $\hat{\eta}(x)$ is infinitely constrained. Hence, because it is not possible to test if a certain point lies in the set Y or not, we cannot compute separator estimates that are always smaller than the exact values with a finite number of operations. A consequence of this is that an implementable algorithm for the problem $P_{T,Q}$ will not necessarily jam up when a point in G is found, since a positive separator estimate may be computed at this point.

We recall that :

$$\hat{\eta}(x) = \min\{\|y-x\|_\infty \mid y \in Y\} \tag{18}$$

and

$$\hat{\delta}(x) = 1 - \hat{\eta}(x). \tag{19}$$

Let :

$$\eta(x) \triangleq \min\{1, \hat{\eta}(x)\} \tag{20}$$

and

$$\delta(x) \triangleq \max\{0, \hat{\delta}(x)\}. \tag{21}$$

Note that :

$$\delta(x) = 1 - \eta(x).$$

$\delta(x)$ is also a separator for the problem $P_{T,Q}$ since it is continuous. We have :

$$\delta(x) = \hat{\delta}(x) > 0 \quad \text{for all } x \in G^c$$
$$\delta(x) = 0 \qquad \text{for all } x \in G. \tag{22}$$

Hence the map $x \to B_2(x, \delta(x))^c$ is a cut map. It can be easily shown that :

$$G = \cap \{ B_2(x, \delta(x))^c \mid x \in R^n \} . \tag{23}$$

Now consider the problem of evaluating $\eta(x)$ where :

$$\eta(x) = \min \{ 1, \min \{ \|x-y\|_\infty \mid y \in Y \} \} \tag{24}$$

Suppose that the map $S: R^n \times Z^+ \to R^n$ is defined so that $S(x,j)$ is the result of applying $\tau(j)$ iterations of a certain algorithm to the problem of (24), where $\tau : Z^+ \to Z^+$ is a monotonically increasing truncation function ($\tau(j) \to \infty$ as $j \to \infty$). Let :

$$\bar{\eta}(x,j) \triangleq \|x - S(x,j)\|_\infty \tag{25}$$
$$\bar{\delta}(x,j) \triangleq 1 - \bar{\eta}(x,j) . \tag{26}$$

To obtain convergence, we impose the following condition on S:

A5: For any compact subset X of R^n

$$|\delta(x) - \bar{\delta}(x,j)| \to 0 \quad \text{as } j \to \infty, \text{ uniformly in } x \text{ for } x \in X .$$

Note that A5 does not contain the equivalent of assumption A4(i) of Chapter II. Hence even for very large values of j, $\bar{\delta}(x,j)$ will not necessarily be smaller than or equal to $\delta(x)$. This implies that the

map $x \to B_2(x, \bar{\delta}(x,j))^c$ will not necessarily possess the fundamental property of cut maps, i.e. :

$$G \cap B_2(x, \bar{\delta}(x,j))^c \neq \phi \tag{27}$$

Although A5 is sufficient for establishing convergence properties, (27) implies that the generation of accumulation points may be hindered by the fact that the approximations to the set G will not necessarily be outer approximations. In other words, the cuts generated by an implementable algorithm may contain only some subset of G (or may even be disjoint with G) and hence the generation of accumulation points may become unlikely (or even impossible). To avoid this undesirable feature, it is necessary to keep updating the separator estimates of the points that have not been dropped by the cut dropping scheme, to ensure approximations to the set G of increasing accuracy.

Suppose that $\bar{\delta}_k(x)$ denotes the latest estimate of the separator $\delta(x)$, (i.e. the estimate of $\delta(x)$ at iteration k). As in Chapter II, let an infinite double sequence $\{\varepsilon_j^k\}$ satisfy:

$$\begin{aligned}
&\text{(i)} \quad \varepsilon_j^k > 0 \ \text{ if } \ j < k, \ \ \varepsilon_j^k = 0 \ \text{ otherwise.} \\
&\text{(ii)} \quad \varepsilon_j^k \nearrow \bar{\varepsilon}_j \ \text{ as } \ k \to \infty, \ \text{uniformly in } j. \\
&\text{(iii)} \quad \bar{\varepsilon}_j \searrow 0 \ \text{ as } \ j \to \infty.
\end{aligned} \tag{28}$$

The set of "most important" balls at iteration k which is used for the formation of W_{k+1} (the next approximation to G^c), is defined by:

$$J(k) \triangleq \{ j \leq k \mid \bar{\delta}_k(x_j) > \varepsilon_j^k \} . \tag{29}$$

Hence in the implementable algorithm to be presented the latest estimates of the separators are employed to determine which points are to be dropped. To keep the number of separator re-estimations as small as possible, we also define the set $\hat{J}(k)$ as follows :

$$\hat{J}(k) = \{ j < k \mid \bar{\delta}_{k-1}(x_j) > \varepsilon_j^k \}. \tag{30}$$

$\hat{J}(k)$ contains all these points which according to their separator estimates at iteration k-1 are predicted to belong to J(k). Only the separators of the points in $\hat{J}(k)$ are re-estimated at iteration k (see step 4 of algorithm 2 below). Note that:

$$\hat{J}(k) \subset J(k-1).$$

Suppose that K is some infinite subset of Z^+ (the set of positive integers). For example, K may be the set of multiples of some positive integer. In the algorithm below the separators of the points in $\hat{J}(k)$ are re-estimated only if $k \in K$, so that the number of re-estimations is kept small (see step 4).

Algorithm 2

Step 0: Set k=0, $W_0 = \phi$.

Step 1: Compute any $x_k \in W_k^c$.

Step 2: Compute $\bar{\delta}_k(x_k) = \bar{\delta}(x_k,k)$.

Step 3: If $\bar{\delta}_k(x_k) \leq 0$ set $\bar{\delta}_k(x_j) = \bar{\delta}_{k-1}(x_j)$ for all $j < k$, set $W_{k+1} = W_k$, set $x_{k+1} = x_k$, set k=k+1 and go to step 2. Else proceed to step 4.

Step 4: If $k \in K$ set $\bar{\delta}_k(x_j) = \bar{\delta}(x_j,k)$ for all $j \in \hat{J}(k)$, set $\bar{\delta}_k(x_j) = \bar{\delta}_{k-1}(x_j)$ for all $j \notin \hat{J}(k)$, $j<k$.

Else set $\bar{\delta}_k(x_j) = \bar{\delta}_{k-1}(x_j)$ for all $j<k$.

Step 5: Set $W_{k+1} = \cup \{ B_2(x_j, \bar{\delta}_k(x_j)) \mid j \in J(k) \}$.

Set k=k+1 and go to step 1.

□

Notes

(i) The algorithm is directly implementable since each computation can be performed with a finite number of operations.

(ii) The difference between the above algorithm and algorithm 2 of Chapter II is that the separators of the active points (i.e. the points that have not been dropped) are re-estimated every some finite number of iterations. Hence the cut dropping scheme performs a double role. Firstly, it ensures that the number of re-estimations is small and secondly it keeps the complexity of the subproblem in step 1 low.

(iii) The above algorithm, unlike the algorithms of Chapter II will not necessarily jam up (i.e. start cycling between Steps 2 and 3) when a point in G is generated.

(iv) The subproblem in step 1 has the same form as that in the algorithms of Chapter II and hence all the remarks about its solution made in Chapters II and III are valid.

(v) The cut dropping scheme operates as described in Chapter II except that the latest separator estimates are now utilized in the definition of the set J(k). Once a point is dropped it does not re-appear and its separator is not re-estimated.

(vi) The approximations to the set G can be improved at each iteration by the introduction in step 1 of conventional constraints, either relating to the parameter magnitudes or involving the vertices of the tolerance region. Let $G_k \supset G$

be defined by :

$$G_k \triangleq \{x \in R^n \mid \text{for all } t \in T_k \text{ there exists some } q \in Q \text{ such that } f^j(x+t+r(q)) \leq 0,\ j=1,\ldots,m\} \tag{31}$$

where T_k is a finite subset of T (for example the set V of vertices of T). Suppose that L_k specifies the elements of T_k so that :

$$T_k \triangleq \{t_\ell \in T \mid \ell \in L_k\} \tag{32}$$

where L_k is a finite set . As discussed in [19] , finding a point in the set G_k is equivalent to computing a point x satisfying :

$$f^j(x+t_\ell, q_\ell) \leq 0, \quad \ell \in L_k, \quad j=1,\ldots,m \tag{33}$$

where

$$-\alpha \leq q_\ell \leq \beta, \quad \ell \in L_k.$$

Following Bandler [8] we call the variables $q_\ell, \ell \in L_k$ slack variables. Hence G_k corresponds to a set of continuously differentiable inequality constraints (assuming that the functions f^j, j=1,...,m are continuously differentiable). Step 1 of algorithm 2 can now be modified to Step 1' as follows :

Step 1': Compute any $x_k \in (W_k{}^c \cap G_k \cap H)$

where H is the set corresponding to any other conventional constraints present. Any heuristic rule for specifying T_k can be employed.

Theorem 2

Any accumulation point x^* of a sequence $\{x_k\}$ generated by algorithm 2 lies in G.

Proof

Case 1: Suppose that the algorithm starts cycling between steps 2 and 3 so that $x_k = x_k^* \triangleq x^*$ for all $k \geq k^*$, for some $k^* > 0$. Also suppose, contrary to what is to be proven, that $x^* \notin G^c$ so that $\delta(x^*) = \delta > 0$. By A5 we have :

$$|\delta(x^*) - \bar{\delta}(x^*,k)| \to 0 \quad \text{as} \quad k \to \infty.$$

Hence there exists some $\hat{k} > 0$ such that :

$$\bar{\delta}(x^*,k) \geq \frac{\delta}{2} > 0 \quad \text{for all } k \geq \hat{k}.$$

By the construction of the algorithm we have :

$$\bar{\delta}(x^*,k) = \bar{\delta}_k(x_k) \leq 0 \quad \text{for all } k \geq k^*$$

which is a contradiction. Hence $x^* \in G$.

Case 2: Suppose that $x_k \overset{I}{\to} x^*$. By the construction of the algorithm for any $j, k \in I$, $j < k$ we have :

$$\text{either} \quad x_k \notin B_2(x_j, \bar{\delta}_{k-1}(x_j)) \quad \text{if } j \in J(k-1)$$

$$\text{or} \quad 0 \leq \bar{\delta}_{k-1}(x_j) \leq \varepsilon_j^{k-1} < \bar{\varepsilon}_j \quad \text{if } j \notin J(k-1).$$

Hence for any $j,k \in I$, $j < k$:

$$0 \leq \bar{\delta}_{k-1}(x_j) \leq \max\{\bar{\varepsilon}_j, \| x_j - x_k \|_2\} \leq \bar{\varepsilon}_j + \| x_j - x_k \|_2$$

so that :

$$\bar{\delta}_{k-1}(x_j) \to 0 \quad \text{as} \quad j \to \infty, \quad j \in I. \tag{34}$$

Also by the construction of the algorithm:

$$\bar{\delta}_{k-1}(x_j) = \bar{\delta}(x_j, j(k)) \text{ for some } j(k) \geq j. \tag{35}$$

By assumption A5 and because $j(k) \geq j$:

$$|\bar{\delta}(x_j, j(k)) - \delta(x_j)| \to 0 \quad \text{as} \quad j \to \infty, \; j \in I. \tag{36}$$

Combining (34), (35) and (36) we have that :

$$\delta(x_j) \to 0 \quad \text{as} \quad j \to \infty, \; j \in I.$$

Hence, by the continuity of δ, $\delta(x^*) = 0$ and $x^* \in G$. □

Algorithm 2 solves the tolerance-tuning problem in the sense that it constructs sequences whose accumulation points are solutions to the problem $P_{T,Q}$. In the next Chapter implementation details are given, numerical examples are presented and conclusions about the properties of the algorithm are drawn.

CHAPTER V

IMPLEMENTATION OF THE ALGORITHM FOR TOLERANCE-TUNING PROBLEM

V.1 Introduction.

In this Chapter the implementation details of the algorithm presented in Chapter IV for the tolerance-tuning problem are discussed. As for the algorithms for the pure tolerance problem, these are very important for computational efficiency. In Section 2 the subproblem of Step 1 is examined. Procedures for the infinitely constrained global optimization problem of Step 2 are proposed in Section 3. In Section 4 numerical examples are presented and finally in Section 5 the properties of the algorithm are discussed.

V.2 On the Solution of the feasibility subproblem.

We recall that the general subproblem in Step 1 of algorithm 2 of Chapter IV has the following form:

P_{W_k}: Find a point x_k in the set $W_k^c \cap G_k \cap H$.

As already discussed, P_{W_k} corresponds to a set of continuously differentiable constraints and has exactly the same form as the subproblem of Step 1 of the algorithms of Chapter II. Hence P_{W_k} can be solved by employing the techniques of Section 2 of Chapter III. The same initial point rule should be utilized for the modified Newton algorithm of [22]. However note that in the case of the tolerance-tuning problem, G is not necessarily a subset of each W_k^c. Hence it is important to utilize an interactive program allowing the specification of the sets T_k (that define G_k) and K (that defines the frequency of the separator re-estimations) as the algorithm progresses.

The designer may choose to relax the "vertex constraints" that define G_k or re-estimate the separators of the active points, if difficulties (e.g. jamming) occur when solving the subproblem of Step 1 at a certain iteration. As before, if no "vertex constraints" are present ($G_k = R^n$) the solution of this subproblem is computationally very cheap, since repeated evaluation of the design constraints and their derivatives is not required.

V.3 On the computation of the separator estimates.

The global optimization problem that is involved in the computation of the separator at x is :

$$\eta(x) = \min\{1,\ \min\{\|x-y\|_\infty \mid y\in Y\}\} = \min\{1, \hat{\eta}(x)\} \qquad (1)$$

where

$$Y = \{y\in R^n \mid \hat{\psi}(y,q) \geq 0 \text{ for all } q\in Q\}. \qquad (2)$$

In this section procedures that generate separator estimates and satisfy assumption A5 of Chapter IV will be proposed. We recall that :

$$\bar{\eta}(x,j) = \| x-S(x,j)\|_\infty \quad \text{(estimate of } \eta(x)) \qquad (3)$$

and

$$\bar{\delta}(x,j) = 1 - \bar{\eta}(x,j) \quad \text{(estimate of } \delta(x)) \qquad (4)$$

where $S(x,j)$ is the result of applying $\tau(j)$ iterations of a certain algorithm to the problem of (1) above and $\tau(j)$ is a monotonically increasing truncation function. Suppose that Y_j is an approximation to the set Y generated by one of the following two methods.

Method 1

Discretize Q to form Q_j by considering $\bar{\tau}(j)$ "uniformly spaced" points in Q, where $\bar{\tau}(j) \to \infty$ as $j \to \infty$. Let :

$$Y_j \triangleq \{y \in R^n \mid \hat{\psi}(y,q) \geq 0 \text{ for all } q \in Q_j\} . \tag{5}$$

Method 2

Discretize Q to form Q_j by considering $\bar{\tau}(j)$ points $\{q_1, q_2, \ldots, q_{\bar{\tau}(j)}\}$ in Q spaced so that $\bar{\varepsilon}_j \to 0$ as $j \to \infty$, where :

$$\bar{\varepsilon}_j \triangleq \max_q \min_r \{\|q - q_r\|_\infty \mid q \in Q, \; r = 1, \ldots, \bar{\tau}(j)\}. \tag{6}$$

Let Y_j be defined as above by (5).

Note that is suffices that $\hat{\psi}(y,\hat{q}) < 0$ for some $\hat{q} \in Q_j$ for the point y not to lie in Y_j. Hence testing if a point lies in Y_j or not does not necessarily involve the evaluation of $\hat{\psi}(y,q)$ for all $q \in Q_j$. The following two procedures for computing the separator estimates $\bar{\delta}(x,j)$ will be shown to satisfy A5.

Procedure 1

Compute $\tau(j)$ "uniformly spaced" points $Z_j(x) = \{z_1, \ldots, z_{\tau(j)}\}$ in $(x+T)^o$ where $\tau(j) \to \infty$ as $j \to \infty$. Order these points so that $\|z_i - x\|_\infty$ increases with i. Set $S(x,j) = z_i^*$ ($\bar{\eta}(x,j) = \|x - S(x,j)\|_\infty$), where z_i^* is the first point such that $z_i^* \in Y_j$. If no such point exists, set $\bar{\eta}(x,j) = 1$ so that $\bar{\delta}(x,j) = 0$.

Procedure 2

Compute $\tau(j)$ points $Z_j(x) = \{ z_1,\ldots,z_{\tau(j)} \}$ in $(x+T)^o$ spaced so that $\varepsilon_j \to 0$ as $j \to \infty$ where :

$$\varepsilon_j \triangleq \max_z \min_r \{ \| z-z_r \|_\infty \Big| z\in(x+T),\ r=1,\ldots,\tau(j)\}. \qquad (7)$$

Order these points and compute $\bar{\delta}(x,j)$ as in procedure 1 above.

□

Notes

(i) The sets of points Q_j and $Z_j(x)$ that satisfy (6) and (7) respectively can be computed using a pseudo-random number generator.

(ii) The number of tests $z\in Y_j$ required for the computation of $\bar{\delta}(x,j)$ is i^* where :

$$i^* \doteq \tau(j)\ \bar{\eta}(x,j)^n. \qquad (8)$$

Hence i^* will be small at low yield points. From the ordered points in $Z_j(x)$ that lie in Y_j only the first one will be tested, by the definition of the procedures. Note that it is precisely for these points that the test $z\in Y_j$ is most computationally expensive, since $\hat{\psi}(z,q)$ has to be evaluated for all $q\in Q_j$.

(iii) The procedures proposed compute $\bar{\delta}(x,j)$ given x and j. It is possible that x remains constant while j is increased to j+k. This may happen either because the algorithm cycles between steps 2 and 3, or because the separator at x has to be re-estimated. The mechanism for generating $Z_{j+k}(x)$ should be such that $\tau(j+k) - \tau(j)$ extra points are generated in x+T, i.e. $Z_{j+k}(x) = Z_j(x) \cup Z^e_{j+k}(x)$. From the points in $Z_{j+k}(x)$ all the points in $Z_j(x)$ that have been found not to lie in Y_j (and hence is Y) are discarded. The rest of the points are ordered and S(x,j+k) is set equal to the first point $z_i^{\sim}$ satisfying $z_i^{\sim} \in Y_{j+k}$.

(iv) The artifice that takes advantage of the overlap of successive tolerance regions described in Section 3 of Chapter III can be employed (after a slight modification) to greatly improve computational efficiency. Also the linear searching technique described in the same section can be utilized to improve the separator estimates.

(v) The truncation function $\tau(j)$ should be chosen as in the pure tolerance case. Hence $\tau(j)$ should be kept small as long as this permits the computation of $\bar{\delta}(x_j,j) > 0$ but should be increased considerably if $\bar{\delta}(x_j,j) = 0$. On the contrary, the difference between $\bar{\tau}(j+1)$ and $\bar{\tau}(j)$ should be small if $\bar{\delta}(x_j,j) = 0$. Note that if, as is usually the case, the dimension of Q (i.e. the number of tuned parameters) is small, small values of $\bar{\tau}(j)$ are sufficient for Y_j to be a good approximation to Y. Ideally $\tau(j)$ and $\bar{\tau}(j)$ should be specified interactively

according to the progress of the algorithm.

We next prove some results that are necessary for establishing the fact that procedures 1 and 2 satisfy A5. Let :

$$\phi(y) \triangleq \min \{ \hat{\psi}(y,q) \,|\, q\in Q\} \tag{9}$$

and

$$\phi_j(y) \triangleq \min \{\hat{\psi}(y,q) \,|\, q\in Q_j\} \tag{10}$$

so that

$$Y = \{ y \,|\, \phi(y) \gtrsim 0\} \tag{11}$$

and

$$Y_j = \{ y \,|\, \phi_j(y) \gtrsim 0 \} \ . \tag{12}$$

Also let:

$$\hat{\eta}_j(x) \triangleq \min\{ \| x-y\|_\infty \,|\, y\in Y_j\} \tag{13}$$

and

$$\eta_j(x) \triangleq \min \{1,\hat{\eta}_j(x)\} \ . \tag{14}$$

Note that from all the above definitions the following relations are true :

$$Y\subset Y_j \tag{15}$$

$$\hat{\eta}_j(x) \lesssim \hat{\eta}(x) \tag{16}$$

$$\eta_j(x) \lesssim \eta(x) \lesssim 1 \tag{17}$$

$$\eta_j(x) \lesssim \bar{\eta}(x,j) \lesssim 1 \tag{18}$$

$$\phi(y) \lesssim \phi_j(y). \tag{19}$$

Proposition 1

For any compact set $\hat{Y}$ of R^n:

$$|\phi(y) - \phi_j(y)| \to 0 \quad \text{as} \quad j \to \infty, \text{ uniformly in } y, \text{ for } y \in \hat{Y}.$$

Proof

Firstly note that Method 1 of discretizing Q also satisfies (6). Now suppose that $q \in Q$ is a minimizer associated with y so that $\phi(y) = \hat{\psi}(y,q)$. Choose any $\varepsilon > 0$, then by the continuity of $\hat{\psi}$ there exists some $\delta > 0$ such that :

$$|\hat{\psi}(y,q) - \hat{\psi}(y',q')| \leq \frac{\varepsilon}{2} \quad \text{for all} \quad q' \in B_\infty(q,\delta) \quad \text{and all} \quad y' \in B_\infty(y,\delta)$$

where $B_\infty(q,\delta) = \{q' \in R^\ell \mid \|q-q'\|_\infty < \delta\}$.

By the definition of Q_j (see (6)), there exists some integer $N(y) > 0$ such that at least one mesh point $q_j \in Q_j$ lies in $B_\infty(q,\delta)$ for all $j \geq N(y)$. Hence :

$$\phi(y') \leq \phi_j(y') \leq \hat{\psi}(y',q_j) \leq \hat{\psi}(y,q) + \frac{\varepsilon}{2} = \phi(y) + \frac{\varepsilon}{2} \qquad (20)$$

$$\text{for all } y' \in B_\infty(y,\delta) \text{ and all } j \geq N(y).$$

By the continuity of ϕ, there exists some $\delta' > 0$ such that :

$$|\phi(y') - \phi(y)| \leq \frac{\varepsilon}{2} \quad \text{for all } y' \in B_\infty(y,\delta') \qquad (21)$$

Combining (20) and (21) and taking $\delta_y = \min\{\delta,\delta'\} > 0$ we have that for any $y \in R^n$ and any $\varepsilon > 0$ there exists some integer $N(y) > 0$ and some $\delta_y > 0$ such that :

$$| \phi(y') - \phi_j(y') | \leq \varepsilon \text{ for all } y' \in B_\infty(y,\delta_y) \text{ and all } j \geq N(y).$$

The family of balls $B_\infty(y,\delta_y)$, $y \in \hat{Y}$ forms an open cover for $\hat{Y}$ and, since $\hat{Y}$ is compact, there exists a finite subcover so that :

$$\hat{Y} \subset \cup \{ B_\infty(y,\delta_y) \,|\, y \in \bar{Y} \}$$

where $\bar{Y}$ is some finite subset of $\hat{Y}$. Setting $N_{\hat{Y}} = \max \{ N(y) \,|\, y \in \bar{Y} \}$ we have that:

$$|\phi(y) - \phi_j(y)| \leq \varepsilon \quad \text{for all } y \in \hat{Y} \text{ and all } j \geq N_{\hat{Y}}$$

which proves the proposition.

□

Proposition 2

Suppose that $y_j(x) \in Y_j$ is a minimizer associated with $\hat{\eta}_j(x)$. Then for any compact subset X of R^n and any $\varepsilon > 0$, there exists some $j_X > 0$ such that :

$$\hat{\eta}(y_j(x)) \leq \varepsilon \quad \text{for all } j \geq j_X \text{ and all } x \in X.$$

Proof

Suppose the result is not true. Then there exists some infinite subsequence $\{ y'_j \}$, $j \in J$ and some $\varepsilon > 0$ such that :

$$y'_j = y_j(x) \in Y_j \quad \text{for some } x \in X$$

and (22)

$$\hat{\eta}(y'_j) > \varepsilon \quad \text{for all } j > 0, \ j \in J.$$

Choose any $\hat{y} \in Y$, then since $Y \subset Y_j$ and X is compact we have :

$$\| x - y'_j \|_\infty = \hat{\eta}_j(x) \leq \hat{\eta}(x) \leq \| x - \hat{y} \|_\infty$$
$$\leq M \triangleq \max \{ \| x - \hat{y} \|_\infty \,|\, x \in X \}.$$

Hence, for any $\hat{x} \in R^n$:

$$\| \hat{x}-y_j' \|_\infty \leq \| \hat{x}-x \|_\infty + \| x-y_j' \|_\infty \leq N+M$$

$$\triangleq \max \{ \| \hat{x}-x \|_\infty \mid x \in X \} + M$$

by the compactness of X. Hence the subsequence $\{y_j'\}$ is bounded so that it has accumulation points. Suppose that $y_j' \xrightarrow{J'} y^*$. Since $y_j' \in Y_j$ we have that:

$$\phi_j(y_j') \geq 0.$$

Also by proposition 1:

$$| \phi_j(y_j') - \phi(y_j') | \to 0 \quad \text{as} \quad j \to \infty, \; j \in J' .$$

Hence by the continuity of ϕ we have :

$$\lim_{\substack{j \to \infty \\ j \in J'}} \phi(y_j') = \phi(y^*) \geq 0 \Longrightarrow y^* \in Y. \tag{23}$$

Combining (22) and (23) :

$$\| y_j' - y^* \|_\infty \geq \hat{\eta}(y_j') > \varepsilon > 0 \quad \text{for all } j > 0, \; j \in J'$$

which is a contradiction to $y_j' \xrightarrow{J'} y^*$. Hence the proposition has been proven.

□

Proposition 3

For any compact subset X of R^n :

$$| \hat{\eta}_j(x) - \hat{\eta}(x) | \to 0 \quad \text{as} \quad j \to \infty \text{ , uniformly in x for } x \in X.$$

Proof

Choose any $\varepsilon > 0$. By proposition 2 there exists some $j_X > 0$ such that :

$$\hat{\eta}(y_j(x)) \leq \varepsilon \text{ for all } j \geq j_X \text{ and all } x \in X.$$

Suppose that $y(x) \in Y$ is a minimizer associated with $\hat{\eta}(x)$ and $y_j'(x) \in Y$ with $\hat{\eta}(y_j(x))$ so that $\hat{\eta}(y_j(x)) = \| y_j(x) - y_j'(x) \|_\infty$.
Then :

$$\hat{\eta}_j(x) \leq \hat{\eta}(x) = \| x - y(x) \|_\infty \leq \| x - y_j'(x) \|_\infty \leq$$

$$\| x - y_j(x) \|_\infty + \| y_j(x) - y_j'(x) \|_\infty = \hat{\eta}_j(x) + \hat{\eta}(y_j(x)) \leq$$

$$\hat{\eta}_j(x) + \varepsilon \quad \text{for all} \quad j \geq j_X \text{ and all } x \in X.$$

Hence :

$$|\hat{\eta}_j(x) - \hat{\eta}(x)| \leq \varepsilon \text{ for all } j \geq j_X \text{ and all } x \in X$$

which proves the proposition.

□

Proposition 4

For any compact set X of R^n :

$$|\eta_j(x) - \eta(x)| \to 0 \quad \text{as} \quad j \to \infty \quad \text{uniformly in } x \text{ for } x \in X.$$

Proof

We prove that $|\eta_j(x) - \eta(x)| \leq |\hat{\eta}_j(x) - \hat{\eta}(x)|$ and use proposition 3.

(i) $\min(A,B) - \min(C,D) \leq \max(A-C, B-D)$

[Proof: $\max(A-C,B-D) \geq A-C \geq \min(A,B)-C \longrightarrow$

$C \geq \min(A,B) - \max(A-C,B-D)$.

Similarly: $D \geq \min(A,B) - \max(A-C,B-D)$.

Hence: $\min(C,D) \geq \min(A,B) - \max(A-C,B-D)$] .

(ii) $|\min(A,B) - \min(C,D)| \leq \max(|A-C|, |B-D|)$.

[Proof: $\min(A,B) - \min(C,D) \leq \max(A-C,B-D) \leq \max(|A-C|, |B-D|)$

$\min(C,D) - \min(A,B) \leq \max(C-A,D-B) \leq \max(|A-C|, |B-D|)$.

Hence: $|\min(A,B) - \min(C,D)| \leq \max(|A-C|, |B-D|)]$.

We now have :

$$|\eta_j(x) - \eta(x)| = |\min(1,\hat{\eta}_j(x)) - \min(1,\hat{\eta}(x))| \leq$$

$$\max(|1-1|, |\hat{\eta}_j(x) - \hat{\eta}(x)|) = |\hat{\eta}_j(x) - \hat{\eta}(x)| .$$

□

We finally prove that procedures 1 and 2 that generate the separator estimates satisfy A5.

Theorem 1

For any compact set $X \subset R^n$ the estimates $\bar{\eta}(x,j)$ generated by Procedures 1 and 2 satisfy :

$|\bar{\eta}(x,j) - \eta(x)| \to 0$ as $j \to \infty$, uniformly in x for $x \in X$,

Proof

(i) Choose any $x \in G^c$. Suppose that y is some minimizer associated with $\eta(x) = \hat{\eta}(x)$ $(y \in \hat{w}(x))$. As in the proof of theorem 2 in Chapter III, for any $\varepsilon > 0$ the open set

$$Y_\varepsilon(y) \triangleq B_\infty(y,\frac{\varepsilon}{2}) \cap (x+T)^o \cap Y^o$$

is not empty. Hence there exist some $y' \in R^n$ and some $\varepsilon' > 0$ such that :

$$B_\infty(y',\varepsilon') \subset Y_\varepsilon(y).$$

As in the proof of theorem 2 in Chapter III there exists some integer $N_1(x) > 0$ such that at least one point $y_j \triangleq (x+\sigma_j) \in Z_j(x)$ lies in $B_\infty(y',\frac{\varepsilon'}{2})$ for all $j \geq N_1(x)$. Hence $y'_j \triangleq (x'+\sigma_j) \in Z_j(x')$ lies in $B_\infty(y',\varepsilon')$ (and hence in Y^o and Y_j) for all $x' \in B_\infty(x,\frac{\varepsilon'}{2})$ and all $j \geq N_1(x)$. We then have :

$$\bar{\eta}(x',j) \leq \| x'-y'_j \|_\infty = \| \sigma_j \|_\infty = \| x-y_j \|_\infty \leq$$

$$\| x-y \|_\infty + \| y-y_j \|_\infty \leq \eta(x) + \frac{\varepsilon}{2} \quad \text{for all } x' \in B_\infty(x, \frac{\varepsilon'}{2}) .$$

By the continuity of $\eta(x)$ there exists some $\delta' > 0$ such that :

$$| \eta(x) - \eta(x') | \leq \frac{\varepsilon}{2} \quad \text{for all } x' \in B_\infty(x,\delta').$$

Taking $\delta_x = \min \{\delta', \frac{\varepsilon'}{2}\} > 0$, we have :

$$\bar{\eta}(x',j) \leq \eta(x') + \varepsilon \quad \text{for all } x' \in B_\infty(x,\delta_x) \quad \text{and all } j \geq N_1(x). \tag{24}$$

By proposition 4, there exists an integer $N_2(x) > 0$ such that :

$$\eta(x') \leq \eta_j(x') + \varepsilon \quad \text{for all } x' \in B_\infty(x,\delta_x) \quad \text{and all } j \geq N_2(x). \tag{25}$$

Combining (24) and (25) with (18) and setting $N(x) = \max \{N_1(x), N_2(x)\}$ we get:

$$|\bar{\eta}(x',j) - \eta(x')| \leq \varepsilon \text{ for all } x' \in B_\infty(x,\delta_x) \text{ and all } j \geq N(x).$$

(ii) Take any $x \in G$, so that $\eta(x) = 1$. By the continuity of η, for any $\varepsilon > 0$ there exists some $\delta_x > 0$ such that :

$$|\eta(x) - \eta(x')| \leq \frac{\varepsilon}{2} \quad \text{for all } x' \in B_\infty(x,\delta_x). \tag{26}$$

so that :

$$0 \leq 1 - \eta(x') \leq \frac{\varepsilon}{2} \quad \text{for all } x' \in B_\infty(x,\delta_x).$$

By proposition 4, there exists some integer $N(x) \geq 0$ such that :

$$|\eta(x') - \eta_j(x')| \leq \frac{\varepsilon}{2} \quad \text{for all } x' \in B_\infty(x,\delta_x) \text{ and all } j \geq N(x).$$

Hence :

$$0 \leq 1 - \eta_j(x') \leq \varepsilon \quad \text{for all } x' \in B_\infty(x,\delta_x) \text{ and all } j \geq N(x).$$

Using (18) and the fact that $\eta(x') \leq 1$ we have :

$$\eta(x') - \bar{\eta}(x',j) \leq 1 - \bar{\eta}(x',j) \leq 1 - \eta_j(x') \leq \varepsilon \text{ for all } x' \in B_\infty(x,\delta_x) \text{ and all } j \geq N(x)$$

so that :

$$\eta(x') \leq \bar{\eta}(x',j) + \varepsilon \quad \text{for all } x' \in B_\infty(x,\delta_x) \text{ and all } j \geq N(x). \tag{27}$$

Also since $\bar{\eta}(x',j) \leq 1 = \eta(x)$, we have by (26):

$$\bar{\eta}(x',j) \leq \eta(x) \leq \eta(x') + \varepsilon \quad \text{for all } x' \in B_\infty(x,\delta_x) \text{ and all } j > 0, \tag{28}$$

Combining (27) and (28) we have :

$$|\eta(x') - \bar{\eta}(x',j)| \leq \varepsilon \quad \text{for all } x' \in B_\infty(x,\delta_x) \text{ and all } j \geq N(x).$$

Combining (i) and (ii) we have that for any $x \in R^n$ and any $\varepsilon > 0$ there exist some $\delta_x \neq 0$ and some integer $N(x) > 0$ such that :

$$|\eta(x') - \bar{\eta}(x',j)| \leq \varepsilon, \quad \text{for all } x' \in B_\infty(x,\delta_x) \text{ and all } j \geq N(x).$$

The desired result now follows as in the proof of theorem 2 in Chapter III by the compactness of the set X.

□

V.4 Examples.

Five examples are presented. The feasible sets F are the same as those in Chapter III, but the introduction of tuning allows considerable increases in the parameter tolerances.

Example 1

The set $F \subset R^2$ is defined as in example 1 of Chapter III. The values of the parameters utilized by the algorithm are $a = \frac{1}{3}$, $\gamma = 1.5$, $\delta = 0.7$. The set K was defined as the set {3, 7, 11, 15,... }. Procedure 1 of section 3 that generates the separator estimates was implemented by taking $Z_k(x)$ to be the set of uniformly spaced mesh points in $(x+T)^0$ such that the distance between adjacent points is $\frac{1}{\sigma(k)}$, where :

$$\sigma(k+1) = \sigma(k) + 1 \quad \text{if} \quad x_{k+1} \neq x_k$$

$$\sigma(k+1) = 2\sigma(k) \quad \text{if} \quad x_{k+1} = x_k$$

and $\sigma(0) = 4$. Each set $Q \subset R$ (only one parameter is tuned for each run) was discretized by taking Q_k to be the set of $\bar{\tau}(k)$ uniformly spaced points in Q where :

$$\bar{\tau}(k) = 2k + 3.$$

(Hence method 1 of section 3 was employed). Only the nominal point was included in the definition of each set T_k, so that only one slack variable was introduced. Table 1 lists the tolerance and tuning regions corresponding to each run. Tables 2, 3, 4 and 5 summarize the results for these runs. Separator re-estimations were performed when k=3 in runs 1 and 2 (see Tables 2 and 3). The operation of the algorithm was stopped when $\sigma(k)$ reached the value of 100 and $\bar{\delta}(x_k,k) = 0$. Most of the feasibility subproblems were solved in one iteration. Figures 1,2,3 and 4 illustrate the four runs presented. Note that the tolerances of the parameters are considerably larger than those in example 1 of Chapter III.

Run	TOLERANCES		Tuned parameter	Tuning region
	x^1	x^2		
1	2.00	1.50	x^1	$\pm$ 1.00
2	1.50	1.20	x^1	$\pm$ 0.50
3	2.00	1.75	x^2	$\pm$ 0.75
4	2.50	2.00	x^2	$\pm$ 1.00

TABLE 1

k	$\sigma(k)$	$\bar{\tau}(k)$	x_k		$\bar{\delta}_k(x_k)$	$\bar{\delta}_3(x_k)$
			x^1	x^2		
0	4	3	-2.000	1.000	0.844	0.849
1	5	5	-0.156	2.677	0.892	0.892
2	6	7	0.656	1.294	0.181	-
3	7	9	1.015	0.912	0.192	-
4	8	11	0.485	1.332	0.191	-
5	9	13	0.215	0.744	0.000	
6	18	15	"	"	"	
7	36	17	"	"	"	
8	72	19	"	"	"	
9	100	21	0.215	0.744	0.000	

TABLE 2 (Run 1, Initial point (-2,1)

k	$\sigma(k)$	$\bar{\tau}(k)$	x_k		$\bar{\delta}_k(x_k)$	$\bar{\delta}_3(x_k)$
			x^1	x^2		
0	4	3	-1.000	1.000	0.638	-
1	5	5	-0.713	2.756	0.820	0.902
2	6	7	0.064	1.651	0.858	0.859
3	7	9	1.064	0.509	0.286	-
4	8	11	0.340	0.509	0.000	-
5	16	13	"	"	"	
6	32	15	"	"	"	
7	64	17	"	"	"	
8	100	19	0.340	0.509	0.000	

TABLE 3 (Run 2, Initial point (-1,1))

k	$\sigma(k)$	$\bar{\tau}(k)$	x_k		$\bar{\delta}_k(x_k)$
			x^1	x^2	
0	4	3	1.300	1.300	0.725
1	5	5	-0.086	1.809	0.624
2	6	7	-0.099	0.493	0.000
3	12	9	"	"	"
4	24	11	"	"	0.058
5	25	13	-0.268	0.353	0.000
6	50	15	"	"	"
7	100	17	-0.268	0.353	0.000

TABLE 4 (Run 3, Initial point (1.3,1.3))

k	σ(k)	$\bar{\tau}(k)$	x_k		$\bar{\delta}_k(x_k)$
			x^1	x^2	
0	4	3	1.000	1.000	0.600
1	5	5	-0.500	0.910	0.200
2	6	7	-0.996	0.326	0.000
3	12	9	"	"	0.083
4	13	11	-0.707	0.088	0.000
5	26	13	"	"	"
6	52	15	"	"	"
7	100	17	-0.707	0.088	0.000

TABLE 5 (Run 4, Initial point (1,1))

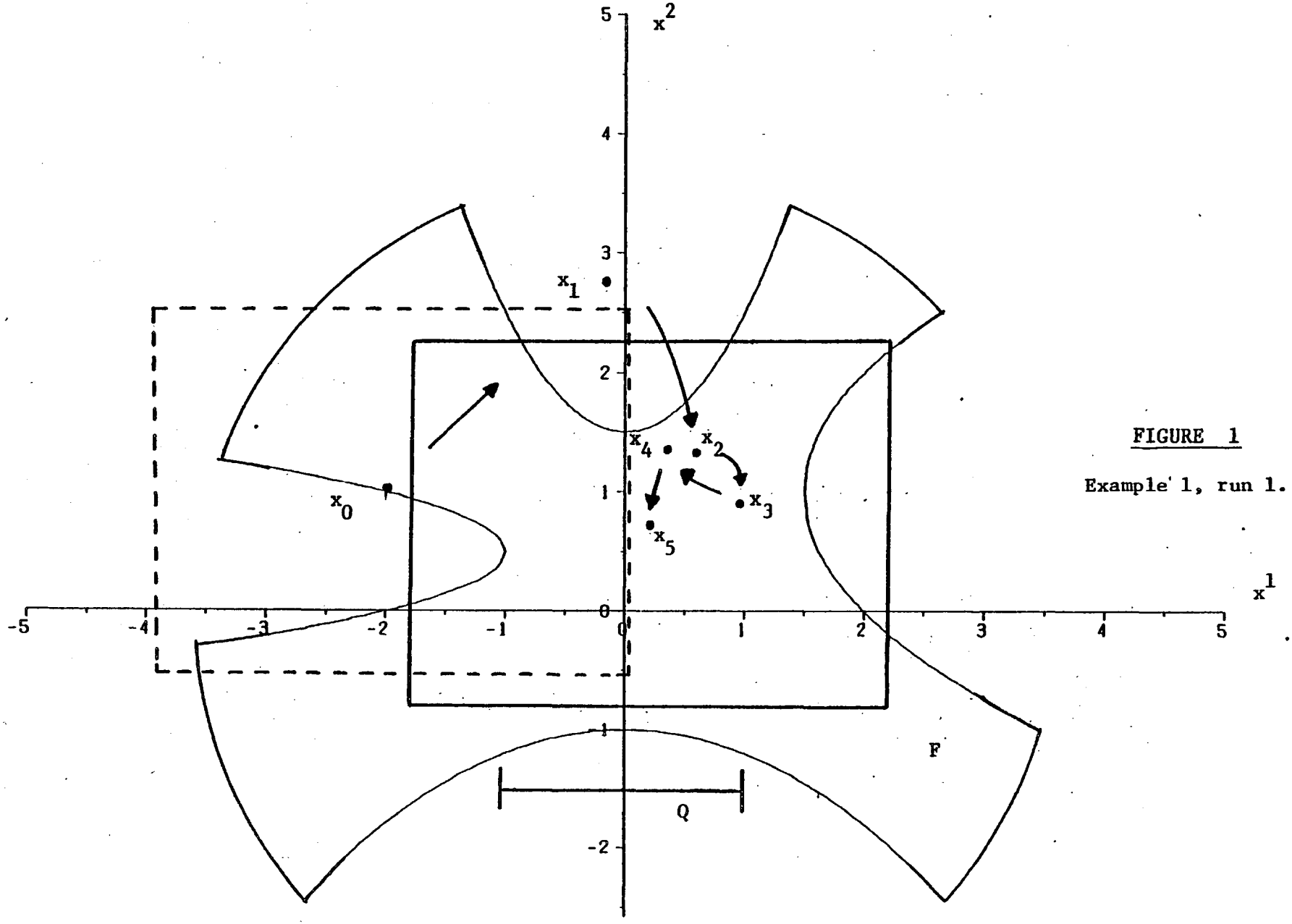


FIGURE 1

Example 1, run 1.

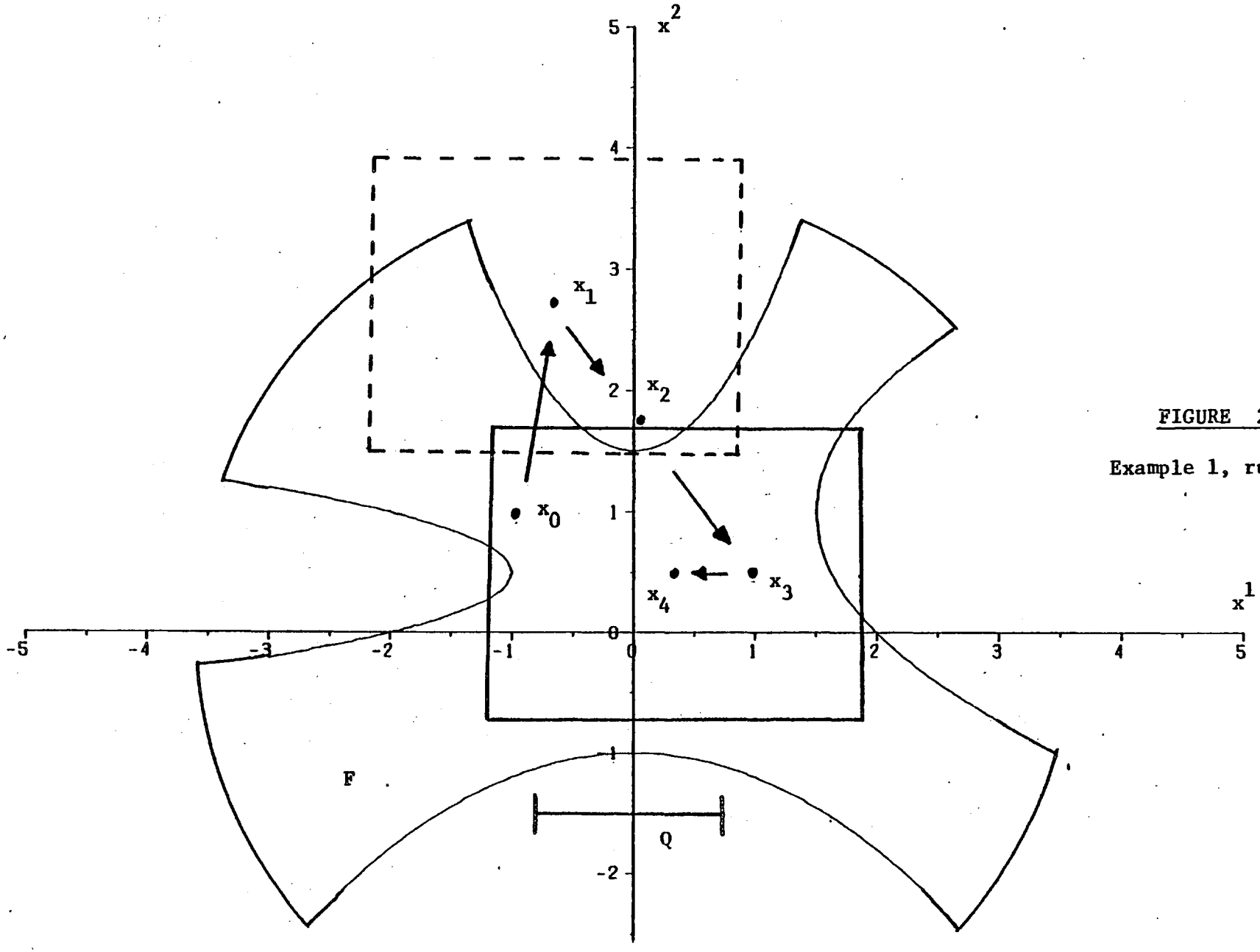


FIGURE 2

Example 1, run 2.

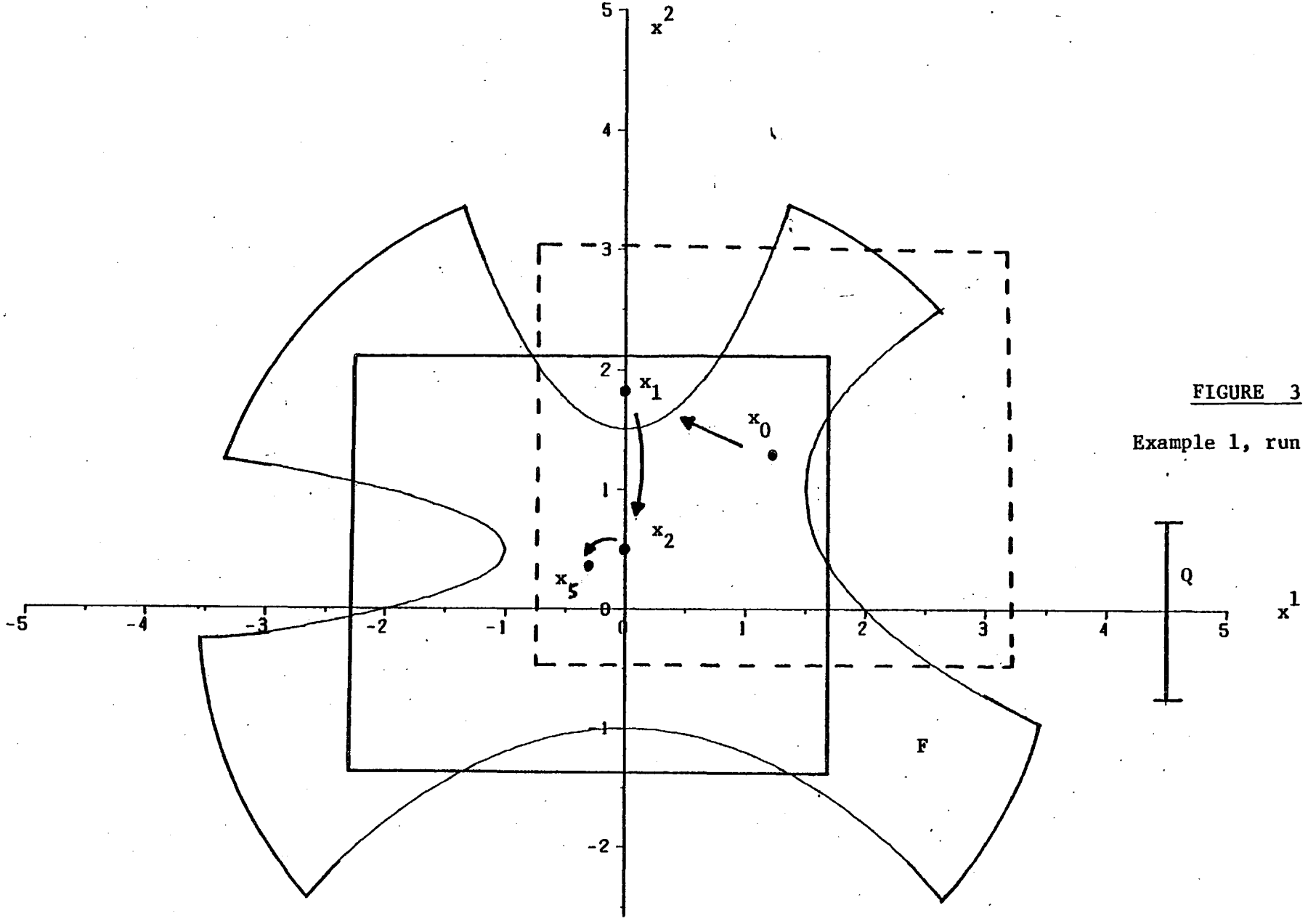


FIGURE 3

Example 1, run 3.

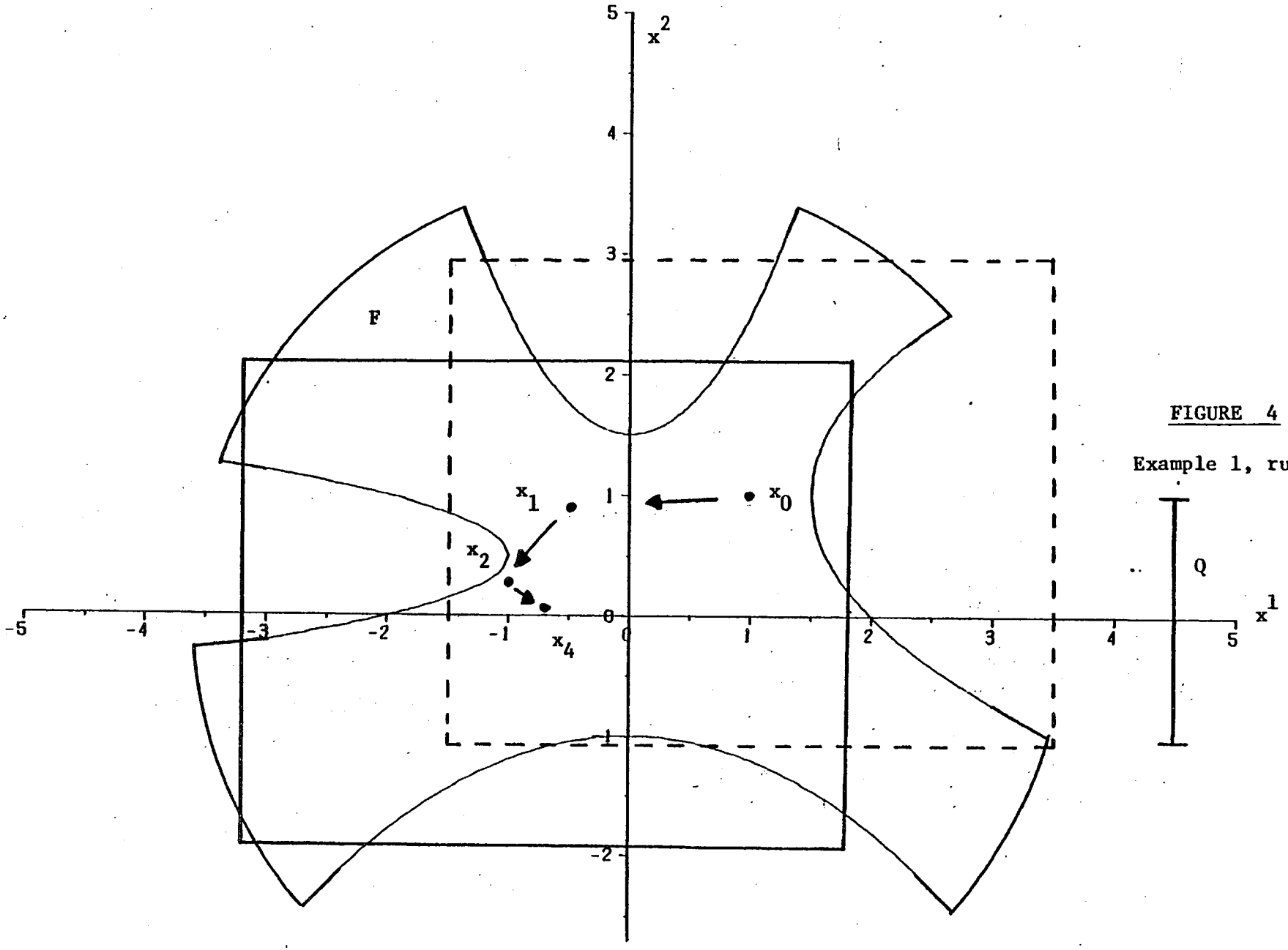


FIGURE 4

Example 1, run 4.

Example 2

The set $F \subset R^2$ is defined as in example2 of Chapter III. All the conventionsand parameters are as in example 1. Table 6 summarizes the tolerance and tuning regions of each run. Tables 7,8,9 and 10 present the results for these runs which are illustrated in figures 5,6,7 and 8 respectively. Note that separator re-estimations are performed only in run 2 when k=3.

Run	TOLERANCES		Tuning parameter	Tuning region
	x^1	x^2		
1	2.00	2.00	x^1	$\pm$2.00
2	1.00	2.00	x^1	$\pm$1.00
3	2.00	1.50	x^2	$\pm$0.75
4	3.00	1.50	x^2	$\pm$ 1.00

TABLE 6

k	$\sigma(k)$	$\bar{\tau}(k)$	x_k		$\bar{\delta}_k(x_k)$
			x^1	x^2	
0	4	3	1.233	1.233	0.613
1	5	5	2.687	-0.220	0.000
2	10	7	"	"	"
3	20	9	"	"	"
4	40	11	"	"	"
5	80	13	"	"	"
6	100	15	2.687	-0.220	0.000

TABLE 7 (Run 1, Initial point (1,1))

k	$\sigma(k)$	$\bar{\tau}(k)$	x_k		$\bar{\delta}_k(x_k)$	$\bar{\delta}_3(x_k)$
			x^1	x^2		
0	4	3	1.233	1.233	0.763	0.731
1	5	5	3.316	2.000	0.996	0.998
2	6	7	4.464	0.405	0.508	0.590
3	7	9	3.675	-0.554	0.571	-
4	8	11	2.590	-0.312	0.000	
5	16	13	"	"	"	
6	32	15	"	"	"	
7	64	17	"	"	"	
8	100	19	2.590	-0.312	0.000	

TABLE 8 (Run 2, Initial point (1,1))

k	$\sigma(k)$	$\bar{\tau}(k)$	x_k		$\bar{\delta}_k(x_k)$
			x^1	x^2	
0	4	3	1.199	1.999	0.306
1	5	5	2.387	0.811	0.200
2	6	7	2.040	0.355	0.000
3	12	9	"	"	
4	24	11	"	"	
5	48	13	"	"	0.043
6	49	15	2.169	0.249	0.000
7	98	17	"	"	"
8	100	19	2.169	0.249	0.000

TABLE 9 (Run 3, Initial Point (1,1)

k	$\sigma(k)$	$\bar{\tau}(k)$	x_k		$\bar{\delta}_k(x_k)$
			x^1	x^2	
0	4	3	1.199	1.199	0.856
1	5	5	3.915	0.386	0.264
2	6	7	3.078	-0.273	0.000
3	12	9	"	"	"
4	24	11	"	"	0.061
5	25	13	2.870	-0.109	0.000
6	50	15	"	"	"
7	100	17	2.870	-0.109	0.000

TABLE 10 (Run 4, Initial point (1,1))

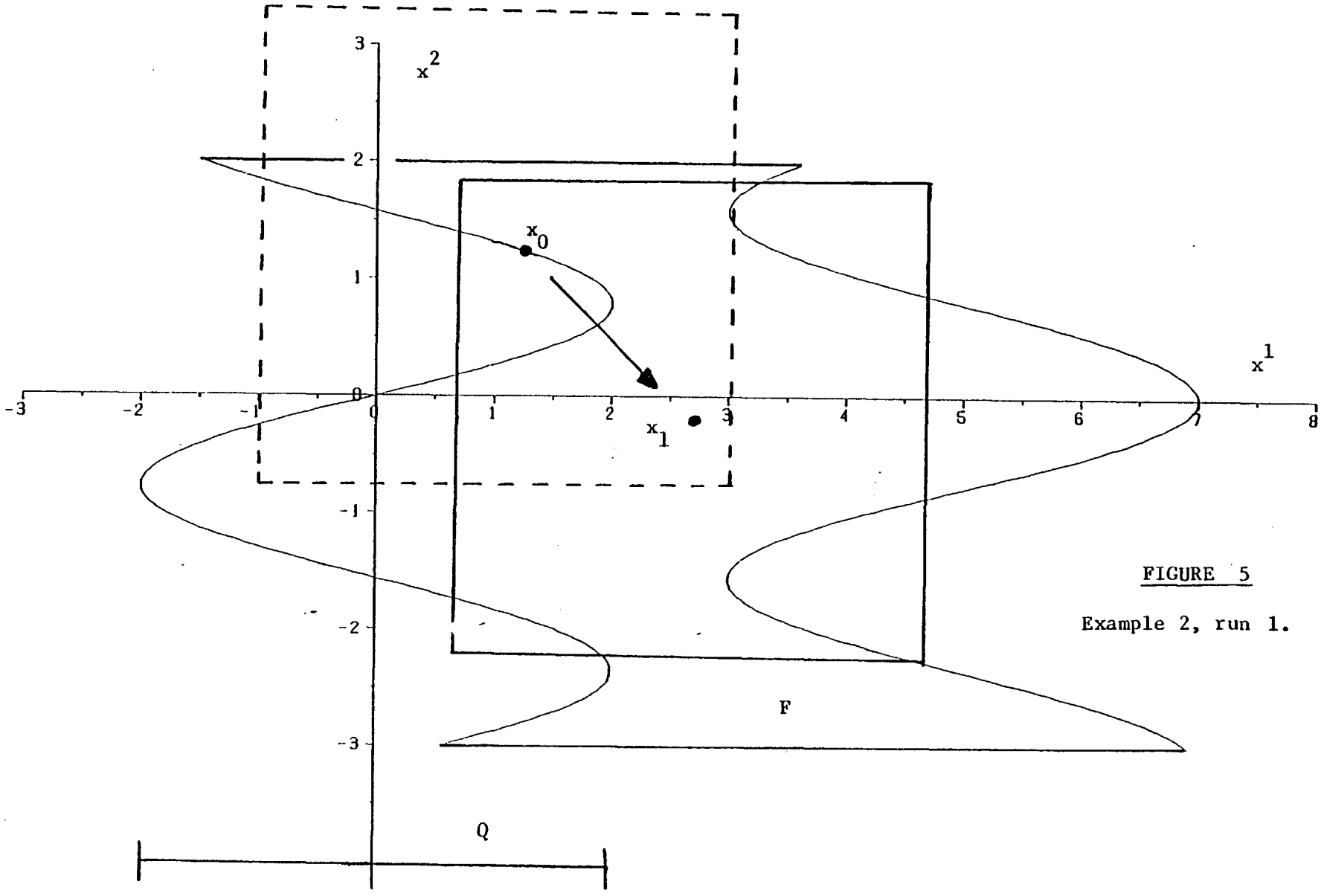


FIGURE 5

Example 2, run 1.

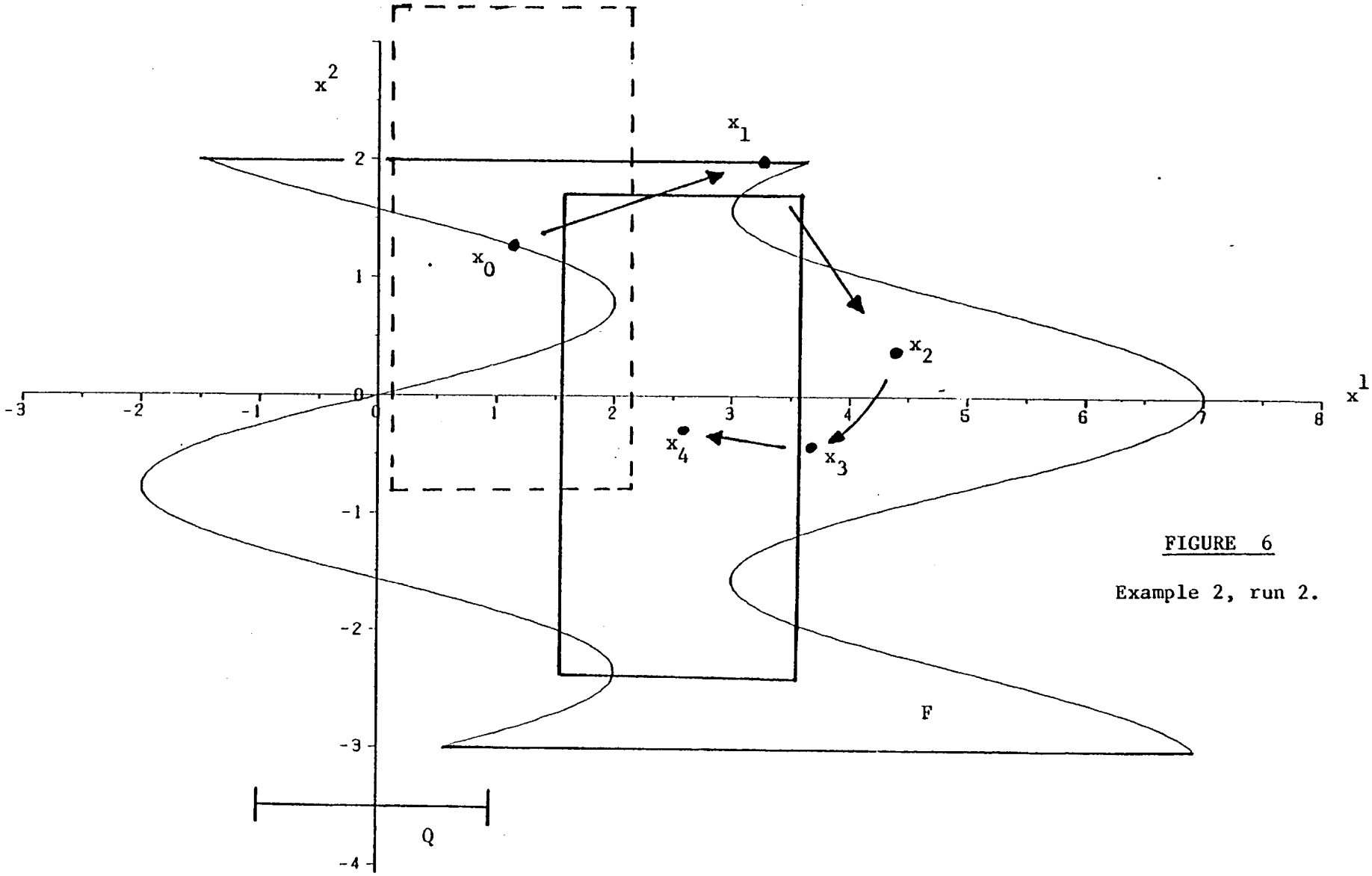


FIGURE 6

Example 2, run 2.

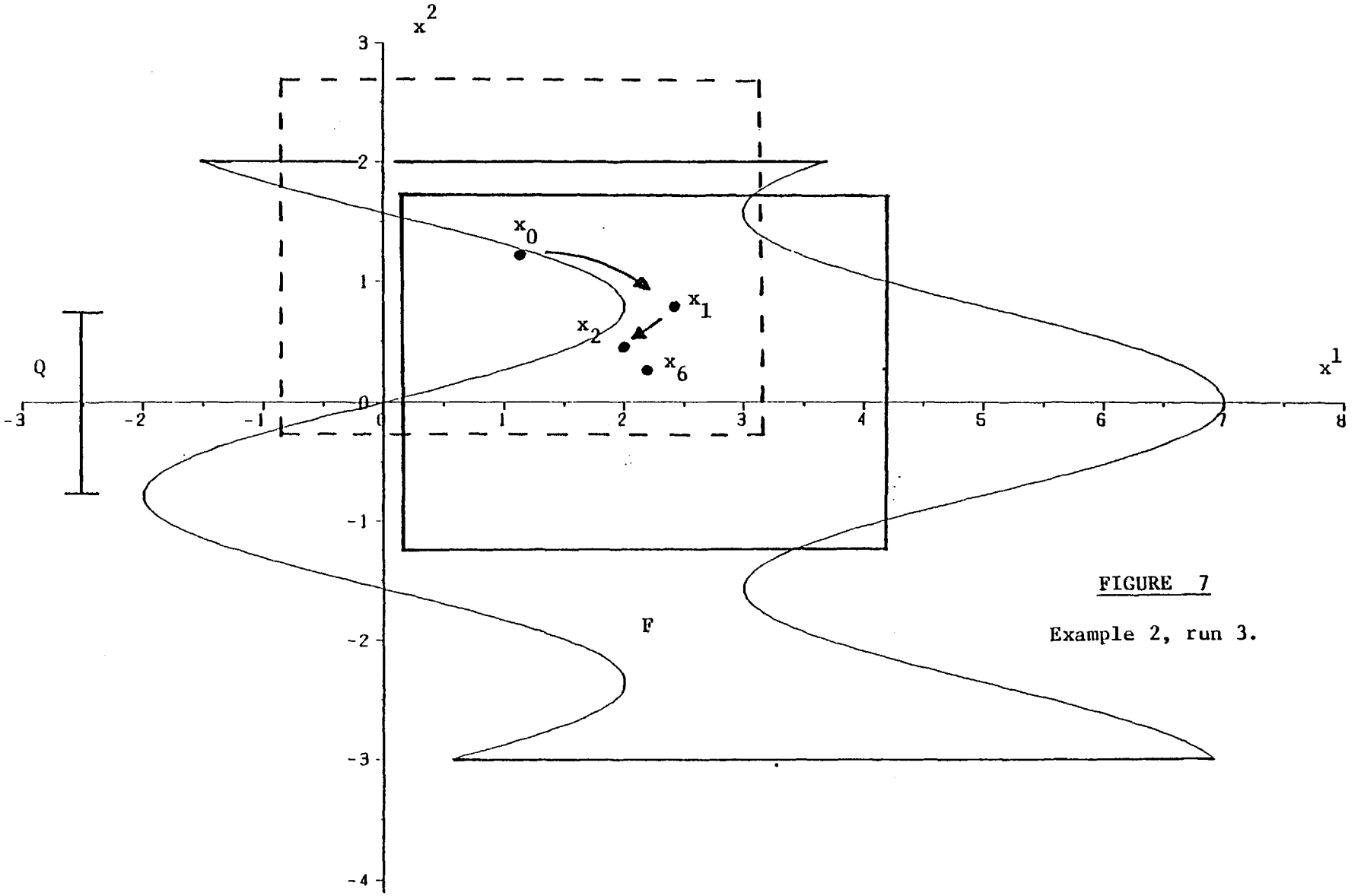


FIGURE 7

Example 2, run 3.

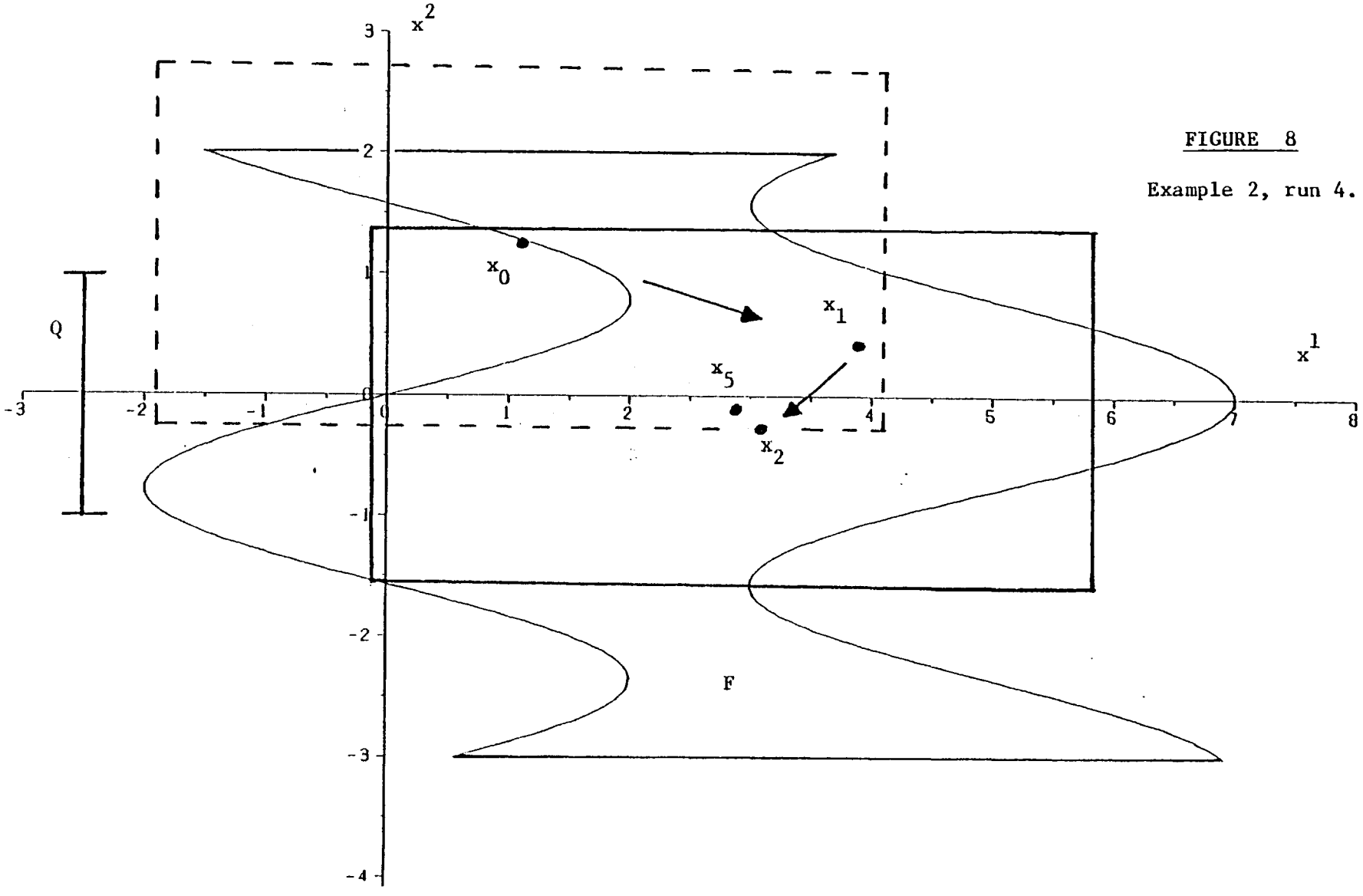


FIGURE 8

Example 2, run 4.

Example 3

The $F \subset R^2$ is defined as in example 3 of Chapter III. All the conventions and parameters are as in examples 1 and 2. Table 11 lists the tolerance and tuning regions for the four runs presented. Tables 12, 13, 14 and 15 summarize the results which are illustrated in figures 9,10,11 and 12 respectively. Separator re-estimationshave been performed in runs 1,2 and 3 when k=3.

Run	TOLERANCES		Tuned parameter	Tuning region
	x^1	x^2		
1	2.50	2.00	x^1	$\pm$ 1.00
2	2.50	1.50	x^1	$\pm$ 0.75
3	2.50	1.50	x^2	$\pm$ 1.00
4	2.00	1.00	x^2	$\pm$ 0.75

TABLE 11

TABLE 12 (Run 1, Initial point (0,0))

k	$\sigma(k)$	$\bar{\tau}(k)$	x_k		$\bar{\delta}_k(x_k)$	$\bar{\delta}_3(x_k)$
			x^1	x^2		
0	4	3	0.000	0.000	0.688	-
1	5	5	0.000	2.302	0.344	-
2	6	7	0.827	2.948	0.292	0.301
3	7	9	1.649	2.194	0.000	
4	14	11	"	"	0.071	
5	15	13	1.446	2.387	0.000	
6	30	15	"	"	0.037	
7	31	17	1.345	2.274	0.000	
8	62	19	"	"	"	
9	100	21	1.345	2.274	0.000	

k	$\sigma(k)$	$\bar{\tau}(k)$	x_k		$\bar{\delta}_k(x_k)$	$\bar{\delta}_3(x_k)$
			x^1	x^2		
0	4	3	2.000	0.000	0.625	-
1	5	5	0.329	1.386	0.216	-
2	6	7	1.239	1.386	0.194	0.155
3	7	9	0.712	1.824	0.155	
4	8	11	1.296	2.287	0.000	
5	16	13	"	"	"	
6	32	15	"	"	"	
7	64	17	"	"	"	
8	100	19	1.296	2.287	0.000	

TABLE 13 (Run 2, Initial point (2,0))

k	$\sigma(k)$	$\bar{\tau}(k)$	x_k		$\bar{\delta}_k(x_k)$	$\bar{\delta}_3(x_k)$
			x^1	x^2		
0	4	3	1.000	1.000	0.638	-
1	5	5	2.705	2.413	0.472	0.478
2	6	7	1.440	3.462	0.706	0.620
3	7	9	-0.064	2.280	0.306	
4	8	11	1.165	2.375	0.000	
5	16	13	"	"	0.104	
6	17	15	0.917	2.146	0.000	
7	34	17	"	"	"	
8	68	19	"	"	0.028	
9	69	21	1.035	2.214	0.000	
10	100	23	"	"	0.010	
11	100	25	1.002	2.183	0.000	

TABLE 14 (Run 3, Initial point (1.1))

k	σ(k)	τ(k)	x_k		$\bar{\delta}_k(x_k)$
			x^1	x^2	
0	4	3	1.000	1.000	0.600
1	5	5	2.262	1.994	0.000
2	10	7	"	"	0.127
3	11	9	1.987	2.212	0.000
4	22	11	"	"	"
5	44	13	"	"	"
6	88	15	"	"	"
7	100	17	1.987	2.212	0.000

TABLE 15 (Run 4, Initial point (1.1))

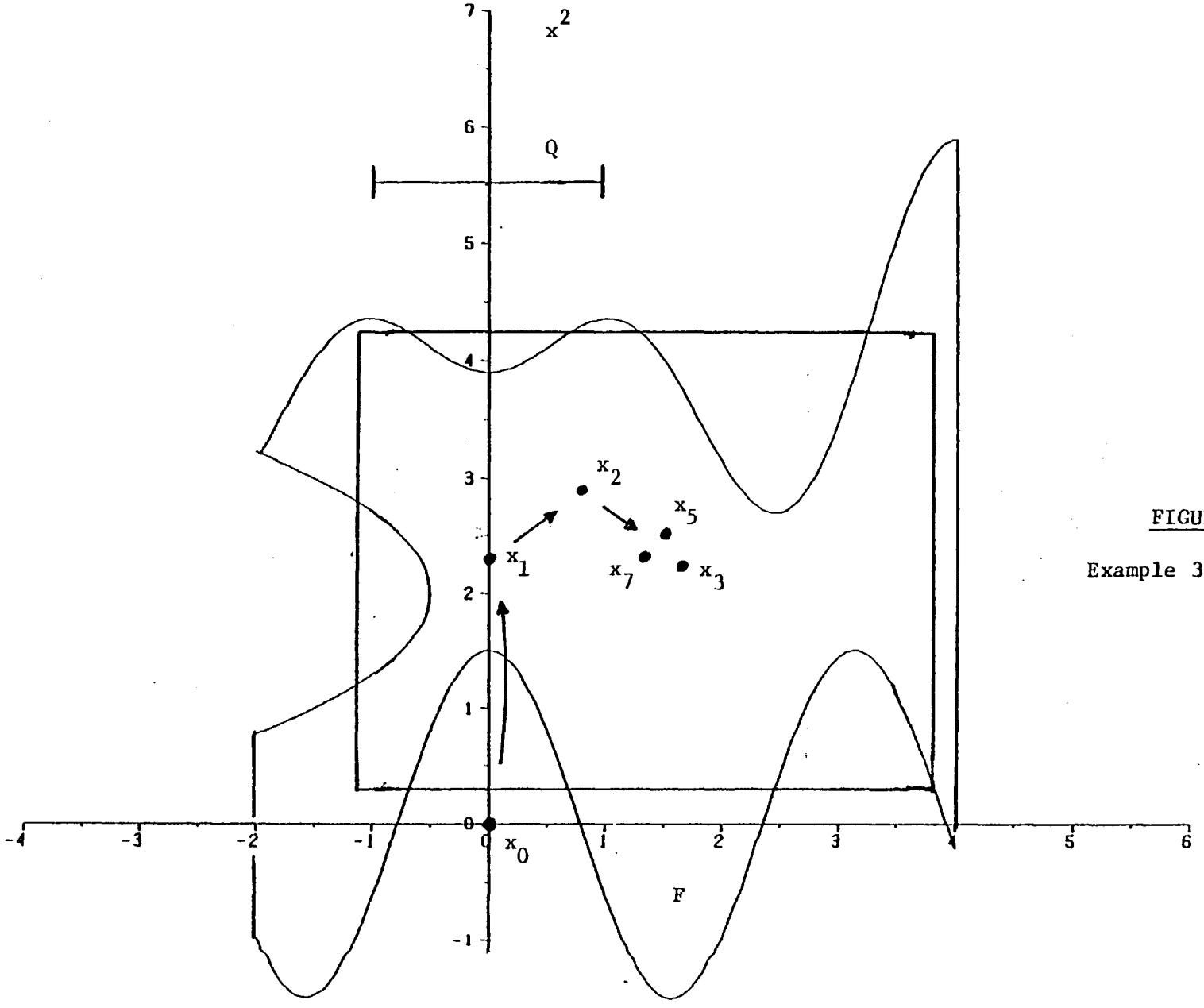


FIGURE 9

Example 3, run 1.

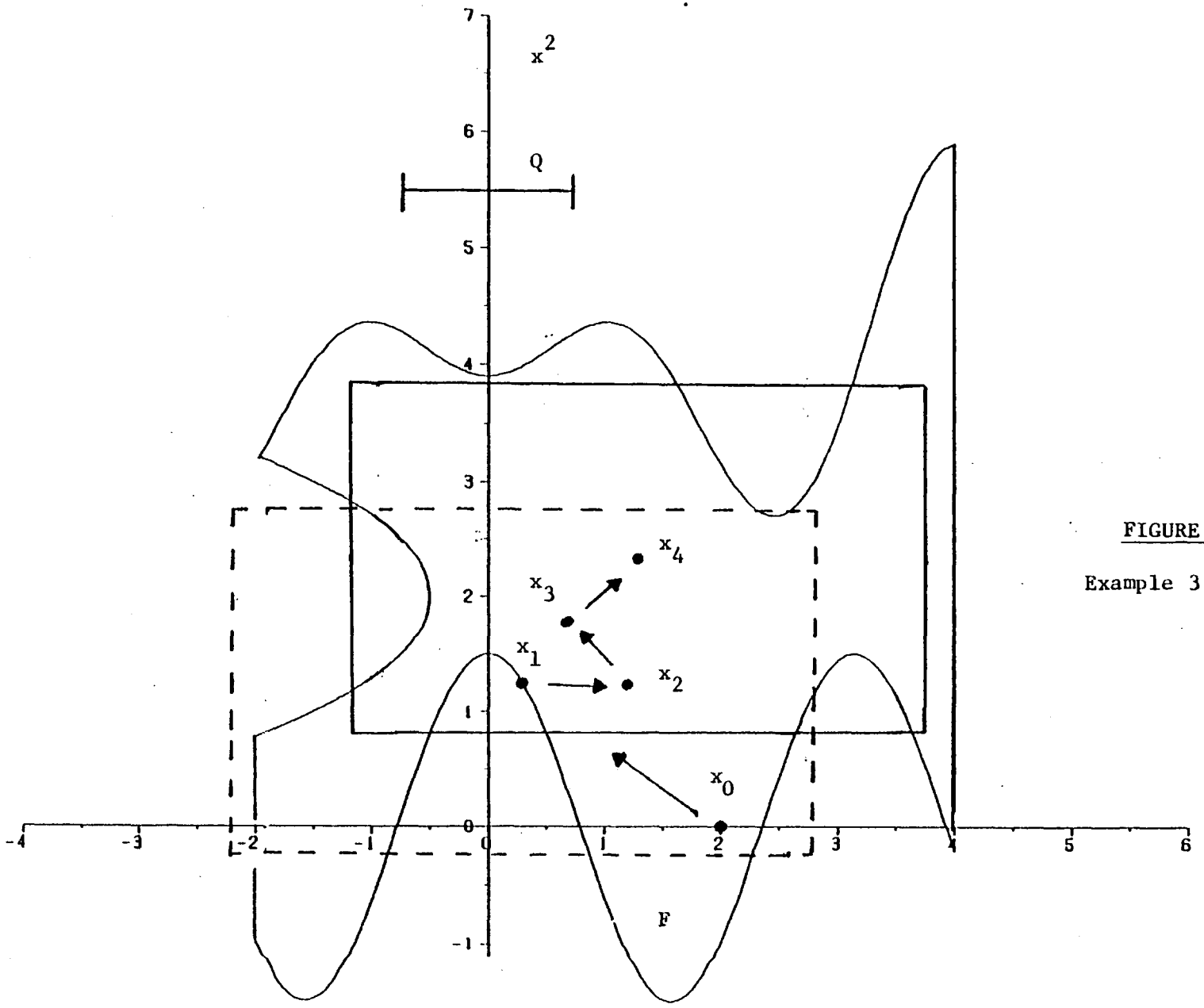


FIGURE 10

Example 3, run 2.

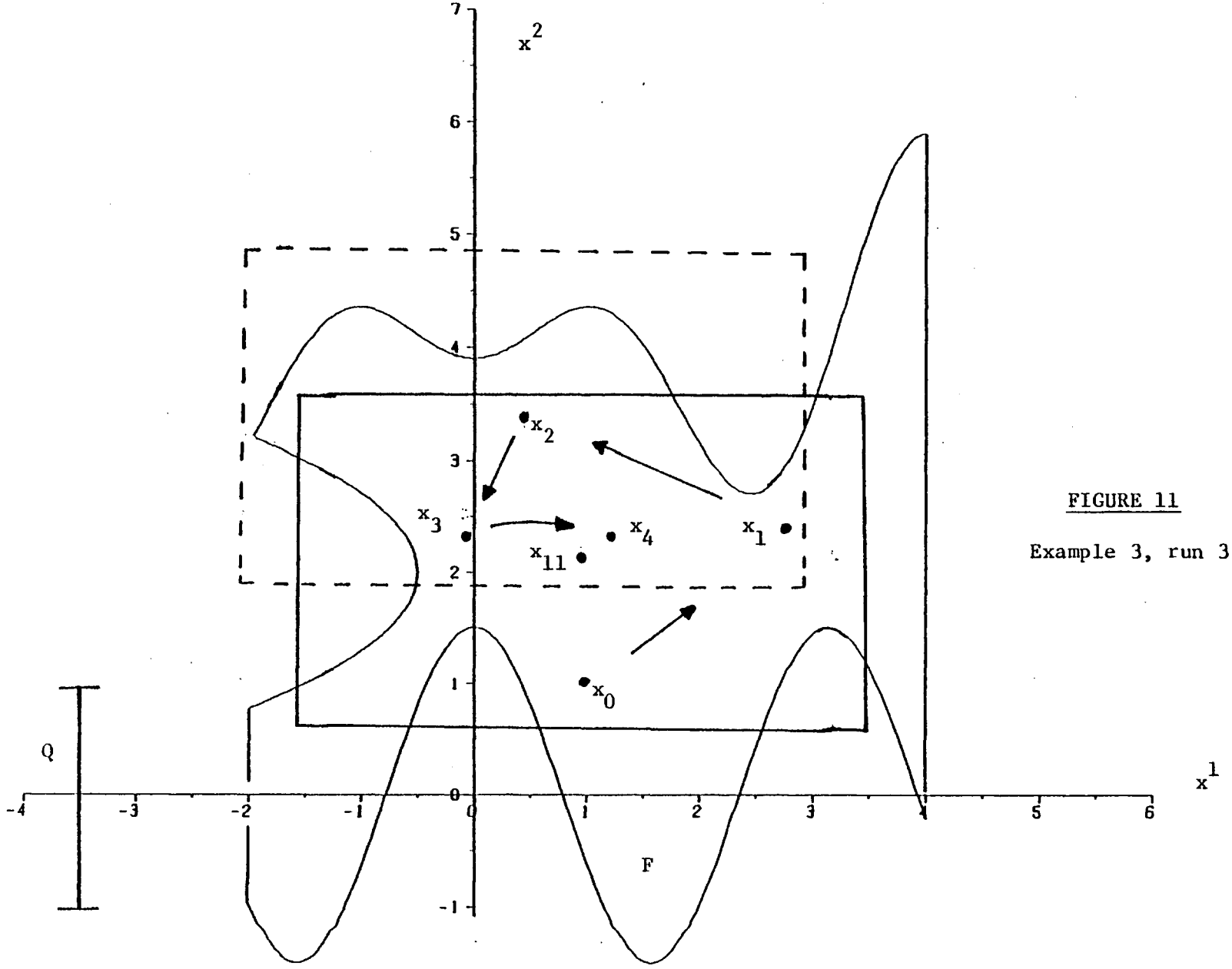


FIGURE 11

Example 3, run 3.

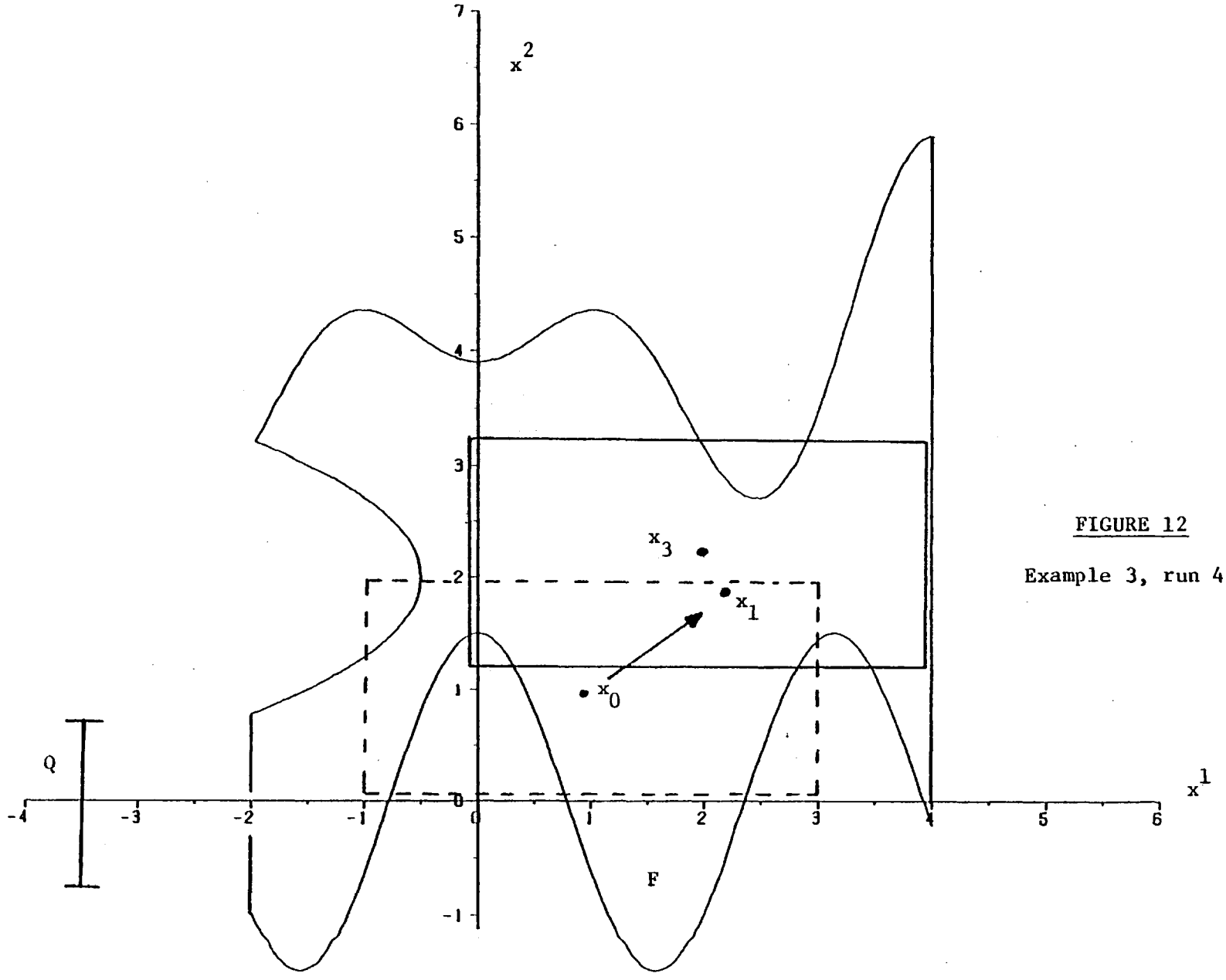


FIGURE 12

Example 3, run 4.

Example 4

This is the three-dimensional low-pass filter problem examined in Chapter III (example 5 of Chapter III). The cases of either L_1 or C tuned have been considered. Procedure 2 of section 3 was employed for the computation of the separator estimates. The points of each set $Z_k(x)$ were generated by a pseudo-random number generator and the truncation function $\tau(k)$ defined in section 3 of Chapter III was utilized. A linear searching technique was also employed to improve the separator estimates. The tuning region Q was discretized using method 1 of section 3, as in examples 1,2 and 3 ($\bar{\tau}(k) = 2k+3$). No conventional constraints were employed at all. The techniques discussed in section 3 concerning the re-estimation of separators and the utilization of the overlap of successive tolerance regions were not employed, so that great improvements on the computational results presented below should be possible (all separator re-estimations were done from scratch without using any previous information). In a practical design procedure a suitable initial point for the tolerance-tuning algorithm would be some nominal point obtained from solving a pure tolerance problem. However the points obtained in Chapter III turned out to be very nearly solutions of the tolerance-tuning problem as well, so that to demonstrate the properties of the algorithm different initial points were chosen. Table 16 summarizes the results of six runs and tables 17,18,19,20,21 and 22 contain details of each run. All the percentages shown are with respect to the final parameter values. The final yields were estimated by Monte Carlo analysis involving at least 400 points and with the tuning region discretized by considering $\bar{\tau}(IT)$ points, where IT denotes the total number of iterations performed by the algorithm for each run. Hence

the yield estimates should be smaller than the exact values.

N denotes the total number of circuit analyses required by the algorithm, where as N(k) the number of circuit analyses performed at iteration k.

Note that in this example no separator re-estimations were necessary.

It can be observed that the introduction of tuning allows considerable increases in the parameter tolerances (see run 4 of example 5 in Chapter III). Most of the feasibility subproblems were solved in one iteration.

Run	TOLERANCES			Tuned parameter	Tuning region	IT	Final Yield	N
	L_1(H)	L_2(H)	C(F)					
1	0.420 20.0%	0.380 20.8%	0.100 11.1%	L_1	±0.50 ±23.8%	3	99.5%	446
2	0.280 15.6%	0.280 13.3%	0.090 10.1%	L_1	±0.20 ±9.5%	3	99.5%	294
3	0.230 12.1%	0.250 11.9%	0.083 9.4%	L_1	±0.10 ±5.3%	3	99.3%	180
4	0.400 22.2%	0.400 21.9%	0.200 17.0%	C	±0.35 ±29.6%	2	99.7%	279
5	0.300 15.7%	0.300 15.6%	0.145 15.0%	C	±0.145 ±15.0%	2	99.9%	147
6	0.260 13.3%	0.260 13.2%	0.100 10.8%	C	±0.075 ±8.1%	2	99.9%	166

TABLE 16

k	x_k			$\bar{\delta}_k(x_k)$	$\tau(k)$	$\bar{\tau}(k)$	N(k)
	L_1(H)	L_2(H)	C(F)				
0	2.150	2.150	1.070	0.808	100	3	36
1	2.264	1.988	0.938	0.300	200	5	112
2	2.155	1.883	0.857	0.171	300	7	298
3	2.100	1.824	0.901				

TABLE 17 (Run 1)

k	x_k			$\bar{\delta}_k(x_k)$	$\tau(k)$	$\bar{\tau}(k)$	N(k)
	L_1(H)	L_2(H)	C(F)				
0	2.150	2.150	1.000	0.688	100	3	15
1	1.974	2.299	0.873	0.447	200	5	74
2	1.868	2.174	0.950	0.283	300	7	205
3	1.795	2.102	0.894				

TABLE 18 (Run 2)

k	x_k			$\bar{\delta}_k(x_k)$	$\tau(k)$	$\bar{\tau}(k)$	N(k)
	L_1(H)	L_2(H)	C(F)				
0	2.150	2.150	1.000	0.816	100	3	16
1	2.046	2.278	0.896	0.386	200	5	55
2	1.961	2.176	0.944	0.352	300	7	109
3	1.896	2.098	0.887				

TABLE 19 (Run 3)

k	x_k			$\bar{\delta}_k(x_k)$	$\tau(k)$	$\bar{\tau}(k)$	N(k)
	L_1(H)	L_2(H)	C(F)				
0	2.150	2.150	0.900	0.729	100	3	32
1	1.887	1.918	1.108	0.231	200	5	247
2	1.798	1.828	1.179				

TABLE 20 (Run 4)

k	x_k			$\bar{\delta}_k(x_k)$	$\tau(k)$	$\bar{\tau}(k)$	N(k)
	L_1(H)	L_2(H)	C(F)				
0	2.200	2.200	0.900	0.708	100	3	29
1	2.008	2.031	1.050	0.369	200	5	118
2	1.910	1.927	0.968				

TABLE 21 (Run 5)

k	x_k			$\bar{\delta}_k(x_k)$	$\tau(k)$	$\bar{\tau}(k)$	N(k)
	L_1(H)	L_2(H)	C(F)				
0	2.200	2.200	0.900	0.563	100	3	52
1	2.056	2.076	1.006	0.439	200	5	114
2	1.956	1.970	0.925				

TABLE 22 (Run 6)

Example 5

This is the seven-dimensional high-pass filter example examined in Chapter III (example 6 of Chapter III). The cases of x_3 or x_6 tuned have been considered. All the conventions and parameters are as in example 4. Table 23 summarizes the results of four runs. Tables 24, 25, 26 and 27 show in detail the progress of the algorithm for these runs. Note that separator re-estimations have been performed in runs 1,2, and 4 when k=3 and these have caused the values of N(k) and N to increase. However, by adopting the techniques of section 3, it should be possible to greatly improve on these results.

Run	Tolerances							Tuned Parameter	Tuning Region	IT	Final Yield	N
	x_1 (nF)	x_2 (nF)	x_3 (H)	x_4 (nF)	x_5 (nF)	x_6 (H)	x_7 (nF)					
1	0.87	2.90	0.30	0.80	7.00	0.21	1.20	x_3	0.30	4	99.4%	458
	8.3%	7.9%	8.0%	8.5%	7.9%	7.3%	7.6%		±8.0%			
2	0.70	2.30	0.25	0.65	5.80	0.20	1.00	x_3	0.16	4	99.6%	448
	6.5%	6.6%	6.1%	7.2%	6.6%	6.7%	6.2%		±3.9%			
3	0.95	2.50	0.26	0.75	6.80	0.22	1.20	x_6	0.35	3	100.0%	241
	8.0%	6.7%	6.8%	7.8%	7.3%	7.7%	8.6%		±12.2%			
4	0.74	2.30	0.24	0.60	6.00	6.18	0.90	x_6	0.18	4	99.6%	259
	5.9%	6.5%	6.1%	6.2%	6.3%	6.4%	6.1%		±6.4%			

TABLE 23

k	x_k							$\bar{\delta}_k(x_k)$	$\bar{\delta}_3(x_k)$	$\tau(k)$	$\bar{\tau}(k)$	N(k)
	x_1 (nF)	x_2 (nF)	x_3 (H)	x_4 (nF)	x_5 (nF)	x_6 (H)	x_7 (nF)					
0	11.00	36.00	4.00	11.00	90.00	3.20	15.00	0.733	0.457	100	3	37
1	12.10	34.76	3.84	9.87	91.86	3.07	16.11	0.461	0.610	200	5	59
2	11.46	35.68	3.76	10.48	90.83	2.99	16.77	0.348	0.370	300	7	57
3	11.02	36.28	3.71	10.05	90.02	2.94	16.35	0.284		400	9	305
4	10.43	36.92	3.77	9.46	88.86	2.87	15.76					

TABLE 24 (Run 1)

k	x_k							$\bar{\delta}_k(x_k)$	$\bar{\delta}_3(x_k)$	$\tau(k)$	$\bar{\tau}(k)$	N(k)
	x_1 (nF)	x_2 (nF)	x_3 (H)	x_4 (nF)	x_5 (nF)	x_6 (H)	x_7 (nF)					
0	11.00	36.00	4.00	11.00	90.00	3.20	15.00	0.360	-	100	3	19
1	11.87	35.11	4.09	10.14	88.47	3.11	15.87	0.369	-	200	5	54
2	11.35	35.68	4.15	9.62	87.36	3.05	16.40	0.421	0.448	300	7	71
3	10.87	35.01	4.09	9.15	88.11	3.00	15.92	0.092		400	9	304
4	10.79	34.84	4.07	9.02	87.96	2.98	16.05					

TABLE 25 (Run 2)

k	x_k							$\bar{\delta}_k(x_k)$	$\tau(k)$	$\bar{\tau}(k)$	N(k)
	x_1 (nF)	x_2 (nF)	x_3 (H)	x_4 (nF)	x_5 (nF)	x_6 (H)	x_7 (nF)				
0	13.00	36.00	4.00	11.40	90.00	3.00	15.00	0.601	100	3	39
1	11.55	37.55	3.85	9.96	92.72	2.85	13.54	0.140	200	5	117
2	11.73	37.79	3.84	9.79	92.38	2.83	13.72	0.185	300	7	85
3	11.94	37.50	3.81	9.58	92.61	2.86	13.94				

TABLE 26 (Run 3)

k	x_k							$\bar{\delta}_k(x_k)$	$\bar{\delta}_3(x_k)$	$\tau(k)$	$\bar{\tau}(k)$	$N(k)$
	x_1 (nF)	x_2 (nF)	x_3 (H)	x_4 (nF)	x_5 (nF)	x_6 (H)	x_7 (nF)					
0	13.00	36.00	4.00	11.00	93.00	3.00	15.00	0.492	-	100	3	11
1	13.61	35.32	3.91	10.38	94.07	2.93	15.61	0.343	-	200	5	28
2	12.88	36.10	3.99	9.64	95.45	2.85	14.87	0.176	0.268	300	7	93
3	12.69	35.83	3.96	9.83	95.17	2.83	15.07	0.216		400	9	127
4	12.48	35.56	3.93	9.62	94.86	2.80	14.85					

TABLE 27 (Run 4)

V.5 Discussion.

In Chapter IV, the ideas presented in Chapter II for the pure tolerance problem are extended to the case in which tuning is also possible. Hence a cut map algorithm for the tolerance-tuning problem has been obtained. The algorithm shares many characreristics with the algorithms of Chapter II. The subproblem of Step 1 has the same simple form and can be very efficiently solved by the modified Newton algorithm of [22]. Conventional constraints either relating to the magnitudes of the parameters or involving the vertices of the tolerance region can be included to improve the convergence properties. The initial point rule increases the probability of convergence, especially when no conventional constraints are present. In such cases the derivatives of the constraints are not required and only "pass-fail" tests are performed in Step 2.

The main difference between the algorithm of Chapter IV and those in Chapter II is that the separator estimates are not necessarily smaller than or equal to their exact values. To avoid poor convergence properties it is thus necessary to keep re-estimating the separators of the active points. Hence the cut dropping scheme keeps the complexity of the subproblem of step 1 low and also ensures that a small number of re-estimations is performed. The optimum frequency of these re-estimations depends on the dimension of the tuning region Q. If, as in many practical problems, the number of tuned parameters is small, a discretization of Q involving only a few points is enough to ensure a good approximation to the set Y and hence to generate separator estimates that are not much larger than the exact values. In such cases few re-estimations will be necessary.

As in the case of the pure tolerance algorithms, computational efficiency is very much dependent on the way the algorithm is implemented. By employing the techniques of section 3 in an interactive program that allows the specification of the truncation functions, the frequency of separator re-estimations and the conventional constraints utilized as the algorithm progresses much better computational results than those presented should be possible. In Chapter VI comparisons between the cut-map algorithm for the tolerance-tuning problem and the rest of the methods in the literature are made.

CHAPTER VI

CONCLUSION

In this thesis specialized algorithms for the infinitely constrained tolerance and tolerance-tuning problems that belong to the class of cut map algorithms of Eaves and Zangwill, have been proposed. The algorithms possess the following general features.

(i) They solve the general non-convex problems P_T and $P_{T,Q}$ and have established convergence properties.

(ii) They are directly implementable since specific truncation rules are given for every infinite operation.

(iii) They approximate the sets of solutions of the problems by the complement of the union of a finite number of very simply described regions (balls). A consequence of this is that the feasibility subproblem at each iteration can be solved very efficiently by standard algorithms, since the evaluation of the design constraints and their derivatives is not required. The initial point rule proposed exploits the geometrical structure of the problem and usually causes rapid convergence.

(iv) The global optimization procedures proposed for the generation of the separator estimates require only "pass-fail" tests and are computationally cheap at low yield points. By employing the techniques described in

Chapters III and V in an interactive programme large increases in yield should be efficiently obtained.

(v) They incorporate a cut dropping scheme to keep the complexity of the feasibility subproblem low.

(vi) They are very suitable for interactive computer-aided design and any extra knowledge about a specific problem can be easily utilized by introducing conventional constraints either involving the vertices of the tolerance region or relating to the parameter magnitudes. The introduction of such constraints improves the convergence properties of the algorithms since it ensures that better approximations to the sets of solutions of the problems are generated.

(vii) They are simple to code.

It should be noted that, as one expects from algorithms with established convergence properties, the cut map algorithms proposed will not converge if the set of solutions is empty (i.e. if 100% yield is not possible). However if (some) vertex constraints are included, they will tend to improve yield and then jam, being unable to solve the subproblem in Step 1.

Most, if not all, of the deterministic methods for the pure tolerance problem in the literature rely on one-dimensional convexity both for their convergence and their efficiency. In Bandler's paper [8], the method of choosing the "worst case"

vertices is not described but seems to rely on a priori information about each particular problem. In [9] , an interval arithmetic approach is presented for solving the worst case problem (WCP) in the general case and a more efficient algorithm for the convex case. The algorithms proposed for the fixed and variable tolerance problems do not have established convergence properties and it seems that jamming is possible. The most advanced and efficient of all the deterministic methods in the literature seems to be the quasi-Newton algorithm proposed by Brayton et al , [10]. Because of the complexity of the procedure utilized to form and update the vertex list, the algorithm is not explicitly stated but only vaguely described. Convergence proofs are not presented and hence it is not guaranteed that cycling will not occur. In the most complex example in this paper 67 circuit analyses and 67 gradient evalutions were required. In the filter examples of Chapter III, 10-200 circuit analyses and less than 10 gradient evaluations were required. Direct comparison with the cut map algorithm is difficult because the examples differ considerably and final yields are not specified in [10].

To summarize, the advantages of the cut map algorithm when compared with all the above methods are that it is suitable for non-convex problems (see for example the strongly non-convex examples of Chapter III), that it has established convergence properties, that it can operate by performing "pass-fail" tests only (i.e. not requiring any gradient evaluations) and that it is very simple to code. It seems that when efficiently implemented the cut map algorithm warrants consideration even for convex problems.

The only other algorithm with established convergence properties that is suitable for the problem P_T seems to be the outer approximations algorithm of [4] . This algorithm should be computationally more expensive than the cut map algorithm for the specialized problem P_T for two reasons. Firstly, it requires repeated estimation of the function $\theta_T(x)$ which is more expensive than the estimation of $\delta(x)$, especially at low yield points (the difference in computational effort between these estimations decreases as x approaches G). Secondly the feasibility subproblem to be solved at each iteration requires repeated evaluation of the design constraints and their derivatives, which implies that the computational effort is larger and the probability of jamming is greater. However, the outer approximations algorithm is more flexible since it solves the general infinitely constrained engineering design problem.

The algorithm of Chapter IV for the problem $P_{T,Q}$ is very similar to the algorithm for the problem P_T. The difference is that the separator estimates are not always smaller than their exact values. Hence it is necessary to keep re-estimating the separators of the active cuts to avoid poor convergence properties. For computational efficiency, it is important to specify the frequency of these re-estimations interactively according to the progress of the algorithm. If the number of tuned parameters is small, a discretization of Q involving only a few points is enough to ensure a good approximation to the set Y. In such cases the operation of the algorithm for the problem $P_{T,Q}$ will be very similar to that of the algorithm for P_T since few or no separator re-estimations will be necessary. Note that the results of Chapters IV and V can be easily extended to cases in which Q has a different form (e.g. when correlated tuning is only permitted).

As opposed to the pure tolerance case, very few methods exist for the solution of the tolerance-tuning problem. Bandler's approach, [8] is restricted to the one-dimensionally convex case and employs heuristics to improve efficiency. Note that it is precisely in strongly non-convex problems that one expects to obtain large increases in the parameter tolerances by the introduction of tuning (e.g. when the feasible set possesses "black holes"). Polak's outer approximations algorithm [19] should be computationally more expensive than the cut-map algorithm for the specialized problem $P_{T,Q}$ for two reasons. Firstly it requires repeated estimation of the function $\theta_{T,Q}(x)$ which is more expensive than the estimation of $\delta(x)$ at both low and high yield points (this is because testing if a point x lies in Y_j or not is usually cheaper than evaluating $\phi_j(x)$; see (10) of Chapter V). Secondly the feasibility subproblem requires repeated evaluation of the design constraints and their derivatives. However the algorithm in [19] is more flexible since it solves the general infinitely constrained engineering design problem with tuning also present. Direct numerical comparison between the three methods is not possible since no numerical results are presented in [19] and those in [8] do not specify the number of circuit analyses.

It is our hope that the simple and robust cut map algorithms proposed in this thesis will prove to be useful for tackling the computationally complex but important tolerance and tolerance-tuning problems.

REFERENCES

1. Becker R.G., Heunis A., Mayne D.Q., "Computer-aided Design of Control Systems via Optimization". Proceedings IEE, Vol. 126, No.6, June 1979.

2. Polak, E. and Mayne D.Q., "An Algorithm for Optimization Problems with Functional Inequality Constraints", IEEE Transactions on Automatic Control, Vol. AC-21, No.2, April 1976.

3. Gonzaga C., Polak E. and Trahan R., "An Improved Algorithm for Optimization Problems with Functional Inequality Constraints", IEEE Transactions on Automatic Control, Vol. AC-25, No.1, February 1980.

4. Mayne D.Q., Polak E. and Trahan R., "An Outer Approximations Algorithm for Computer-Aided Design Problems", JOTA, Vol.28, No.3, 1979.

5. Gonzaga C. and Polak E., "On Constraint Dropping Schemes and Optimality Functions for a Class of Outer Approximations Algorithms", SIAM J. on Control and Optimization, 17, No.4. 1979

6. Bandler J.W., "Optimization of Design Tolerances Using Non-linear Programming", JOTA, Vol. 14, No.1, July 1974.

7. Bandler J.W., Liu P.C. and Chen J.H.K., "Worst Case Network Tolerance Optimization", IEEE Transactions on Microwave Theory and Techniques", Vol. MTT-23 No.8, 1975.

8. Bandler J.W., Liu P.C. and Tromp H., "Non-linear Programming Approach to Optimal Design Centering, Tolerancing and Tuning", IEEE Transactions on Circuits and Systems, Vol. CAS-23, No.3 March 1976.

9. Schjaer-Jacobsen H. and Madsen K., "Algorithms for Worst-case Tolerance Optimization", IEEE Transactions on Circuits and Systems, Vol. CA5-26, No.9, Sept. 1979.

10. Brayton R.K., Director S.W.,Hachtel G.D. and Vidigal L.M., "A New Algorithm for Statistical Circuit Design Based on Quasi-Newton Methods and Function Splitting", IEEE Transactions on Circuits and Systems, Vol. CAS-26, No.9, September 1979.

11. Director S.W. and Hachtel G.D., "The Simplicial Approximation Approach to Design Centering", IEEE Transactions on Circuits and Systems, Vol. CAS-24, No.7, July 1977.

12. Brayton, R.K., Director S.W., and Hachtel G.D., "Yield Maximization and Worst-Case Design with Arbitrary Statistical Distributions", IEEE Transactions on Circuit and Systems, Vol. CAS -27, No.9, September 1980.

13. Iyer R.K. and Downs T., "A Variance Minimization Approach to Tolerance Design", IEEE Transactions on Circuits and Systems, Vol. CAS-27, No.9, Sept. 1980.

14. Pinel J.F. and Roberts K.A., "Tolerance Assignment in Linear Networks Using Nonlinear Programming", IEEE Transactions on Circuit Theory, Vol. 19, No.5, September 1972.

15. Soin R.S. and Spence R., "Manufacturing Yield Optimization by Statistical Exploration", Proc. of the Conference on the CAD of Electronic Components, Circuits and Systems, Brighton 1979, (IEE Conference publication 175).

16. Pinel J.F., "Computer-Aided Network Tuning", IEEE Transactions on Circuit Theory, January 1971.

17. Lopresti P.V., "Optimum Design of Linear Tuning Algorithms", IEEE Transactions on Circuits and Systems, Vol. CAS-24, No.3, March 1977.

18. Polak E. and Sangiovanni-Vincetelli A., "Theoretical and Computational Aspects of the Optimal Design Centering, Tolerancing and Tuning Problem", IEEE Transactions on Circuits and Systems, Vol. CAS-26, No.9, September 1979.

19. Polak E., "An Implementable Algorithm for the Optimal Design Centering, Tolerancing and Tuning Problem", Dept. of Electrical Engineering and Computer Sciences and the Electronics Research Laboratory, University of California, Berkeley, California 94720.

20. Eaves B.C. and Zangwill W.I., "Generalized Cutting Plane Algorithms", SIAM J. on Control, Vol.9, No.4, 1971.

21. Hogan W.W., "Applications of a General Convergence Theory for Outer Approximations Algorithms", Mathematical Programming 5 (1973).

22. Mayne D.Q., Polak E. and Heunis A., "Solving Non-linear Inequalities in a Finite Number of Iterations", Publication No: 79/3, Dept. of Computing and Control, Imperial College, London SW.7. England.

23. Dixon L.C.W. and Szego G.P., "Towards Global Optimization", North-Holland Publishing Co. 1975.

24. Dixon L.C.W. and Szego G.P., "Towards Global Optimization II", North Holland Publishing Co.

25. Dixon L.C.W., Spedicato E. and Szego G.P., "Non-linear Optimization Theory and Algorithms", Birkhauser 1980.

26. Polak E., "Computational Methods in Optimization", Academic Press, 1971.

27. Rudin W.,"Principles of Mathematical Analysis", McGraw Hill, 1976.

28. Berge C.,"Topological Spaces", Oliver and Boyd Ltd., 1963.

APPENDIX I

We now briefly state the feasibility algorithm of [22] for completeness. Suppose that we want to find a point in the set W defined by :

$$W \triangleq \{x \in R^n \mid g^j(x) \leq 0, \quad j=1,.....,m\} . \tag{A.1}$$

Let:

$$\psi(x) \triangleq \max \{g^j(x) \mid j = 1,....,m\} \tag{A.2}$$

$$\hat{\psi}(x,p) \triangleq \max\{g^j(x)+ \langle \nabla g^j(x),p\rangle \mid j = 1,...,m\} \tag{A.3}$$

$$\theta(x,p) \triangleq \hat{\psi}(x,p) - \psi(x). \tag{A.4}$$

For any $x \in R^n$, a Newton step is any vector in the solution set of the following linear program $L^1(x)$:

$$\min_p \{ \| p \|_\infty \mid \hat{\psi}(x,p) \leq 0\} . \tag{A.5}$$

For any $x \in R^n$, any Newton step p^1 we define an additional step p^2 which is any vector in the solution set of the following linear program $L^2(x,p^1)$:

$$\theta^2(x,p^1) \triangleq \min_p\{ \theta(x,p^1+p) \mid \| p \|_\infty \leq \varepsilon \} \tag{A.6}$$

for some $\varepsilon > 0$. Hence for any $x \in R^n$ the modified Newton step generated by the algorithm is :

$$p = p^1 + p^2. \tag{A.7}$$

The Newton step may not exist or may be unsatisfactory. In such cases the algorithm employs a first order step p^3 which is any vector in the solution set of the following linear program $L^3(x)$:

$$\theta^3(x) = \min_p \{\theta(x,p) \mid \|p\|_\infty \leq 1\} . \qquad (A.8)$$

A standard test on the magnitude of the Newton step is employed to judge whether it is satisfactory or not. We are now in a position to state our algorithm.

Algorithm A1

Parameters: $\gamma \in (0,\frac{1}{2})$, $\beta \in (0,1)$, $\varepsilon > 0$, $L > 1$.

Data: $x_0 \in R^n$

Step 0: Set i=0.

Step 1: If $\psi(x_i) \leq 0$ stop.

Step 2: Solve $L^1(x_i)$ to obtain p^1. If a solution exists and $\|p^1\|_\infty \leq L$, solve $L^2(x_i, p^1)$ to obtain p^2. Set $p_i = p^1 + p^2$.
If $L^1(x_i)$ has no solution or $\|p^1\|_\infty \geq L$ solve $L^3(x_i)$ to obtain p^3 and set $p_i = p^3$.

Step 3: Determine the smallest integer $k_i \geq 0$ such that :
$\psi(x_i + \beta^{k_i} p_i) - \psi(x_i) \leq \gamma\beta^{k_i} \theta(x_i,p_i)$.

Step 4: Set $x_{i+1} = x_i + \beta^{k_i} p_i$.
Set i= i+1 and go to step 1.

□

Theorem A1 [22]

Suppose that :

(i) The functions $g^j : R^n \to R$, $j=1,\ldots,m$ are continuously differentiable.

(ii) The set $\{\nabla g^j(x) \mid j \in I(x)\}$ where

$$I(x) \triangleq \{ j \in \{1,\ldots,m\} \mid g^j(x) = \psi(x)\}$$

is positive linearly independent for all x such that $\psi(x) \geq 0$.

Then, any accumulation point generated by algorithm A1 lies in the interior of the set W.

□

Notes

(i) A direct consequence of theorem A1 is that if a bounded sequence $\{x_i\}$ is generated by the algorithm, then there exists a finite integer j such that $x_j \in W$.

(ii) Condition (ii) in the statement of the above theorem ensures that at all non-feasible points the maximum of the constraints (ψ) can be reduced by the algorithm and hence the algorithm cannot jam up at a non-feasible

point. If this is not satisfied, the algorithm may jam up at such a point. A criterion of jamming can be obtained by testing the value of $\theta(x_i, p_i)$ at iteration i since this is a first order estimate of how much ψ can be reduced at this iteration.

(iii) The following values for the parameters can be employed :

$$\beta = 0.1, \quad \gamma = 0.1, \quad \varepsilon = 0.01 \quad , \quad L = 10.$$

APPENDIX II

1. The low-pass filter.

The circuit diagram and specifications for the low pass filter utilized in the examples of this thesis are shown in Figure I. The frequencies of interest are {0.45, 0.50, 0.55, 0.60, 1.0, 2.5} rad/sec, (6 constraints). The magnitude of the transfer function can be calculated as :

$$\left| \frac{V_1}{V_2}(j\omega) \right|^2 = (1 - (L_1+L_2)\ C\ \omega^2/2)^2 + \omega^2(L_1 + L_2 + C - L_1\ L_2\ C\ \omega^2)^2/4.$$

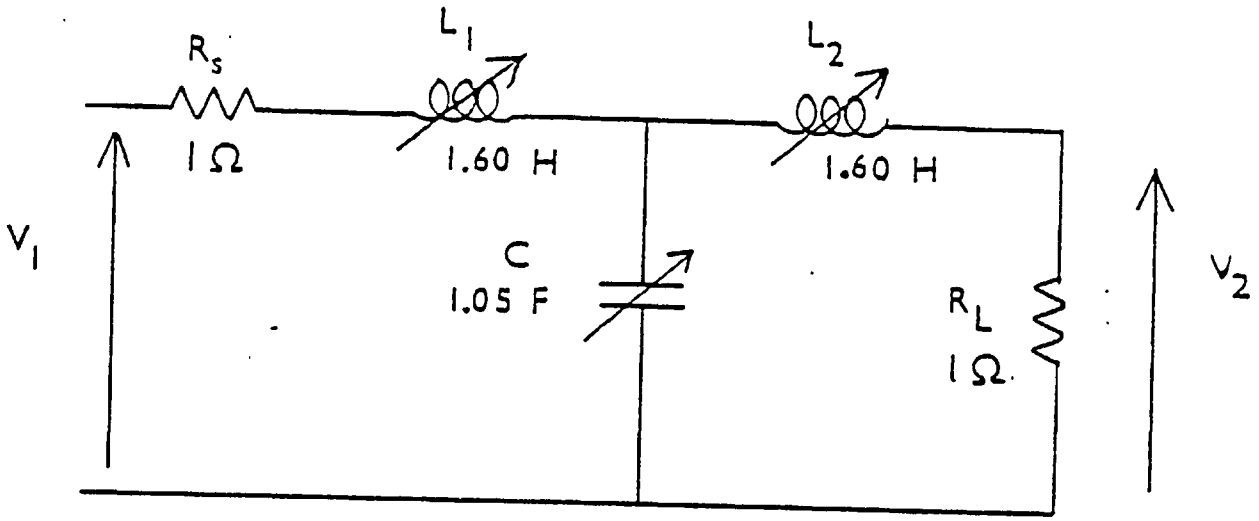
The insertion loss is defined as :

$$I_{loss}(j\omega) \triangleq 20\ \log_{10} \left| \frac{V_1}{V_2}(j\omega) \right| \text{ db}$$

and the relative insertion loss as

$$\bar{I}_{loss}(j\omega) \triangleq I_{loss}(j\omega) - I_{loss}(0j) \text{ db} .$$

($\omega = 0$ is the reference frequency). Toleranced components are marked with an arrow.



Low – pass filter.

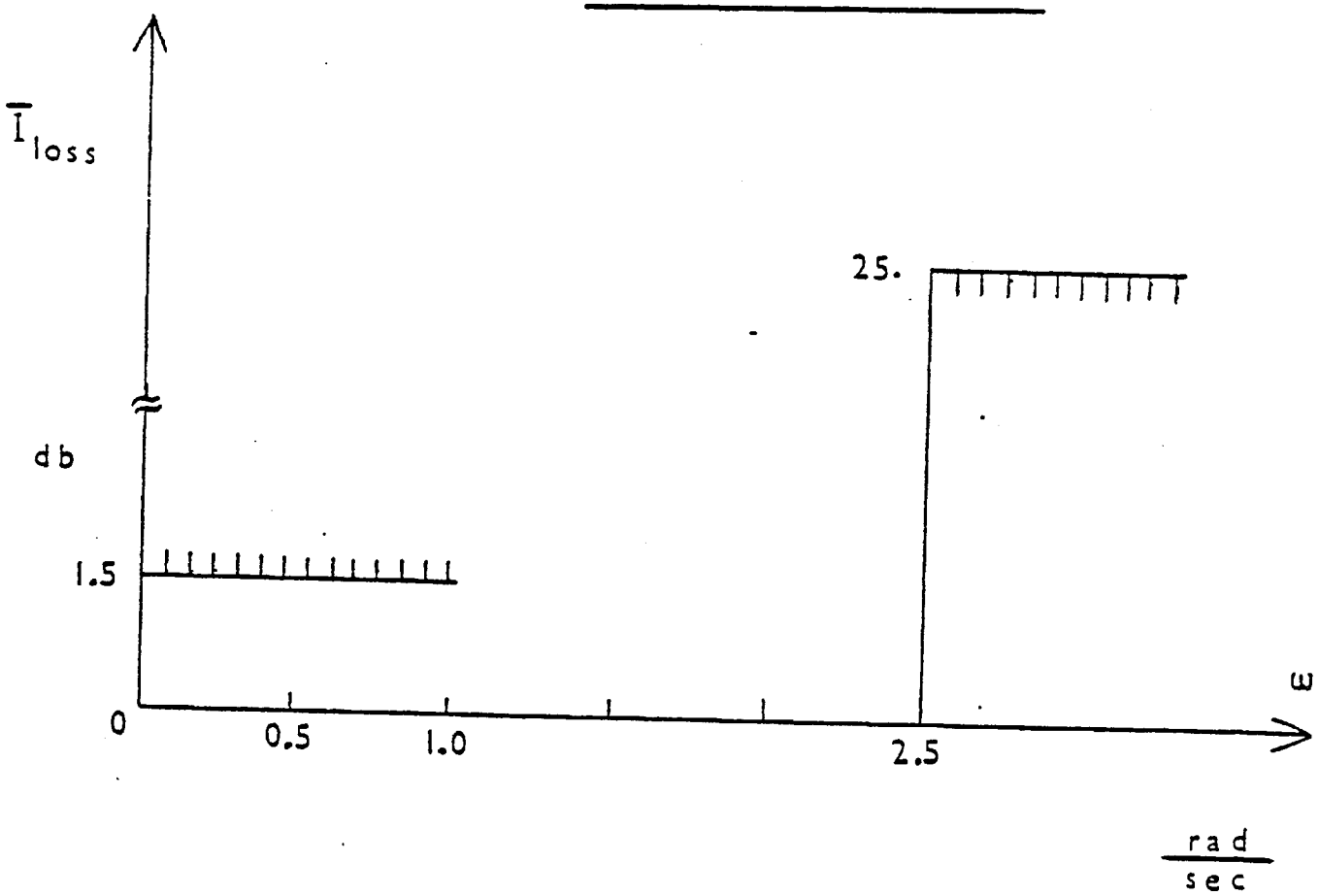


FIGURE 1

2. The high-pass filter.

The circuit diagram and specifications for the high-pass filter utilized in the examples of this thesis are shown in Figure II . The frequencies of interest are : { 170, 350, 440, 630, 650, 720, 740, 750, 940, 1040, 1800 } Hz, (12 constraints). Table I summarizes the specification constraints :

	RELATIVE INSERTION LOSS	
f	Upper Bound (db)	Lower bound (db)
170	-	+ 45.00
350	-	+ 49.00
440	-	+ 42.00
630	+4.00	- 0.75
650	-	- 0.75
720	+1.75	-
740	+1.75	-
750	+1.75	-
940	-	- 0.75
1040	-	- 0.75
1800	+1.75	-

TABLE I

The relative insertion loss is defined as for the low pass filter except that now the reference frequency is 990 Hz. Toleranced components are marked with an arrow. The transfer function can be calculated by using the following relations :

$$z_1 \triangleq R_L - j/\omega\, x_7$$

$$z_2 \triangleq j\omega\, x_6 - j/\omega\, x_5$$

$$z_3 \triangleq - j/\omega\, x_4$$

$$z_4 \triangleq j\omega x_3 - j/\omega x_2$$

$$z_5 \triangleq R_s - j/\omega x_1$$

$$a = z_1\, z_2 \,/\, z_1 + z_2$$

$$b = a + z_3$$

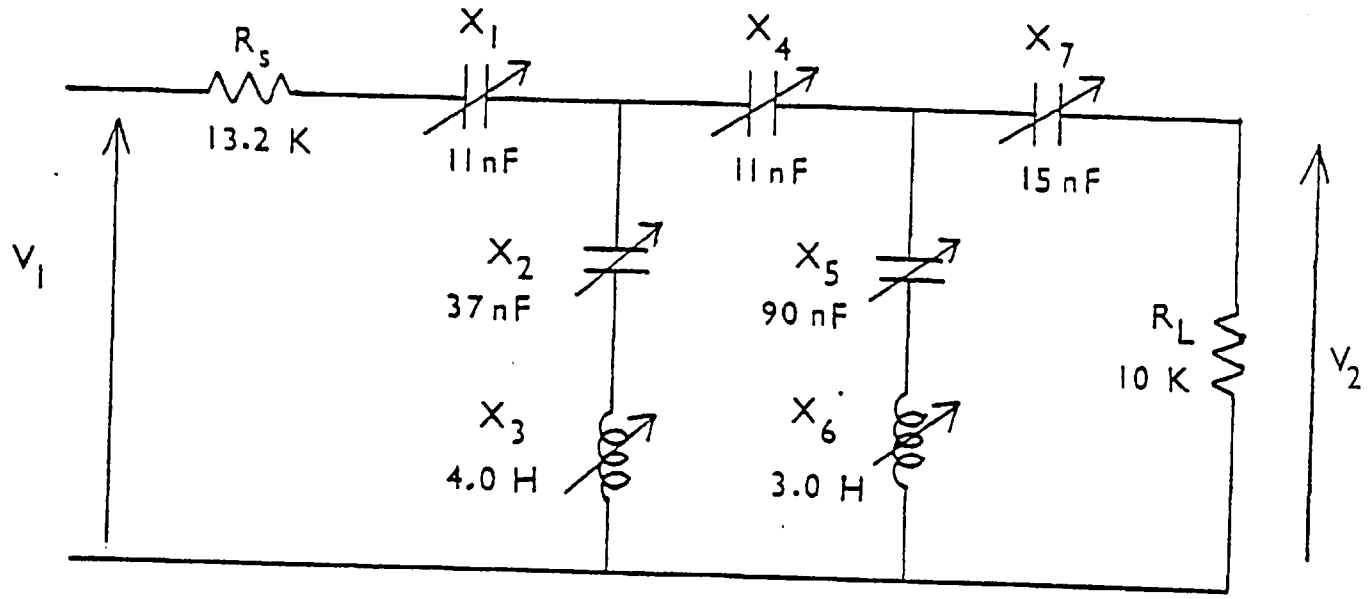
$$c = b\, z_4 / \, b + z_4$$

$$d = z_2\, R_L / z_4\, (z_1 + z_2)$$

$$e = \frac{z_4 - c}{z_5 + c} \quad .$$

Then :

$$\frac{V_2}{V_1}\,(j\omega) = d.e$$


R_s
13.2 K
X_1
11 nF
X_4
11 nF
X_7
15 nF
V_1
X_2
37 nF
X_5
90 nF
R_L
10 K
V_2
X_3
4.0 H
X_6
3.0 H

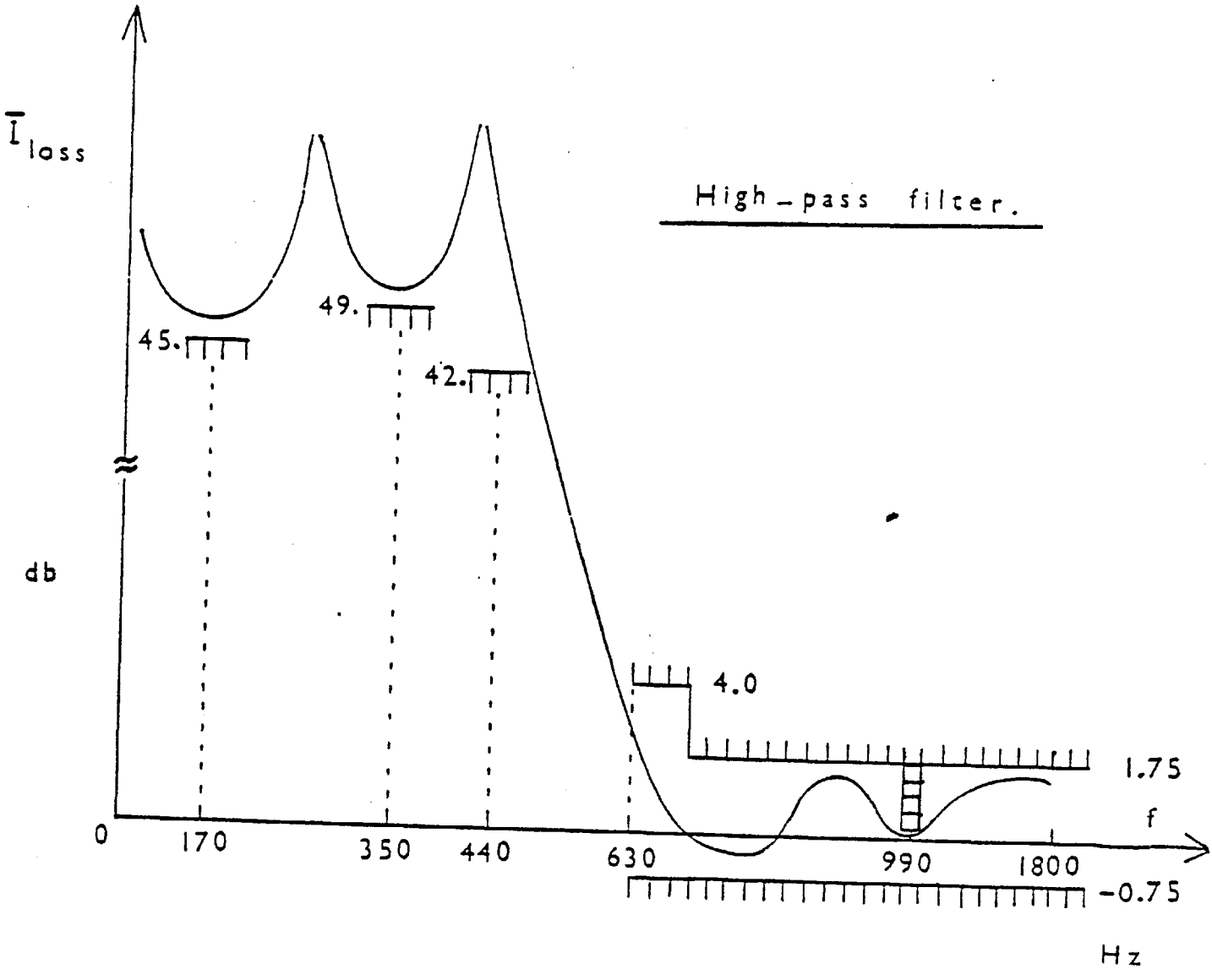

$\bar{I}_{loss}$
High – pass filter.
45.
49.
42.
4.0
1.75
db
f
0
170
350
440
630
990
1800
-0.75
Hz

FIGURE 11