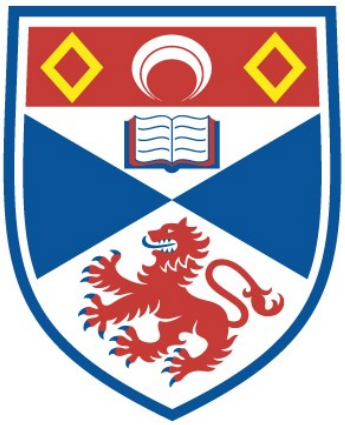
GLOBAL OPTIMIZATION USING INTERVAL ARITHMETIC

Ismail Bin Mohd

A Thesis Submitted for the Degree of PhD
at the
University of St Andrews



1987

Global Optimization Using Interval Arithmetic

Ismail Bin Mohd

Thesis submitted for the degree of Doctor of Philosophy

of the University of St Andrews

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 – 1346

Abstract

This thesis contains a description of algorithm, MW, for bounding the global minimizers and globally minimum value of a twice continuously differentiable function $f : R^n \to R^1$ in a compact sub-interval of R^n . The algorithm MW is similar to the algorithm H of Hansen [Han-80a] in that interval arithmetic is used together with certain of Hansen's ideas, but is different from Hansen's algorithm in that MW bounds the Kuhn Tucker points corresponding to the global minimizers of f in the given sub-interval. The Kuhn Tucker points are bounded with prescribed precision by using either of the algorithms $KMSW$ [SheW-85c] or MAP [SheW-85b].

Numerical results which are obtained from Triplex [BaCM-82a] [MorC-83a] implementations of H and MW are presented.

Acknowledgements

In the Name of Allah, the Most Beneficent, the Most Merciful. Praise be to Allah, the Lord of the Worlds, and Peace be upon the Master of the Apostles, his Family and Companions.

I am pleased to thank my supervisor Mr. M. A. Wolfe for all his help and encouragement during my time as a research student. I am grateful to the University of Agriculture Malaysia and the Department of Public Service Malaysia for their financial support.

I also wish to thank my mother, my father, my wife and my children for their patience.

I Ismail Bin Mohd hereby certify that this thesis which is approximately 30,000 words long has been written by me, that it is a record of work carried out by me, and that it has not been submitted in any previous application for a higher degree.

6th June 1986

I hereby certify that the candidate has fulfilled the conditions of Resolution and Regulations appropriate to the degree of Ph.D. of the University of St Andrews and that he is qualified to submit this thesis in application for that degree.

6th June 1986

I was admitted as a research student under Ordinance No. 12 on 17th December 1982 and as a candidate for the degree of Ph.D. on 17th December 1982; the higher study for which this is a record was carried out in the University of St Andrews between 1982 and 1986.

6th June 1986

Contents

CHAPTER 1

Introduction

There are three types of error in numerical computation, namely data error, rounding error, and truncation error [Han–69a]. The first error is caused by lack of precision in the data, the second error is caused by computing with numbers rounded or truncated to a finite number of digits, and the third error is caused by truncating infinite sequences of algebraic operations after a finite number of steps. More detail can be found in the book of Vandergraft [Van–78a].

Data error, rounding error, and truncation error can be bounded by using rounded interval arithmetic instead of ordinary machine arithmetic ([Moo–66a][Moo–79a]).

In this thesis we consider the use of interval arithmetic for solving the global optimization problem a description of which is given in §1.1. In order to solve the global optimization problem there are available methods which do not use interval arithmetic : an outline of these methods is given in §1.2. An outline of existing methods in which interval arithmetic is used is given in §1.3. In §1.4 an outline of a new interval arithmetic method for global optimization is given. In §1.5 a plan of the remainder of the thesis is given.

1.1 Description of the Global Optimization Problem

The following definitions make it possible to distinguish between unconstrained and global optimization.

Definition 1.1 : Let $f : D \subseteq R^n \rightarrow R^1$ be a given function and let $S \subseteq R^n$ be a given set. The point $x^* \in S$ is a *global minimizer* of f in S if and only if

$$f(x^*) \leq f(x) \ (\forall x \in S \cap D). \ \square$$

Definition 1.2 : Let $f : D \subseteq R^n \rightarrow R^1$ be a given function and let $S \subseteq R^n$ be a given set. Then $x^* \in S$ is an *unconstrained local minimizer* of f in S if and only if $\exists \varepsilon > 0$ such that

$$f(x^*) \leq f(x) \ (\forall x \in B(x^*, \varepsilon) \subset S \cap D)$$

where the open ball $B(x^*, \varepsilon)$ with centre x^* and radius ε is defined by

$$B(x^*, \varepsilon) = \{x \in R^n \mid \|x - x^*\| < \varepsilon\}. \ \square$$

The global optimization problem consists of determining a global minimizer x^* of $f : D \subseteq R^n \rightarrow R^1$ in a given set $S \subseteq R^n$. If $x^* \in \partial S$ where ∂S is the boundary

of S, then the determination of x^* is a constrained global optimization problem. If $x^* \in int(S)$ where $int(S)$ is the interior of S, then the determination of x^* is an unconstrained global optimization problem, and x^* is an unconstrained local minimizer of f in S, as well as being a global minimizer of f in S.

If f is strictly convex in S and $x^* \in S$ is an unconstrained local minimizer of f then x^* is unique in S and is the global minimizer of f in S. In general, however, f might have several global minimizers $x^{*1}, \ldots, x^{*m} \in S$, so that $f(x^{*1}) = \ldots = f(x^{*m})$ and for $i = 1, \ldots, m$

$$f(x^{*i}) \leq f(x) \ (\forall x \in S \cap D).$$

The global optimization problem which is considered in this thesis may be expressed as follows.

$$\left.\begin{array}{ll} \textit{minimize} & f(x) \\ \textit{subject to} & x \in S \end{array}\right\} \qquad 1.1$$

where $f : D \subseteq R^n \rightarrow R^1$ is a given function and $S \subseteq D$ is a box (See A.27.).

1.2 Algorithms in which Interval Arithmetic is not used

Algorithms for the solution of global optimization problems in which interval arithmetic is not used may be divided into two classes, namely the class of deterministic methods, and the class of probabilistic methods [Gom–77a].

The class of deterministic methods may be subdivided into the classes of space-covering methods, region-of-attraction methods, trajectory methods, and function-modification methods.

Of the space-covering methods, which are based upon the Lipschitz Condition

$$| f(x) - f(y) | \leq L \|x - y\| \ (x, y \in S), \qquad 1.2$$

Shubert's method [Shu–72a] is very efficient for one-dimensional problems, and Evtushenko's method [Evt–71a] is useful for up to 2 variables only. Other space-covering methods are described by Archetti and Betro [ArcB-78a].

Region-of-attraction methods are based on how to identify the set of initial points from which it is possible to estimate particular minimizers by means of some local minimization algorithm. The methods which use this idea are described by Treccani, Trabattoni, Szegö [TrTS-72a], and Corles [Cor–75a].

Trajectory methods are based on integration of a differential equation, whose trajectories lead either to critical points of the objective function or to points from

which minimizers of this function can be obtained by a local minimization algorithm. The methods which use this idea are described by Branin [Bra–71a] [Bra–72a] [BraH-72a], and Hardy [Har–75a].

Function-modification methods, which are based upon the idea of descent from a local minimizer, are described by Goldstein and Price in [GolP-71a]. They define a method for determining the global minimizer of a polynomial and then generalize the method to an analytic function of n dimensions.

The probabilistic methods use the fact that if sufficient points are distributed uniformly but at random over any set of finite measure then the likelihood of any particular subset, with positive measure, containing at least one point tends to unity as the number of points tends to infinity. The methods which use this fact are the pure random method [Bro–58a] [And–72a], the Monte Carlo method [RubW-77a] and the Chichinadze method [Chi–67a].

In order to reduce the cost, probabilistic methods are combined with deterministic methods such as the multistart method [DixS-78a], Hartman's method [Hart-72a], the method of Becker and Lago [BecL-70a], Törn's method [Tor–78a], Price's method [Pri–78a], Gomulka's method [Gom–78a], and the method of Fagiuoli, Pianca, and Zecchin [FaPZ-78a].

Another approach which belongs to the class of probabilistic methods is the method of Biase and Frontini [BiaF-78a] and the Bayesian method the application of which is described in [MoTZ-78a].

1.3 Algorithms in which Interval Arithmetic is used

In this section, outlines of the algorithms in which interval arithmetic is used for bounding solutions of the global optimization problem given by 1.1 are described.

1.3.1 The Method of Robinson

Robinson [Rob–73a] has shown how to use interval arithmetic for computing error bounds for an approximate Kuhn-Tucker point $(\tilde{x}^T, \tilde{u}^T, \tilde{w}^T)^T$ of the nonlinear programming problem

$$\left.\begin{array}{lll} \textit{minimize} & f(x) & \\ \textit{subject to} & & \\ & g_i(x) \leq 0 & (i = 1, \ldots, m) \\ \textit{and} & & \\ & h_j(x) = 0 & (j = 1, \ldots, r) \end{array}\right\} \qquad 1.3$$

where $\tilde{x} \in D \subseteq R^n$, $\tilde{u} \in R^m$, $\tilde{w} \in R^r$ and $f : D \subseteq R^n \rightarrow R^1$, $g_i : R^n \rightarrow R^1$ $(i = 1, \ldots, m)$, and $h_j : R^n \rightarrow R^1$ $(j = 1, \ldots, r)$ are given continuously differentiable but not necessarily convex functions.

Suppose that 1.3 is solved approximately to obtain a numerical solution $(\tilde{x}^T, \tilde{u}^T, \tilde{w}^T)^T$ at which the Kuhn-Tucker conditions [FiaM-68a]

$$\left.\begin{aligned} f'(x) + u^T g'(x) + w^T h'(x) &= 0 \\ g(x) &\leq 0 \\ u^T g(x) &= 0 \\ h(x) &= 0 \\ u &\geq 0 \end{aligned}\right\} \qquad 1.4$$

are approximately satisfied and that $\hat{\underline{z}} = (\tilde{x}^T, \tilde{u}^T, \tilde{w}^T)^T + \hat{\underline{d}}$ where $\hat{\underline{d}}$ is an (n+m+r)-interval vector with $m(\hat{\underline{d}}_i) = 0$ $(i = 1, \ldots, n+m+r)$ and $w(\hat{\underline{d}}_1) = \ldots = w(\hat{\underline{d}}_{n+m+r})$. Computable sufficient conditions which are based on the interval Newton operator $\underline{N}$ [Nic–71a] for the existence of a unique Kuhn-Tucker point $(x^{*T}, u^{*T}, w^{*T})^T$ in $\hat{\underline{z}}$ which satisfies 1.4 have been given by Robinson [Rob–73a].

Moore [Moo–79a] has suggested that the interval Newton operator $\underline{N}$ [Nic–71a] which has been used by Robinson [Rob–73a] could be replaced by the interval Krawczyk operator $\underline{K}$ [Kra–69a].

1.3.2 The Method of Skelboe

Skelboe [Ske–74a] has described an algorithm for computing a lower bound on the value of the objective function f given in 1.1. The algorithm is applied to $-f$ for

computing an upper bound on the value of f.

The method of Skelboe [Ske-74a] is based on the following definitions and theorems.

Definition 1.3 : Let $f : D \subseteq R^n \rightarrow R^1$ be a given mapping and let f be continuous in $\hat{D}$ where $\hat{D} \subseteq D$ is a convex set. The mapping $\underline{f} : I(\hat{D}) \rightarrow I(R)$ defined by

$$\underline{f}(\underline{x}) = \{f(x) \mid x \in \underline{x}\} \ (\underline{x} \in I(\hat{D})) \qquad 1.5$$

is called the *united extension* of f. □

Definition 1.4 : Let $f : D \subseteq R^n \rightarrow R^1$ be a rational function. A rational interval function $\underline{f} : I(D) \rightarrow I(R)$ which is obtained from the rational real expression which represents f by replacing, for $i = 1, \ldots, n$, $x_i \in R$ with $\underline{x}_i \in I(R)$ and real arithmetic operations with the corresponding interval arithmetic operations is called the *natural interval extension* of f. □

Theorem 1.1 : If (1) $f : D \subseteq R^n \rightarrow R^1$ is a rational function; (2) $\underline{f} : I(D) \rightarrow I(R)$ is a natural interval extension of f then $(\forall \underline{x} = (\underline{x}_1, \ldots, \underline{x}_n) \in I(D))$

$$(\underline{x}'_1 \subseteq \underline{x}_1, \ldots, \underline{x}'_n \subseteq \underline{x}_n) \Rightarrow (\underline{f}(\underline{x}'_1, \ldots, \underline{x}'_n) \subseteq \underline{f}(\underline{x}_1, \ldots, \underline{x}_n)).$$

Proof : A proof of Theorem 1.1 is given in [Moo–66a]. □

Theorem 1.2 : If (1) $f : D \subseteq R^n \to R^1$ is a rational function; (2) $\underline{f} : I(D) \to I(R)$ is a natural interval extension of f; (3) $\underline{f}(\underline{x})$ $(\underline{x} \in I(D))$ is in centred form (See A.26.) then $\exists \alpha > 0$ such that

$$\underline{f}(\underline{x}) = \bar{f}(\underline{x}) + \underline{e} \tag{1.6}$$

where $w(\underline{e}) \leq \alpha \left\| w(\underline{x}) \right\|^2$, $0 \in \underline{e}$ and $\bar{f}(\underline{x})$ is given by 1.5.

Proof : A proof of Theorem 1.2 is given in [Han–69a]. □

Theorem 1.3 : If (1) $f : D \subseteq R^n \to R^1$ is a rational function; (2) $\underline{f} : I(D) \to I(R)$ is a natural interval extension of f; (3) $\underline{x}_j$, $j \in \{1, \ldots, n\}$ occurs only once in the expression $\underline{f}(\underline{x})$ and to the first power only, and

$$\underline{x}_j = \bigcup_{i=1}^{N} \underline{x}_j^{(i)} \tag{1.7}$$

then

$$\underline{f}(\underline{x}_1, \ldots, \underline{x}_{j-1}, \underline{x}_j, \underline{x}_{j+1}, \ldots, \underline{x}_n) = \bigcup_{i=1}^{N} \underline{f}(\underline{x}_1, \ldots, \underline{x}_{j-1}, \underline{x}_j^{(i)}, \underline{x}_{j+1}, \ldots, \underline{x}_n). \tag{1.8}$$

Proof : A proof of Theorem 1.3 is given in [Ske–74a]. □

Theorem 1.4 : If (1) $f : D \subseteq R^n \to R^1$ is a rational function; (2) $\underline{f} : I(D) \to I(R)$ is a natural interval extension of f, written in centred form; (3) each of the variables $\underline{x}_{p+1}, \ldots, \underline{x}_n$ occurs exactly once in the function $\underline{f}$ and to the first power only; (4) each of the variables $\underline{x}_1, \ldots, \underline{x}_p$ is subdivided into N intervals of equal width so that

$$\underline{x}_i = \bigcup_{j=1}^{N} \underline{x}_i^{(j)} \text{ with } w(\underline{x}_i^{(j)}) = w(\underline{x}_i)/N \ (i = 1, \ldots, p), \tag{1.9}$$

then $\exists \alpha > 0$ such that

$$\bigcup_{i_1=1}^{N} \cdots \bigcup_{i_p=1}^{N} \underline{f}(\underline{x}_1^{(i_1)}, \ldots, \underline{x}_p^{(i_p)}, \underline{x}_{p+1}, \ldots, \underline{x}_n) = \overline{f}(\underline{x}_1, \ldots, \underline{x}_n) + \underline{e}_N \tag{1.10}$$

where $0 \in \underline{e}_N$ and

$$w(\underline{e}_N) \leq (\alpha/N^2) \, \|w(\underline{x})\|^2 . \tag{1.11}$$

Proof : A proof of **Theorem 1.4** is given in [Ske–74a]. □

It follows from **Theorem 1.4** that if $\underline{x} \in I(R^n)$ is a given box then $\overline{f}(\underline{x})$ can be computed with a precision which is limited only by the precision of the machine arithmetic and the available memory by taking N sufficiently large. Let

$$\underline{y}^{(N)} = \bigcup_{i_1=1}^{N} \cdots \bigcup_{i_p=1}^{N} \underline{f}(\underline{x}_1^{(i_1)}, \ldots, \underline{x}_p^{(i_p)}, \underline{x}_{p+1}, \ldots, \underline{x}_n).$$

Then $\overline{\underline{f}}(\underline{x}) \subseteq \underline{y}^{(N)}$, and $\underline{y}^{(N)} \to \overline{\underline{f}}(\underline{x})$ $(N \to \infty)$. Let $m \geq 1$ and $\varepsilon > 0$ be given. For $N^{(k)} = 2^k m$ $(k = 0, 1, 2, \ldots)$ compute $\underline{y}^{(N^{(k)})}$, until for some $k \geq 1$, $| w(\underline{y}^{(N^{(k)})}) - w(\underline{y}^{(N^{(k-1)})}) | \leq \varepsilon w(\underline{y}^{(N^{(k)})})$. Then $\underline{y}^{(N^{(k)})}$ bounds $\overline{\underline{f}}(\underline{x})$ to the required precision. This procedure for bounding $\overline{\underline{f}}(\underline{x})$ requires a prohibitively large number of evaluations of $\underline{f}$.

If p of the variables of f are to be bisected as in 1.9 then N^p subdivisions are obtained. Therefore, N^p function evaluations are needed to compute the expression on the left-hand side of 1.10.

Suppose that it is required to determine a lower bound on $f : D \subseteq R^1 \to R^1$ over $\underline{x} \in I(D)$. The interval $\underline{x}$ is divided into N sub-intervals $\underline{x}^{(N,1)}, \ldots, \underline{x}^{(N,N)}$ with $x_S^{(N,i)} = x_I^{(N,i+1)}$ $(i = 1, \ldots, N-1)$ and $w(\underline{x}^{(N,i)}) = (1/N) w(\underline{x})$ $(i = 1, \ldots, N)$.

Suppose that $\overline{\underline{f}} = \overline{\underline{f}}(\underline{x})$, that $\underline{f}^{(N,i)} = \underline{f}(\underline{x}^{(N,i)})$ $(i = 1, \ldots, N)$, and that $\underline{f}^{(2,1)}$ and $\underline{f}^{(2,2)}$ have been computed. Now by **Definition 1.3** and **Theorem 1.1**,

$$\overline{\underline{f}} \subseteq \underline{y}^{(2)} = \underline{f}^{(2,1)} \cup \underline{f}^{(2,2)} \subseteq \underline{f}(\underline{x})$$

so

$$\overline{f}_I \geq min\{f_I^{(2,1)}, f_I^{(2,2)}\}.$$

Suppose that $f_I^{(2,1)} < f_I^{(2,2)}$. let $L = (f_I^{(2,1)}, f_I^{(2,2)})$ represent an ordered linked list of infima. Then $f_I^{(2,1)}$ is currently the best lower bound on $\bar{f}_I$.

Since $f_I^{(2,1)} < f_I^{(2,2)}$, we compute $\underline{f}^{(4,1)}$ and $\underline{f}^{(4,2)}$. We do not need to compute $\underline{f}^{(4,3)}$ and $\underline{f}^{(4,4)}$ because since $\underline{x}^{(2,2)} = \underline{x}^{(4,3)} \cup \underline{x}^{(4,4)}$ it follows that

$$\underline{f}^{(4,3)} \cup \underline{f}^{(4,4)} \subseteq \underline{f}^{(2,2)}.$$

Therefore

$$\underline{f} \subseteq \underline{y}^{(4)} = \bigcup_{i=1}^{4} \underline{f}^{(4,i)}$$

$$\subseteq \underline{f}^{(4,1)} \cup \underline{f}^{(4,2)} \cup \underline{f}^{(2,2)}$$

whence

$$\bar{f}_I \geq min\{f_I^{(4,1)}, f_I^{(4,2)}, f_I^{(2,2)}\}.$$

Suppose that $f_I^{(4,2)} < f_I^{(4,1)}$. Now $\underline{f}^{(4,1)} \subset \underline{f}^{(2,1)}$ and $\underline{f}^{(4,2)} \subset \underline{f}^{(2,1)}$ so $f_I^{(2,1)} \leq f_I^{(4,2)} < f_I^{(4,1)}$. Since $f_I^{(4,2)} \leq \bar{f}_I$, $f_I^{(2,1)}$ is deleted from L and $f_I^{(4,1)}$ and $f_I^{(4,2)}$ are inserted into L so as to retain the ordering. Thus if $f_I^{(4,1)} \leq f_I^{(2,2)}$ then

$$L = (f_I^{(4,2)}, f_I^{(4,1)}, f_I^{(2,2)}), \qquad 1.12$$

if $f_I^{(4,2)} \leq f_I^{(2,2)} \leq f_I^{(4,1)}$ then

$$L = (f_I^{(4,2)}, f_I^{(2,2)}, f_I^{(4,1)}), \tag{1.13}$$

and if $f_I^{(2,2)} \leq f_I^{(4,2)}$ then

$$L = (f_I^{(2,2)}, f_I^{(4,2)}, f_I^{(4,1)}). \tag{1.14}$$

If L is given by 1.12 or 1.13 then $\underline{f}^{(8,3)}$ and $\underline{f}^{(8,4)}$ are computed. If L is given by 1.14 then $\underline{f}^{(4,3)}$ and $\underline{f}^{(4,4)}$ are computed. This procedure is continued until the difference between the first elements in successive lists L is less in magnitude than a pre-assigned tolerance. Skelboe's algorithm is easily generalized to functions of several variables and the Algol W program which he gives in [Ske–74a] is able to bound the united extensions of functions of up to 30 variables.

1.3.3 The Method of Dussel

Dussel [Dus–72a] has described an algorithm for bounding x^*, the global minimizer and f^*, the global minimum value of the continuously differentiable strictly convex function $f : D \subset R^n \rightarrow R^1$.

It is assumed that $\underline{f} : I(D) \to I(R^1)$, and $\partial_i \underline{f} : I(D) \to I(R^1)$ $(i = 1, \ldots, n)$ are inclusion monotonic interval extensions of $f : D \subset R^n \to R^1$ and of $\partial_i f : D \subset R^n \to R^1$ $(i = 1, \ldots, n)$ respectively. Let

$$sign(\underline{h}) = \begin{cases} 1 & (0 < h_I) \\ 0 & (h_I \leq 0 \leq h_S) \\ -1 & (h_S < 0) \end{cases} \qquad 1.15$$

where $\underline{h} = [h_I, h_S]$. For degenerate intervals, this reduces to the ordinary sign function for real numbers (a degenerate interval is an interval containing one point only).

Let

$$\underline{b}(i, a) = \{x \mid x \in \underline{\hat{x}} \in I(D),\ x_i = a\} \qquad 1.16$$

for $i = 1, \ldots, n$ and $\hat{x}_{iI} < a < \hat{x}_{iS}$. Clearly $\underline{b}(i, a) = (\underline{\hat{x}}_1, \ldots, \underline{\hat{x}}_{i-1}, a, \underline{\hat{x}}_{i+1}, \ldots, \underline{\hat{x}}_n)$. The function f has a unique minimizer in each $\underline{b}(i, a)$ $(i = 1, \ldots, n)$. Let

$$sign(z) = \begin{cases} 1 & (z > 0) \\ 0 & (z = 0). \\ -1 & (z < 0) \end{cases} \qquad 1.17$$

The following result is proved in [Dus–72a].

Theorem 1.5 : If $f(y) = \min\{f(x) \mid x \in \underline{b}(i,a)\ i \in \{1,\ldots,n\}\}$ and x^* is the global minimizer of f then

$$sign(\partial_i f(y)) = sign(a - x_i^*). \quad \square$$

The method of Dussel consists of a sequence of cyclic bisections of $\hat{\underline{x}}$ (cyclic choice of co-ordinate directions), choosing at each bisection the half which still contains x^*. This half can be chosen by using 1.17. In order to compute $sign(\partial_i f(y))$, we note that $(sign(\partial_i \underline{f}(\hat{\underline{x}})) = \pm 1) \Rightarrow (sign(\partial_i f(y)) = \pm 1\ (\forall y \in \hat{\underline{x}}))$. That is, if $sign(\partial_i \underline{f}(\hat{\underline{x}})) \neq 0$ then $sign(\partial_i \underline{f}(\hat{\underline{x}})) = sign(\partial_i f(y))\ (\forall y \in \hat{\underline{x}})$. The problem of finding y to minimize $f(x)$ for $x \in \underline{b}(i,a)$ is again a problem of the type 1.1, but of dimension $n-1$. We do not have to find y accurately, but only need to know a box $\hat{\underline{x}}$ containing y such that $sign(\partial_i \underline{f}(\hat{\underline{x}})) \neq 0$. This box can be obtained by applying cyclic bisection to the box $\underline{b}(i,a)$.

Dussel [Dus–72a] has given a computer program written in Triplex Algol 60 which implements his algorithm.

1.3.4 The Method of Moore

Moore [Moo–76a] has discussed several techniques which have been used in algorithms for bounding the range of values of a function $f : R^n \to R^1$ in a given box $\hat{\underline{x}} \in I(R^n)$, and has considered how the techniques which he has discussed might be incorporated into a method for bounding the range of values of f in $\hat{\underline{x}}$.

Moore mentions the following ideas.

(i) $\underline{\bar{f}}(\hat{\underline{x}}) \subseteq \underline{f}(\hat{\underline{x}})$ ([Moo–62a][Moo–66a]).

(ii) If $\hat{\underline{x}}_{ij}^{(N)}$ $(i = 1, \ldots, n)$ are given by

$$\hat{\underline{x}}_{ij}^{(N)} = \hat{x}_{iI} + [j-1, j]w(\hat{\underline{x}}_i)/N \ (j = 1, \ldots, N) \tag{1.18}$$

and

$$[L^{(N)}, U^{(N)}] = \bigcup_{j_1=1}^{N} \cdots \bigcup_{j_n=1}^{N} \underline{f}(\hat{\underline{x}}_{1j_1}^{(N)}, \ldots, \hat{\underline{x}}_{nj_n}^{(N)}), \tag{1.19}$$

then $\underline{\bar{f}}(\hat{\underline{x}}) \subseteq [L^{(N)}, U^{(N)}]$. Furthermore, $\exists \alpha > 0$, independent of N, such that

$$\max(U^{(N)} - (\underline{\bar{f}}(\hat{\underline{x}}))_S, (\underline{\bar{f}}(\hat{\underline{x}}))_I - L^{(N)}) \leq \alpha/N \tag{1.20}$$

([Moo–62a][Moo–66a]).

(iii) If each component of $\hat{\underline{x}}$ occurs at most once and to the first power only in $\underline{f}(\hat{\underline{x}})$ then ([Moo–62a][Moo–66a])

$$\underline{\bar{f}}(\hat{\underline{x}}) = \underline{f}(\hat{\underline{x}}). \tag{1.21}$$

(iv) If $f_c(x)$ is the centred form (See A.26.) of $f(x)$ then [Moo-66a] $\underline{f}(\hat{\underline{x}}) \subseteq [L_c^{(N)}, U_c^{(N)}]$, where

$$[L_c^{(N)}, U_c^{(N)}] = \bigcup_{j_1=1}^{N} \cdots \bigcup_{j_n=1}^{N} \underline{f}_c(\hat{\underline{x}}_{1j_1}^{(N)}, \ldots, \hat{\underline{x}}_{nj_n}^{(N)}) \qquad 1.22$$

and $\exists \beta > 0$, independent of N [Han–69a] such that

$$\max(U_c^{(N)} - (\underline{f}(\hat{\underline{x}}))_S, (\overline{f}(\hat{\underline{x}}))_I - L_c^{(N)}) \leq \beta/N^2. \qquad 1.23$$

(v) If the components $\hat{\underline{x}}_{p+1}, \ldots, \hat{\underline{x}}_n$, $(1 \leq p \leq n)$ occur at most once and to the first power only in $\underline{f}(\hat{\underline{x}})$ then $\overline{f}(\hat{\underline{x}}) \subseteq [L_c^{(N)}, U_c^{(N)}]$, where

$$[L_c^{(N)}, U_c^{(N)}] = \bigcup_{j_1=1}^{N} \cdots \bigcup_{j_p=1}^{N} \underline{f}_c(\hat{\underline{x}}_{1j_1}^{(N)}, \ldots, \hat{\underline{x}}_{pj_p}^{(N)}, \hat{\underline{x}}_{p+1}, \ldots, \hat{\underline{x}}_n) \qquad 1.24$$

and $\exists \gamma > 0$, independent of N [Ske–74a], such that

$$\max(U_c^{(N)} - (\underline{f}(\hat{\underline{x}}))_S, (\overline{f}(\hat{\underline{x}}))_I - L_c^{(N)}) \leq \gamma/N^2. \qquad 1.25$$

(vi) Suppose that $\underline{g} : I(D) \to I(R^n)$ is the natural interval extension of the gradient $g : D \subseteq R^n \to R^n$ of $f : D \subseteq R^n \to R^1$. Recall the monotonicity property of f, namely that if $g_i(x) \geq 0$ $(\forall x \in \hat{\underline{x}})$ for some $i \in \{1, \ldots, n\}$ then $(\forall x \in \hat{\underline{x}})$

$$f(x_1, \ldots, x_{i-1}, \hat{x}_{iI}, x_{i+1}, \ldots, x_n) \leq f(x), \tag{1.26}$$

and

$$f(x_1, \ldots, x_{i-1}, \hat{x}_{iS}, x_{i+1}, \ldots, x_n) \geq f(x). \tag{1.27}$$

Therefore, if for some i $(1 \leq i \leq n)$, $(\underline{g}_i(\hat{\underline{x}}))_I \geq 0$ then the minimizer and maximizer of f occur on the boundaries $\partial_1^i(\hat{\underline{x}})$ and $\partial_2^i(\hat{\underline{x}})$ respectively, where

$$\partial_1^i(\hat{\underline{x}}) = (\hat{\underline{x}}_1, \ldots, \hat{\underline{x}}_{i-1}, [\hat{x}_{iI}, \hat{x}_{iI}], \hat{\underline{x}}_{i+1}, \ldots, \hat{\underline{x}}_n)^T, \tag{1.28}$$

and

$$\partial_2^i(\hat{\underline{x}}) = (\hat{\underline{x}}_1, \ldots, \hat{\underline{x}}_{i-1}, [\hat{x}_{iS}, \hat{x}_{iS}], \hat{\underline{x}}_{i+1}, \ldots, \hat{\underline{x}}_n)^T. \tag{1.29}$$

Similar statements are valid if $(\underline{g}_i(\hat{\underline{x}}))_S \leq 0$. Furthermore, we can intersect those parts of the boundary of $\hat{\underline{x}}$ in which the minimum and maximum values of f are known to lie. Thus, if the minimum value of f occurs on the boundaries $\partial_{j_1}^{i_1}(\hat{\underline{x}}), \ldots, \partial_{j_k}^{i_k}(\hat{\underline{x}})$ $(1 \leq k \leq n)$ then it occurs on the intersection

$$\partial_{j_1}^{i_1}(\hat{\underline{x}}) \cap \ldots \cap \partial_{j_k}^{i_k}(\hat{\underline{x}}) \tag{1.30}$$

where $j_1 = \ldots = j_k = 1$ or $j_1 = \ldots = j_k = 2$, whichever is appropriate.

These observations can be used to reduce the dimensionality of the problem in particular examples. This technique can also be used on the sub-boxes of $\hat{\underline{x}}$ obtained by using 1.18.

(vii) Suppose that for every $i = 1, \ldots, n$ we have either

$$(\underline{g}_i(\hat{\underline{x}}_{1j_1}^{(N)}, \ldots, \hat{\underline{x}}_{nj_n}^{(N)}))_S \leq 0 \tag{1.31}$$

or

$$(\underline{g}_i(\hat{\underline{x}}_{1j_1}^{(N)}, \ldots, \hat{\underline{x}}_{nj_n}^{(N)}))_I \geq 0 \tag{1.32}$$

where $\hat{\underline{x}}_{ij_i}^{(N)}$ $(i = 1, \ldots, n)$ are given by 1.18. Suppose that S and T are defined by

$$S = \{i \mid (\underline{g}_i(\hat{\underline{x}}_{1j_1}^{(N)}, \ldots, \hat{\underline{x}}_{nj_n}^{(N)}))_S \leq 0,\ 1 \leq i \leq n\} \tag{1.33}$$

and

$$T = \{i \mid (\underline{g}_i(\hat{\underline{x}}_{1j_1}^{(N)}, \ldots, \hat{\underline{x}}_{nj_n}^{(N)}))_I \geq 0,\ 1 \leq i \leq n\} \tag{1.34}$$

and that

$$S \cup T = \{1, \ldots, n\}. \tag{1.35}$$

Then f attains its minimum value in

$$(\underline{\hat{x}}_{1j_1}^{(N)}, \ldots, \underline{\hat{x}}_{nj_n}^{(N)})^T \tag{1.36}$$

at the point $u = (u_i)_{n\times 1}$ with

$$u_i = \begin{cases} \hat{x}_{iI} + (j_i - 1)(\hat{x}_{iS} - \hat{x}_{iI})/N & (i \in T) \\ \hat{x}_{iI} + j_i(\hat{x}_{iS} - \hat{x}_{iI})/N & (i \in S) \end{cases} \tag{1.37}$$

and its maximum values in the same sub-box at the point $v = (v_i)_{n\times 1}$ with

$$v_i = \begin{cases} \hat{x}_{iI} + j_i(\hat{x}_{iS} - \hat{x}_{iI})/N & (i \in T) \\ \hat{x}_{iI} + (j_i - 1)(\hat{x}_{iS} - \hat{x}_{iI})/N & (i \in S) \end{cases}. \tag{1.38}$$

We can obtain bounds on the values of f at u and v using $\underline{f}$ as in (i). Thus

$$f(u) \in \underline{f}(u)$$

and

$$f(v) \in \underline{f}(v).$$

Also, if $\underline{f}(u) \subseteq [a_I, a_S]$ and $\underline{f}(v) \subseteq [b_I, b_S]$ where $[a_I, a_S]$ and $[b_I, b_S]$ are obtained

by using rounded interval arithmetic evaluation of $\underline{f}$ at u and v then on the sub-box given by 1.36 the values of f are contained in the interval $[a_I, b_S]$.

(viii) Suppose that by using the technique of (vi) the sub-box $\underline{\tilde{x}}$ of $\underline{\hat{x}}$ is obtained but $\underline{\tilde{x}}$ is not a point box (i.e. a box which contains only one point). Suppose that the minimum and maximum values of f on $\underline{\tilde{x}}$ lie in faces of dimension $1 \leq p \leq n$ (if $p = n$ then this is $\underline{\tilde{x}}$ itself).

In the sub-box $\underline{\tilde{x}}$, $n-p$ arguments of f are fixed real numbers, and the remaining p arguments $x_{i_1}, \ldots, x_{i_p}$ of f $(1 \leq i_1 < \ldots < i_p \leq n)$ range over intervals. Suppose that $y_1 = x_{i_1}, \ldots, y_p = x_{i_p}$. Let

$$h(y_1, \ldots, y_p) = f|_{\underline{\tilde{x}}} \qquad 1.39$$

where $f|_{\underline{\tilde{x}}}$ denotes that f is restricted to $\underline{\tilde{x}}$. Then $\partial_j h(y)$ $(j = 1, \ldots, p)$ attains both positive and negative values on $\underline{\tilde{x}}$.

Consider the matrix

$$J(y) = J(y_1, \ldots, y_p) = (\partial_j \partial_k h(y))_{p \times p} \qquad 1.40$$

$(j,k = 1,\ldots,p)$. If $\exists \underline{Q}$ ([Moo-62a][Moo-66a][Han-65a][Han-69b][HanS-67a]) such that

$$\{(J^{-1}(y))_{jk} \mid y \in \tilde{\underline{x}}\} \subseteq \underline{Q}_{jk} \qquad 1.41$$

$(j,k = 1,\ldots,p)$ then there is at most one local extremum of f in $\tilde{\underline{x}}$ and, if in $\tilde{\underline{x}}$, it is also in

$$\tilde{\underline{x}}' = \tilde{\underline{x}} \cap \{m(\tilde{\underline{x}}) - \underline{Q}\underline{h}'(m(\tilde{\underline{x}}))\} \qquad 1.42$$

where $\underline{h}'$ is an interval extension h' of h. This gives rise to an interval version of Newton's method [Moo-66a]. If $\tilde{\underline{x}}' = \emptyset$ then f has no extremum in the interior of $\tilde{\underline{x}}'$ and the minimum and the maximum values of f on $\tilde{\underline{x}}$ occur on the boundary of $\tilde{\underline{x}}$.

If $\nexists \underline{Q}$ such that 1.41 holds then it might be possible to find one for a sub-box of $\tilde{\underline{x}}$ upon further subdivision. Once having found a suitable $\underline{Q}$ and $\tilde{\underline{x}}'$, 1.42 can be iterated with or without re-evaluation of $\underline{Q}$ on the possibly smaller sub-box $\tilde{\underline{x}}'$. We can stop at any iterate and evaluate $\underline{f}$ or $\underline{f}_c$, the centred form, on $\tilde{\underline{x}}'$ to obtain bounds on an extremal value of f in the original box $\tilde{\underline{x}}$.

(ix) We can accomplish the results of the technique (viii) without using $\underline{Q}$. We can use an interval extension $\underline{J}$ of J itself and solve the system of p linear algebraic equations with interval coefficients

$$\underline{M}\underline{b} = -\underline{h}'(m(\tilde{\underline{x}})) \tag{1.43}$$

where $\underline{M} = (\underline{m}_{jk})_{p\times p}$ and

$$(\underline{J}(\tilde{\underline{x}}))_{jk} \subseteq \underline{m}_{jk} \ (j,k = 1\ldots,p). \tag{1.44}$$

We can use the methods of Hansen [Han–65a] [HanS-67a] [Han–69b] to find a p–dimensional interval vector $\underline{b}$ containing the set of solutions to the system corresponding to real matrices chosen from the interval matrix $\underline{M}$. We can then take

$$\tilde{\underline{x}}' = \tilde{\underline{x}} \cap \{m(\tilde{\underline{x}}) + \underline{b}\} \tag{1.45}$$

(instead of $\tilde{\underline{x}}' = \tilde{\underline{x}} \cap \{m(\tilde{\underline{x}}) - \underline{Q}\underline{h}'(m(\tilde{\underline{x}}))\}$ as in 1.42). Again, if $\tilde{\underline{x}}' = \emptyset$, then f has its extrema on the boundary of $\tilde{\underline{x}}$. Otherwise we can continue to iterate 1.45, stopping at any desired iterate to evaluate $\underline{f}$ or $\underline{f}_c$ on $\tilde{\underline{x}}'$.

Another alternative interval version of Newton's method for systems of equations is the method of Krawczyk [Kra–69a]. We can use the iteration formula

$$\tilde{\underline{x}}' = \tilde{\underline{x}} \cap \{m(\tilde{\underline{x}}) - Y\underline{h}'(m(\tilde{\underline{x}})) + \underline{R}(\tilde{\underline{x}} - m(\tilde{\underline{x}}))\} \qquad 1.46$$

where

$$\underline{R} = I - Y\underline{J}(\tilde{\underline{x}}), \qquad 1.47$$

I is the identity matrix, and Y is an arbitrary nonsingular real matrix. We can choose Y as an approximate inverse of $\underline{J}(m(\tilde{\underline{x}}))$. The choice of $m(\tilde{\underline{x}})$ is not essential.

(x) The *method of Moore* can be summarized as follows.

If there is a way of re-writing the given expression for $f(x_1, \ldots, x_n)$ in such a way that each variable occurs at most once and to the first power only then (iii) can be used to find the range of values of f in one function evaluation using interval arithmetic. However, in general, this will not be possible.

If $(\underline{g}_i(\hat{\underline{x}}))_S \leq 0$ $(i \in S)$ and $(\underline{g}_i(\hat{\underline{x}}))_I \geq 0$ $(i \in T)$ then we can proceed as follows.

(1) If $T \cup S = \{1, \ldots, n\}$ then the technique of (vii) can be used directly to obtain the range of values with two function evaluations to the accuracy of the machine arithmetic used;

(2) if $T \cup S = \emptyset$ then we can attempt to apply the techniques of (viii) - (ix) and Skelboe's method [Ske–74a] on $\underline{\hat{x}}$;

(3) otherwise the technique (vi) can be used to find a part of the boundary of $\underline{\hat{x}}$ containing an argument giving the minimum value of f on $\underline{\hat{x}}$ (and another part for the maximum of f).

In case (2), if the first iteration of Krawczyk's formula does not produce a smaller box than $\underline{\hat{x}}$, then we could bisect the box $\underline{\hat{x}}$ in a co-ordinate direction for which $g_i(\underline{\hat{x}})$ has maximum width.

In case (2), f must also be bounded on the boundary of $\underline{\hat{x}}$ consisting of $2n$ faces of dimension $n-1$.

We can then, as Skelboe [Ske–74a] does, find the lower bound of f and then repeat the process with $-f$ to find the upper bound.

More detail can be found in [Moo–76a].

1.3.5 The Method of Mancini and McCormick

Mancini and McCormick [ManM-76a] have given computable sufficient conditions for the existence of a unique critical point x^* of a given twice continuously

differentiable function $f : R^n \to R^1$ in a convex compact set D; they prove the following theorem.

Theorem 1.6 : If (1) $f : D \subseteq R^n \to R^1$ is a given mapping with $f \in C^2(D)$; (2) $\hat{D} \subseteq D$ is a convex compact set; (3) $f''(x)$ is positive definite on $\hat{D}$; (4) for some $\hat{x} \in \hat{D}$

$$N(\hat{x}) = \{\hat{x} - \{\int_0^1 f''(\hat{x} + (x - \hat{x})s)ds\}^{-1} f'(\hat{x})^T \mid x \in \hat{D}\} \qquad 1.48$$

and $N(\hat{x}) \subseteq \hat{D}$, then $\exists x^* \in N(\hat{x})$ such that

$$f'(x^*) = 0, \qquad 1.49$$

$$f(x^*) = \inf_{x \in \hat{D}} f(x), \qquad 1.50$$

and x^* is the unique critical point of f in $\hat{D}$. Furthermore, if (5) $\hat{x} \in int(\hat{D})$ or (6) $N(\hat{x}) \subseteq int(\hat{D})$, then

$$tf'(\hat{x}) \in U, \ (\forall t \in [0, 1]) \qquad 1.51$$

where

$$U = \{u \in R^n \mid u = f'(x)\ (x \in \hat{D})\}$$

and

$$f(\hat{x}) - f(x^*) = f'(\hat{x}) \int_0^1 t\{f''(g(tf'(\hat{x})))\}^{-1} dt f'(\hat{x})^T, \qquad 1.52$$

in which $g : R^n \to R^1$ is such that

$$g(f'(x)) = x\ (\forall x \in \hat{D}),$$

and

$$f'(g(u)) = u\ (\forall u \in U). \square$$

Clearly the application of **Theorem 1.6** depends on the ability to computationally verify the hypotheses and to compute lower and upper bounds on the right-hand side of 1.52. According to Mancini and McCormick, if the function f is factorable [Mc—83a] and the set D is a box then interval arithmetic can be used to verify the hypotheses and to compute an interval bound on the right-hand side of 1.52.

1.3.6 The Method of Ichida and Fujii

Ichida and Fujii [IchF-79a] have introduced three algorithms, namely Algorithms A and B for bounding the global minimizer of the objective function, and Algorithm C for bounding the global minimizer subject to constraints.

An outline of Algorithm A is as follows.

Suppose that $\underline{f}: I(D) \to I(R)$ is an interval extension of $f: D \subseteq R^n \to R^1$. Let $\underline{\hat{x}} \in I(D)$ be a given box and let $\varepsilon > 0$. L_1 and L_2 are linked lists of pairs $(\underline{x}, \underline{f}(\underline{x}))$, ordered so that

$$((\underline{x}^{(1)}, \underline{f}(\underline{x}^{(1)})) \leq (\underline{x}^{(2)}, \underline{f}(\underline{x}^{(2)}))) \Leftrightarrow ((\underline{f}(\underline{x}^{(1)}))_I \leq (\underline{f}(\underline{x}^{(2)}))_I).$$

L_1, initially empty, ultimately contains $(\underline{x}, \underline{f}(\underline{x}))$ such that $\|w(\underline{x})\| > \varepsilon$ and L_2, initially empty, ultimately contains $(\underline{x}, \underline{f}(\underline{x}))$ such that $\|w(\underline{x})\| \leq \varepsilon$. Compute $\underline{\hat{f}} = \underline{f}(\underline{\hat{x}})$ and insert $(\underline{\hat{x}}, \underline{\hat{f}})$ into L_1. Then proceed as follows.

(1) If $L_1 = \emptyset$ then extract all the pairs from L_2. If L_2 contains

$$(\underline{x}^{*1}, \underline{f}(\underline{x}^{*1})), \ldots, (\underline{x}^{*m}, \underline{f}(\underline{x}^{*m}))$$

then $\bigcup_{i=1}^{m} \underline{x}^{*i}$ and $\bigcup_{i=1}^{m} \underline{f}(\underline{x}^{*i})$ which contain x^* and f^* respectively, are formed and the algorithm is terminated. If $L_1 \neq \emptyset$ then extract $(\underline{x}, \underline{f}(\underline{x}))$ which has the smallest lower bound $(\underline{f}(\underline{x}))_I$ from L_1.

(2) Determine $k \in \{1, \ldots, n\}$ such that

$$w = w(\underline{x}_k) = \max_{1 \le i \le n} \{w(\underline{x}_i)\}$$

If $w \le \varepsilon$ then insert $(\underline{x}, \underline{f}(\underline{x}))$ into L_2 and go to (1); otherwise form the sub-boxes

$$\underline{x}^{(1)} = (\underline{x}_1, \ldots, \underline{x}_{k-1}, [x_{kI}, x_m], \underline{x}_{k+1}, \ldots, \underline{x}_n)^T$$

and

$$\underline{x}^{(2)} = (\underline{x}_1, \ldots, \underline{x}_{k-1}, [x_m, x_{kS}], \underline{x}_{k+1}, \ldots, \underline{x}_n)^T$$

where $x_m = m(\underline{x}_k)$, and then compute

$$\underline{f}^{(1)} = \underline{f}(\underline{x}^{(1)}) \text{ and } \underline{f}^{(2)} = \underline{f}(\underline{x}^{(2)}).$$

(3) Let $x_m^{(1)} = m(\underline{x}^{(1)})$ and let $x_m^{(2)} = m(\underline{x}^{(2)})$. Compute $\underline{f}_m^{(1)} = \underline{f}(x_m^{(1)})$. If $f_{mS}^{(1)} < f_I^{(2)}$ then $(\underline{x}^{(2)}, \underline{f}^{(2)})$ can be eliminated because $\underline{x}^{(2)}$ and $\underline{f}^{(2)}$ do not contain x^* and f^* respectively. Therefore if $f_{mS}^{(1)} < f_I^{(2)}$ then insert $(\underline{x}^{(1)}, \underline{f}^{(1)})$ into L_1 and go to (1). Otherwise, go to (4).

(4) Compute $\underline{f}_m^{(2)} = \underline{f}(x_m^{(2)})$. If $f_{mS}^{(2)} < f_I^{(1)}$ then $(\underline{x}^{(1)}, \underline{f}^{(1)})$ can be eliminated because $\underline{x}^{(1)}$ and $\underline{f}^{(1)}$ do not contain x^* and f^* respectively. Therefore if $f_{mS}^{(2)} < f_I^{(1)}$ then insert $(\underline{x}^{(2)}, \underline{f}^{(2)})$ into L_1 and go to (1). Otherwise, go to (5).

(5) Insert $(\underline{x}^{(1)}, \underline{f}^{(1)})$ and $(\underline{x}^{(2)}, \underline{f}^{(2)})$ into L_1 and go to (1). □

If the objective function contains many global minimizers, say $x^{*1}, \ldots, x^{*m}$, then the solution $\bigcup_{i=1}^{m} \underline{x}^{*i}$ obtained from (1) of **Algorithm** A is very wide and must be divided into subgroups separately by using **Algorithm** B [IchF-79a].

An outline of the **Algorithm** B [IchF-79a] is as follows.

(1)′ Suppose that the solutions $\underline{x}^{*1}, \ldots, \underline{x}^{*m}$ are obtained after applying **Algorithm** A to the box $\underline{\hat{x}}$.

(2)′ Select $\underline{x}^{*1}$ as the first member of a group of sub-boxes and add a sub-box $\underline{x}^{*j}$ $(j = 2, \ldots, m)$ to that group if it has a common boundary with a sub-box which is already in the group. This process is repeated until all the sub-boxes $\underline{x}^{*i}$ $(i = 1, \ldots, m)$ are collected into groups.

(3)′ Compute the range of values of f for each group and the union of these ranges gives the desired global minimum bound. The global minimizer bounds and the global minimum bound can be made sufficiently small by using Newton's method [Moo–66a] on the groups mentioned in (2)′. □

If S in 1.1 is irregular in shape [IchF-79a] then **Algorithms** A and B cannot be used unless S can be transformed into a box. However, the transformation can be done in special cases only. To overcome this disadvantage, Ichida and Fujii [IchF-79a] have suggested the use of the Lagrange-multiplier technique which is described in **Algorithm** C.

Recently Ichida and Fujii [IchF-85a] have described another global optimization algorithm in which an interval Newton method is used.

1.3.7 The Methods of Hansen

Hansen [Han–79a] has described an algorithm for bounding the global minimizer of a function $f : D \subseteq R^1 \to R^1$ with $f \in C^2(\hat{D})$ where $\hat{D} \subset D$ is a bounded closed interval. Hansen assumes that f' and f'' have only a finite number of isolated zeros in the given initial interval $\underline{\hat{x}} \in I(\hat{D})$. An outline of this method is as follows.

Hansen's algorithm [Han–79a] iteratively deletes subintervals of $\underline{\hat{x}}$ until the remaining intervals are sufficiently small and contain the global minimizers of f. In order to delete subintervals of $\underline{\hat{x}}$ the following techniques are used.

An interval $\underline{\bar{f}} \in I(R)$ such that the global minimum $f^* \leq \bar{f}_S$, is computed initially from $\underline{\bar{f}} = [(\underline{f}(\hat{x}_I))_S, (\underline{f}(\hat{x}_I))_S]$ if $(\underline{f}(\hat{x}_I))_S \leq (\underline{f}(\hat{x}_S))_S$ and $\underline{\bar{f}} = [(\underline{f}(\hat{x}_S))_S, [(\underline{f}(\hat{x}_S))_S]$ otherwise, and is updated continually. The interval $\underline{\bar{f}}$ is used to delete subintervals $\underline{x}$ of $\underline{\hat{x}}$ such that $x^* \notin \underline{x}$. If $f(x) > \bar{f}_S$ $(\forall x \in \underline{x})$ then $f(x) > f^*$ $(\forall x \in \underline{x})$ so $x^* \notin \underline{x}$.

The concavity property of f is used by Hansen [Han–79a] to delete the subinterval $\underline{x}$ of $\underline{\hat{x}}$ by computing $\underline{f''} = \underline{f''}(\underline{x})$. If $f''_S < 0$ then f is concave in $\underline{x}$ and f cannot have a minimizer in the interior of $\underline{x}$. Therefore the interior of $\underline{x}$ can be deleted.

The interval Newton method [Moo–66a] and the extended interval Newton method [Han–78a] are also used to delete subintervals of $\underline{\hat{x}}$.

Hansen [Han–79a] also has used a quadratic approximation of the function f in order to delete subintervals of $\underline{\hat{x}}$. More detail can be found in [Han–79a]. Hansen's algorithm [Han–79a] is implemented in [Moh–84a].

Hansen's algorithm [Han–79a] has been extended by Hansen [Han–80a] in order to bound the global minimizers of $f : R^n \rightarrow R^1$. The implementation of Hansen's algorithm H for the n-dimensional case [Han–80a] is described in Chapter 3 of this thesis.

1.3.3 The Method of Hansen and Sengupta

Hansen and Sengupta [HanS-80a] have extended Hansen's algorithm, H, [Han–80a] to solve the problem

$$\left.\begin{array}{ll} \textit{minimize} & f(x) \\ \textit{subject to} & p_i(x) \leq 0 \quad (i = 1, \ldots, m) \end{array}\right\} \qquad 1.53$$

where $f \in C^2(\hat{D})$ and $p_i \in C^1(\hat{D})$ $(i = 1, \ldots, m)$ in which $\hat{D} \subset D$ is an open convex subset of the feasible set (Chapter 4).

In order to give an outline of this method we need the following definitions.

Definition 1.5 : The point $x \in \underline{x} \in I(\hat{D})$ is *certainly feasible* if and only if $(\underline{p}_i(x))_S \leq 0$ $(i = 1, \ldots, m)$. □

Definition 1.6 : The box $\underline{x} \in I(\hat{D})$ is *certainly feasible* if and only if $(\underline{p}_i(\underline{x}))_S \leq 0$ $(i = 1, \ldots, m)$. □

It follows from **Definitions 1.3** and **1.4** that if $\underline{x}$ is certainly feasible then $(\forall x \in \underline{x})$, x is certainly feasible.

Hansen and Sengupta [HanS-80a] have shown that the monotonicity and non-convexity tests which are described in [Han–80a] can be used in the constrained case.

If $\underline{x}$ is not certainly feasible then the interval Newton method ([Moo–66a][HanS-81a]) cannot be used (at least not easily) [HanS-80a]. For this reason, Hansen and Sengupta [HanS-80a] have used a linear interval extension rather than a quadratic interval extension [Han–80a] for f in order to delete parts of $\underline{x}$ which do not contain a global minimizer of f. Hansen and Sengupta [HanS-80a] have also described how to use the constraints given in 1.53 for deleting points which are not feasible.

In order to improve the value of the upper bound $\bar{f}$ where $\bar{f}$ is computed as a degenerate interval $\underline{\bar{f}}$ which is to be used as described in §1.3.7, Hansen and Sengupta [HanS-80a] have suggested using a non-interval algorithm to find a better replacement for $\bar{f}$. They have also described in detail how to use a line search for updating $\bar{f}$. Furthermore, Hansen and Sengupta [HanS-80a] have explained briefly how to use their method to solve linear programming and integer programming problems.

1.3.9 The Method of Asaithambi, Shen and Moore

Let $f : D \subseteq R^n \to R^1$ be a given mapping with $f \in C^1(\hat{D})$ where $\hat{D} \subset D$ is an open convex set. Let $\underline{f} : I(\hat{D}) \to I(R^1)$ and $\underline{f}' : I(\hat{D}) \to I(R^n)$ be continuous inclusion monotonic interval extensions of $f : \hat{D} \to R^1$ and of $f' : \hat{D} \to R^n$ respectively. Let $\hat{\underline{x}} \in I(\hat{D})$ be given.

Asaithambi, Shen and Moore [AsSM-82a] have described a method for computing the range of values

$$\overline{f}(\hat{\underline{x}}) = \{f(x_1, \ldots, x_n) \mid x_i \in \hat{\underline{x}}_i \ (i = 1, \ldots, n)\}. \tag{1.54}$$

By Theorem 1.2 we obtain

$$\overline{f}(\hat{\underline{x}}) \subseteq \underline{f}(\hat{\underline{x}}), \tag{1.55}$$

but the width of $\underline{f}(\hat{\underline{x}})$ may exceed the width of the exact range of values 1.54 by an unacceptable amount. Define the excess width e of an interval $[a, b]$ on the range of values $\overline{f}(\hat{\underline{x}}) \subseteq [a, b]$ as

$$e = w([a, b]) - w(\overline{f}(\hat{\underline{x}})). \tag{1.56}$$

By 1.54 and 1.56 a and b are the minimum and maximum values of f in $\hat{\underline{x}}$ respectively if and only if $e = 0$.

In order to obtain the smallest possible value of e, the following ideas are used by Asaithambi, Shen and Moore.

(1) Compute a lower bound on the minimum value of f over $\hat{\underline{x}}$ and then apply the algorithm to $-f$ to bound the maximum value of f.

(2) From the initial box $\hat{\underline{x}}$, a list of sub-boxes is generated whose union must contain the global minimizer. The elements in the list are generated in pairs; each pair is the result of a bisection in a single co-ordinate direction of some previous box in the list. The elements are entered into the list in order of increasing lower bounds of f.

(3) The sub-box $\underline{x}$ of $\hat{\underline{x}}$ is bisected at the first direction in which $\underline{x}$ has maximum width.

(4) Suppose that the sub-box $\underline{x}$ of $\hat{\underline{x}}$ is bisected into $\underline{x}^{(1)}$ and $\underline{x}^{(2)}$. If $(\underline{f}(\underline{x}^{(1)}))_I > v$ where $v = f(x)$ for some $x \in \underline{x}^{(2)}$, say the midpoint of $\underline{x}^{(2)}$, then $\underline{x}^{(1)}$ can be deleted since the global minimizer must lie in $\underline{x}^{(2)}$. This is called the *midpoint test*.

(5) If for $1 \leq i \leq n$ $(\underline{f}'_i(\underline{x}))_I \geq 0$ then replace $\underline{x}_i$ with $[x_{iI}, x_{iI}]$ and if $(\underline{f}'_i(\underline{x}))_S \leq 0$ then replace $\underline{x}_i$ with $[x_{iS}, x_{iS}]$.

(6) The *mean value form* [Moo–79a]

$$\underline{f}_{MV}(\underline{x}) = f(m(\underline{x})) + \sum_{i=1}^{n} \underline{f}_i'(\underline{x})(\underline{x}_i - m(\underline{x}_i)) \qquad 1.57$$

corresponding to f is combined with (5) to produce the *monotonicity test form* [Moo–79a]

$$\underline{f}_{MT}(\underline{x}) = [f(u), f(v)] + \sum_{i \in T} \underline{f}_i'(\underline{x})(\underline{x}_i - m(\underline{x}_i)) \qquad 1.58$$

where T is the set of integers i such that $\underline{f}_i'(\underline{x})$ properly contains zero, and for $i = 1, \ldots, n$,

$$(u_i, v_i) = \begin{cases} (x_{iI}, x_{iS}) & ((\underline{f}_i'(\underline{x}))_I \geq 0) \\ (x_{iS}, x_{iI}) & ((\underline{f}_i'(\underline{x}))_S \leq 0)\,. \\ (m(\underline{x}_i), m(\underline{x}_i)) & (i \in T) \end{cases} \qquad 1.59$$

(7) The computation is terminated when there is no further increase in the current lower bound on the interval extension.

(8) During the computation of $\underline{f}_{MT}(\underline{x})$ given by 1.58, the set T and the reduced box $\underline{y}$ with

$$\underline{y}_i = \begin{cases} [u_i, u_i] & (i \notin T) \\ \underline{x}_i & (i \in T) \end{cases} \qquad 1.60$$

are found.

The algorithm in which the preceding ideas are used is given in [AsSM-82a].

1.3.10 The Method of Cornelius and Lohner

Let $f : [a,b] \subseteq R^1 \to R^1$ be a given mapping. Cornelius and Lohner [CorL-84a] have described a method for finding an interval $\underline{f}_v(\underline{x})$ which contains the range of values of f, that is

$$\overline{f}(\underline{x}) = \{f(x) \mid x \in \underline{x}\} \subseteq \underline{f}_v(\underline{x}), \qquad 1.61$$

on a subinterval $\underline{x} \subseteq [a,b]$.

In order to obtain high accuracy f must satisfy additional assumptions. Let f be m times differentiable and let each $f^{(k)}$ $(k = 1, \ldots, m)$ have an interval extension for any interval $\underline{x} \subseteq [a,b]$.

If $\sigma(.,.) : I(R) \times I(R) \to R$ is a metric (See A22.) then $\sigma(\overline{f}(\underline{x}), \underline{f}_v(\underline{x}))$ is a measure of the amount by which $\underline{f}_v(\underline{x})$ overestimates $\overline{f}(\underline{x})$. If $\underline{f}_v(\underline{x})$ satisfies

$$\sigma(\underline{\bar{f}}(\underline{x}), \underline{f}_v(\underline{x})) \leq c(w(\underline{x}))^n \qquad 1.62$$

with fixed $n \in N$ and $c = c([a,b]) \geq 0$ ($c([a,b])$ means that c depends on $[a,b]$) then $\underline{f}_v(\underline{x})$ is called an n-th order approximation of $\underline{\bar{f}}(\underline{x})$.

The following theorems the proofs of which can be found in [Moo–66a] and [Moo–79a], are needed in this discussion.

Theorem 1.7 : If $f : [a,b] \subseteq R^1 \rightarrow R^1$ satisfies the Lipschitz condition ([Moo–66a][Moo–79a])

$$| f(x) - f(y) | \leq L | x - y | \quad (x, y \in [a,b])$$

where L is a constant then the natural interval extension $\underline{f} : I([a,b]) \rightarrow I(R)$ of f satisfies ($\forall \underline{x} \in I([a,b])$)

$$\underline{\bar{f}}(\underline{x}) \subseteq \underline{f}(\underline{x}), \qquad 1.63a$$

$$\sigma(\underline{\bar{f}}(\underline{x}), \underline{f}(\underline{x})) \leq c_1 w(\underline{x}) \ (c_1 \geq 0), \qquad 1.63b$$

and

$$w(\underline{f}(\underline{x})) \leq c_2 w(\underline{x}) \ (c_2 \geq 0). \ \square \qquad 1.63c$$

From **Theorem 1.7** it follows that if for some sequence $(\underline{x}^{(k)})$, $w(\underline{x}^{(k)}) \to 0 \ (k \to \infty)$, then $\sigma(\bar{\underline{f}}(\underline{x}^{(k)}), \underline{f}(\underline{x}^{(k)})) \to 0 \ (k \to \infty)$ at least linearly with $w(\underline{x}^{(k)})$.

Theorem 1.8 : Let $f : [a,b] \subseteq R^1 \to R^1$ be differentiable and let $\underline{f}' : I([a,b]) \to I(R^1)$ be a continuous inclusion monotonic interval extension of $f' : [a,b] \to R^1$ that satisfies **Theorem 1.7**. Then for the mean value form

$$\underline{f}_y(\underline{x}) = f(y) + \underline{f}'(\underline{x})(\underline{x} - y) \qquad 1.64$$

for fixed $y \in \underline{x}$, there holds $(\forall \underline{x} \in I([a,b]))$

$$\bar{\underline{f}}(\underline{x}) \subseteq \underline{f}_y(\underline{x}), \qquad 1.65a$$

$$\sigma(\bar{\underline{f}}(\underline{x}), \underline{f}_y(\underline{x})) \leq c_3 (w(\underline{x}))^2 \ (c_3 \geq 0). \ \square \qquad 1.65b$$

Thus if for some sequence $(\underline{x}^{(k)})$, $w(\underline{x}^{(k)}) \to 0 \ (k \to \infty)$ then $\sigma(\bar{\underline{f}}(\underline{x}^{(k)}), \underline{f}_y(\underline{x}^{(k)})) \to 0$ $(k \to \infty)$ at least quadratically with $w(\underline{x}^{(k)})$.

We shall describe the basic approach used in the method of Cornelius and Lohner.

Let f have a representation of the form

$$f(x) = g(x) + r(x) \ (\forall x \in [a,b]) \qquad 1.66$$

with continuous functions g and r. Let $\underline{r} : I([a,b]) \to I(R)$ be such that

$$r(x) \in \underline{r}(\underline{x}) \subseteq \underline{r}([a,b]) \ (\forall x \in \underline{x} \subseteq [a,b]). \qquad 1.67$$

The function g can be interpreted as an approximation of f, and r as the corresponding remainder term. The intervals $\underline{r}(\underline{x})$ and $\underline{r}([a,b])$ bound the remainder term over $\underline{x}$ and $[a,b]$ respectively.

The remainder form $\underline{f}_v : I([a,b]) \to I(R)$ of the representation 1.66 of f is defined by

$$\underline{f}_v(\underline{x}) = \overline{g}(\underline{x}) + \underline{r}(\underline{x}). \qquad 1.68$$

In order to compute $\underline{f}_v(\underline{x})$ we do not use an interval extension of $g(x)$ but rather we use the exact range of g on $\underline{x}$. However, $\underline{r}(\underline{x})$ can be an interval extension of $r(x)$

or any other inclusion for $r(x)$ on $\underline{x}$ [CorL-84a]. The use of $\underline{g}(\underline{x})$ implies that, in practice, we can choose only very simple functions g, for example the polynomials of degree at most 5 or monotone functions.

Theorem 1.9 : Let the continuous function $f : [a,b] \subseteq R^1 \to R^1$ have the representation 1.66 and let $\underline{f}_v(\underline{x})$ be the remainder form 1.68. Then $(\forall \underline{x} \in I([a,b]))$

$$\overline{f}(\underline{x}) \subseteq \underline{f}_v(\underline{x}), \tag{1.69a}$$

and

$$\sigma(\overline{f}(\underline{x}), \underline{f}_v(\underline{x})) \leq w(\underline{r}(\underline{x})) \leq 2|\underline{r}(\underline{x})|. \tag{1.69b}$$

Proof : A proof of Theorem 1.9 is given in [CorL-84a]. □

For the numerical applications of **Theorem 1.9**, it is important to use simple functions g in order to be able to compute $\underline{g}(\underline{x})$. On the other hand $w(\underline{r}(\underline{x}))$ should be sufficiently small for $\underline{f}_v(\underline{x})$ to be a close bound for $\overline{f}(\underline{x})$. Cornelius and Lohner [CorL-84a] have suggested using the *Hermite interpolating polynomial* for this purpose.

Let $p_s(x)$ be the unique polynomial of degree $s \geq 0$ such that

$$p_s^{(j)}(x_i) = f^{(j)}(x_i) \ (j = 0, \ldots, m_i - 1; \ i = 0, \ldots, k), \tag{1.70}$$

where $x_0, \ldots, x_k \in \underline{x}$ are $k+1$ distinct points and $m_0, \ldots, m_k$ are such that $s+1 = \sum_{i=0}^{k} m_i$. If f is $s+1$ times continuously differentiable, then $(\forall x \in \underline{x})$

$$f(x) = p_s(x) + (1/(s+1)!) f^{(s+1)}(\xi(x)) \prod_{i=0}^{k} (x - x_i)^{m_i}, \tag{1.71}$$

where $\xi(x) \in \underline{x}$. Let $\underline{d}_{s+1}(\underline{x})$ and $\underline{d}_{s+1}([a,b])$ be intervals with

$$f^{(s+1)}(x) \in \underline{d}_{s+1}(\underline{x}) \subseteq \underline{d}_{s+1}([a,b]) \ (\forall x \in \underline{x} \in I([a,b])). \tag{1.72}$$

If 1.71 is taken as a representation of f of the form 1.66 then the interpolation form $\underline{f}_{v_s}(\underline{x})$ according to 1.68 is defined by

$$\underline{f}_{v_s}(\underline{x}) = \underline{p}_s(\underline{x}) + (1/(s+1)!) \underline{d}_{s+1}(\underline{x}) \prod_{i=0}^{k} (\underline{x} - x_i)^{m_i}, \tag{1.73}$$

where

$$(\underline{x} - x_i)^{m_i} = \prod_{j=1}^{m_i} (\underline{x} - x_i).$$

If $f^{(s+1)}$ has an interval extension $\underline{f}^{(s+1)}$ over $\underline{x}$ satisfying Theorem 1.7, and if $h^{(s+1)}$ is an arbitrary fixed point in $\underline{f}^{(s+1)}(\underline{x})$ so that

$$h^{(s+1)} \in \underline{f}^{(s+1)}(\underline{x}), \tag{1.74}$$

then we use

$$f(x) = q_{s+1}(x) + (1/(s+1)!)(f^{(s+1)}(\xi(x)) - h^{(s+1)}) \prod_{i=0}^{k} (x - x_i)^{m_i}, \tag{1.75}$$

where

$$q_{s+1}(x) = p_s(x) + (h^{(s+1)}/(s+1)!) \prod_{i=0}^{k} (x - x_i)^{m_i}, \tag{1.76}$$

as a representation of the form 1.66 and define the interpolation form $\underline{f}_{u_s}(\underline{x})$ according to

$$\underline{f}_{u_s}(\underline{x}) = \underline{q}_{s+1}(\underline{x}) + (1/(s+1)!)(\underline{f}^{(s+1)}(\underline{x}) - h^{(s+1)}) \prod_{i=0}^{k} (\underline{x} - x_i)^{m_i}. \tag{1.77}$$

The following result shows that both 1.73 and 1.77 can be approximations of high order.

Theorem 1.10 : Let $\underline{f}_{v_s}(\underline{x})$ be defined as in 1.73 such that 1.72 holds, and let $\underline{f}_{u_s}(\underline{x})$ be defined as in 1.77 such that Theorem 1.7 holds for $f^{(s+1)}$. Then $(\forall \underline{x} \in I([a,b]))$

$$\underline{\bar{f}}(\underline{x}) \subseteq \underline{f}_{v_s}(\underline{x}), \tag{1.78a}$$

$$\sigma(\underline{\bar{f}}(\underline{x}), \underline{f}_{v_s}(\underline{x})) \leq \alpha_{s+1}(w(\underline{x}))^{s+1} \tag{1.78b}$$

with $\alpha_{s+1} = \alpha_{s+1}([a,b]) \geq 0$ and

$$\underline{\bar{f}}(\underline{x}) \subseteq \underline{f}_{u_s}(\underline{x}), \tag{1.79a}$$

$$\sigma(\underline{\bar{f}}(\underline{x}), \underline{f}_{u_s}(\underline{x})) \leq \beta_{s+2}(w(\underline{x}))^{s+2} \tag{1.79b}$$

with $\beta_{s+2} = \beta_{s+2}([a,b]) \geq 0$.

Proof : A proof of Theorem 1.10 is given in [CorL-84a]. □

The approximations $\underline{f}_{v_s}(\underline{x})$ and $\underline{f}_{u_s}(\underline{x})$ can be improved by refining the method of evaluating the products in the remainder terms and this is desribed in detail in [CorL-84a].

One of the most commonly-used forms for the approximation of $\underline{f}(\underline{x})$ is the mean value form $\underline{f}_y(\underline{x})$ given by 1.64, whose convergence is quadratic when f' satisfies **Theorem 1.7**. The discussion of this approach is given in [CorL-84a].

Cornelius and Lohner [CorL-84a] state that their method can be applied to the n-dimensional case.

1.4 A New Algorithm for Global Optimization in which Interval Arithmetic is used

Recently, Shearer and Wolfe [SheW-85a] have described some computable existence, uniqueness, and convergence tests for systems of nonlinear algebraic equations, and have also described an improved form, $KMSW$, [SheW-85c] of the Krawczyk-Moore algorithm [Qi—80a] and an improved form, MAP, [SheW-85b] of the Alefeld-Platzöder algorithm [AleP-83a] for bounding solutions of systems of nonlinear algebraic equations.

This thesis contains a description of a new algorithm, MW, for the global optimization problem 1.1. The algorithm MW incorporates ideas due to Robinson [Rob–73a], Hansen [Han 30a], and to Shearer and Wolfe [SheW-85a], [SheW-85b], [SheW-85c].

Computational experience indicates that MW is usually more efficient than the algorithm H of Hansen [Han–80a].

1.5 Plan of Thesis

Chapter 2 contains preliminary results which are used throughout the thesis. An implementation of algorithm H is described in Chapter 3. In Chapter 4, problem 1.1 is expressed as a system of nonlinear equations and inequalities in a manner similar to that which has been used by Robinson [Rob–73a]. The system of nonlinear algebraic equations and inequalities which is obtained in Chapter 4 is solved by using the methods due to Shearer and Wolfe [SheW-85a], [SheW-85b], [SheW-85c]; this is described in Chapter 5. The new algorithm, MW, in which the ideas which have been described in Chapters 3, 4, and 5 are used, is described in Chapter 6. Numerical results are given in Chapter 7.

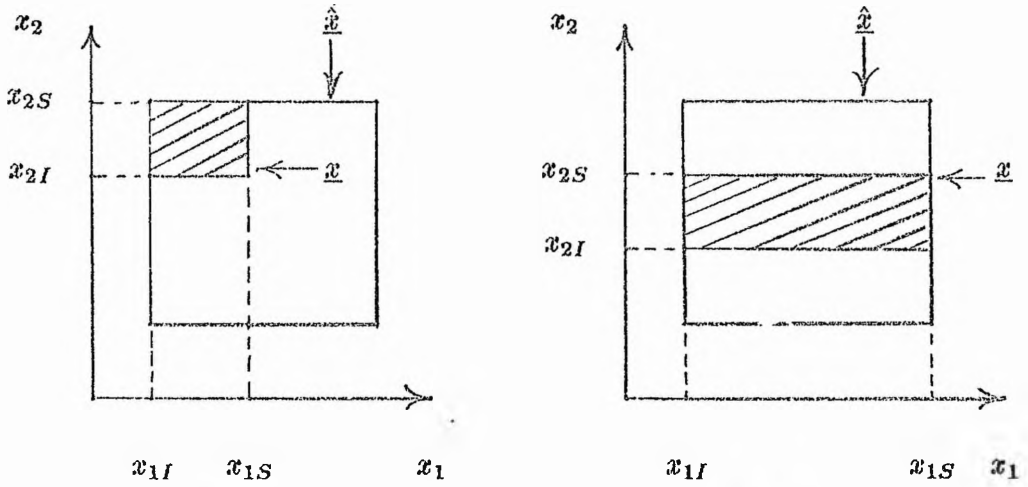
The notation which is used throughout this thesis is described in Appendix A. The algorithms in this thesis are expressed in a pseudo-code which is based on S-algol [ColM-82a] and which is described in Appendix B. Appendix C contains the global optimization problems corresponding to the numerical results which are given in Chapter 7. Appendix D contains pseudo-code from which the functions, gradients, and Hessians corresponding to the problems which are given in Appendix C can be computed and an explanation of the procedures *time* and *decode* [ColM-82a].

CHAPTER 2

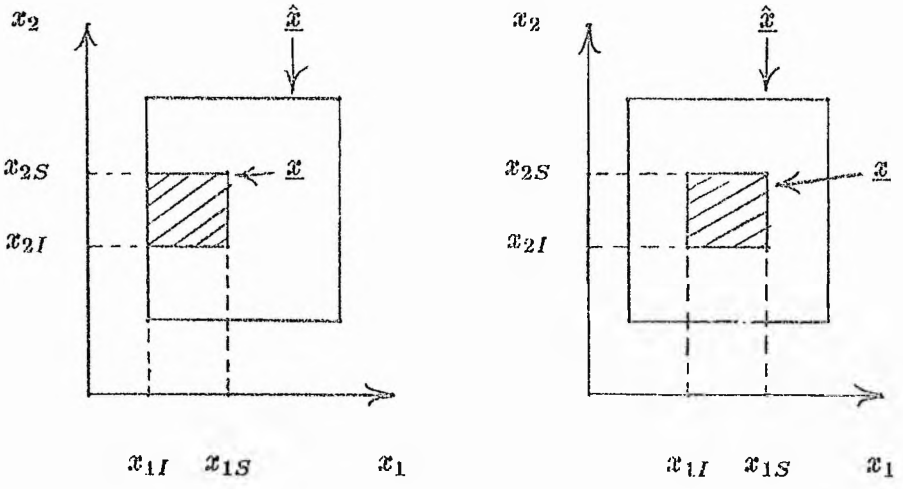
Preliminary Results

This chapter contains certain results which are used throughout this thesis. The notation which is used throughout this thesis is described in Appendix A.

Configuration 2.1 : Suppose that $\underline{x} \in I(R^n)$ is a sub-box of $\hat{\underline{x}}$. Then the smallest box $\underline{x}'$ containing the boundary points of $\hat{\underline{x}}$ which lie in $\underline{x}$ is given by the following configurations (for $n = 2$).



(a) $\underline{x}' = \underline{x}$ (b) $\underline{x}' = \underline{x}$



(c) $\underline{x}' = ([x_{1I}, x_{1I}], x_2)^T$ (d) $\underline{x}' = \emptyset$

There are three other configurations similar to (a) and one other configuration similar to (b). In both (a) and (b) and similar configurations $\underline{x}' = \underline{x}$. There are three other configurations similar to (c), where in all of these configurations $\underline{x}' \subset \underline{x}$ and in configuration (c) $\underline{x}' = ([x_{1I}, x_{1I}], \underline{x}_2)^T$. In configuration (d) $\underline{x}' = \emptyset$. □

Furthermore, for $n > 2$, if $\underline{x}$ and $\hat{\underline{x}}$ share at least two common faces then $\underline{x}' = \underline{x}$, if $\underline{x}$ and $\hat{\underline{x}}$ share exactly one common face then $\underline{x}' \subset \underline{x}$, and if $\underline{x}$ and $\hat{\underline{x}}$ share no common face then $\underline{x}' = \emptyset$.

Lemma 2.1 : If $f: D \subseteq R^n \to R^1$ is a given mapping with $f \in C^1(\hat{D})$ where $\hat{D} \subset D$ is an open convex set, then $(\forall x, y \in \hat{D})\ \exists \theta_i \in [0,1]$ $(i = 1, \ldots, n)$ such that

$$f(y) = f(x) + \sum_{i=1}^{n} \partial_i f(y_1, \ldots, y_{i-1}, \xi_i, x_{i+1}, \ldots, x_n)(y_i - x_i) \qquad 2.1$$

where $\xi_i = x_i + \theta_i(y_i - x_i)$.

Proof: By Taylor's theorem for $i = 1, \ldots, n$ $\exists \xi_1, \xi_2, \xi_3$ such that

$$f(y_1, x_2, \ldots, x_n) = f(x) + \partial_1 f(\xi_1, x_2, \ldots, x_n)(y_1 - x_1),$$

$$f(y_1, y_2, x_3, \ldots, x_n) = f(y_1, x_2, \ldots, x_n) + \partial_2 f(y_1, \xi_2, x_3, \ldots, x_n)(y_2 - x_2)$$

$$= f(x) + \partial_1 f(\xi_1, x_2, \ldots, x_n)(y_1 - x_1) +$$

$$\partial_2 f(y_1, \xi_2, x_3, \ldots, x_n)(y_2 - x_2),$$

and

$$f(y_1, y_2, y_3, x_4, \ldots, x_n) = f(y_1, y_2, x_3, \ldots, x_n) +$$

$$\partial_3 f(y_1, y_2, \xi_3, x_4, \ldots, x_n)(y_3 - x_3)$$

$$= f(x) + \sum_{i=1}^{3} \partial_i f(y_1, \ldots, y_{i-1}, \xi_i, \ldots, x_n)(y_i - x_i).$$

In general, we have, for $i = 1, \ldots, n$

$$f(y) = f(x) + \sum_{i=1}^{n} \partial_i f(y_1, \ldots, y_{i-1}, \xi_i, x_{i+1}, \ldots, x_n)(y_i - x_i) \qquad 2.2$$

where for some $\theta_i \in [0,1]$, $\xi_i = x_i + \theta_i(y_i - x_i)$. □

Lemma 2.2 : If $f: D \subseteq R^n \to R^n$ is a given mapping with $f \in C^1(\hat{D})$ where $\hat{D} \subset D$ is an open convex set, then $(\forall x, y \in \hat{D})\ \exists \theta_{ij} \in [0,1]$ $(i,j = 1, \ldots, n)$ such that

$$f_i(y) = f_i(x) + \sum_{j=1}^{n} \partial_j f_i(y_1, \ldots, y_{j-1}, \eta_{ij}, x_{j+1}, \ldots, x_n)(y_j - x_j), \qquad 2.3$$

$(i = 1, \ldots, n)$ where $\eta_{ij} = x_j + \theta_{ij}(y_j - x_j)$ $(i,j = 1, \ldots, n)$.

Proof : Lemma 2.2 is an immediate consequence of Lemma 2.1. □

Definition 2.1 : Suppose that $\underline{f}: I(D) \subset I(R^n) \to I(R^1)$ is a given mapping. Then $\underline{f}$ is *inclusion monotonic* if and only if $(\underline{x} \subseteq \underline{y} \in I(D)) \Rightarrow (\underline{f}(\underline{x}) \subseteq \underline{f}(\underline{y}))$. □

Lemma 2.3 : If (1) $f: D \subseteq R^n \to R^n$ is a given mapping with $f \in C^1(\hat{D})$ where $\hat{D} \subset D$ is an open convex set; (2) $\underline{f}': I(\hat{D}) \to I(M(R^n))$ is an inclusion monotonic interval extension of $f': \hat{D} \to M(R^n)$; (3) $\hat{\underline{x}} \in I(\hat{D})$, then $(\forall x, y \in \hat{\underline{x}})$

$$f(y) \in f(x) + \underline{f}'(\hat{\underline{x}})(y - x)$$

$$\subseteq f(x) + \underline{f}'(\hat{\underline{x}})(\hat{\underline{x}} - x).$$

Proof: Let $x, y \in \hat{\underline{x}}$ be given. By Lemma 2.2, for $(i, j = 1, \ldots, n)$, $\exists \theta_{ij} \in [0, 1]$ such that

$$f_i(y) = f_i(x) + \sum_{j=1}^{n} \partial_j f_i(y_1, \ldots, y_{j-1}, \eta_{ij}, x_{j+1}, \ldots, x_n)(y_j - x_j),$$

where $\eta_{ij} = x_j + \theta_{ij}(y_j - x_j)$. If

$$\underline{f}'(\hat{\underline{x}}) = (\partial_j \underline{f}_i(\hat{\underline{x}}))_{n \times n}$$

then for $i, j = 1, \ldots, n$,

$$\partial_j f_i(y_1, \ldots, y_{j-1}, \eta_{ij}, x_{j+1}, \ldots, x_n) \in \partial_j \underline{f}_i(\hat{\underline{x}}),$$

since by convexity of $\hat{\underline{x}}$, $\eta_{ij} \in \hat{\underline{x}}_j$. Therefore

$$f_i(y) \in f_i(x) + \sum_{j=1}^{n} \partial_j \underline{f}_i(\hat{\underline{x}})(y_j - x_j) \ (i = 1, \ldots, n),$$

whence

$$f(y) \in f(x) + \underline{f}'(\hat{\underline{x}})(y - x).$$

Finally, $(y \in \hat{\underline{x}}) \Rightarrow (\underline{f}'(\hat{\underline{x}})(y - x) \subseteq \underline{f}'(\hat{\underline{x}})(\hat{\underline{x}} - x))$. So

$$f(y) \in f(x) + \underline{f}'(\hat{\underline{x}})(\hat{\underline{x}} - x). \quad \square$$

Lemma 2.4 : If (1) $f: D \subseteq R^n \to R^n$ is a given mapping with $f \in C^1(\hat{D})$ where $\hat{D} \subset D$ is an open convex set; (2) $\underline{f}': I(\hat{D}) \to I(M(R^n))$ is an inclusion monotonic interval extension of $f': \hat{D} \to M(R^n)$; (3) $\hat{\underline{x}} \in I(\hat{D})$, then $(\forall x, y \in \hat{\underline{x}})$

$$f(y) \in f(x) + \underline{f}'_H(\hat{\underline{x}}, y)(y - x)$$

$$\subseteq f(x) + \underline{f}'_H(\hat{\underline{x}}, y)(\hat{\underline{x}} - x)$$

$$\subseteq f(x) + \underline{f}'(\hat{\underline{x}})(\hat{\underline{x}} - x),$$

where $\underline{f}'_H: I(\hat{D}) \times I(\hat{D}) \to I(M(R^n))$ is defined by

$$\underline{f}'_H(\underline{x}, \underline{y}) = (\partial_j \underline{f}_i(\underline{x}_1, \cdots, \underline{x}_j, \underline{y}_{j+1}, \cdots, \underline{y}_n))_{n \times n}.$$

Proof : The proof of Lemma 2.4 is similar to that of Lemma 2.3. □

Lemma 2.5 : If (1) $f: D \subseteq R^n \rightarrow R^n$ is a given mapping with $f \in C(\hat{D})$ where $\hat{D} \subseteq D$ is a convex compact set; (2) $(\hat{x} \in \hat{D}) \Rightarrow (f(\hat{x}) \in \hat{D})$, then $\exists x^* \in \hat{D}$ such that $x^* = f(x^*)$.

Proof : A proof of Lemma 2.5 is given by Brouwer in [Brou-12a]. □

Definition 2.2 : The sequence $(\underline{x}^{(k)})$ in $I(R^n)$ is nested if and only if $\underline{x}^{(k+1)} \subseteq \underline{x}^{(k)}$ $(\forall k \geq 0)$. □

Lemma 2.6 : If $(\underline{x}^{(k)})$ is a nested sequence in $I(R^n)$ then $\exists \underline{x}^* \in I(R^n)$ such that $\underline{x}^* \subseteq \underline{x}^{(k)}$ $(\forall k \geq 0)$ and $\underline{x}^{(k)} \rightarrow \underline{x}^*$ $(k \rightarrow \infty)$.

Proof : $\underline{x}^{(k+1)} \subseteq \underline{x}^{(k)} \subseteq \underline{x}^{(0)}$ implies

$$x_{iI}^{(0)} \leq x_{iI}^{(k)} \leq x_{iI}^{(k+1)} \leq x_{iS}^{(k+1)} \leq x_{iS}^{(k)} \leq x_{iS}^{(0)}$$

$(\forall k \geq 0 \wedge i = 1, \ldots, n)$. Therefore, the sequence $(x_{iI}^{(k)})$ $(i = 1, \ldots, n)$ is monotonic increasing and bounded above by $x_{iS}^{(0)}$. Therefore $\exists x_{iI}^* \leq x_{iS}^{(0)}$ such that $x_{iI}^{(k)} \uparrow x_{iI}^*$ $(k \rightarrow \infty)$. Similarly $\exists x_{iS}^* \geq x_{iI}^{(0)}$ such that $x_{iS}^{(k)} \downarrow x_{iS}^*$ $(k \rightarrow \infty)$. If for some i, $x_{iI}^* > x_{iS}^*$ it is easily shown that $\exists \hat{k} \geq 0$ such that $x_{iI}^{(k)} > x_{iS}^{(k)}$ $(\forall k \geq \hat{k})$ which contradicts the hypothesis that $\underline{x}^{(k)} \in I(R^n)$ $(\forall k \geq 0)$. Therefore $x_{iI}^* \leq x_{iS}^*$ $(i = 1, \ldots, n)$ and the result follows. □

Theorem 2.1 : If (1) $f: D \subseteq R^n \to R^n$ is a given mapping with $f \in C^1(\hat{D})$ where $\hat{D} \subseteq D$ is an open convex set; (2) $\underline{f}': I(\hat{D}) \to I(M(R^n))$ is a continuous inclusion monotonic interval extension of $f': \hat{D} \to M(R^n)$; (3) $\hat{\underline{x}} \in I(\hat{D})$ is given and $\hat{A} = \{m(\underline{f}'(\hat{\underline{x}}))\}^{-1}$ exists; (4) $\underline{K}(\hat{x}, \hat{A}, \hat{\underline{x}}) \subset int(\hat{\underline{x}})$, or $\underline{K}(\hat{x}, \hat{A}, \hat{\underline{x}}) \subseteq \hat{\underline{x}}$ and $\left\|\hat{\underline{R}}\right\|_{w(\underline{x})} < 1$ where $\hat{x} = m(\hat{\underline{x}})$,

$$\underline{K}(\hat{x}, \hat{A}, \hat{\underline{x}}) = \hat{x} - \hat{A}f(\hat{x}) + (I - \hat{A}\underline{f}'(\hat{\underline{x}}))(\hat{\underline{x}} - \hat{x}),$$

$$= \hat{x} - \hat{A}f(\hat{x}) + \hat{\underline{R}}(\hat{\underline{x}} - \hat{x}),$$

and

$$\left\|\hat{\underline{R}}\right\|_{w(\underline{x})} = \max_{1 \le i \le n} \{\sum_{j=1}^{n} | \hat{\underline{r}}_{ij} | \, w(\hat{\underline{x}}_j)/w(\hat{\underline{x}}_i)\};$$

(5) the sequence $(x^{(k)})$ is generated from

$$x^{(k+1)} = x^{(k)} - \hat{A}f(x^{(k)}) \ (k \ge 0)$$

with $x^{(0)} \in \hat{\underline{x}}$ arbitrary, then (a) $\exists x^* \in \underline{K}(\hat{x}, \hat{A}, \hat{\underline{x}})$ such that $f(x^*) = 0$ and x^* is unique in $\hat{\underline{x}}$; (b) $x^{(k)} \to x^*$ $(k \to \infty)$.

Proof : A proof of Theorem 2.1 is given by Qi in [Qi—80a]. □

Note 2.1 : Let $A \in M(R^n)$, $\underline{b} \in I(R^n)$ and $n \in N$ be given. The result of applying the Gauss algorithm ([AleH-83a] [AleP-83a]) to the pair $(A, \underline{b})$ is represented by $\underline{g}(A, \underline{b}) \in I(R^n)$. □

Definition 2.3 : Let $f : D \subseteq R^n \to R^n$ be a given mapping with $f \in C^1(\hat{D})$ where $\hat{D} \subset D$ is an open convex set. Let $\underline{f} : I(\hat{D}) \to I(R^n)$ and $\underline{f'} : I(\hat{D}) \to I(M(R^n))$ be inclusion monotonic interval extensions of $f : \hat{D} \to R^n$ and of $f' : \hat{D} \to M(R^n)$ respectively. The Krawczyk operator $K : I(R^n) \times M(R^n) \to I(R^n)$ is defined by

$$\underline{K}(\underline{x}, A) = m(\underline{x}) - Af(m(\underline{x})) + (I - A\underline{f'}(\underline{x}))(\underline{x} - m(\underline{x})) \qquad 2.4$$

where I is the $n \times n$ unit matrix. □

Lemma 2.7 : Let $A \in M(R^n)$ be nonsingular. Then for $\underline{b} \in I(R^n)$ arbitrary

$$A^{-1}\underline{b} \subseteq \underline{g}(A, \underline{b}).$$

Proof : A proof of Lemma 2.7 is given by Alefeld and Platzöder in [AleP-83a]. □

Lemma 2.8 : If (1) $f : D \subseteq R^n \to R^n$ is a given mapping with $f \in C^1(\hat{D})$ where $\hat{D} \subset D$ is an open convex set; (2) $\underline{f} : I(\hat{D}) \to I(R^n)$ and $\underline{f'} : I(\hat{D}) \to I(M(R^n))$ are continuous inclusion monotonic interval extensions of $f : \hat{D} \to R^n$ and of $f' : \hat{D} \to M(R^n)$ respectively; (3) the Krawczyk operator $K : I(R^n) \times M(R^n) \to I(R^n)$ is

defined by 2.4 and the Alefeld and Platzöder operator $\underline{K}_N : I(R^n) \times I(M(R^n)) \times M(R^n) \to I(R^n)$ is defined by

$$\underline{K}_N(\underline{x}, \underline{M}, A) = m(\underline{x}) - \underline{g}(A, \{f(m(\underline{x})) - (A - \underline{M})(\underline{x} - m(\underline{x}))\}) \qquad 2.5$$

then for any nonsingular matrix $A \in M(R^n)$,

$$\underline{K}(\underline{x}, A^{-1}) \subseteq \underline{K}_N(\underline{x}, \underline{f}'(\underline{x}), A). \qquad 2.6$$

Proof: By Lemma 2.7 and by 2.5 with $\underline{M} = \underline{f}'(\underline{x})$,

$$A^{-1}\{f(m(\underline{x})) - (A - \underline{f}'(\underline{x}))(\underline{x} - m(\underline{x}))\} \subseteq \underline{g}(A, \{f(m(\underline{x})) - (A - \underline{f}'(\underline{x}))(\underline{x} - m(\underline{x}))\})$$

$$= m(\underline{x}) - \underline{K}_N(\underline{x}, \underline{f}'(\underline{x}), A). \qquad 2.7$$

Furthermore by 2.4,

$$A^{-1}\{f(m(\underline{x})) - (A - \underline{f}'(\underline{x}))(\underline{x} - m(\underline{x}))\} = A^{-1}f(m(\underline{x})) - A^{-1}(A - \underline{f}'(\underline{x}))(\underline{x} - m(\underline{x}))$$

$$= A^{-1}f(m(\underline{x})) - (I - A^{-1}\underline{f}'(\underline{x}))(\underline{x} - m(\underline{x}))$$

$$= m(\underline{x}) - \underline{K}(\underline{x}, A^{-1}). \qquad 2.8$$

The result 2.6 follows immediately from 2.7 and 2.8. □

Lemma 2.9 ([AleP-83a]) : If (1) $f: D \subseteq R^n \to R^n$ is a given mapping with $f \in C^1(\hat{D})$ where $\hat{D} \subseteq D$ is an open convex set; (2) $\underline{f}: I(\hat{D}) \to I(R^n)$ and $\underline{f}': I(\hat{D}) \to I(M(R^n))$ are continuous inclusion monotonic interval extensions of $f: \hat{D} \to R^n$ and of $f': \hat{D} \to M(R^n)$ respectively; (3) $\underline{x}^{(0)} \in I(\hat{D})$ is such that $w(\underline{x}^{(0)}) > 0$; (4) $B^{(0)} = m(\underline{f}'(\underline{x}^{(0)}))$ is nonsigular; (5) $\exists \alpha \in [0,1)$ such that

$$w(\underline{K}_N(\underline{x}^{(0)}, \underline{f}'(\underline{x}^{(0)}), B^{(0)})) \leq \alpha w(\underline{x}^{(0)}),$$

then $\underline{f}'(\underline{x}^{(0)})$ does not contain any singular point matrix.

Proof : A proof of Lemma 2.9 is given by Alefeld and Platzöder in [AleP-83a]. □

Lemma 2.10 : If (1) $f : D \subseteq R^n \to R^n$ is a given mapping with $f \in C^1(\hat{D})$ where $\hat{D} \subseteq D$ is an open convex set; (2) $\underline{f} : I(\hat{D}) \to I(R^n)$ and $\underline{f}' : I(\hat{D}) \to I(M(R^n))$ are continuous inclusion monotonic interval extensions of $f : \hat{D} \to R^n$ and of $f' : \hat{D} \to M(R^n)$ respectively; (3) $\hat{\underline{x}} \in I(\hat{D})$ and $\hat{Y} \in M(R^n)$ are given, and $\hat{Y}$ is nonsingular; (4) $P : \hat{D} \to R^n$ is defined by

$$P(x) = x - \hat{Y} f(x)$$

then

$$(x \in \underline{\hat{x}}) \Rightarrow (P(x) \in \underline{K}(\underline{\hat{x}}, \hat{Y})).$$

Proof : Let $\hat{x} = m(\underline{\hat{x}})$. Then by Lemma 2.3 and Equation 2.4,

$$\begin{aligned} P(x) &= x - \hat{Y} f(x) \\ &= \hat{x} - \hat{Y} f(\hat{x}) + x - \hat{x} - \hat{Y}(f(x) - f(\hat{x})) \\ &\in \hat{x} - \hat{Y} f(\hat{x}) + x - \hat{x} - \hat{Y} \underline{f'}(\underline{\hat{x}})(x - \hat{x}) \\ &\subseteq \hat{x} - \hat{Y} f(\hat{x}) + (I - \hat{Y} \underline{f'}(\underline{\hat{x}}))(\underline{\hat{x}} - \hat{x}) \\ &= \underline{K}(\underline{\hat{x}}, \hat{Y}). \end{aligned}$$

□

CHAPTER 3

Hansen's Global Optimization Algorithm

This chapter contains an implementation of Hansen's algorithm [Han–80a] for bounding the global minimizer(s) of $f: R^n \rightarrow R^1$ in a box $\underline{\hat{x}} \in I(D)$, where $D \subseteq R^n$ is an open convex set and $f \in C^2(D)$. The problem of bounding the global minimizer(s) of f in $\underline{\hat{x}}$ defined by 1.1 is referred to in this thesis as *Problem P*.

Hansen's algorithm may be divided into 10 steps, and these are described in §3.1 – §3.10. The steps of Hansen's algorithm are described by using a pseudo-code which is explained in Appendix B.

3.1 Step 1. The Stopping Criterion

Three queues L_i $(i = 1, 2, 3)$ are used in Hansen's algorithm. The head and tail of queue L_i are denoted by h_i and t_i respectively.

The queue L_1, which initially is empty, ultimately contains boxes $\underline{x} \in I(R^n)$ for which $\|w(\underline{x})\| \leq \varepsilon_2$, where ε_2 is given, and which might contain the global minimizer(s) of f.

The queue L_2, which initially contains the box $\underline{\hat{x}} \in I(R^n)$ in which it is required to bound the global minimizer(s) of f, will contain boxes $\underline{x} \in I(R^n)$ for which

$\|w(\underline{x})\| > \varepsilon_2$ and which are yet to be processed by the algorithm.

The queue L_3, which initially is empty, will contain boxes $\underline{x} \in I(R^n)$ which have been completely reduced; a box $\underline{x} = (\underline{x}_i)_{n\times 1}$ is completely reduced if and only if $w(\underline{x}_i) = 0$ $(i = 1, \ldots, n)$. The boxes in L_3 might contain the global minimizer(s) of f.

Since a minimizer of f in $\hat{\underline{x}}$ might lie in a face of $\hat{\underline{x}}$, we must be able to retain boundary points of $\hat{\underline{x}}$ if necessary. We store the boundary points of $\hat{\underline{x}}$ in an $n \times 2$ interval matrix $\underline{B} = (\underline{b}_{ij})_{n\times 2}$, with

$$\underline{b}_{i1} = [\hat{x}_{iI}, \hat{x}_{iI}] \quad \text{and} \quad \underline{b}_{i2} = [\hat{x}_{iS}, \hat{x}_{iS}] \quad (i = 1, \ldots, n). \tag{3.1}$$

An upper bound $\bar{f}$ for f^*, the global minimum of f in $\hat{\underline{x}}$, is determined in Hansen's algorithm and is updated continually. This bound is computed as a degenerate interval $\underline{\bar{f}} \in I(R)$, and $f^* \leq \bar{f}_S$. Initially, $\bar{f}$ is computed from

$$\underline{\bar{f}} = \underline{f}(m(\hat{\underline{x}})). \tag{3.2}$$

$\underline{\bar{f}}$ is also used in order to write the boxes in L_3 which might contain the global minimizer(s) of f as follows.

procedure write.L.3$(L_3 \in Q, h_3, t_3 \in N, \underline{\bar{f}} \in I(R))$

! This procedure writes the boxes $\underline{x}^{(j)}$ $(j = h_3, \ldots, t_3)$ such that $f_S^{(j)} \leq \bar{f}_S$ where

! $\underline{f}^{(j)} = \underline{f}(\underline{x}^{(j)})$,

! from the queue L_3.

! On entry, h_3 and t_3 are the head and tail of L_3 respectively.

1. *for* $j = h_3$ *to* t_3 *do*

   1.1. $L_3 \longrightarrow \underline{x}^{(j)}$! *Extract* $\underline{x}^{(j)}$ *from* L_3.

   1.2. $\underline{f} := \underline{f}(\underline{x}^{(j)})$

   1.3. *if* $fs \leq \bar{f}s$ *do*

      1.3.1. *write* $\underline{x}^{(j)}$

2. *return* □

We sometimes extract a box at the ith position from the queue L_1 to be processed where $h_1 \leq i \leq t_1$. Therefore the ith position in L_1 is empty. Therefore the boxes in L_1 need to be rearranged so that all the positions from the (h_1+1)th position to the t_1th position are filled. The procedure *close.up* which implements this arrangement is as follows.

procedure $close.up(i, h_1, t_1 \in N\ ;\ L_1 \in Q\ :\ \underline{x}^{(i)} \in I(R^n))$

! This procedure rearranges the boxes in L_1 such that all the positions from the

! (h_1+1)th position to the t_1th position are filled after extracting one of the

! boxes in L_1.

! On entry, i is the position number between h_1 and t_1.

! On return, $\underline{x}^{(i)}$ is the box which is extracted from L_1.

1. *case* *true* *of*

   $h_1 < i$ *and* $i \leq t_1$:

   1.1. $L_1 \longrightarrow \underline{x}^{(i)}$! *Extract* $\underline{x}^{(i)}$ *from* L_1.

   1.2. *for* $j = i$ *to* $h_1 + 1$ *by* -1 *do*

      1.2.1. $L_{1j} := L_{1j-1}$! L_{1j} *is in the* j*th position in* L_1.

   $h_1 = i$ *and* $i \leq t_1$:

   1.3. $L_1 \longrightarrow \underline{x}^{(i)}$! *Extract* $\underline{x}^{(i)}$ *from* L_1.

   default :

   1.4. *write* "*Error in close.up.*"

   1.5. *stop*

2. *return* □

The preceding notation and ideas are used in the procedure *terminate*, which is an implementation of the stopping criterion which is used in Hansen's algorithm.

procedure $terminate(\underline{f} \in I(R), \varepsilon_0, \varepsilon_1 \in R, L_3 \in Q, h_3, t_1, t_2, t_3, n \in N$; L_1,

$L_2 \in Q, h_1, h_2 \in N$: $\underline{x} \in I(R^n), \hat{w} \in R, f.too.wide \in B)$

! This procedure implements the stopping criterion for Hansen's algorithm.

! On entry, $\underline{f}$, L_3, h_3, t_1, t_2, t_3 and n are as already explained, $\varepsilon_0 > 0$, and $\varepsilon_1 \geq 0$.

! The input-output parameters L_i, and h_i $(i = 1, 2)$ are as already explained.

! On return, $\underline{x}$ is a box which is assumed to contain x^*, the real number $\hat{w}$ is the

! width of the box $\underline{x}$, and the Boolean *f.too.wide* has the value *true* if for at least

! one i $(h_1 \leq i \leq t_1)$, $w(\underline{f}(\underline{x}^{(i)})) > \varepsilon_0$, where $\underline{x}^{(i)} (i = h_1, \ldots, t_1)$ are the boxes in the

! queue L_1 which might contain the global minimizer(s) of f.

1. $\underline{x} := (\underline{0})_{n \times 1}$

2. $\hat{w} := 0$

3. *f.too.wide* := *false*

4. *if* $L_2 = \emptyset$

   then

   ! *No boxes remain to be processed.*

   4.1. *if* $L_1 = \emptyset$

      then

      ! *There is no box* $\underline{x}$ *with* $\|w(\underline{x})\| \leq \varepsilon_2$

      ! *and which contains a minimizer of* f.

      4.1.1. *if* $\varepsilon_1 > 0$

         then

! *The minimum value of* f, *rather than the*

! *minimizer, is to be bounded.*

4.1.1.1. $\tilde{f} := \bar{f}_S - \varepsilon_1$

4.1.1.2. *write* $[\tilde{f}, \bar{f}_S]$

4.1.1.3. $write.L.3(L_3, h_3, t_3, \underline{f})$! §3.1.

4.1.1.4. *stop*

else

! $\varepsilon_1 = 0$ *and* $L_1 = \emptyset$.

! *We need to check the program because if this*

! *is the case then we have lost the global minimizer.*

4.1.1.5. *write* "*Error in terminate.*"

4.1.1.6. *stop*

else

! $L_1 \neq \emptyset$. *There is at least one box* $\underline{x}$ *with* $\|w(\underline{x})\| \leq \varepsilon_2$

! *which might contain a minimizer of* f.

4.1.2. *if* $\varepsilon_1 > 0$

then

! *We need to process all the boxes in* L_1 *since we wish to*

! *bound the minimum value of* f *only.*

4.1.2.1. $L_1 \longrightarrow \underline{x}$! *Extract* $\underline{x}$ *from* L_1.

4.1.2.2. $h_1 := h_1 + 1$

4.1.2.3. $\hat{w} := \|w(\underline{x})\|$

else

! $\varepsilon_1 = 0$ *and* $L_1 \neq \emptyset$. *We need to check whether or not*

! $w(f(\underline{x}^{(i)})) \leq \varepsilon_0$ *for* $\underline{x}^{(i)} \in L_1$ $(i = h_1, \ldots, t_1)$.

! *If there exists at least one* i *such that* $w(f(\underline{x}^{(i)})) > \varepsilon_0$

! *then we process the box* $\underline{x}^{(i)}$. *If there is no*

! *such* i *then we write all the boxes from* L_1, *and stop.*

4.1.2.4. *for* $i = h_1$ *to* t_1 *do*

4.1.2.4.1. $L_1 \longrightarrow \underline{x}^{(i)}$

4.1.2.4.2. $\underline{f}^{(i)} := \underline{f}(\underline{x}^{(i)})$

4.1.2.5. $j := h_1$

4.1.2.6. *repeat*

4.1.2.6.1. $f.too.wide := w(\underline{f}^{(j)}) > \varepsilon_0$

while $\sim f.too.wide$ *and* $j < t_1$ *do*

4.1.2.6.2. $j := j + 1$

4.1.2.7. *if* $\sim f.too.wide$ *do*

! L_1 *contains only boxes* $\underline{x}^{(i)}$ $(i = h_1, \ldots, t_1)$

! *with* $w(\underline{f}(\underline{x}^{(i)})) \leq \varepsilon_0$, *and* $\left\|w(\underline{x}^{(i)})\right\| \leq \varepsilon_2$.

4.1.2.7.1. *for* $i = h_1$ *to* t_1 *do*

4.1.2.7.1.1. $L_1 \longrightarrow \underline{x}^{(i)}$

4.1.2.7.1.2. *write* $\underline{x}^{(i)}$

4.1.2.7.2. $\tilde{f} := \min_{h_1 \le i \le t_1} \{f_I^{(i)}\}$

4.1.2.7.3. *write* $[\tilde{f}, \bar{f}_S]$

4.1.2.7.4. $write.L.3(L_3, h_3, t_3, \underline{f})$! §3.1.

4.1.2.7.5. *stop*

! *Extract, from* L_1, *the* $\underline{x}^{(j)}$ *for which* $w(\underline{f}(\underline{x}^{(j)})) > \varepsilon_0$.

4.1.2.8. $close.up(j, h_1, t_1 \ ; \ L_1 \ : \ \underline{x}^{(j)})$! §3.1.

4.1.2.9. $\underline{x} := \underline{x}^{(j)}$

4.1.2.10. $h_1 := h_1 + 1$

4.1.2.11. $\hat{w} := \|w(\underline{x})\|$

else

! *Extract* $\underline{x}$ *from* L_2.

4.2. $L_2 \longrightarrow \underline{x}$

4.3. $h_2 := h_2 + 1$

4.4. $\hat{w} := \|w(\underline{x})\|$

5. *return* □

3.2 Step 2. Monotonicity Test

Let $\underline{x} \in I(R^n)$ be a current sub-box of $\hat{\underline{x}}$ resulting from Step 1. We shall use the monotonicity property of the function f to delete the whole box $\underline{x}$ or to replace the box $\underline{x}$ either with a degenerate box (i.e a box with at least one of its component a degenerate interval) or with a point box (i.e a box containing just one point in R^n or a box which has been completely reduced (§3.1)).

If $f : R^n \to R^1$ has gradient $g : R^n \to R^n$ and $g_i(x) \geq 0$ $(\forall x \in \underline{x} \in I(R^n))$ for some $i \in \{1, \ldots, n\}$ then

$$f(x_1, \ldots, x_{i-1}, x_{iI}, x_{i+1}, \ldots, x_n) \leq f(x) \quad (\forall x \in \underline{x}). \tag{3.3}$$

Therefore if for some i $(1 \leq i \leq n)$, $(\underline{g}_i(\underline{x}))_I \geq 0$, where $\underline{g} : I(R^n) \to I(R^n)$ is a continuous inclusion monotonic interval extension of $g : R^n \to R^n$, then we can delete all of $\underline{x}$ save those points in $\underline{x}$ with $x_i = x_{iI}$; that is, we replace $\underline{x}$ with $(\underline{x}_1, \ldots, \underline{x}_{i-1}, [x_{iI}, x_{iI}], \underline{x}_{i+1}, \ldots, \underline{x}_n)^T$. If $(\underline{g}_i(\underline{x}))_I \geq 0$ $(i = 1, \ldots, n)$ then we may replace $\underline{x}$ with the point box $([x_{iI}, x_{iI}])_{n \times 1}$. If $(\underline{g}_i(\underline{x}))_I > 0$, for some $i \in \{1, \ldots, n\}$ then we may delete the whole of $\underline{x}$ unless the boundary plane $x_i = x_{iI}$ contains boundary points of the initial box $\hat{\underline{x}}$. However, if $(\underline{g}_i(\underline{x}))_I > 0$ and the global minimizer is in the interior of the initial box $\hat{\underline{x}}$ then we can delete the whole box $\underline{x}$. Similar statements are valid if $(\underline{g}_i(\underline{x}))_S \leq 0$ or if $(\underline{g}_i(\underline{x}))_S < 0$.

These ideas are embodied in the procedure *monotonicity.test* which follows.

procedure monotonicity.test($\underline{B} \in I(M(R^n, R^2)), n \in N, ignore.boundary \in B$;

$$L_3 \in Q, \underline{x} \in I(R^n), \underline{f} \in I(R), t_3 \in N : delete.x \in B)$$

! This procedure implements step 2 of Hansen's global optimization algorithm.

! On entry, $\underline{B}$ and n are as in §3.1; the Boolean *ignore.boundary* has the value *true*

! if the global minimizer is in the interior of the initial box $\hat{\underline{x}}$ and it has the value

! *false* otherwise.

! The input-output parameters L_3, $\underline{x}$, $\underline{\bar{f}}$ and t_3 are as in §3.1.

! On return, delete.x = *true* if $\underline{x}$ can be deleted or if $\underline{x}$ is completely reduced;

! otherwise delete.x = *false*.

1. *delete.x* := *false*

2. $\underline{g} := \underline{g}(\underline{x})$

3. $k := 0$

! *k is the number of co – ordinate directions i along which* $\underline{x}_i$ *is reduced*

! *to a degenerate interval.*

4. $i := 1$

5. *if ignore.boundary*

   then

   ! *The minimizer of f lies in the interior of* $\hat{\underline{x}}$.

   5.1. *repeat*

      5.1.1. *if* $0 < g_{iI}$ *or* $g_{iS} < 0$ *do*

         5.1.1.1. *delete.x* := *true*

      while ~ *delete.x and* $i < n$ *do*

      5.1.2. $i := i + 1$

   else

! *The minimizer of f might lie on the boundary*

! *of* $\hat{\underline{x}}$ *so boundary points must not be deleted.*

5.2. *repeat*

5.2.1. *case true of*

$0 < g_{iI}$: ! $0 < (\underline{g}_i(\underline{x}))_I$.

5.2.1.1. *if* $x_{iI} \notin \underline{b}_{i1}$

then

! x_{iI} *is not in the boundary of* $\hat{\underline{x}}$ *so the whole of* $\underline{x}$ *may be*

! *deleted because if* x^* *is the minimizer of f then*

! $g(x^*) = 0$.

5.2.1.1.1. *delete.x* := *true*

else

! *All points in the* x_i *– direction save the lower*

! *boundary point may be deleted.*

5.2.1.1.2. $\underline{x}_i := \underline{b}_{i1}$

5.2.1.1.3. $k := k + 1$

$0 \leq g_{iI}$: ! $0 \leq (\underline{g}_i(\underline{x}))_I$.

! *The minimizer of f could lie in the face* $x_i = x_{iI}$ *of* $\underline{x}$.

5.2.1.2. $\underline{x}_i := [x_{iI}, x_{iI}]$

5.2.1.3. $k := k + 1$

$g_{iS} < 0$: ! $(\underline{g}_i(\underline{x}))_S < 0$.

5.2.1.4. *if* $x_{iS} \notin \underline{b}_{i2}$

then

! x_{iS} *is not in the boundary of* $\hat{\underline{x}}$ *so the whole of* $\underline{x}$ *may be*

! *deleted because if* $\underline{x}^*$ *is the minimizer of* f *then*

! $g(x^*) = 0$.

5.2.1.4.1. $delete.x := \underline{true}$

else

! *All points in the* x_i *– direction save the*

! *upper boundary point may be deleted.*

5.2.1.4.2. $\underline{x}_i := \underline{b}_{i2}$

5.2.1.4.3. $k := k + 1$

$g_{iS} \leq 0$: ! $(\underline{g}_i(\underline{x}))_S \leq 0$.

! *The minimizer of* f *could lie in the face* $x_i = x_{iS}$ *of* $\underline{x}$.

5.2.1.5. $\underline{x}_i := [x_{iS}, x_{iS}]$

5.2.1.6. $k := k + 1$

default :

5.2.1.7. { }

while $\sim delete.x$ *and* $i < n$ *do*

5.2.2. $i := i + 1$

5.3. *if* $\sim delete.x$ *and* $k = n$ *do*

! *If* $k = n$ *then* $\underline{x}$ *is completely reduced to a point box*

! *which is either deleted or is inserted into* L_3.

5.3.1. *delete.*$x := \underline{true}$

5.3.2. $\underline{f} := \underline{f}(\underline{x})$

5.3.3. *if* $f_I \leq \bar{f}_S$ *do*

! *Insert* $\underline{x}$ *into* L_3.

5.3.3.1. *if* $f_S \leq \bar{f}_I$ *do*

! *update* $\bar{f}$.

5.3.3.1.1. $\bar{f} := [f_S, f_S]$

5.3.3.2. $\underline{x} \longrightarrow L_3$

5.3.3.3. $t_3 := t_3 + 1$

6. *return* □

3.3 Step 3. Non-Convexity Test

Let $\underline{x} \in I(R^n)$ be a sub-box of $\hat{\underline{x}}$. Suppose that $\underline{G}(\underline{x}, x) \in I(M(R^n))$ is defined by

$$\underline{G}_{ij}(\underline{x}, x) = \begin{cases} \partial_i \partial_i \underline{f}(\underline{x}_1, \ldots, \underline{x}_{i-1}, \underline{x}_i, x_{i+1}, \ldots, x_n) & (i = j (i = 1, \ldots, n)) \\ 2\partial_j \partial_i \underline{f}(\underline{x}_1, \ldots, \underline{x}_{j-1}, \underline{x}_j, x_{j+1}, \ldots, x_n) & (j < i (i = 1, \ldots, n; \\ & \quad j = 1, \ldots, i-1)) \\ 0 & (\text{otherwise}) \end{cases} \tag{3.4}$$

where $\partial_i \partial_j \underline{f} : I(R^n) \rightarrow I(R^1)$ is an inclusion monotonic interval extension of $\partial_i \partial_j f : R^n \rightarrow R^1$ $(1 \leq i, j \leq n)$.

In the non-convexity test which is described in this section we need to examine the value of $\underline{G}_{ii}(\underline{x}, \underline{x})$. By 3.4, all the diagonal elements of $\underline{G}(\underline{x}, x)$ have arguments different from $(\underline{x}_1, \ldots, \underline{x}_n)$ save the element in position (n, n). Therefore in our implementation we need a separate procedure to compute $\underline{G}_{ii}(\underline{x}, \underline{x})$ $(i = 1, \ldots, n)$.

We evaluate $\underline{G}_{ii} = \underline{G}_{ii}(\underline{x}, \underline{x})$ $(i = 1, \ldots, n)$. If for some i, $(i = 1, \ldots, n)$, $G_{iiS} < 0$ then there is no point in $\underline{x}$ for which the Hessian of f is positive semi-definite. Therefore f is not convex in any subset of $\underline{x}$, and so f has no minimizer in the interior of $\underline{x}$. Therefore we can delete all of $\underline{x}$ save the boundary points of $\hat{\underline{x}}$ which lie in $\underline{x}$.

Suppose that $\underline{x}'$ is the smallest box containing the boundary points of $\hat{\underline{x}}$ which lie in $\underline{x}$ (See Configuration 2.1.). In our implementation, either $\underline{x}' = \underline{x}$, $\underline{x}'$ is a degenerate box of dimension less than that of $\underline{x}$, or $\underline{x}' = \emptyset$. Therefore, before we test the non-convexity of f, we need to examine the box $\underline{x}'$ as follows.

If $\underline{x}' = \underline{x}$ then the non-convexity test need not be used because, whether or not f is convex in $\underline{x}$, we cannot reduce $\underline{x}$ further without losing boundary points of $\hat{\underline{x}}$. If $\underline{x}' = \emptyset$ then we delete all of $\underline{x}$ when f is not convex in any subset of $\underline{x}$. If $\underline{x}'$ is a degenerate box then we replace $\underline{x}$ with $\underline{x}'$ when f is not convex in any subset of $\underline{x}$.

These ideas are embodied in the procedure *non.convexity.test* which follows.

procedure non.convexity.test($\underline{B} \in I(M(R^n, R^2)), n \in N, \varepsilon_2 \in R,$

ignore.boundary $\in B$; $L_1, L_2 \in Q, \underline{x} \in I(R^n)$,

$t_1, t_2 \in N \ : \ delete.x \in B)$

! This procedure implements step 3 of Hansen's global optimization algorithm.

! On entry, $\underline{B}$, n, and ε_2 are as in §3.1, and *ignore.boundary* is as in §3.2.

! The input-output parameters L_1, L_2, t_1 and t_2 are as in §3.1 and $\underline{x}$ is the current

! box.

! On return, $delete.x = \underline{true}$ if f is not convex in any subset of $\underline{x}$, and $\underline{x}$ and $\underline{B}$ do

! not share any common point.

1. $delete.x := \underline{false}$
2. $x.is.x.prime := \underline{true}$
3. $x.prime.empty := \underline{true}$
4. $f.not.convex := \underline{false}$
5. *if ignore.boundary*

   then

   ! *The minimizer of f is in the interior of* $\underline{\hat{x}}$.

   5.1. $i := 1$

   5.2. *repeat*

      5.2.1. $f.not.convex := (\underline{G_{ii}}(\underline{x}, \underline{x}))_S < 0$

   while $\sim f.not.convex$ *and* $i < n$ *do*

      5.2.2. $i := i + 1$

5.3. *if* $f.not.convex$ *do*

5.3.1. $delete.x :=$ *true*

else

! *The minimizer of* f *might be on the boundary of* $\underline{\hat{x}}$.

5.4. $j := 0$

5.5. $k := 0$

5.6. $l := 0$

! j *is the number of faces shared by* $\underline{x}$ *and* $\underline{\hat{x}}$, *and* k *is the co – ordinate*

! *direction in which* $\underline{\hat{x}}$ *and* $\underline{x}$ *share a face.* $l \in \{1, 2\}$ *where* $l = 1$ *and* $l = 2$

! *denote the lower boundary point and the upper boundary point corresponding*

! *to* $\underline{b}_{k1}$ *and* $\underline{b}_{k2}$ *respectively, where* $\underline{b}_{k1}$ *and* $\underline{b}_{k2}$ *are defined by* 3.1.

5.7. *for* $i = 1$ *to* n *do*

! *We calculate* j, *the number of faces shared by* $\underline{x}$ *and* $\underline{\hat{x}}$.

! *If* $j = 1$ *then* k, *the co – ordinate direction in which* $\underline{x}$ *and* $\underline{\hat{x}}$

! *share a face, is needed.*

5.7.1. *if* $x_{iI} \in \underline{b}_{i1}$ *do*

5.7.1.1. $j := j + 1$

5.7.1.2. $k := i$

5.7.2. *if* $x_{iS} \in \underline{b}_{i2}$ *do*

5.7.2.1. $j := j + 1$

5.7.2.2. $k := i$

5.8. *case true of*

$j > 1$: ! $x.is.x.prime = \underline{true}$.

5.8.1. $x.prime.empty :=$ *false*

$j = 1$:

5.8.2. $x.is.x.prime :=$ *false*

5.8.3. $x.prime.empty :=$ *false*

default : ! $j = 0$, *and* $x.prime.empty = \underline{true}$.

5.8.4. $x.is.x.prime :=$ *false*

5.9. *if* $\sim x.is.x.prime$ *do*

! *Here* $j \leq 1$ *and* k *is the co – ordinate direction*

! *in which* $\underline{x}$ *and* $\hat{\underline{x}}$ *share a face.*

5.9.1. $i := 1$

5.9.2. *repeat*

5.9.2.1. $f.not.convex := (\underline{G}_{ii}(\underline{x}, \underline{x}))_S < 0$

while $\sim f.not.convex$ *and* $i < n$ *do*

5.9.2.2. $i := i + 1$

5.9.3. *if* $f.not.convex$ *do*

5.9.3.1. *if* $\sim x.prime.empty$ *do*

! *If* $x.prime.empty = \underline{true}$ *then we delete*

! $\underline{x}$; *else we replace* $\underline{x}_k$ *with* $\underline{b}_{kl}$ $(l \in \{1, 2\})$ *and*

! *insert* $\underline{x}$ *into* L_1 *or* L_2, *whichever is appropriate.*

5.9.3.1.1. *if* $x_{kI} \in \underline{b}_{k1}$

then ! x_{kI} *lies in the face* $\underline{b}_{k1}$ *of* $\underline{\hat{x}}$.

5.9.3.1.1.1. $l := 1$

else ! x_{kS} *lies in the face* $\underline{b}_{k2}$ *of* $\underline{\hat{x}}$.

5.9.3.1.1.2. $l := 2$

5.9.3.1.2. $\underline{x}_k := \underline{b}_{kl}$

5.9.3.1.3. *if* $\|w(\underline{x})\| \leq \varepsilon_2$

then

5.9.3.1.3.1. $\underline{x} \longrightarrow L_1$

5.9.3.1.3.2. $t_1 := t_1 + 1$

else

5.9.3.1.3.3. $\underline{x} \longrightarrow L_2$

5.9.3.1.3.4. $t_2 := t_2 + 1$

5.9.3.2. $delete.x :=$ *true*

6. *return* □

3.4 Step 4. Sub-box Deletion Using Function Values (1)

Let $\underline{x} \in I(R^n)$ be a sub-box of $\underline{\hat{x}}$ resulting from Step 1 or from Step 3. In Step 4 we reduce or delete $\underline{x}$ as explained in §7 of [Han–80a].

Suppose that $\underline{J}(\underline{x}) = \underline{J}(\underline{x}, x)$ denotes the Jacobian matrix with elements

$$\underline{J}_{ij}(\underline{x}, x) = \partial_j \partial_i \underline{f}(\underline{x}_1, \ldots, \underline{x}_{j-1}, \underline{x}_j, x_{j+1}, \ldots, x_n) \tag{3.5}$$

$(i, j = 1, \ldots, n)$. We compute $\underline{G}(\underline{x}) = \underline{G}(\underline{x}, x)$, where $\underline{G}(\underline{x}, x)$ is given by 3.4, by using $\underline{J}(\underline{x}, x)$, according to

$$\underline{G}_{ij}(\underline{x}, x) = \begin{cases} 0 & (i < j \ (j = 1, \ldots, n;\ i = 1, \ldots, j-1)) \\ \underline{J}_{ii}(\underline{x}, x) & (i = j \ (i, j = 1, \ldots, n)). \\ 2\underline{J}_{ij}(\underline{x}, x) & (i > j \ (i = 1, \ldots, n;\ j = 1, \ldots, i-1)) \end{cases}$$

According to §7 of [Han–80a], $\underline{x}$ can be reduced in one dimension, say the k-th, at a time by deleting all points y from $\underline{x}$ such that $f(y) > \bar{f}_S - \varepsilon_1$, and this can be done by solving the quadratic relation

$$\underline{a}_k + \underline{b}_k \tilde{y}_k + \underline{c}_k \tilde{y}_k^2 \leq 0 \tag{3.6}$$

where

$$\underline{a}_k = \sum_{\substack{j=1 \\ j \neq k}}^{n} g_j(x) \tilde{\underline{x}}_j + \frac{1}{2} \sum_{\substack{j=1 \\ j \neq k}}^{n} \sum_{\substack{i=j \\ i \neq k}}^{n} \underline{G}_{ij}(\underline{x}) \tilde{\underline{x}}_i \tilde{\underline{x}}_j - \underline{E}, \tag{3.7}$$

$$\underline{b}_k = g_k(x) + \frac{1}{2} \sum_{j=1}^{k-1} \underline{G}_{kj}(\underline{x}) \tilde{\underline{x}}_j + \frac{1}{2} \sum_{j=k+1}^{n} \underline{G}_{jk}(\underline{x}) \tilde{\underline{x}}_j, \tag{3.8}$$

$$\underline{c}_k = \frac{1}{2} \underline{G}_{kk}(\underline{x}), \tag{3.9}$$

$\tilde{y} = y - x \quad (y,\ x \in \underline{x})$, $\tilde{\underline{x}}_i = \underline{x}_i - x_i \quad (i = 1, \ldots, n)$ and $\underline{E} = \underline{\bar{f}} - \underline{f}(x) - \varepsilon_1$. If $\varepsilon_1 > 0$ then we do not need to know the point(s) x^* at which f is globally minimum; otherwise we know x^* within the tolerance ε_2.

According to §7 of [Han–80a], by using 3.6, $\underline{x}_k$ can be reduced to two intervals, to one interval, or to no interval. In Step 4 of Hansen's algorithm, the case in which $0 \leq G_{kkI}$, $k \in \{1, \ldots, n\}$ is considered. The case in which $G_{kkI} < 0$, $k \in \{1, \ldots, n\}$ is considered in Step 6.

In Step 4, for those values of $k \in \{1, \ldots, n\}$, for which $0 \leq G_{kkI}$ we use 3.6 to determine $\underline{y}^{(1)}$ and $\underline{y}^{(2)}$. Now in all cases save one (See page 256 (7.10) of [Han–80a].) only one of $\underline{y}^{(1)}$ and $\underline{y}^{(2)}$ will be non-empty. If only one of $\underline{y}^{(1)}$ and $\underline{y}^{(2)}$, say $\underline{y}^{(1)}$, is non-empty then we replace $\underline{x}_k$ with $\underline{y}^{(1)}$. If both $\underline{y}^{(1)}$ and $\underline{y}^{(2)}$ are non-empty then we leave $\underline{x}_k$ unchanged because Hansen says nothing about what to do for this case.

These ideas are embodied in the procedure *solve.quadratic*.1 which follows.

procedure *solve.quadratic*.1($\underline{f} \in I(R), \varepsilon_1 \in R, n \in N$; $\underline{x} \in I(R^n)$:

$\underline{J}, \underline{G} \in I(M(R^n)), delete.x \in B$)

! This procedure implements step 4 of Hansen's global optimization algorithm.

! On entry, $\underline{f}$, ε_1 *and* n are as in §3.1 *and* §3.2.

! On return, $\underline{J}$, *and* $\underline{G}$ are the Jacobian and Hessian respectively computed over the

! current box $\underline{x}$ and $delete.x = \underline{true}$ if $\underline{y}^{(1)} = \emptyset$ and $\underline{y}^{(2)} = \emptyset$; $delete.x = \underline{false}$

! otherwise.

1. $\tilde{\underline{x}} := \underline{x} - m(\underline{x})$

2. $delete.x := \underline{false}$

3. $\underline{J} := \underline{J}(\underline{x})$

4. $\underline{for}\ i = 1\ \underline{to}\ n\ \underline{do}$

   4.1. $\underline{G}_{ii} := \underline{J}_{ii}$

   4.2. $\underline{for}\ j = 1\ \underline{to}\ i-1\ \underline{do}$

      4.2.1. $\underline{G}_{ij} := 2\underline{J}_{ij}$

   4.3. $\underline{for}\ j = i+1\ \underline{to}\ n\ \underline{do}$

      4.3.1. $\underline{G}_{ij} := [0, 0]$

5. $\underline{g} := \underline{g}(m(\underline{x}))$! *g is the gradient of f* (§3.2).

6. $\underline{f} := \underline{f}(m(\underline{x}))$

7. $\underline{E} := \overline{f} - \underline{f} - [\epsilon_1, \epsilon_1]$

8. $k := 1$

9. *repeat*

   9.1. $\underline{if}\ \underline{G}_{kkI} \geq 0\ \underline{do}$

      9.1.1. $\underline{a}_k := \sum_{\substack{j=1 \\ j\neq k}}^{n} \underline{g}_j \tilde{\underline{x}}_j + \frac{1}{2} \sum_{\substack{j=1 \\ j\neq k}}^{n} \sum_{\substack{i=j \\ i\neq k}}^{n} \underline{G}_{ij} \tilde{\underline{x}}_i \tilde{\underline{x}}_j - \underline{E}$! *equation 3.7*

      9.1.2. $\underline{b}_k := \underline{g}_k + \frac{1}{2} \sum_{j=1}^{k-1} \underline{G}_{kj} \tilde{\underline{x}}_j + \frac{1}{2} \sum_{j=k+1}^{n} \underline{G}_{jk} \tilde{\underline{x}}_j$! *equation 3.8*

      9.1.3. $\underline{c}_k := \frac{1}{2} \underline{G}_{kk}$! *equation 3.9*

      9.1.4. *Solve* $\underline{a}_k + \underline{b}_k \tilde{y}_k + \underline{c}_k \tilde{y}_k^2 \leq 0$ *(relation 3.6) to give* $\underline{y}^{(1)}$ *and* $\underline{y}^{(2)}$, *using the method which has been described by Hansen*[*Han*−−80*a*].

      9.1.5. $\underline{case}\ \underline{true}\ \underline{of}$

$\underline{y}^{(1)} \neq \emptyset$ *and* $\underline{y}^{(2)} = \emptyset$:

9.1.5.1. $\underline{x}_k := \underline{y}^{(1)}$

$\underline{y}^{(1)} = \emptyset$ *and* $\underline{y}^{(2)} \neq \emptyset$:

9.1.5.2. $\underline{x}_k := \underline{y}^{(2)}$

$\underline{y}^{(1)} \neq \emptyset$ *and* $\underline{y}^{(2)} \neq \emptyset$: ! $\underline{x}_k$ *is unchanged.*

9.1.5.3. { }

default : ! $\underline{y}^{(1)} = \emptyset$ *and* $\underline{y}^{(2)} = \emptyset$

9.1.5.4. $delete.x :=$ *true*

while $\sim$ *delete.x and* $k < n$ *do*

9.2. $k := k + 1$

10. *return* □

3.5 Step 5. Sub-box Deletion Using Interval Newton Methods (1)

Let $\underline{x} \in I(R^n)$ be a sub-box of $\hat{\underline{x}}$ resulting from Step 4. In Step 5 we use the interval Newton Methods which are described in [Han–80a] and [HanS-81a] to reduce or to delete $\underline{x}$. The interval matrix $\underline{J} = \underline{J}(\underline{x})$ has already been computed in Step 4 in order to compute $\underline{G} = \underline{G}(\underline{x})$. So we can determine $B = \{m(\underline{J})\}^{-1}$ if it exists. Hansen says nothing about what to do if $m(\underline{J})$ is singular.

According to [Han–80a] and [HanS-81a] $\underline{x}$ can be reduced in one dimension, say the k-th, at a time by computing $\underline{x}'_k$ from

$$\underline{y}_k = x_k - (\underline{a}_{kk})^{-1}\{b_k + \sum_{j=1}^{k-1} \underline{a}_{kj}(\underline{x}_j - x_j) + \sum_{j=k+1}^{n} \underline{a}_{kj}(\underline{x}_j - x_j)\} \qquad 3.10$$

$$\underline{x}'_k = \underline{y}_k \cap \underline{x}_k \quad k \in \{1, \ldots, n\} \qquad 3.11$$

where $\underline{A} = (\underline{a}_{ij})_{n\times n} = B\underline{J}$, $b = Bf'(m(\underline{x}))^T$, and $x_k = m(\underline{x}_k)$ $(k = 1, \ldots, n)$.

According to [Han–80a] and [HanS-81a] if $0 \notin \underline{a}_{kk}$ then $\underline{y}_k$ consists of one interval and hence $\underline{x}'_k$ consists of at most one interval provided that $\underline{x}'_k \neq \emptyset$, but if $0 \in \underline{a}_{kk}$ then $\underline{y}_k$ consists of at most two intervals. In step 5 of Hansen's algorithm, the case in which $0 \notin \underline{a}_{kk}$, $k \in \{1, \ldots, n\}$ is considered. The case in which $0 \in \underline{a}_{kk}$, $k \in \{1, \ldots, n\}$ is considered in step 7.

In Step 5, for those values of $k \in \{1, \ldots, n\}$ such that $0 \notin \underline{a}_{kk}$ we use 3.10 and 3.11 to determine a replacement for $\underline{x}_k$. If a reduction of $\underline{x}_k$ will delete boundary points of $\hat{\underline{x}}$ from the box $\underline{x}$ then we retain $\underline{x}$ unchanged. However, if we know that the global minimizer $x^* \in int(\hat{\underline{x}})$ then we replace $\underline{x}_k$ with $\underline{x}'_k$, because we know that x^* cannot be a boundary point of $\hat{\underline{x}}$.

In order to describe the procedure *newton.method*.1 which implements Step 5 we need the following procedure.

procedure $newton(\underline{B} \in I(M(R^n, R^2)), \underline{A} \in I(M(R^n)), B \in M(R^n), \underline{x} \in I(R^n)$,

$j, k, n \in N, ignore.boundary \in B : \underline{x}'^{(1)}, \underline{x}'^{(2)} \in I(R),$

$delete.x \in B)$

! This procedure determines $\underline{x}'_k$ where $\underline{x}'_k$ is given by 3.11 and consists of either

! two intervals, one interval, or no interval.

! On entry, $\underline{B}$ is as in §3.1, $\underline{A} = B\underline{J}$ where $B = \{m(\underline{J})\}^{-1}$, j is the number of faces

! which are common to $\underline{x}$ and $\hat{\underline{x}}$, k is such that $\underline{x}_k$ is to be reduced, n is the number

! of components in $\underline{x}$, the Boolean *ignore.boundary* is as in §3.2 and $\underline{x}$ is the

! current box.

! On return, $\underline{x}'^{(1)}$ and $\underline{x}'^{(2)}$ are the intervals which are obtained from the interval $\underline{x}_k$

! by using 3.10 and 3.11, and $delete.x = \underline{true}$ if both $\underline{x}'^{(1)}$ and $\underline{x}'^{(2)}$ are empty.

1. $delete.x := \underline{false}$

2. $x := m(\underline{x})$

3. $b := Bf'(x)^T$

4. $\underline{y}_k := x_k - (\underline{a}_{kk})^{-1}\{b_k + \sum_{j=1}^{k-1} \underline{a}_{kj}(\underline{x}_j - x_j) + \sum_{j=k+1}^{n} \underline{a}_{kj}(\underline{x}_j - x_j)\}$

! *If* $0 \notin \underline{a}_{kk}$ *then* $\underline{y}_k$ *is one interval,*

! *else* $\underline{y}_k$ *might consist of two intervals* $\underline{y}_k^{(1)}$ *and* $\underline{y}_k^{(2)}$.

5. $\underline{x}'^{(1)} := \underline{y}_k^{(1)} \cap \underline{x}_k$

6. $\underline{x}'^{(2)} := \underline{y}_k^{(2)} \cap \underline{x}_k$

7. *if ignore.boundary*

then

7.1. *if* $\underline{x}'^{(1)} = \emptyset$ *and* $\underline{x}'^{(2)} = \emptyset$ *do*

7.1.1. *delete.x* := *true*

else

7.2. *case true of*

$j = 0$: ! $\underline{x}$ *and* $\underline{B}$ *are disjoint.*

7.2.1. *if* $\underline{x}'^{(1)} = \emptyset$ *and* $\underline{x}'^{(2)} = \emptyset$ *do*

7.2.1.1. *delete.x* := *true*

$j = 1$: ! $\underline{x}$ *and* $\hat{\underline{x}}$ *share exactly one face.*

7.2.2. *if* $x_{kI} \in \underline{b}_{k1}$

then

! *Only the lower face is common to* $\underline{x}_k$ *and* $\underline{b}_{k1}$.

! *If a reduction of* $\underline{x}_k$ *will delete points in* $\underline{b}_{k1}$

! *from* $\underline{x}_k$ *then we retain* $\underline{x}_k$ *unchanged.*

7.2.2.1. *case true of*

$\underline{x}'^{(1)} \neq \emptyset$ *and* $\underline{x}'^{(2)} \neq \emptyset$:

7.2.2.1.1. *if* $x'^{(1)}_I > b_{k1S}$ *do* ! $\underline{x}_k$ *is unchanged.*

7.2.2.1.1.1. $\underline{x}'^{(1)} := \underline{x}_k$

7.2.2.1.2. *if* $x'^{(2)}_I > b_{k1S}$ *do* ! $\underline{x}_k$ *is unchanged.*

7.2.2.1.2.1. $\underline{x}'^{(2)} := \underline{x}_k$

$\underline{x}'^{(1)} \neq \emptyset$ *and* $\underline{x}'^{(2)} = \emptyset$:

7.2.2.1.3. *if* $x'^{(1)}_I > b_{k1S}$ *do* ! $\underline{x}_k$ *is unchanged.*

7.2.2.1.3.1. $\underline{x}'^{(1)} := \underline{x}_k$

$\underline{x}'^{(1)} = \emptyset$ *and* $\underline{x}'^{(2)} \neq \emptyset$:

7.2.2.1.4. *if* $x'^{(2)}_I > b_{k1S}$ *do* ! $\underline{x}_k$ *is unchanged.*

7.2.2.1.4.1. $\underline{x}'^{(2)} := \underline{x}_k$

default : ! $\underline{x}'^{(1)}$ *and* $\underline{x}'^{(2)}$ *are empty.*

7.2.2.1.5. $\underline{x}'^{(1)} := \underline{x}_k$

else

! *Only the upper face is common to* $\underline{x}_k$ *and* $\underline{b}_{k2}$.

! *If a reduction of* $\underline{x}_k$ *will delete points in* $\underline{b}_{k2}$

! *from* $\underline{x}_k$ *then we retain* $\underline{x}_k$ *unchanged.*

7.2.2.2. *case true of*

$\underline{x}'^{(1)} \neq \emptyset$ *and* $\underline{x}'^{(2)} \neq \emptyset$:

7.2.2.2.1. *if* $x'^{(1)}_S < b_{k2I}$ *do* ! $\underline{x}_k$ *is unchanged.*

7.2.2.2.1.1. $\underline{x}'^{(1)} := \underline{x}_k$

7.2.2.2.2. *if* $x'^{(2)}_S < b_{k2I}$ *do* ! $\underline{x}_k$ *is unchanged.*

7.2.2.2.2.1. $\underline{x}'^{(2)} := \underline{x}_k$

$\underline{x}'^{(1)} \neq \emptyset$ *and* $\underline{x}'^{(2)} = \emptyset$:

7.2.2.2.3. *if* $x'^{(1)}_S < b_{k2I}$ *do* ! $\underline{x}_k$ *is unchanged.*

7.2.2.2.3.1. $\underline{x}'^{(1)} := \underline{x}_k$

$\underline{x}'^{(1)} = \emptyset$ *and* $\underline{x}'^{(2)} \neq \emptyset$:

7.2.2.2.4. *if* $x'^{(2)}_S < b_{k2I}$ *do* ! $\underline{x}_k$ *is unchanged.*

7.2.2.2.4.1. $\underline{x}'^{(2)} := \underline{x}_k$

default : ! $\underline{x}'^{(1)}$ *and* $\underline{x}'^{(2)}$ *are empty.*

7.2.2.2.5. $\underline{x}'^{(1)} := \underline{x}_k$

default : ! $j > 1$.

7.2.3. $\underline{x}'^{(1)} := \underline{x}_k$

7.2.4. $\underline{x}'^{(2)} := \emptyset$

8. *return* □

procedure newton.method.1($\underline{B} \in I(M(R^n, R^2))$, $\underline{J} \in I(M(R^n))$, $n \in N$,

ignore.boundary $\in B$; $\underline{x} \in I(R^n)$: $\underline{A} \in I(M(R^n))$,

$B \in M(R^n)$, *singular*, *delete.x* $\in B$)

! This procedure implements step 5 of Hansen's global optimization algorithm.

! On entry, $\underline{B}$ is as in §3.1, $\underline{J}$ is obtained from the procedure *solve.quadratic*.1, n is

! the number of components in $\underline{x}$, and the Boolean *ignore.boundary* is as in §3.2.

! The input-output parameter $\underline{x}$ is the current box.

! On return, $B = \{m(\underline{J})\}^{-1}$, $\underline{A} = B\underline{J}$, *singular* = *true* if $m(\underline{J})$ is singular and

! *delete.x* = *true* if $\underline{x}$ does not contain a global minimizer of f even on its

! boundary.

1. $A := (\underline{0})_{n \times n}$

2. $B := (0)_{n \times n}$

3. *delete.x* := *false*

4. *singular* := *false*

5. $j := 0$! *j is the number of faces shared by* $\underline{x}$ *and* $\hat{\underline{x}}$.

6. *if* $m(\underline{J})$ *is singular*

   then

   6.1. *singular* := *true*

   ! *The value of singular is determined in the*

   ! *procedure which is used to compute* $\{m(\underline{J})\}^{-1}$.

   else

   6.2. *for* $i = 1$ *to* n *do*

   ! *We calculate j, the number of faces shared by* $\underline{x}$ *and* $\hat{\underline{x}}$.

   6.2.1. *if* $x_{iI} \in \underline{b}_{i1}$ *do*

   6.2.1.1. $j := j + 1$

   6.2.2. *if* $x_{iS} \in \underline{b}_{i2}$ *do*

   6.2.2.1. $j := j + 1$

   6.3. $B := \{m(\underline{J})\}^{-1}$

   6.4. $\underline{A} := B\underline{J}$

   6.5. $b := Bf'(m(\underline{x}))^T$

6.6. $k := 1$

6.7. *repeat*

6.7.1. *if* $0 \notin \underline{a}_{kk}$ *do*

6.7.1.1. $newton(\underline{B}, \underline{A}, B, \underline{x}, j, k, n, ignore.boundary : \underline{x}'^{(1)}, \underline{x}'^{(2)}, delete.x)$

! *If* $delete.x = true$ *then both* $\underline{x}'^{(1)}$ *and* $\underline{x}'^{(2)}$ *are empty.*

6.7.1.2. *if* $\sim delete.x$ *do*

6.7.1.2.1. *case true of*

$\underline{x}'^{(1)} \neq \emptyset$ *and* $\underline{x}'^{(2)} = \emptyset$:

6.7.1.2.1.1. $\underline{x}_k := \underline{x}'^{(1)}$

$\underline{x}'^{(1)} = \emptyset$ *and* $\underline{x}'^{(2)} \neq \emptyset$:

6.7.1.2.1.2. $\underline{x}_k := \underline{x}'^{(2)}$

default : ! $\underline{x}'^{(1)} \neq \emptyset$ *and* $\underline{x}'^{(2)} \neq \emptyset$

! *This case should never happen because* $0 \notin \underline{a}_{kk}$.

! *Therefore we stop the computation.*

6.7.1.2.1.3. *write* "*Error in newton.method.1.*"

6.7.1.2.1.4. *stop*

while $\sim delete.x$ *and* $k < n$ *do*

6.7.2. $k := k + 1$

7. *return* □

3.6 Step 6. Sub-box Deletion Using Function Values (2)

Let $\underline{x} \in I(R^n)$ be a sub-box of $\hat{\underline{x}}$ resulting from Step 5. In Step 6 we complete the method for reducing or deleting $\underline{x}$ which is described in §3.4.

For those values of $k \in \{1, \ldots, n\}$ such that $G_{kkI} < 0$ where $\underline{G}(\underline{x})$ has already been computed in Step 4, we solve 3.6 to obtain at most two non-empty intervals $\underline{y}^{(1)}$ and $\underline{y}^{(2)}$ (See §7 of [Han–80a].). However if only one, say $\underline{y}^{(1)}$, is non-empty then we replace $\underline{x}_k$ with $\underline{y}^{(1)}$. If both $\underline{y}^{(1)}$ and $\underline{y}^{(2)}$ are non-empty then we leave $\underline{x}_k$ unchanged but we save $\underline{y}^{(1)}$ and $\underline{y}^{(2)}$ for use in Step 8.

These ideas give rise to the procedure *solve.quadratic.*2 which follows.

*procedure solve.quadratic.*2($\underline{G} \in I(M(R^n)), \bar{\underline{f}} \in I(R), \varepsilon_1 \in R, n \in N$; $\underline{x} \in I(R^n)$:
$\underline{y}^{(1)}, \underline{y}^{(2)} \in I(R^n), delete.x \in B, y.double \in B^n$)

! This procedure implements step 6 of Hansen's global optimization algorithm.

! On entry, $\underline{G}$ is as computed in the procedure *solve.quadratic.*1, and $\bar{\underline{f}}$, ε_1 *and* n

! are as in the preceding sub-sections.

! The input-output parameter $\underline{x}$ is the current box.

! On return, $\underline{y}^{(1)}$ *and* $\underline{y}^{(2)}$ are such that $\underline{y}_k^{(1)}$ and $\underline{y}_k^{(2)}$ $(k = 1, \ldots, n)$ are equal to

! $\underline{x}_k^{(1)}$ and $\underline{x}_k^{(2)}$ respectively if $\underline{x}_k$ is reduced to $\underline{x}_k^{(1)}$ and $\underline{x}_k^{(2)}$; otherwise $\underline{y}_k^{(1)} = \emptyset$

! and $\underline{y}_k^{(2)} = \emptyset$. If $\underline{y}_k^{(1)} \neq \emptyset$ and $\underline{y}_k^{(2)} \neq \emptyset$ then $y.double_k = \underline{true}$. If $\underline{x}$ is deleted then

! $delete.x = \underline{true}$.

1. $\underline{\tilde{x}} := \underline{x} - m(\underline{x})$

2. $delete.x := \underline{false}$

3. $y.double := (\ \underline{false}\ )_{n\times 1}$

4. $\underline{y}^{(1)} := (\underline{0})_{n\times 1}$

5. $\underline{y}^{(2)} := (\underline{0})_{n\times 1}$

6. $\underline{g} := \underline{g}(m(\underline{x}))$

7. $\underline{f} := \underline{f}(m(\underline{x}))$

8. $\underline{E} := \underline{\bar{f}} - \underline{f} - [\varepsilon_1, \varepsilon_1]$

9. $k := 1$

10. *repeat*

    10.1. *if* $G_{kkI} < 0$ *do*

        10.1.1. $\underline{a}_k := \sum_{\substack{j=1\\ j\neq k}}^{n} \underline{g}_j \underline{\tilde{x}}_j + \frac{1}{2} \sum_{\substack{j=1\\ j\neq k}}^{n} \sum_{\substack{i=j\\ i\neq k}}^{n} \underline{G}_{ij} \underline{\tilde{x}}_i \underline{\tilde{x}}_j - \underline{E}$! *equation* 3.7

        10.1.2. $\underline{b}_k := \underline{g}_k + \frac{1}{2} \sum_{j=1}^{k-1} \underline{G}_{kj} \underline{\tilde{x}}_j + \frac{1}{2} \sum_{j=k+1}^{n} \underline{G}_{jk} \underline{\tilde{x}}_j$! *equation* 3.8

        10.1.3. $\underline{c}_k := \frac{1}{2} \underline{G}_{kk}$! *equation* 3.9

        10.1.4. *Solve* $\underline{a}_k + \underline{b}_k t + \underline{c}_k t^2 \leq 0$ (*relation* 3.6) *to give* $\underline{t}^{(1)}$ *and* $\underline{t}^{(2)}$, *using the method which has been described by Hansen* [*Han* – –80*a*].

        10.1.5. *case true of*

        $\underline{t}^{(1)} \neq \emptyset$ *and* $\underline{t}^{(2)} = \emptyset$:

10.1.5.1. $\underline{x}_k := \underline{t}^{(1)} + m(\underline{x}_k)$

$\underline{t}^{(1)} = \emptyset$ *and* $\underline{t}^{(2)} \neq \emptyset$:

10.1.5.2. $\underline{x}_k := \underline{t}^{(2)} + m(\underline{x}_k)$

$\underline{t}^{(1)} \neq \emptyset$ *and* $\underline{t}^{(2)} \neq \emptyset$:

10.1.5.3. $y.double_k :=$ *true*

10.1.5.4. $\underline{y}_k^{(1)} := \underline{t}^{(1)} + m(\underline{x}_k)$

10.1.5.5. $\underline{y}_k^{(2)} := \underline{t}^{(2)} + m(\underline{x}_k)$

default : ! $\underline{t}^{(1)} = \emptyset$ *and* $\underline{t}^{(2)} = \emptyset$.

10.1.5.6. $delete.x :=$ *true*

while $\sim delete.x$ *and* $k < n$ *do*

10.2. $k := k + 1$

11. *return* □

3.7 Step 7. Sub-box Deletion Using Interval Newton Methods (2)

Let $\underline{x} \in I(R^n)$ be a sub-box of $\hat{\underline{x}}$. Now $\underline{J}$ and B have been computed in Step 4 and Step 5 respectively. So, if $m(\underline{J})$ is non-singular then we use 3.10 and 3.11 as in the procedure *newton* (§3.5) for those values of $k \in \{1, \ldots, n\}$ such that $0 \in (B\underline{J})_{kk}$ to determine $\underline{x}_k$ which consists of at most two intervals $\underline{y}^{(1)}$ and $\underline{y}^{(2)}$. If only one, say $\underline{y}^{(1)}$, is non-empty then we repalce $\underline{x}_k$ with $\underline{y}^{(1)}$. If both $\underline{y}^{(1)}$ and $\underline{y}^{(2)}$ are non-empty then we save them for use in Step 8.

If the reduction of $\underline{x}_k$ would delete boundary points of $\hat{\underline{x}}$ from the box $\underline{x}$ then we retain the box $\underline{x}$ unchanged. Therefore we proceed as in Step 5 save that here we

obtain two subintervals of $\underline{x}_k$ which are saved for use in Step 8.

These ideas give rise to the procedure *newton.method.2* which follows.

procedure *newton.method.2*($\underline{B} \in I(M(R^n, R^2)), \underline{A} \in I(M(R^n)), B \in M(R^n)$,
singular, *ignore.boundary* $\in B, n \in N$; $\underline{x} \in I(R^n)$:
$\underline{y}'^{(1)}, \underline{y}'^{(2)} \in I(R^n)$, *delete.x* $\in B, y'.double \in B^n$)

! This procedure implements step 7 of Hansen's global optimization algorithm.

! On entry, $\underline{B}$ is as in §3.1, $\underline{A}$, B and *singular* are as computed in the procedure

! *newton.method.1*, *ignore.boundary* is as in §3.2, and n is the number of components

! of the current box $\underline{x}$.

! On return, $\underline{y}'^{(1)}$ *and* $\underline{y}'^{(2)}$ are such that $\underline{y}'^{(1)}_k = \underline{y}^{(1)}$ *and* $\underline{y}'^{(2)}_k = \underline{y}^{(2)}$ for

! $k \in \{1, \ldots, n\}$ where $\underline{x}_k$ is reduced to two intervals $\underline{y}^{(1)}$ and $\underline{y}^{(2)}$. If $\underline{y}'^{(1)}_k \neq \emptyset$

! and $\underline{y}'^{(2)}_k \neq \emptyset$ then $y'.double_k = \underline{true}$.

1. *delete.x* := *false*
2. $\underline{y}'^{(1)} := (\underline{0})_{n\times 1}$
3. $\underline{y}'^{(2)} := (\underline{0})_{n\times 1}$
4. $y'.double := (\underline{false})_{n\times 1}$
5. *if* $\sim$ *singular* *do*

5.1. $j := 0$

5.2. *for* $i = 1$ *to* n *do*

! *Calculate the number of faces shared by* $\underline{x}$ *and* $\hat{\underline{x}}$.

5.2.1. *if* $x_{iI} \in \underline{b}_{i1}$ *do*

5.2.1.1. $j := j + 1$

5.2.2. *if* $x_{iS} \in \underline{b}_{i2}$ *do*

5.2.2.1. $j := j + 1$

5.3. $k := 1$

5.4. *repeat*

5.4.1. *if* $0 \in a_{kk}$ *do*

5.4.1.1. $newton(\underline{B}, \underline{A}, B, \underline{x}, j, k, n, ignore.boundary :$

$\underline{y}^{(1)}, \underline{y}^{(2)}, delete.x)$

! *If* $delete.x =$ *true then both* $\underline{y}^{(1)}$ *and* $\underline{y}^{(2)}$ *are empty.*

5.4.1.2. *if* $\sim delete.x$ *do*

5.4.1.2.1. *case true of*

$\underline{y}^{(1)} \neq \emptyset$ *and* $\underline{y}^{(2)} = \emptyset$:

5.4.1.2.1.1. $\underline{x}_k := \underline{y}^{(1)}$

$\underline{y}^{(1)} = \emptyset$ *and* $\underline{y}^{(2)} \neq \emptyset$:

5.4.1.2.1.2. $\underline{x}_k := \underline{y}^{(2)}$

default : ! $\underline{y}^{(1)} \neq \emptyset$ *and* $\underline{y}^{(2)} \neq \emptyset$

5.4.1.2.1.3. $y'.double_k :=$ *true*

5.4.1.2.1.4. $\underline{y'}_k^{(1)} := \underline{y}^{(1)}$

5.4.1.2.1.5. $\underline{y'}_k^{(2)} := \underline{y}^{(2)}$

while $\sim$ *delete.x* *and* $k < n$ *do*

5.4.2. $k := k + 1$

6. *return* □

3.3 Step 8. Determination of the Largest Gap

In Step 8 we combine the results which are obtained in steps 6 and 7 corresponding to those $\underline{x}_i$ $(i \in \{1, \ldots, n\})$ which have been divided into two subintervals. We find the intersection $\underline{y}_i''$ of $\underline{y}_i$ $(= \underline{y}_i^{(1)} \cup \underline{y}_i^{(2)})$ from Step 6 and $\underline{y}_i'$ $(= \underline{y'}_i^{(1)} \cup \underline{y'}_i^{(2)})$ from Step 7. If in Step 6 or Step 7 $\underline{x}_i$ $(i \in \{1, \ldots, \})$ is not divided into two subintervals then $\underline{y}_i$ or $\underline{y}_i'$ does not exist. If $m(\underline{J})$ is singular then $\underline{y}_i'$ does not exist because $\underline{y}_i'$ cannot be computed in Step 7. Therefore if one of the subintervals $\underline{y}_i$ and $\underline{y}_i'$ does not exist then we set $\underline{y}_i'' = \underline{x}_i$ except when $m(\underline{J})$ is singular, in which case we set $\underline{y}_i'' = \underline{y}_i$ if $\underline{y}_i$ exists and we set $\underline{y}_i'' = \underline{x}_i$ otherwise. The intersection $\underline{y}_i \cap \underline{y}_i'$ $(i \in \{1, \ldots, n\})$ will produce at most three subintervals because $\underline{y}_i^{(1)} \cap \underline{y}_i^{(2)} = \emptyset$ and $\underline{y'}_i^{(1)} \cap \underline{y'}_i^{(2)} = \emptyset$ so that only one of $\underline{y}_i^{(1)}$ and $\underline{y}_i^{(2)}$ can have a non-empty intersection with both $\underline{y'}_i^{(1)}$ and $\underline{y'}_i^{(2)}$.

If $\underline{y}_i''$ is composed of three intervals [Han–80a], then we set either $\underline{y}_i'' = \underline{y}_i$ or $\underline{y}_i'' = \underline{y}_i'$, whichever has the intersection with $\underline{x}_i$ of least width. But we know that $\underline{y}_i \subseteq \underline{x}_i$ and $\underline{y}_i' \subseteq \underline{x}_i$, so if $w(\underline{y}_i) < w(\underline{y}_i')$ then we set $\underline{y}_i'' = \underline{y}_i$, and otherwise we set $\underline{y}_i'' = \underline{y}_i'$. If $\underline{y}_i''$ is composed of two intervals then we set $x.divided = \underline{true}$; this will be used in Step 9.

Hansen says nothing about what to do if $\underline{y}_i$ and $\underline{y}'_i$ are disjoint. Therefore in our implementation, if $\underline{y}_i$ and $\underline{y}'_i$ are disjoint, then the algorithm is terminated. Presumably if $\underline{y}_i \cap \underline{y}'_i = \emptyset$ then either $\underline{y}_i$ or $\underline{y}'_i$ does not contain x_i^*. This should not be possible.

Suppose that $\underline{y}''_i$ $(i = 1, \ldots, s)$ which are obtained from the intersection of $\underline{y}_i$ and $\underline{y}'_i$ $(i = 1, \ldots, n)$, are composed of two subintervals where $s \leq n$. Since we do not want more than one interval $\underline{x}_i$ $(i = 1, \ldots, n)$ to be divided into two subintervals, we save the $\underline{y}''_i$ which deletes the largest subinterval of $\underline{x}_i$. Suppose that $\underline{y}''_j$ is saved. Then for $i = 1, \ldots, s$ and $i \neq j$ we replace $\underline{y}''_i$ with $\underline{x}_i$; that is, we ignore the fact that part of $\underline{x}_i$ $(i = 1, \ldots, s,\ i \neq j)$ could be deleted.

The preceding ideas give rise to the procedure *d.l.gap* (determine.largest.gap).

procedure $d.l.gap(\underline{y}^{(1)}, \underline{y}^{(2)}, \underline{y}'^{(1)}, \underline{y}'^{(2)}, \underline{x} \in I(R^n), y.double, y'.double \in B^n,$
$singular \in B : \underline{y}''^{(1)}, \underline{y}''^{(2)} \in I(R^n), j \in N, x.divided \in B)$

! This procedure implements step 8 of Hansen's global optimization algorithm.
! On entry, $\underline{y}^{(1)}$, $\underline{y}^{(2)}$ are the boxes which are computed in the procedure
! *solve.quadratic.2*, $\underline{y}'^{(1)}$, $\underline{y}'^{(2)}$ are the boxes which are computed in the
! procedure *newton.method.2*, *y.double* and *y'.double* are the Boolean vectors
! which are computed in the procedures *solve.quadratic.2* and *newton.method.2*
! respectively, and *singular* is the Boolean which is computed in the procedure

! *newton.method*.1.

! In this procedure we compute

! $(\underline{y}^{(1)} \cup \underline{y}^{(2)}) \cap (\underline{y}'^{(1)} \cup \underline{y}'^{(2)}) =$

! $(\underline{y}^{(1)} \cap \underline{y}'^{(1)}) \cup (\underline{y}^{(1)} \cap \underline{y}'^{(2)}) \cup (\underline{y}^{(2)} \cap \underline{y}'^{(1)}) \cup (\underline{y}^{(2)} \cap \underline{y}'^{(2)})$.

! If three sub-boxes are obtained we retain either $\underline{y}^{(1)} \cup \underline{y}^{(2)}$ or $\underline{y}'^{(1)} \cup \underline{y}'^{(2)}$

! whichever has the smallest width.

! On return, $\underline{y}''^{(1)}$ and $\underline{y}''^{(2)}$ are the boxes which result from this procedure, j denotes

! which component of $\underline{x}$ is divided into two sub-boxes, and if $\underline{x}$ is divided into

! two sub-boxes then $x.divided = \underline{true}$.

1. $x.divided :=$ *false*
2. $w_c := 0$
3. $j := 0$
4. $\underline{y}''^{(1)} := (\underline{0})_{n\times 1}$
5. $\underline{y}''^{(2)} := (\underline{0})_{n\times 1}$
6. *for* $i = 1$ *to* n *do*

   ! *Check each component of the box* $\underline{x}$.

   6.1. *if singular*

   then

   ! *The Newton method cannot be used; therefore we can only check*

   ! *whether or not the procedure solve.quadratic.2 has produced two*

! *sub – boxes* $\underline{y}^{(1)}$ *and* $\underline{y}^{(2)}$. *If* $y.double_i =$ *true then we set* $\underline{y''}_i^{(1)} = \underline{y}^{(1)}$

! *and* $\underline{y''}_i^{(2)} = \underline{y}^{(2)}$.

6.1.1. *if* $y.double_i$ *do*

6.1.1.1. $x.divided :=$ *true*

6.1.1.2. $\underline{y''}_i^{(1)} := \underline{y}_i^{(1)}$

6.1.1.3. $\underline{y''}_i^{(2)} := \underline{y}_i^{(2)}$

6.1.1.4. $w := w(\underline{x}_i) - w(\underline{y}_i^{(1)}) - w(\underline{y}_i^{(2)})$

! *If* $w > w_c$ *then* $\underline{x}_i$ *is reduced more than*

! $\underline{x}_k$ $(1 \leq k \leq i-1)$.

6.1.1.5. *if* $w > w_c$ *do*

6.1.1.5.1. $j := i$

6.1.1.5.2. $w_c := w$

else

! *Here, singular =* *false* ; *therefore check whether or not*

! $\underline{y}_i^{(k)} \neq \emptyset$ *and* $\underline{y'}_i^{(k)} \neq \emptyset$ $(k = 1, 2)$. *If so then proceed with*

! *the following steps; otherwise go to step* 6.1 *with* $i := i + 1$.

6.1.2. *if* $y.double_i$ *and* $y'.double_i$ *do*

6.1.2.1. $\underline{a}_1 := \underline{y}_i^{(1)}$

6.1.2.2. $\underline{a}_2 := \underline{y}_i^{(2)}$

6.1.2.3. $\underline{b}_1 := \underline{y'}_i^{(1)}$

6.1.2.4. $\underline{b}_2 := \underline{y'}_i^{(2)}$

6.1.2.5. $\underline{c} := (\underline{0})_{3\times 1}$

6.1.2.6. $t := 1$

6.1.2.7. $m := 0$

! *m denotes the number of intervals which are produced by the*

! *intersection of* $(\underline{a}_1 \cup \underline{a}_2)$ *and* $(\underline{b}_1 \cup \underline{b}_2)$.

6.1.2.8. *for* $k = 1$ *to* 2 *do*

6.1.2.8.1. *for* $l = 1$ *to* 2 *do*

6.1.2.8.1.1. *if* $\underline{a}_k \cap \underline{b}_l \neq \emptyset$ *do*

6.1.2.8.1.1.1. $\underline{c}_t := \underline{a}_k \cap \underline{b}_l$

6.1.2.8.1.1.2. $t := t + 1$

6.1.2.8.1.1.3. $m := m + 1$

6.1.2.9. *case* m *of*

3 :

! *For this case we retain either*

! $(\underline{y}^{(1)} \cup \underline{y}^{(2)})$ *or* $(\underline{y}'^{(1)} \cup \underline{y}'^{(2)})$.

6.1.2.9.1. $x.divided := true$

6.1.2.9.2. $w := w(\underline{y}_i^{(1)}) + w(\underline{y}_i^{(2)})$

6.1.2.9.3. $w' := w(\underline{y}'^{(1)}_i) + w(\underline{y}'^{(2)}_i)$

! *If* $w \leq w'$ *then retain* $\underline{y}_i^{(1)} \cup \underline{y}_i^{(1)}$;

! *else retain* $\underline{y}'^{(1)}_i \cup \underline{y}'^{(2)}_i$.

6.1.2.9.4. *if* $w \leq w'$

then

6.1.2.9.4.1. $\underline{y}''^{(1)}_i := \underline{y}^{(1)}_i$

6.1.2.9.4.2. $\underline{y}''^{(2)}_i := \underline{y}^{(2)}_i$

6.1.2.9.4.3. $w_m := w$

else

6.1.2.9.4.4. $\underline{y}''^{(1)}_i := \underline{y}'^{(1)}_i$

6.1.2.9.4.5. $\underline{y}''^{(2)}_i := \underline{y}'^{(2)}_i$

6.1.2.9.4.6. $w_m := w'$

6.1.2.9.5. $w := w(\underline{x}_i) - w_m$

6.1.2.9.6. *if* $w > w_c$ *do*

6.1.2.9.6.1. $j := i$

6.1.2.9.6.2. $w_c := w$

2 :

! *For this case we have only* $\underline{c}_1 \neq \underline{0}$ *and* $\underline{c}_2 \neq \underline{0}$.

6.1.2.9.7. *x.divided* := *true*

6.1.2.9.8. $\underline{y}''^{(1)}_i := \underline{c}_1$

6.1.2.9.9. $\underline{y}''^{(2)}_i := \underline{c}_2$

6.1.2.9.10. $w := w(\underline{x}_i) - w(\underline{c}_1) - w(\underline{c}_2)$

6.1.2.9.11. *if* $w > w_c$ *do*

6.1.2.9.11.1. $j := i$

6.1.2.9.11.2. $w_c := w$

1 :

! *For this case only* $\underline{c}_1 \neq \underline{0}$.

6.1.2.9.12. $\underline{y}''^{(1)}_i := \underline{c}_1$

6.1.2.9.13. $w := w(\underline{x}_i) - w(\underline{c}_1)$

6.1.2.9.14. *if* $w > w_c$ *do*

6.1.2.9.14.1. $j := i$

6.1.2.9.14.2. $w_c := w$

default :

! *In this case the procedures solve.quadratic.2 and*

! *newton.method.2 have produced two disjoint intervals.*

! *Therefore we should stop the computation because*

! *this should not happen unless* $\underline{x}_i$ *does not contain* x_i^*.

6.1.2.9.15. *write* "*Two disjoint intervals are produced in d.l.gap.*"

6.1.2.9.16. *stop*

7. *for* $i = 1$ *to* n *do*

7.1. *if* $i \neq j$ *do*

! *If* $j \neq 0$ *then for* $1 \leq i \leq n$ *and* $i \neq j$ *we assign to* $\underline{y}''^{(1)}_i$

! *and* $\underline{y}''^{(2)}_i$ *the interval* $\underline{x}_i$.

7.1.1. $\underline{y}''^{(1)}_i := \underline{x}_i$

7.1.2. $\underline{y}''^{(2)}_i := \underline{x}_i$

8. *return* □

3.9 Step 9. Construction of New Sub-boxes

If $\underline{y}''_j$ exists; that is, if at least one interval $\underline{x}_j$ $(j = 1, \ldots, n)$ was divided into two subintervals, say $\underline{y}''^{(1)}_j$ and $\underline{y}''^{(2)}_j$ in Step 8 then we subdivide the box $\underline{x}$ into two sub-boxes $\underline{x}^{(1)}$ and $\underline{x}^{(2)}$, where

$$\underline{x}^{(1)} = (\underline{x}_1, \ldots, \underline{x}_{j-1}, \underline{y}''^{(1)}_j, \underline{x}_{j+1}, \ldots, \underline{x}_n)^T,$$

and

$$\underline{x}^{(2)} = (\underline{x}_1, \ldots, \underline{x}_{j-1}, \underline{y}''^{(2)}_j, \underline{x}_{j+1}, \ldots, \underline{x}_n)^T.$$

Otherwise, we might wish to subdivide the current box. Let $\underline{x}$ denote the box chosen in Step 1 with width $\hat{w} = \|w(\underline{x})\|$ and let $\underline{x}''$ denote the current box resulting from applying steps 2–8 to $\underline{x}$ with width $\tilde{w} = \|w(\underline{x}'')\|$. If $\tilde{w} > 0.75\hat{w}$ then we bisect $\underline{x}''$ along its widest dimension. The number 0.75 was suggested by Hansen.

This preceding ideas give rise to the procedure *c.n.s.boxes* (*construct.new.sub. boxes*).

procedure $c.n.s.boxes(\underline{x}, \underline{y}''^{(1)}, \underline{y}''^{(2)} \in I(R^n), \hat{w} \in R, j, n \in N$; $x.divided \in B$:

$$\underline{x}^{(1)}, \underline{x}^{(2)} \in I(R^n))$$

! This procedure implements step 9 of Hansen's global optimization algorithm.

! On entry, $\underline{x}$ is the current box, $\underline{y}''^{(1)}$ and $\underline{y}''^{(2)}$ are computed from the procedure
! *d.l.gap*, $\hat{w}$ is the width of the box which is extracted from the queue L_1 or L_2 in
! the procedure *terminate* (Step 1), j is computed from the procedure *d.l.gap* and
! n is the number of components of the initial box $\hat{\underline{x}}$.
! The input-output Boolean *x.divided* is computed from the procedure *d.l.gap*
! and *x.divided* = *true* if the box $\underline{x}$ is divided into two sub-boxes $\underline{x}^{(1)}$ and $\underline{x}^{(2)}$.

1. $\underline{x}^{(1)} := (0)_{n\times 1}$
2. $\underline{x}^{(2)} := (0)_{n\times 1}$
3. *if* $j = 0$

   then

   ! *For this case* $\underline{y}''^{(1)}$ *and* $\underline{y}''^{(2)}$ *are both empty so we need to check whether*
   ! $\tilde{w} = \|w(\underline{x})\| > 0.75\hat{w}$. *If* $\tilde{w} > 0.75\hat{w}$ *then we divide* $\underline{x}$ *along the direction*
   ! *of maximum width.*

   3.1. $j := 1$

   3.2. $\tilde{w} := w(\underline{x}_1)$

   3.3. *for* $i = 2$ *to* n *do*

      3.3.1. $w := w(\underline{x}_i)$

      3.3.2. *if* $w > \tilde{w}$ *do*

         3.3.2.1. $\tilde{w} := w$

         3.3.2.2. $j := i$

3.4. *if* $\tilde{w} > 0.75\hat{w}$ *do*

3.4.1. *for* $i = 1$ *to* n *do*

3.4.1.1. $\underline{x}_i^{(1)} := \underline{x}_i$

3.4.1.2. $\underline{x}_i^{(2)} := \underline{x}_i$

3.4.2. $\underline{x}_j^{(1)} := [x_{jI}, m(\underline{x}_j)]$

3.4.3. $\underline{x}_j^{(2)} := [m(\underline{x}_j), x_{jS}]$

3.4.4. $x.divided :=$ *true*

else

! *If* $x.divided =$ *true* *set* $\underline{x}^{(1)} := \underline{y}''^{(1)}$ *and* $\underline{x}^{(2)} := \underline{y}''^{(2)}$; *otherwise*

! *check the width of* $\underline{y}''^{(1)}$ *because* $\underline{y}''^{(1)} \neq \emptyset$ *whereas* $\underline{y}''^{(2)} = \emptyset$.

3.5. *if* $x.divided$

then

3.5.1. $\underline{x}^{(1)} := \underline{y}''^{(1)}$

3.5.2. $\underline{x}^{(2)} := \underline{y}''^{(2)}$

else

! $j \neq 0$ *and only one interval is obtained from* $(\underline{a}_1 \cup \underline{a}_2) \cap (\underline{b}_1 \cup \underline{b}_2)$

! *in the procedure d.l.gap.*

3.5.3. $\underline{x}_j := \underline{y}''^{(1)}_j$

3.5.4. $j := 1$

3.5.5. $\tilde{w} := w(\underline{x}_1)$

3.5.6. *for* $i = 2$ *to* n *do*

3.5.6.1. $w := w(\underline{x}_i)$

3.5.6.2. *if* $w > \tilde{w}$ *do*

3.5.6.2.1. $\tilde{w} := w$

3.5.6.2.2. $j := i$

3.5.7. *if* $\tilde{w} > 0.75\hat{w}$ *do*

3.5.7.1. *for* $i = 1$ *to* n *do*

3.5.7.1.1. $\underline{x}_i^{(1)} := \underline{x}_i$

3.5.7.1.2. $\underline{x}_i^{(2)} := \underline{x}_i$

3.5.7.2. $\underline{x}_j^{(1)} := [x_{jI}, m(\underline{x}_j)]$

3.5.7.3. $\underline{x}_j^{(2)} := [m(\underline{x}_j), x_{jS}]$

3.5.7.4. $x.divided :=$ *true*

4. *return* □

3.10 Step 10. Insertion of New Sub-boxes into the Appropriate Queue

In Step 10 we insert the box or boxes resulting from Step 9 into the appropriate queue; that is into either L_1 or L_2. We also update $\overline{f}$ if possible as described in §4 of [Han–80a]. Let $\underline{x} \in I(R^n)$ be a box resulting from Step 9. We evaluate $\underline{f}$ at the centre of $\underline{x}$. If $f_S < \overline{f}_I$ then we set $\overline{f} = [f_S, f_S]$; otherwise $\overline{f}$ is not changed. Here, although $\overline{f}$ can be reduced, $\underline{x}$ cannot be deleted because $\underline{f}$ is computed at $m(\underline{x})$. If $\|w(\underline{x})\| \leq \varepsilon_2$ then we insert the box $\underline{x}$ into the queue L_1; otherwise insert the box $\underline{x}$ into the queue L_2 and restart the process at Step 1.

The preceding ideas give rise to the procedure *insert.into.queue.*

procedure insert.into.queue($\underline{x}^{(1)}, \underline{x}^{(2)}, \underline{x} \in I(R^n), \varepsilon_2 \in R, n \in N, x.divided \in B$;

$L_1, L_2 \in Q, t_1, t_2 \in N, \underline{\bar{f}} \in I(R)$)

! This procedure implements step 10 of Hansen's global optimization algorithm.

! On entry, L_1, L_2, t_1 and t_2 are as in §3.1, $\underline{x}^{(1)}$, $\underline{x}^{(2)}$, and *x.divided* are as

! computed from the procedure *c.n.s.boxes*, $\underline{x}$ is the current box, and $\varepsilon_2 > 0$.

! On return $\underline{\bar{f}}$ is such that $\bar{f}_S > f^*$ where f^* is the global minimum of f.

1. *if x.divided*

   then

   ! *Compute* $\underline{f}^{(i)} = \underline{f}(x^{(i)})$ *where* $x^{(i)} = m(\underline{x}^{(i)})$ $(i = 1, 2)$, *and update the value*

   ! *of* $\underline{\bar{f}}$.

   1.1. $\underline{f}^{(1)} := \underline{f}(m(\underline{x}^{(1)}))$

   1.2. $\underline{f}^{(2)} := \underline{f}(m(\underline{x}^{(2)}))$

   1.3. *if* $f_S^{(1)} < \bar{f}_I$ *do*

      1.3.1. $\underline{\bar{f}} := [f_S^{(1)}, f_S^{(1)}]$

   1.4. *if* $f_S^{(2)} < \bar{f}_I$ *do*

      1.4.1. $\underline{\bar{f}} := [f_S^{(2)}, f_S^{(2)}]$

   ! *Insert* $\underline{x}^{(1)}$ *and* $\underline{x}^{(2)}$ *into either* L_1 *or* L_2.

   1.5. *if* $\left\| w(\underline{x}^{(1)}) \right\| \leq \varepsilon_2$

then

1.5.1. $\underline{x}^{(1)} \longrightarrow L_1$

1.5.2. $t_1 := t_1 + 1$

else

1.5.3. $\underline{x}^{(1)} \longrightarrow L_2$

1.5.4. $t_2 := t_2 + 1$

1.6. *if* $\left\| w(\underline{x}^{(2)}) \right\| \leq \varepsilon_2$

then

1.6.1. $\underline{x}^{(2)} \longrightarrow L_1$

1.6.2. $t_1 := t_1 + 1$

else

1.6.3. $\underline{x}^{(2)} \longrightarrow L_2$

1.6.4. $t_2 := t_2 + 1$

else

! *Update* $\underline{\bar{f}}$ *at* $m(\underline{x})$ *and then insert* $\underline{x}$ *into either* L_1 *or* L_2.

1.7. $\underline{f} := \underline{f}(m(\underline{x}))$

1.8. *if* $f_S < \bar{f}_I$ *do*

1.8.1. $\underline{\bar{f}} := [f_S, f_S]$

1.9. *if* $\|w(\underline{x})\| \leq \varepsilon_2$

then

1.9.1. $\underline{x} \longrightarrow L_1$

1.9.2. $t_1 := t_1 + 1$

else

1.9.3. $\underline{x} \longrightarrow L_2$

1.9.4. $t_2 := t_2 + 1$

2. *return* □

3.11 Hansen's Global Optimization Algorithm

In order to describe Hansen's global optimization algorithm, the following procedure is needed, in addition to those which have been described in §3.1 – §3.10.

procedure *execute.Hansen.steps*($\underline{B} \in I(M(R^n, R^2)), n, h_3 \in N, \varepsilon_0, \varepsilon_1, \varepsilon_2 \in R,$
ignore.boundary $\in B$; $L_1, L_2, L_3 \in Q,$
$\underline{f} \in I(R), h_1, t_1, h_2, t_2, t_3 \in N$)

! This procedure combines all the procedures in Steps 1 – 10.

! $\underline{B}$, ε_0, ε_1, and h_3 are as in Step 1, ε_2 is as in Step 3, *ignore.boundary* is

! as in Step 2 and n is the number of components in the initial box $\underline{\hat{x}}$.

! $\underline{f}$, h_1, t_1, h_2, t_2, t_3, L_1, L_2 and L_3 are as explained in §3.1.

1. *delete.x* := *false*

2. *singular* := *false*

3. *terminate*($\underline{\bar{f}}, \epsilon_0, \epsilon_1, L_3, h_3, t_1, t_2, t_3, n$; L_1, L_2, h_1, h_2 :

$\underline{x}, \hat{w}$, *f.too.wide*) ! §3.1.

4. *if* ~ *f.too.wide* *do*

! *If f.too.wide* = *false* *then four possible cases occur in the*

! *procedure terminate* : (*i*) $L_2 \longrightarrow \underline{x}$ *if* $L_2 \neq \emptyset$, (*ii*) $L_2 = \emptyset$, $L_1 \longrightarrow \underline{x}$ *if* $\epsilon_1 > 0$,

! (*iii*) $L_2 = \emptyset$, $L_1 \longrightarrow \underline{x}$ *if* $w(\underline{f}(\underline{x})) > \epsilon_0$, *and* (*iv*) *the algorithm H*

! *terminates. Therefore if f.too.wide* = *false* *then either the box* $\underline{x}$ *is*

! *processed or the algorithm is terminated. If f.too.wide* = *true* *then*

! *go to step* 5 [*Han* – –80*a*], *because f.too.wide* = *true* *only when* $L_2 = \emptyset$

! *and the box* $\underline{x}$ *which has been extracted from* L_1 *in the procedure*

! *terminate probably contains a minimizer, so the procedures*

! *monotonicity.test and non.convexity.test are unlikely to delete* $\underline{x}$.

4.1. *monotonicity.test*($\underline{B}, n$, *ignore.boundary* ; $L_3, \underline{x}, \underline{\bar{f}}, t_3$:

delete.x) ! §3.2.

4.2. *if* ~ *delete.x* *do*

4.2.1. *non.convexity.test*($\underline{B}, n, \epsilon_2$, *ignore.boundary* ; $L_1, L_2, \underline{x}, t_1, t_2$:

delete.x) ! §3.3.

5. *if* ~ *delete.x* *do*

5.1. *solve.quadratic.1*($\underline{\bar{f}}, \epsilon_1, n$; $\underline{x}$:

$\underline{J}, \underline{G}$, *delete.x*) ! §3.4.

5.2. *if* ~ *delete.x* *do*

5.2.1. *newton.method.1*($\underline{B}, \underline{J}, n$, *ignore.boundary* ; $\underline{x}$:

$\underline{A}, B, singular, delete.x$) ! §3.5.

! *If singular* = *true* *then the procedures newton.method.1 and*

! *newton.method.2 cannot be used.*

5.2.2. *if* ~ *delete.x* *do*

5.2.2.1. *solve.quadratic.2*($\underline{G}, \underline{\bar{f}}, \epsilon_1, n$; $\underline{x}$:

$\underline{y}^{(1)}, \underline{y}^{(2)}, delete.x$, *y.double*) ! §3.6.

5.2.2.2. *if* ~ *delete.x* *do*

5.2.2.2.1. *newton.method.2*($\underline{B}, \underline{A}, B$, *singular*,

ignore.boundary, n ; $\underline{x}$:

$\underline{y'}^{(1)}, \underline{y'}^{(2)}, delete.x$,

$y'.double$) ! §3.7.

5.2.2.2.2. *if* ~ *delete.x* *do*

5.2.2.2.2.1. *d.l.gap*($\underline{y}^{(1)}, \underline{y}^{(2)}, \underline{y'}^{(1)}, \underline{y'}^{(2)}$,

$\underline{x}$, *y.double*, $y'.double$,

singular :

$\underline{y''}^{(1)}, \underline{y''}^{(2)}, j$,

x.divided) ! §3.8.

! *If x.divided* = *true* *then* $\underline{y''}^{(1)} \neq \emptyset$

! *and* $\underline{y''}^{(2)} \neq \emptyset$

5.2.2.2.2.2. *c.n.s.boxes*($\underline{x}, \underline{y}''^{(1)}, \underline{y}''^{(2)}$,

$\hat{w}, j, n$; *x.divided* :

$\underline{x}^{(1)}, \underline{x}^{(2)}$) ! §3.9.

5.2.2.2.2.3. *insert.into.queue*($\underline{x}^{(1)}, \underline{x}^{(2)}$,

$\underline{x}, \varepsilon_2, n$,

x.divided ;

L_1, L_2, t_1, t_2

$\bar{f}$) ! §3.10.

6. *return* □

Hansen's Global Optimization Algorithm

Data :

$L_i \in Q,\ L_i = \emptyset\ (i = 1, 2, 3)$,

$h_i, t_i \in N,\ (h_i = 1,\ t_i = 0)\ (i = 1, 2, 3)$,

$n \in N,\ (n \geq 2)$,

$\varepsilon_0, \varepsilon_1, \varepsilon_2 \in R\ (\varepsilon_0 > 0,\ \varepsilon_1 \geq 0,\ \varepsilon_2 \geq 0)$,

$\hat{\underline{x}} \in I(R^n)$,

ignore.boundary $\in B$.

! *If ignore.boundary* = *true then* $x^* \in int(\hat{\underline{x}})$; *else* $x^* \in \hat{\underline{x}}$.

1. *for* $i = 1$ *to* n *do*

! *Determine the boundary points of* $\hat{\underline{x}}$.

1.1. $\underline{b}_{i1} := [\hat{x}_{iI}, \hat{x}_{iI}]$

1.2. $\underline{b}_{i2} := [\hat{x}_{iS}, \hat{x}_{iS}]$

2. $\overline{f} := \underline{f}(m(\hat{\underline{x}}))$

3. *if* $\|w(\hat{\underline{x}})\| \leq \varepsilon_2$

then

3.1. $\hat{\underline{x}} \longrightarrow L_1$! *Insert* $\hat{\underline{x}}$ *into* L_1.

3.2. $t_1 := 1$

else

3.3. $\hat{\underline{x}} \longrightarrow L_2$! *Insert* $\hat{\underline{x}}$ *into* L_2.

3.4. $t_2 := 1$

4. *while true do*

4.1. *execute.Hansen.steps*$(\underline{B}, n, h_3, \varepsilon_0, \varepsilon_1, \varepsilon_2, ignore.boundary$;

$L_1, L_2, L_3, \overline{f}, h_1, t_1, h_2, t_2, t_3)$! §3.11.

5. *stop* □

CHAPTER 4

Computable Error Bounds For Nonlinear Programming

In this chapter *Problem P* (Chapter 3) is shown to be related to the problem of solving a system of nonlinear algebraic equations and inequalities.

4.1 The Global Optimization Problem Expressed as a System of Nonlinear Equations and Inequalities

Consider the nonlinear programming problem $NP.1$

$$\left.\begin{array}{lll} & \textit{minimize} & f(x)\ (x \in \hat{D} \subseteq R^n) \\ & \textit{subject to} & \\ & c_i(x) \geq 0 & (i = 1, \ldots, m) \\ \textit{and} & & \\ & h_j(x) = 0 & (j = 1, \ldots, r) \end{array}\right\} NP.1$$

where $f : D \subseteq R^n \to R^1$, $c_i : R^n \to R^1$ $(i = 1, \ldots, m)$, $h_j : R^n \to R^1$ $(j = 1, \ldots, r)$ are given continuously differentiable functions and $\hat{D} \subset D$ is an open set containing the points which satisfy the constraints. In order to establish the first order necessary conditions for $NP.1$ we need the following definitions and theorems ([FiaM-68a] [Gal–51a] [ManF-67a]).

Definition 4.1 : Let $f : D \subseteq R^n \rightarrow R^1$ be a given function and let $\hat{D} \subset D$ be a given set. The point $x^* \in \hat{D}$ is a *strong local minimizer* of f in $\hat{D}$ if and only if $\exists \varepsilon > 0$ such that

$$f(x^*) < f(x) \ (\forall x \in ((B(x^*, \varepsilon) - \{x^*\}) \cap \hat{D}))$$

where $B(x^*, \varepsilon)$ is a neighbourhood of x^* with radius ε. □

Definition 4.2 : A point $x \in D$ is a *feasible point* for $NP.1$ if and only if x satisfies the constraints $c_i(x) \geq 0 \ (i = 1, \ldots, m)$ and $h_j(x) = 0 \ (j = 1, \ldots, r)$. □

Definition 4.3 : For $NP.1$ the *feasible set* D is defined by

$$D = \{x \in R^n \mid c_i(x) \geq 0 \ (i = 1, \ldots, m) \wedge h_j(x) = 0 \ (j = 1, \ldots, r)\}.$$ □

Definition 4.4 : The constraints $c_i(x) \geq 0 \ (i = 1, \ldots, m)$ corresponding to $NP.1$ are called *inequality constraints* and the constraints $h_j(x) = 0 \ (j = 1, \ldots, r)$ are called *equality constraints*. □

Definition 4.5 : The inequality constraint $c_i(x) \geq 0 \ (1 \leq i \leq m)$ for $NP.1$ is *active* (*binding*) at $\hat{x} \in R^n$ if and only if $c_i(\hat{x}) = 0$. □

Definition 4.6 : Let $f : D \subseteq R^n \rightarrow R^1$, $c_i : R^n \rightarrow R^1 \ (i = 1, \ldots, m)$ and $h_j : R^n \rightarrow R^1 \ (j = 1, \ldots, r)$ be given mappings. Suppose that $\partial_k f(\hat{x}) \ (k = 1, \ldots, n)$, $\partial_k c_i(\hat{x}) \ (i = 1, \ldots, m; \ k = 1, \ldots, n)$ and $\partial_k h_j(\hat{x}) \ (j = 1, \ldots, r; \ k = 1, \ldots, n)$ exist, where

$\hat{x} \in D$ is given. Then the gradients $\nabla f(\hat{x}) \in R^n$ of f, $\nabla c_i(\hat{x}) \in R^n$ of c_i $(i = 1, \ldots, m)$ and $\nabla h_j(\hat{x}) \in R^n$ of h_j $(j = 1, \ldots, r)$ at $\hat{x}$ are defined by $\nabla f(\hat{x}) = (\partial_k f(\hat{x}))_{n\times 1}$, $\nabla c_i(\hat{x}) = (\partial_k c_i(\hat{x}))_{n\times 1}$ and $\nabla h_j(\hat{x}) = (\partial_k h_j(\hat{x}))_{n\times 1}$ respectively. □

Definition 4.7 : The *Lagrangian function* $L : R^n \times R^m \times R^r \to R^1$ corresponding to $NP.1$ is defined by

$$L(x, u, w) = f(x) - \sum_{i=1}^{m} u_i c_i(x) + \sum_{j=1}^{r} w_j h_j(x). \quad \square$$

Suppose that $x^* \in R^n$ is a feasible point for $NP.1$ and that $f : R^n \to R^1$, $c_i : R^n \to R^1$ $(i = 1, \ldots, m)$ and $h_j : R^n \to R^1$ $(j = 1, \ldots, r)$ have first partial derivatives at x^*. Let $c_i^* = c_i(x^*)$ $(i = 1, \ldots, m)$, etc, and let

$$B^* = \{i \in N_+ \mid c_i(x^*) = 0\}, \tag{4.1}$$

$$Z_1^* = \{z \in R^n \mid z^T \nabla c_i^* \geq 0 \ (\forall i \in B^*) \wedge z^T \nabla h_j^* = 0 \ (j = 1, \ldots, r) \wedge z^T \nabla f^* \geq 0\}, \tag{4.2}$$

$$Z_2^* = \{z \in R^n \mid z^T \nabla c_i^* \geq 0 \ (\forall i \in B^*) \wedge z^T \nabla h_j^* = 0 \ (j = 1, \ldots, r) \wedge z^T \nabla f^* < 0\}, \tag{4.3}$$

$$Z_3^* = \{z \in R^n \mid (\exists i \in B^*, \ z^T \nabla c_i^* < 0) \vee (\exists j \in \{1, \ldots, r\}, z^T \nabla h_j^* \neq 0)\}, \tag{4.4}$$

where $\nabla c_i^* = \nabla c_i(x^*)$ $(i = 1, \ldots, m)$, $\nabla h_j^* = \nabla h_j(x^*)$ $(i = 1, \ldots, r)$ and $\nabla f^* = \nabla f(x^*)$. The sets Z_1^*, Z_2^* and Z_3^* are disjoint and $Z_1^* \cup Z_2^* \cup Z_3^* = R^n$.

Observe that all feasible directions from x^* must be contained in $Z_1^* \cup Z_2^*$. Furthermore, $f(x)$ initially decreases along $z \in Z_2^*$ and initially increases or is constant along $z \in Z_1^*$. Thus if $Z_2^* \neq \emptyset$ we would not expect x^* to be a local minimizer. We shall see that this expectation is correct, but only if we make a suitable additional assumption.

Definition 4.8 : Let $\alpha : [a, b] \subset R^1 \rightarrow R^n$ be a given mapping. Then $\{\alpha(\theta) \in R^n \mid \theta \in [a, b]\}$ is a *curve* in R^n. □

Definition 4.9 : A curve $\{\alpha(\theta) \in R^n \mid \theta \in [a, b]\}$ is *once differentiable* at $\hat{\theta} \in [a, b]$ if and only if $D\alpha(\hat{\theta}) \in R^n$ defined by

$$D\alpha(\hat{\theta}) = \left(\frac{d}{d\theta}\alpha_i(\hat{\theta})\right)_{n \times 1}$$

exists. □

Definition 4.10 : An *arc* is a differentiable curve $\{\alpha(\theta) \in R^n \mid \theta \in [0, \varepsilon]\}$. □

Definition 4.11 : The *tangent* T to the arc $\{\alpha(\theta) \in R^n \mid \theta \in [0, \varepsilon]\}$ at $\hat{\theta} \in [0, \varepsilon]$ is defined by

$$T = \{z \in R^n \mid z = \lambda D\alpha(\hat{\theta}) \wedge \lambda > 0\}.$$ □

Definition 4.12 : A curve $\{\alpha(\theta) \in R^n \mid \theta \in [a, b]\}$ *passes through* $\hat{x} \in R^n$ if and only if $\exists \hat{\theta} \in [a, b]$ such that $\hat{x} = \alpha(\hat{\theta})$. □

Definition 4.13 : An arc $\{\alpha(\theta) \in R^n \mid \theta \in [0, \varepsilon]\}$ is *feasible* if and only if $\alpha(\theta)$ is feasible $(\forall \theta \in [0, \varepsilon])$. □

Theorem 4.1 : If T is the tangent to a feasible arc for $NP.1$ through x^*, at x^*, then $T \subset Z_1^* \cup Z_2^*$ for $\lambda > 0$ sufficiently small. □

Theorem 4.2 (Existence of Generalized Lagrange Multipliers) : If (1) $x^* \in R^n$ is a feasible point for $NP.1$; (2) $f \in C^1(\hat{D})$, $c_i \in C^1(\hat{D})$ $(i = 1, \ldots, m)$ and $h_j \in C^1(\hat{D})$ $(j = 1, \ldots, r)$ where $\hat{D} \subset D$ is an open set containing the points which satisfy the constraints; (3) $Z_2^* = \emptyset$, then $\exists u^* \in R^m$ and $w^* \in R^r$ such that

$$c_i(x^*) \geq 0 \ (i = 1, \ldots, m), \tag{4.5}$$

$$h_j(x^*) = 0 \ (j = 1, \ldots, r), \tag{4.6}$$

$$u_i^* c_i(x^*) = 0 \ (i = 1, \ldots, m), \tag{4.7}$$

$$u_i^* \geq 0 \ (i = 1, \ldots, m), \tag{4.8}$$

and

$$\nabla_x L(x^*, u^*, w^*) = 0. \tag{4.9}$$

□

Definition 4.14 : The point $(x^{*T}, u^{*T}, w^{*T})^T \in R^s$ where $s = n + m + r$ is a *Kuhn – Tucker* (KT) *point* for $NP.1$ if and only if $4.5 - 4.9$ hold. □

In applying Theorem 4.2 one must be able to determine whether the set Z_2^* is empty. Clearly, assuming that the functions are differentiable, $Z_2^* = \emptyset$ is a necessary and sufficient condition for the existence of the *generalized Lagrange multipliers* u^* and w^*.

Several conditions have been imposed to ensure that the set Z_2^* be empty at a local minimizer. We give three of them in the following discussion. First we state a condition that may be required to hold at a candidate for a local solution of $NP.1$ [FiaM-68a].

Definition 4.15 (First-Order Constraint Qualification) : Let $x^* \in R^n$ be a feasible point for $NP.1$ and suppose that $c_i \in C^1(\hat{D})$ $(i = 1, \ldots, m)$ and $h_j \in C^1(\hat{D})$ $(i = 1, \ldots, r)$ where $\hat{D} \subset D$ is an open set containing the points which satisfy the constraints. The *first order constraint qualification* holds at x^* if and only if $((z \neq 0) \wedge (z^T \nabla c_i^* \geq 0 \ (\forall i \in B^*)) \wedge (z^T \nabla h_j^* = 0 \ (j = 1, \ldots, r))) \Rightarrow$ (z is tangential to a once differentiable arc emanating from x^* and contained in the feasible region). □

There are situations in which $NP.1$ can be solved to give the unique solution x^* but the first-order constraint qualification does not hold at x^* [FiaM-68a].

Theorem 4.3 (Kuhn-Tucker Necessity Theorem) : If (1) $x^* \in R^n$ is a solution of $NP.1$; (2) $f \in C^1(\hat{D})$, $c_i \in C^1(\hat{D})$ $(i = 1, \ldots, m)$, $h_j \in C^1(\hat{D})$ $(j = 1, \ldots, r)$ where $\hat{D} \subset D$ is an open set containing the points which satisfy the constraints; (3) the first-order constraint qualification holds at x^*, then $\exists u^* \in R^m$, $w^* \in R^r$ such that $(x^{*T}, u^{*T}, w^{*T})^T$ is a KT point for $NP.1$. □

The Kuhn-Tucker constraint qualification is one of the best-known conditions implying that $Z_2^* = \emptyset$ or equivalently, implying the existence of the generalized Lagrange Multipliers u^*, w^*.

Neither the first-order constraint qualification nor $Z_2^* = \emptyset$ are usually capable of direct computational verification. A second condition which implies that $Z_2^* = \emptyset$ is contained in the following Theorem [FiaM-68a].

Theorem 4.4 (Interiority-Independence NecessityTheorem) : If (1) $x^* \in R^n$ is a solution of $NP.1$; (2) $f \in C^1(\hat{D})$, $c_i \in C^1(\hat{D})$ $(i = 1, \ldots, m)$, $h_j \in C^1(\hat{D})$ $(j = 1, \ldots, r)$ where $\hat{D} \subset D$ is an open set containing the points which satisfy the constraints; (3) $\exists s \in R^n$ such that $s^T \nabla c_i^* > 0$ $(\forall i \in B^*)$, $s^T \nabla h_j^* = 0$ $(j = 1, \ldots, r)$; (4) $\{\nabla h_1^*, \ldots, \nabla h_r^*\}$ are linearly independent, then $\exists u^* \in R^m$, $w^* \in R^n$ such that $(x^{*T}, u^{*T}, w^{*T})^T$ is a KT point for $NP.1$. □

A final condition implying the existence of Lagrange multipliers u^*, w^* for $NP.1$ is the linear independence of the gradients af all active constraints at a solution x^*. The following theorem contains a stronger result which implies the validity of the first-order constraint qualification.

Theorem 4.5 (Sufficient Conditions for the First-Order Constraint Qualification) : If (1) $x^* \in R^n$ is a feasible point for $NP.1$; (2) $f \in C^1(\hat{D})$, $c_i \in C^1(\hat{D})$ $(i = 1, \ldots, m)$, $h_j \in C^1(\hat{D})$ $(j = 1, \ldots, r)$ where $\hat{D} \subset D$ is an open set containing the points which satisfy the constraints; (3) $\{\nabla c_i^* \mid i \in B^*\} \cup \{\nabla h_j^* \mid j = 1, \ldots, r\}$ is linearly independent, then the first order constraint qualification holds at x^*. □

We cannot say that the first order constraint qualification is both necessary and sufficient for $Z_2^* = \emptyset$ as shown by the examples in [FiaM-68a].

The first order constraint qualification is usually not a useful criterion for testing whether or not $(x^{*T}, u^{*T}, w^{*T})^T$ is a KT point for $NP.1$. The criteria in Theorems 4.4 and 4.5 are more useful because they can be tested. However, it is easy to show that the first order constraint qualification holds for the special case of $NP.1$ the solution of which is considered in this thesis.

The criterion $Z_2^* = \emptyset$ is weaker than the first order constraint qualification because the latter implies the former (Theorem 4.3) but not conversely. Furthermore, the criterion $Z_2^* = \emptyset$ gives an explicit characterization of the described condition directly in terms of the problem functions at a solution x^* of $NP.1$.

The condition $Z_2^* = \emptyset$ is actually the fulfilment of the hypotheses of Farkas' Lemma which are both necessary and sufficient conditions for the existence of Lagrange multipliers. Because of the necessity of $Z_2^* = \emptyset$ it is clear that it cannot be weakened.

Theorem 4.2 is a first order characterization of local minimizers in that it involves first partial derivatives of the problem functions only. It does not take into account the curvature of the problem functions, which is measured by their second partial derivatives. There is a need for curvature analysis to provide a finer characterization of local minimizers in solutions in which the first order necessary conditions fail to give complete information.

Suppose that for the problem *NP*.1 we have $f \in C^2(\hat{D})$, $c_i \in C^2(\hat{D})$ $(i = 1, \ldots, m)$ and $h_j \in C^2(\hat{D})$ $(j = 1, \ldots, r)$. Let $z = (x^T, u^T, w^T)^T$ where $z \in R^s$, $x \in R^n$, $u \in R^m$, $w \in R^r$ and $s = n + m + r$. Then by Definition 4.14 $z^* = (x^{*T}, u^{*T}, w^{*T})^T$ is a KT point if and only if

$$F(z^*) = 0, \tag{4.10}$$

$$c(x^*) \geq 0, \tag{4.11}$$

and

$$u^* \geq 0 \tag{4.12}$$

where $F : R^s \to R^s$, $c : R^n \to R^m$ and $h : R^n \to R^r$ are defined by

$$F(z) = \begin{pmatrix} (f'(x) - u^T c'(x) + w^T h'(x))^T \\ u_1 c_1(x) \\ \vdots \\ u_m c_m(x) \\ h_1(x) \\ \vdots \\ h_r(x) \end{pmatrix} \tag{4.13}$$

and

$$c(x) = (c_1(x), \ldots, c_m(x))^T. \tag{4.14}$$

In 4.13,

$$f'(x) = (\partial_1 f(x), \ldots, \partial_n f(x)), \tag{4.15}$$

$$c'(x) = (\partial_j c_i(x))_{m \times n}, \tag{4.16}$$

and

$$h'(x) = (\partial_j h_i(x))_{r \times n}. \tag{4.17}$$

Consider the following special case of the problem $NP.1$.

$$\left.\begin{array}{ll} \textit{minimize} & f(x)\ (x \in \hat{D} \subseteq D \subseteq R^n) \\ \textit{subject to} & \\ c_i(x) \geq 0 & (i = 1, \ldots, m) \end{array}\right\} NP.2$$

in which

$$c_i(x) = x_i - \hat{x}_{iI}\ (i = 1, \ldots, n) \tag{4.18}$$

and

$$c_i(x) = \hat{x}_{i-nS} - x_{i-n}\ (i = n+1, \ldots, 2n) \tag{4.19}$$

where $\underline{\hat{x}} \in I(\hat{D})$ is given, so that $m = 2n$. The problem of bounding the solutions of $NP.2$ is equivalent to *Problem P*.

The solution of $NP.2$ lies either in the interior of $\underline{\hat{x}}$ or on the boundary of $\underline{\hat{x}}$. For example suppose that $n = 2$. Then

$$c_1(x) = x_1 - \hat{x}_{1I}, \tag{4.20}$$

$$c_2(x) = x_2 - \hat{x}_{2I}, \tag{4.21}$$

$$c_3(x) = \hat{x}_{1S} - x_1, \tag{4.22}$$

and

$$c_4(x) = \hat{x}_{2S} - x_2. \tag{4.23}$$

If $x^* \in int(\underline{\hat{x}})$, then (See 4.1.) $B^* = \emptyset$. Let Z_c be the set of vectors which are tangential to a once differentiable arc emanating from x^* and contained in the feasible region. Since $B^* = \emptyset$ and there are no equality constraints then we have only

$$Z_c = \{z \in R^2 \mid z \neq 0\}.$$

Then $z = (z_1, z_2)^T$ is tangent to the arc $\alpha(\theta) = (x_1^* + \theta z_1, x_2^* + \theta z_2)^T$, which is contained in the feasible region when θ satisfies

$$\left.\begin{aligned} -(x_1^* - \hat{x}_{1I}) \;\leq\; \theta z_1 \;\leq\; \hat{x}_{1S} - x_1^*, \\ -(x_2^* - \hat{x}_{2I}) \;\leq\; \theta z_2 \;\leq\; \hat{x}_{2S} - x_2^*, \end{aligned}\right\} \tag{4.24}$$

for $\alpha(\theta)$ has tangent $D\alpha(\theta)$ where $D\alpha(\theta) = (z_1, z_2)^T$. Therefore the first order constraint qualification holds at x^* for $x^* \in int(\underline{\hat{x}})$.

Suppose that x^* lies on the boundary of $\underline{\hat{x}}$. Then from 4.20-4.23 we obtain

$$\nabla c_1^* = (1,0)^T, \tag{4.25}$$

$$\nabla c_2^* = (0,1)^T, \tag{4.26}$$

$$\nabla c_3^* = (-1,0)^T, \tag{4.27}$$

and

$$\nabla c_4^* = (0,-1)^T. \tag{4.28}$$

Clearly, for this case B^* is a member of the set

$$\{\{1\},\{2\},\{3\},\{4\},\{1,2\},\{1,4\},\{2,3\},\{3,4\}\}. \tag{4.29}$$

Therefore the set $\{\nabla c_i^* \mid i \in B^*\}$ which is obtained by using at most two of 4.25-4.28 according to the elements of 4.29, is linearly independent. Therefore according to Theorem 4.5, the first order constraint qualification holds at x^* for $x^* \in \partial(\underline{\hat{x}})$. A similar argument is valid for $n > 2$.

We conclude that the first order constraint qualification holds at x^* for the problem $NP.2$. Therefore, by Theorem 4.2, $\exists u^* \in R^{2n}$, such that

$$F(z^*) = 0, \tag{4.30}$$

$$c(x^*) \geq 0, \tag{4.31}$$

and

$$u^* \geq 0 \tag{4.32}$$

where $z^* = (x^{*T}, u^{*T})^T$, and $F : R^{3n} \to R^{3n}$, $c : R^n \to R^{2n}$ are defined by

$$F(z) = \begin{pmatrix} (f'(x) - u^T c'(x))^T \\ u_1 c_1(x) \\ \vdots \\ u_{2n} c_{2n}(x) \end{pmatrix}. \tag{4.33}$$

and

$$c(x) = (c_1(x), \ldots, c_{2n}(x))^T. \tag{4.34}$$

Moreover, if we differentiate $F(z)$ with respect to z where $z = (z_1, \ldots, z_{3n})^T = (x_1, \ldots, x_n, u_1, \ldots, u_{2n})^T$ then we obtain

$$\dot{F}'(z) = \left(\begin{array}{c|c|c} \nabla^2 f(z_1, \ldots, z_n) & -I_n & +I_n \\ \hline D^{(1)} & D^{(2)} & 0 \\ \hline D^{(3)} & 0 & D^{(4)} \end{array}\right), \tag{4.35}$$

where

$$\nabla^2 f(z_1, \ldots, z_n) = (\partial_i \partial_j f(z_1, \ldots, z_n))_{n \times n} \qquad 4.36$$

and $D^{(i)} \in M(R^n)$ $(i = 1, \ldots, 4)$ are defined by

$$D^{(1)} = diag(z_{n+1}, \ldots, z_{2n}),$$

$$D^{(2)} = diag(z_1 - \hat{x}_{1I}, \ldots, z_n - \hat{x}_{nI}),$$

$$D^{(3)} = diag(-z_{2n+1}, \ldots, -z_{3n}),$$

and

$$D^{(4)} = diag(\hat{x}_{1S} - z_1, \ldots, \hat{x}_{nS} - z_n).$$

The KT points for $NP.2$ satisfy

$$F(z) = 0, \qquad 4.37$$

$$c(x) \geq 0, \qquad 4.38$$

and

$$u \geq 0. \qquad 4.39$$

Also, the global minimizers of f in $\hat{\underline{x}}$ are KT points for $NP.2$. Furthermore, a KT point for $NP.2$ might correspond to a global maximizer or to a global minimizer of f in $\hat{\underline{x}}$. Therefore an algorithm which bounds the solutions of 4.37-4.39 which correspond to minimizers of f solves *Problem P*. This idea is used to construct the algorithm MW for solving *Problem P* which is described in Chapter 6.

4.2 Strict Complementary Slackness

Definition 4.16 : *Strict Complementary Slackness* is said to hold at a KT point $z^* = (x^{*T}, u^{*T}, w^{*T})^T$ if and only if for $i = 1, \ldots, m$, $u_i^* > 0$ if $c_i(x^*) = 0$ and $u_i^* = 0$ if $c_i(x^*) > 0$. □

That strict complementary slackness does not always hold at a KT point $(x^{*T}, u^{*T})^T$ of $NP.2$ is apparent from the following example. Let $f : R^1 \to R^1$ be defined by

$$f(x) = x^2$$

and let $\hat{\underline{x}} = [0, 1]$. Then by 4.18 and 4.19

$$c_1(x) = x,$$

and

$$c_2(x) = 1 - x,$$

and the solution of $NP.2$ is $x^* = 0$. Therefore $c_1(x^*) = 0$ and $c_2(x^*) > 0$. Now by 4.33 and 4.37,

$$F_1(z) = 2x^* - u_1^* + u_2^*$$

$$= 0,$$

whence $u_1^* = u_2^*$. Also by 4.33 and 4.37,

$$u_2^* c_2(x^*) = 0$$

so $u_2^* = 0$, whence $u_1^* = 0$. Therefore strict complementary slackness does not hold at x^*. The preceding example illustrates part (b) of the following theorem.

Theorem 4.6 : (a) If $x^* \in int(\hat{\underline{x}})$ is a solution of $NP.2$ then strict complementary slackness holds at x^*; (b) if $x^* \in \partial(\hat{\underline{x}})$ is a solution of $NP.2$ then strict complementary slackness holds at x^* if and only if $\partial_i f(x^*) > 0 \; \forall i \in \{1, \ldots, n\}$ such that $x_i^* = \hat{x}_{iI}$ and $\partial_i f(x^*) < 0 \; \forall i \in \{1, \ldots, n\}$ such that $x_i^* = \hat{x}_{iS}$.

Proof : (a) If $x^* \in int(\hat{\underline{x}})$ then $c_i(x^*) > 0$ $(i = 1, \ldots, 2n)$ whence by 4.33 and 4.37 $u_i^* = 0$ $(i = 1, \ldots, 2n)$. Therefore strict complementary slackness holds at x^*.

(b) Suppose that $x_i^* = \hat{x}_{iI}$. Then $\partial_i f(x^*) \geq 0$, for if $\partial_i f(x^*) < 0$ then $Z_2^* \neq \emptyset$ since $e^{(i)} \in Z_2^*$ where $e^{(i)}$ is column i of the $n \times n$ unit matrix. Therefore x^* is not a solution of $NP.2$, contrary to hypothesis. Also if $x_i^* = \hat{x}_{iI}$ then $c_i(x^*) = 0$ and $c_{i+n}(x^*) > 0$, whence by 4.33 and 4.37, $u_{i+n}^* = 0$.

If strict complementary slackness holds at x^* then $u_i^* > 0$. Therefore

$$\partial_i f(x^*) = u_i^* - u_{i+n}^*$$

$$> 0.$$

Conversely, if $\partial_i f(x^*) > 0$ then

$$u_i^* = u_{i+n}^* + \partial_i f(x^*)$$

$$> 0,$$

whence strict complementary slackness holds at x^*. A similar argument is valid when $x_i^* = \hat{x}_{iS}$. □

Definition 4.17 : Let $\underline{a} = [a_I, a_S] \in I(R)$ be given. Then

$$\underline{a}^0 = \begin{cases} [a_I, 0] & (a_S \leq 0) \\ \underline{a} & (a_S > 0). \end{cases} \quad \square \tag{4.40}$$

Theorem 4.7 : Let $\underline{a}, \underline{b} \in I(R)$. Then $(\underline{a} \subseteq \underline{b}) \Rightarrow (\underline{a}^0 \subseteq \underline{b}^0)$.

Proof : Now

$$(\underline{a} \subseteq \underline{b}) \Longleftrightarrow (b_I \leq a_I \leq a_S \leq b_S). \tag{4.41}$$

Suppose that $b_S \leq 0$. Then by 4.41 $a_S \leq 0$, so by 4.40, $\underline{a}^0 = [a_I, 0]$ and $\underline{b}^0 = [b_I, 0]$ whence by 4.41, $\underline{a}^0 \subseteq \underline{b}^0$. Suppose that $b_S > 0$. Then by 4.40, $\underline{b}^0 = \underline{b}$. If $a_S > 0$ then by 4.40 $\underline{a}^0 = \underline{a} \subseteq \underline{b} = \underline{b}^0$. If $a_S \leq 0$ then by 4.40, $\underline{a}^0 = [a_I, 0] \subset \underline{b} = \underline{b}^0$. So $(\underline{a} \subseteq \underline{b} \wedge b_S \leq 0) \Rightarrow (\underline{a}^0 \subseteq \underline{b}^0)$, and $(\underline{a} \subseteq \underline{b} \wedge b_S > 0) \Rightarrow (\underline{a}^0 \subseteq \underline{b}^0)$. Therefore $(\underline{a} \subseteq \underline{b}) \Rightarrow (\underline{a}^0 \subseteq \underline{b}^0)$. $\square$

Theorem 4.8 : If $\underline{a}, \underline{b}, \underline{c}, \underline{d} \in I(R)$, $\underline{c} \subseteq \underline{a}$, $\underline{d} \subseteq \underline{b}$ and $0 \notin \underline{a}^0 \cap \underline{b}^0$ then $0 \notin \underline{c}^0 \cap \underline{d}^0$.

Proof : By Theorem 4.7 $(\underline{c} \subset \underline{a}) \Rightarrow (\underline{c}^0 \subseteq \underline{a}^0)$ and $(\underline{d} \subseteq \underline{b}) \Rightarrow (\underline{d}^0 \subseteq \underline{b}^0)$. Therefore $\underline{c}^0 \cap \underline{d}^0 \subseteq \underline{a}^0 \cap \underline{b}^0$. Therefore $(0 \notin \underline{a}^0 \cap \underline{b}^0) \Rightarrow (0 \notin \underline{c}^0 \cap \underline{d}^0)$. $\square$

In order to obtain the KT point $z^* = (x^{*T}, u^{*T})^T$ for $NP.2$ we solve the system of equations $F(z) = 0$ where $F(z)$ is given by 4.33.

Let $F : R^{3n} \rightarrow R^{3n}$ be defined by 4.33 and let $\underline{\hat{z}} = (\underline{\hat{x}}^T, \underline{\hat{u}}^T)^T$ be given, where $\underline{\hat{x}} \in I(R^n)$ is the box in which a global minimizer x^* of $f : R^n \rightarrow R^1$ is sought, and

$\underline{\hat{u}} \in I(R^{2n})$ is a box containing the Lagrange multiplier values $u^* \in R^{2n}$ corresponding to the global minimizer x^*.

Let $c_i : R^n \to R^1$ $(i = 1, \ldots, 2n)$ be defined by 4.18 and 4.19, and let $\underline{c}_i : I(R^n) \to I(R^1)$ $(i = 1, \ldots, 2n)$ be inclusion monotonic interval extensions of $c_i : R^n \to R^1$ $(i = 1, \ldots, 2n)$ respectively. If

$$0 \notin \underline{c}_i(\underline{\hat{x}})^0 \cap \underline{\hat{u}}_i^0 \quad (i = 1, \ldots, 2n) \tag{4.42}$$

and $\underline{z} = (\underline{x}^T, \underline{u}^T)^T \subseteq \underline{\hat{z}}$ then by **Theorem 4.8**

$$0 \notin \underline{c}_i(\underline{x})^0 \cap \underline{u}_i^0 \quad (i = 1, \ldots, 2n). \tag{4.43}$$

Therefore if we could generate a sequence $(\underline{z}^{(k)})$ such that $\underline{z}^{(k)} = (\underline{x}^{(k)^T}, \underline{u}^{(k)^T})^T \in I(R^{3n})$ and $\underline{z}^{(k+1)} \subseteq \underline{z}^{(k)}$ $(\forall k \geq 0)$ where $\underline{z}^{(0)} = (\underline{\hat{x}}^T, \underline{\hat{u}}^T)^T$ and 4.42 holds then $(\forall k \geq 0)$

$$0 \notin \underline{c}_i(\underline{x}^{(k)})^0 \cap \underline{u}_i^{(k)o} \quad (i = 1, \ldots, 2n). \tag{4.44}$$

Therefore, if 4.42 holds and $\underline{z}^{(k)} \to z^*$ $(k \to \infty)$ where $F(z^*) = 0$ and $z^* =$

$(x^{*T}, u^{*T})^T$ then strict complementary slackness holds at z^* and z^* satisfies the Kuhn-Tucker conditions. This idea is used in the algorithm MW which is described in Chapter 6.

4.3 Robinson's Theorem

If for $NP.2$ $f \in C^2(\hat{D})$ and $c_i \in C^2(\hat{D})$ $(i = 1, \ldots, 2n)$ then $F(z) = 0$, where $F(z)$ is given by 4.33, can be solved by using the Newton operator $N : R^{3n} \to R^{3n}$ [Nic–71a], which is defined by

$$N(z) = z - (F'(z))^{-1}F(z), \tag{4.45}$$

where $F'(z)$ is given by 4.35.

Let $\underline{F} : I(\Omega) \to I(R^{3n})$ be a continuous inclusion monotonic interval extension of $F : \Omega \subseteq R^{3n} \to R^{3n}$. The interval Newton operator $\underline{N} : I(\Omega) \times \Omega \to I(R^{3n})$ [Nic–71a] is defined by

$$\underline{N}(\underline{z}, z) = z - (\underline{F'}(\underline{z}, z))^{-1}F(z) \ (z \in \underline{z}) \tag{4.46}$$

where $\underline{F'} : I(\Omega) \times \Omega \to I(M(R^{3n}))$ is a continuous inclusion monotonic interval extension of $F' : \Omega \times \Omega \to M(R^{3n})$,

$$(\underline{F}'(\underline{z}, z))_{ij} = \partial_j \underline{F}_i(\underline{z}_1, \ldots, \underline{z}_{j-1}, \underline{z}_j, z_{j+1}, \ldots, z_{3n}) \ (i, j = 1, \ldots, 3n), \qquad 4.47$$

and $(\underline{F}'(\underline{z}, z))^{-1} \in I(M(R^{3n}))$ is such that

$$(\underline{F}'(\underline{z}, z))^{-1} \supseteq \{(F'(y, z))^{-1} \mid y \in \underline{z}\},$$

in which $F' : \Omega \times \Omega \to M(R^{3n})$ is defined by

$$(F'(z, y))_{ij} = \partial_j F_i(z_1, \ldots, z_{j-1}, z_j, y_{j+1}, \ldots, y_{3n}) \ (i, j = 1, \ldots, 3n).$$

Note that $\underline{N}$ is not uniquely defined because it contains an interval inverse.

The following result is a special case of a theorem which has been proved by Robinson [Rob–73a] for problem *NP*.1.

Theorem 4.9 : If (1) $f : D \subset R^n \to R^1$, $c : D \subset R^n \to R^{2n}$ are given mappings with $f \in C^2(\hat{D})$, $c \in C^2(\hat{D})$ where $\hat{D} \subset D$ is an open convex set; (2) $\underline{z} \in I(\hat{D})$, $\underline{u} \in I(R^{2n})$ are given; (3) $\underline{c} : I(D) \to I(R^{2n})$ is a continuous inclusion monotonic interval extension of $c : D \to R^{2n}$; (4) $\underline{F} : I(\Omega) \to I(R^{3n})$ is a continuous inclusion

monotonic interval extension of $F : \Omega \to R^{3n}$ where F is given by 4.33 and $\Omega = D \times R^{2n}$; (5) $\underline{F}' : I(\Omega) \to I(M(R^{3n}))$ is a continuous inclusion monotonic interval extension of $F' : \hat{\Omega} \to M(R^{3n})$ where F' is given by 4.35, $\hat{\Omega} = \hat{D} \times R^{2n}$, $\underline{F}'(\underline{z}, z)$ is nonsingular, and $\underline{z} = (\underline{x}^T, \underline{u}^T)^T$; (6) $0 \notin \underline{c}_i(\underline{x})^0 \cap \underline{u}_i^0$ $(i = 1, \ldots, 2n)$; (7) $\exists \tilde{z} \in \underline{z}$ such that $\underline{N}(\underline{z}, \tilde{z}) \subseteq \underline{z}$, then $\underline{N}(\underline{z}, \tilde{z})$ contains a KT point $z^* = (x^{*T}, u^{*T})^T$ of $NP.2$ at which strict complementary slackness holds with linear independence of the gradients to the active constraints. Furthermore, z^* is the only KT point of the problem $NP.2$ in $\underline{z}$. □

Robinson [Rob–73a] has used the more general form of **Theorem 4.9** to bound a KT point z^* of $NP.1$ given an initial (small) box containing z^*.

In algorithm MW the operators which are described in [SheW-85c] and [SheW-85b] are used instead of the operator $\underline{N}$ to obtain arbitrarily sharp bounds on a KT point z^* for $NP.2$, given a box $\hat{\underline{z}}$, which need not be small, such that $z^* \in \hat{\underline{z}}$.

CHAPTER 5

The Algorithms *MAP* and *KMSW*

In this chapter, the algorithms *MAP* [SheW-85b] and *KMSW* [SheW-85c] are described. These algorithms are used in the algorithm *MW* as explained in Chapter 6.

5.1 The Algorithm *MAP*

Let $A \in M(R^t)$ and $\underline{b} \in I(R^t)$ be given. The result of applying the Gauss algorithm [AleH-83a][AleP-83a] to the pair $(A, \underline{b})$ is represented by $\underline{g}(A, \underline{b}) \in I(R^t)$. If A^{-1} exists then $\exists M_A \in M(R^t)$ depending only on A, such that

$$w(\underline{g}(A, \underline{b})) = M_A w(\underline{b}) \tag{5.1}$$

and

$$A^{-1}\underline{b} \subseteq \underline{g}(A, \underline{b}). \tag{5.2}$$

Let $\underline{K}_N : I(R^t) \times I(M(R^t)) \times M(R^t) \rightarrow I(R^t)$ be defined by

$$\underline{K}_N(\underline{z}, \underline{G}, A) = m(\underline{z}) - \underline{g}(A, \{F(m(\underline{z})) - (A - \underline{G})(\underline{z} - m(\underline{z}))\}), \tag{5.3}$$

in which $F : D \subseteq R^t \to R^t$ is a given mapping with $F \in C^1(\hat{D})$ where $\hat{D} \subseteq D$ is an open convex set and $t \in N$. Let $\underline{F} : I(\hat{D}) \to I(R^t)$ and $\underline{F}' : I(\hat{D}) \to I(M(R^t))$ be continuous inclusion monotonic interval extensions of $F : \hat{D} \to R^t$ and of $F' : \hat{D} \to M(R^t)$ respectively.

Let $\hat{\underline{z}} \in I(\hat{D})$ be given. The algorithm MAP [SheW-85b] for bounding a zero of F in $\hat{\underline{z}}$ is as follows.

Algorithm MAP

Data : $\hat{\underline{z}} \in I(\hat{D})$, $\alpha \in [0,1)$, t, $p^{(k)} \in N$ $(k \geq 0)$.

1. $\underline{z}^{(0)} := \hat{\underline{z}}$
2. $\underline{F}'^{(0)} := \underline{F}'(\underline{z}^{(0)})$
3. $B^{(0)} := m(\underline{F}'^{(0)})$
4. $\underline{z}^{(0,0)} := \underline{z}^{(0)}$
5. *for* $m = 0$ *to* $p^{(0)}$ *do*
   5.1. $\underline{z}^{(0,m+1)} := \underline{K}_N(\underline{z}^{(0,m)}, \underline{F}'^{(0)}, B^{(0)}) \cap \underline{z}^{(0,m)}$
6. $\underline{z}^{(1)} := \underline{z}^{(0,p^{(0)}+1)}$
7. $k := 1$
8. *while true do*
   8.1. $\underline{F}'^{(k)} := \underline{F}'(\underline{z}^{(k)})$

8.2. $A^{(k)} := m(\underline{F}'^{(k)})$

8.3. $\underline{u}^{(k)} := \underline{K}_N(\underline{z}^{(k)}, \underline{F}'^{(k)}, A^{(k)})$

8.4. $\underline{z}^{(k)} := \underline{u}^{(k)} \cap \underline{z}^{(k)}$

8.5. *if* $w(\underline{u}^{(k)}) \leq \alpha w(\underline{z}^{(k)})$

then

8.5.1. $B^{(k)} := A^{(k)}$

8.5.2. $\underline{z}^{(k,1)} := \underline{x}^{(k)}$

8.5.3. *for* $m = 1$ *to* $p^{(k)}$ *do*

8.5.3.1. $\underline{u}^{(k,m)} := \underline{K}_N(\underline{z}^{(k,m)}, \underline{F}'^{(k)}, A^{(k)})$

8.5.3.2. $\underline{z}^{(k,m+1)} := \underline{u}^{(k,m)} \cap \underline{z}^{(k,m)}$

8.5.4. $\underline{z}^{(k+1)} := \underline{z}^{(k,p^{(k)}+1)}$

else

8.5.5. $B^{(k)} := B^{(k-1)}$

8.5.6. $\underline{v}^{(k)} := \underline{K}_N(\underline{z}^{(k)}, \underline{F}'^{(k)}, B^{(k)})$

8.5.7. $\underline{z}^{(k,1)} := \underline{v}^{(k)} \cap \underline{x}^{(k)}$

8.5.8. *for* $m = 1$ *to* $p^{(k)}$ *do*

8.5.8.1. $\underline{v}^{(k,m)} := \underline{K}_N(\underline{z}^{(k,m)}, \underline{F}'^{(k)}, A^{(k)})$

8.5.8.2. $\underline{z}^{(k,m+1)} := \underline{v}^{(k,m)} \cap \underline{z}^{(k,m)}$

8.5.9. $\underline{z}^{(k+1)} := \underline{z}^{(k,p^{(k)}+1)}$

8.6. $k := k + 1$ □

The determination of $p^{(k)}$ is described in §5.3.

5.2 Theoretical Results for MAP

This section contains proofs of the existence, uniqueness, and convergence results for MAP which are given in [SheW-85b].

Lemma 5.1 : (i) If $\underline{z}$, $\underline{y} \in I(R^t)$ and $\beta > 0$ are such that $w(\underline{y}) \leq \beta w(\underline{z})$, then $(\forall \underline{G} \in I(M(R^t)))\ (\forall A \in M(R^t))$

$$w(\underline{K}_N(\underline{y}, \underline{G}, A)) \leq \beta w(K_N(\underline{z}, \underline{G}, A)).$$

(ii) If $\underline{U}$, $\underline{V} \in I(M(R^t))$ are such that $\underline{U} \subseteq \underline{V}$, then $(\forall \underline{z} \in I(R^t))\ (\forall A \in M(R^t))$

$$w(\underline{K}_N(\underline{z}, \underline{U}, A)) \leq w(\underline{K}_N(\underline{z}, \underline{V}, A)).$$

Proof : (i) By 5.1,

$$\underline{K}_N(\underline{y}, \underline{G}, A) = m(\underline{y}) - \underline{g}(A, \{F(m(\underline{y})) - (A - \underline{G})(\underline{y} - m(\underline{y}))\})$$

$$\Rightarrow w(\underline{K}_N(\underline{y}, \underline{G}, A)) = w(\underline{g}(A, \{F(m(\underline{y})) - (A - \underline{G})(\underline{y} - m(\underline{y}))\}))$$

$$= M_A w(F(m(\underline{y})) - (A - \underline{G})(\underline{y} - m(\underline{y})))$$

$$= M_A \mid A - \underline{G} \mid w(\underline{y}).$$

So if $w(\underline{y}) \leq \beta w(\underline{z})$ then

$$w(\underline{K}_N(\underline{y}, \underline{G}, A)) \leq M_A \mid A - \underline{G} \mid \beta w(\underline{z})$$

$$= \beta w(\underline{K}_N(\underline{z}, \underline{G}, A)).$$

(ii)

$$w(\underline{K}_N(\underline{z}, \underline{U}, A)) = M_A \mid A - \underline{U} \mid w(\underline{z})$$

$$\leq M_A \mid A - \underline{V} \mid w(\underline{z})$$

$$= w(\underline{K}_N(\underline{z}, \underline{V}, A)),$$

since

$$(\underline{U} \subseteq \underline{V}) \Rightarrow (A - \underline{U} \subseteq A - \underline{V})$$

$$\Rightarrow (\mid A - \underline{U} \mid \leq \mid A - \underline{V} \mid).$$

Lemma 5.2 : (i) For each $\underline{G} \in I(M(R^t))$ $\exists U_G \in M(R^t)$ such that $(\forall \underline{z} \in I(R^t))$

$$w(\underline{K}_N(\underline{z}, \underline{G}, m(\underline{G}))) \leq U_G w(\underline{G}) w(\underline{z}).$$

(ii) Let $\underline{V} \in I(M(R^t))$ be such that $(V \in \underline{V}) \Rightarrow (\exists V^{-1})$. Then $\exists U \in M(R^t)$ such that $(\forall \underline{G} \subseteq \underline{V})$ $(\forall \underline{z} \in I(R^t))$

$$w(\underline{K}_N(\underline{z}, \underline{G}, m(\underline{G}))) \leq U\, w(\underline{G}) w(\underline{z}).$$

Proof: (i) By 5.1, $\exists M_G \in M(R^t)$ such that

$$w(\underline{K}_N(\underline{z}, \underline{G}, m(\underline{G}))) = M_G \mid m(\underline{G}) - \underline{G} \mid w(\underline{z})$$

$$= \frac{1}{2} M_G w(\underline{G}) w(\underline{z}).$$

Set $U_G = \frac{1}{2} M_G$.

(ii) Since $(V \in \underline{V}) \Rightarrow (\exists V^{-1})$, $\underline{g}(V, \underline{b})$ exists $(\forall\, V \in \underline{V})$ $(\forall \underline{b} \in I(R^t))$. So $\underline{K}_N(\underline{z}, \underline{G}, m(\underline{G}))$ exists $(\forall \underline{G} \subseteq \underline{V})$, $(\forall \underline{z} \in I(R^t))$ and by (i) $\exists U_G \in M(R^t)$ such that

$$w(\underline{K}_N(\underline{z}, \underline{G}, m(\underline{G}))) \leq U_G w(\underline{G}) w(\underline{z}),$$

and U_G depends only on $m(\underline{G})$. Now $\underline{V}$ is a compact set in $M(R^t)$ and $(\underline{G} \subseteq \underline{V}) \Rightarrow (m(\underline{G}) \in \underline{V})$. Since U_G depends continuously on $m(\underline{G})$ it follows that U_G attains a maximum value U on $\underline{V}$. Therefore $(\forall \underline{G} \subseteq \underline{V})$ $(\forall \underline{z} \in I(R^t))$

$$w(\underline{K}_N(\underline{z}, \underline{G}, m(\underline{G}))) \leq U_G w(\underline{G}) w(\underline{z})$$

$$\leq U\, w(\underline{G}) w(\underline{z}). \quad \square$$

Lemma 5.3 : Suppose that $\exists \lambda > 0$ such that $(\forall \underline{z} \subseteq \hat{\underline{z}} \in I(\hat{D}))$

$$\|w(\underline{F}'(\underline{z}))\| \leq \lambda \|w(\underline{z})\| .$$

Then $\exists \mu > 0$ such that $(\forall \underline{z} \subseteq \hat{\underline{z}} \in I(\hat{D}))$

$$\|w(\underline{K}_N(\underline{z}, \underline{F}'(\underline{z}), m(\underline{F}'(\underline{z}))))\| \leq \mu \|w(\underline{z})\|^2 .$$

Proof : By **Lemma 5.2** (ii), $\exists U \in M(R^t)$ such that $(\forall \underline{z} \subseteq \hat{\underline{z}})$

$$w(\underline{K}_N(\underline{z}, \underline{F}'(\underline{z}), m(\underline{F}'(\underline{z})))) \leq U w(\underline{F}'(\underline{z}))w(\underline{z})$$

since by inclusion monotonicity of $\underline{F}'$,

$$(\underline{z} \subseteq \hat{\underline{z}}) \Rightarrow (\underline{F}'(\underline{z}) \subseteq \underline{F}'(\hat{\underline{z}}))$$

$$\Rightarrow (m(\underline{F}'(\underline{z})) \in \underline{F}'(\hat{\underline{z}})).$$

So

$$\|w(\underline{K}_N(\underline{z}, \underline{F}'(\underline{z}), m(\underline{F}'(\underline{z}))))\| \leq \|U w(\underline{F}'(\underline{z}))w(\underline{z})\|$$

$$\leq \|U\| \left\|w(\underline{F}'(\underline{z}))\right\| \|w(\underline{z})\|$$

$$\leq \lambda \|U\| \|w(\underline{z})\|^2 .$$

Set $\mu = \lambda \|U\|$. □

Theorem 5.1 : Suppose that (1) $F : D \subseteq R^t \to R^t$ is a given mapping with $F \in C^1(\hat{D})$ where $\hat{D} \subseteq D$ is an open convex set; (2) $\underline{F} : I(\hat{D}) \to I(R^t)$ and $\underline{F}' : I(\hat{D}) \to I(M(R^t))$ are continuous inclusion monotonic interval extensions of $F : \hat{D} \to R^t$ and of $F' : \hat{D} \to M(R^t)$ respectively; (3) $\underline{z}^{(0)} \in I(\hat{D})$ is such that $w(\underline{z}^{(0)}) > 0$; (4) $B^{(0)} = m(\underline{F}'(\underline{z}^{(0)}))$ is nonsingular; (5) $\exists \alpha \in [0, 1)$ such that

$$w(\underline{K}_N(\underline{z}^{(0)}, \underline{F}'(\underline{z}^{(0)}), B^{(0)})) \leq \alpha w(\underline{z}^{(0)}). \qquad 5.4$$

(a) If $\underline{z}^{(0)}$ contains a zero z^* of F, then the algorithm MAP is well defined and the sequence $(\underline{z}^{(k)})$ converges to z^*. (b) If $\exists \lambda > 0$ such that $\left\|w(\underline{F}'(\underline{z}))\right\| \leq \lambda \|w(\underline{z})\|$ $(\forall \underline{z} \in I(\hat{D}))$ then $\exists \mu^{(k)} > 0$ such that

$$\left\|w(\underline{z}^{(k+1)})\right\| \leq \mu^{(k)} \left\|w(\underline{z}^{(k)})\right\|^{2+p^{(k)}} \quad (k \geq 0). \qquad 5.5$$

(c) If MAP terminates because of an empty intersection then there is no zero of F in $\underline{z}^{(0)}$.

Proof : (a) By Lemma 2.9, $\{A^{(k)}\}^{-1}$ exists $(\forall k \geq 1)$ so MAP is well defined provided that no empty intersections occur. If $\underline{z}^{(k)}$ contains a zero z^* of F, then (See

2.6.) $z^* \in \underline{K}(\underline{z}^{(k)}, \{A^{(k)}\}^{-1}) \subseteq \underline{K}_N(\underline{z}^{(k)}, \underline{F}'^{(k)}, A^{(k)}) = \underline{u}^{(k)}$, so $z^* \in \underline{z}^{(k)} \neq \emptyset$. If $w(\underline{u}^{(k)}) \leq \alpha w(\underline{z}^{(k)})$ then $z^* \in \underline{z}^{(k,1)} = \underline{x}^{(k)}$ and by finite induction on m, $z^* \in \underline{z}^{(k,m)}$ $(m = 1, \ldots, p^{(k)} + 1)$, whence $z^* \in \underline{z}^{(k+1)} \neq \emptyset$. If $w(\underline{u}^{(k)}) > \alpha w(\underline{z}^{(k)})$ then $(z^* \in \underline{z}^{(k)}) \Rightarrow (z^* \in \underline{K}(\underline{z}^{(k)}, \{B^{(k)}\}^{-1}) \subseteq \underline{K}_N(\underline{z}^{(k)}, \underline{F}'^{(k)}, B^{(k)}) = \underline{v}^{(k)})$ and $z^* \in \underline{x}^{(k)}$ so $z^* \in \underline{z}^{(k,1)} \neq \emptyset$.

By finite induction on m, $z^* \in \underline{z}^{(k,m)}$ $(m = 1, \ldots, p^{(k)} + 1)$ so $z^* \in \underline{z}^{(k+1)} \neq \emptyset$. So by induction on k, if $z^* \in \underline{z}^{(0)}$ then $z^* \in \underline{z}^{(k)} \neq \emptyset$ $(\forall k \geq 0)$.

We shall show that $(\forall k \geq 0)$

$$w(\underline{z}^{(k)}) \leq \alpha^k w(\underline{z}^{(0)}). \qquad 5.6$$

By 5.4

$$w(\underline{z}^{(0,1)}) \leq w(\underline{K}_N(\underline{z}^{(0)}, \underline{F}'^{(0)}, B^{(0)}))$$

$$\leq \alpha w(\underline{z}^{(0)}).$$

Furthermore, for $m = 0, \ldots, p^{(0)}$, $\underline{z}^{(0,m+1)} \subseteq \underline{z}^{(0,m)}$, whence $w(\underline{z}^{(1)}) = w(\underline{z}^{(0,p^{(0)}+1)}) \leq \alpha w(\underline{z}^{(0)})$. Suppose that for some $k \geq 1$, 5.6 holds. If $w(\underline{u}^{(k)}) \leq \alpha w(\underline{z}^{(k)})$ then

$$w(\underline{z}^{(k,1)}) = w(\underline{x}^{(k)})$$

$$\leq w(\underline{u}^{(k)})$$

$$\leq \alpha w(\underline{z}^{(k)})$$

so because for $m = 0, \ldots, p^{(k)}$, $\underline{z}^{(k,m+1)} \subseteq \underline{z}^{(k,m)}$, we have $w(\underline{z}^{(k+1)}) = w(\underline{z}^{(k,p^{(k)}+1)})$ $\leq \alpha w(\underline{z}^{(k)})$.

If $w(\underline{u}^{(k)}) > \alpha w(\underline{z}^{(k)})$ then

$$w(\underline{z}^{(k,1)}) \leq w(\underline{v}^{(k)})$$

$$= w(\underline{K}_N(\underline{z}^{(k)}, \underline{F}'^{(k)}, B^{(k)})).$$

Now $\exists \tilde{k}$ such that $0 < \tilde{k} \leq k-1$ and $B^{(k)} = A^{(\tilde{k})} = m(\underline{F}'^{(\tilde{k})})$ and by step 8.5 of *MAP*,

$$w(\underline{K}_N(\underline{z}^{(\tilde{k})}, \underline{F}'^{(\tilde{k})}, A^{(\tilde{k})})) \leq \alpha w(\underline{z}^{(\tilde{k})}). \qquad 5.7$$

Therefore $w(\underline{z}^{(\tilde{k},1)}) \leq \alpha w(\underline{z}^{(\tilde{k})})$ whence because for $m = 0, \ldots, p^{(\tilde{k})}$, $\underline{z}^{(\tilde{k},m+1)} \subseteq$ $\underline{z}^{(\tilde{k},m)}$, we have $w(\underline{z}^{(\tilde{k}+1)}) \leq \alpha w(\underline{z}^{(\tilde{k})})$.

We shall show that for $j = 0, \ldots, k - \tilde{k}$,

$$w(\underline{z}^{(\tilde{k}+j+1)}) \leq \alpha^{j+1} w(\underline{z}^{(\tilde{k})}). \tag{5.8}$$

We have shown that 5.8 holds for $j = 0$. Suppose that 5.8 holds for some $j \geq 0$. Then by step 8.5.6 of MAP, since $B^{(\tilde{k}+j+1)} = B^{(k)}$ $(j = 0, \ldots, k - \tilde{k})$ we have

$$w(\underline{v}^{(\tilde{k}+j+1)}) = w(\underline{K}_N(\underline{z}^{(\tilde{k}+j+1)}, \underline{F}'^{(\tilde{k}+j+1)}, B^{(\tilde{k}+j+1)}))$$

$$= w(\underline{K}_N(\underline{z}^{(\tilde{k}+j+1)}, \underline{F}'^{(\tilde{k}+j+1)}, B^{(k)}))$$

$$\leq w(\underline{K}_N(\underline{z}^{(\tilde{k}+j+1)}, \underline{F}'^{(\tilde{k})}, A^{(\tilde{k})}))$$

$$\leq \alpha^{j+1} w(\underline{K}_N(\underline{z}^{(\tilde{k})}, \underline{F}'^{(\tilde{k})}, A^{(\tilde{k})}))$$

$$\leq \alpha^{j+2} w(\underline{z}^{(\tilde{k})}).$$

So by finite induction on j, 5.8 holds for $j = 0, \ldots, k - \tilde{k}$. Therefore by 5.8 with $j = k - \tilde{k}$, we have

$$w(\underline{z}^{(k+1)}) \leq \alpha^{k-\tilde{k}+1} w(\underline{z}^{(\tilde{k})}). \tag{5.9}$$

By 5.6 with $k = \tilde{k}$, we have

$$w(\underline{z}^{(\tilde{k})}) \leq \alpha^{\tilde{k}} w(\underline{z}^{(0)}).$$

So by 5.9

$$w(\underline{z}^{(k+1)}) \leq \alpha^{k+1} w(\underline{z}^{(0)}).$$

Therefore by induction on k,

$$w(\underline{z}^{(k)}) \leq \alpha^{k} w(\underline{z}^{(0)}) \ (\forall k \geq 1).$$

Therefore since $\alpha \in [0,1)$, it follows that $w(\underline{z}^{(k)}) \to 0$ $(k \to \infty)$. Therefore $\underline{z}^{(k)} \to z^*$ $(k \to \infty)$ because $z^* \in \underline{z}^{(k)}$ $(\forall k \geq 0)$.

(b) If $w(\underline{u}^{(k)}) \leq \alpha w(\underline{z}^{(k)})$ then

$$w(\underline{z}^{(k,1)}) = w(\underline{x}^{(k)})$$

$$\leq w(\underline{u}^{(k)})$$

$$= w(\underline{K}_N(\underline{z}^{(k)}, \underline{F}'^{(k)}, A^{(k)})),$$

whence by Lemma 5.3,

$$\left\| w(\underline{z}^{(k,1)}) \right\| \leq \mu \left\| w(\underline{z}^{(k)}) \right\|^2 .$$

By step 8.5.3 of MAP, for $m = 1, \ldots, p^{(k)}$,

$$w(\underline{z}^{(k,m+1)}) \leq w(\underline{u}^{(k,m)})$$

$$= w(\underline{K}_N(\underline{z}^{(k,m)}, \underline{F}'^{(k)}, A^{(k)}))$$

$$\leq M_{A^{(k)}} \mid \underline{F}'^{(k)} - A^{(k)} \mid w(\underline{z}^{(k,m)})$$

$$\leq \frac{1}{2} M_{A^{(k)}} w(\underline{F}'^{(k)}) w(\underline{z}^{(k,m)}),$$

whence

$$\left\| w(\underline{z}^{(k,m+1)}) \right\| \leq \frac{\lambda}{2} \| M_{A^{(k)}} \| \left\| w(\underline{z}^{(k)}) \right\| \left\| w(\underline{z}^{(k,m)}) \right\| .$$

So

$$\left\| w\left(\underline{z}^{(k+1)}\right) \right\| = \left\| w\left(\underline{z}^{(k,p^{(k)}+1)}\right) \right\|$$

$$\leq \left(\frac{\lambda}{2} \left\| M_{A^{(k)}} \right\| \left\| w\left(\underline{z}^{(k)}\right) \right\| \right)^{p^{(k)}} \left\| w\left(\underline{z}^{(k,1)}\right) \right\|$$

$$\leq \left(\frac{\lambda}{2} \left\| M_{A^{(k)}} \right\| \right)^{p^{(k)}} \mu \left\| w\left(\underline{z}^{(k)}\right) \right\|^{p^{(k)}+2}.$$

Let

$$\mu^{(k)} = \left(\frac{\lambda}{2} \left\| M_{A^{(k)}} \right\| \right)^{p^{(k)}} \mu.$$

Then

$$\left\| w\left(\underline{z}^{(k+1)}\right) \right\| \leq \mu^{(k)} \left\| w\left(\underline{z}^{(k)}\right) \right\|^{p^{(k)}+2}.$$

If

$$w\left(\underline{u}^{(k)}\right) > \alpha w\left(\underline{z}^{(k)}\right)$$

then

$$w\left(\underline{z}^{(k,1)}\right) \leq w\left(\underline{x}^{(k)}\right)$$

whence as before

$$\left\| w\left(\underline{z}^{(k,1)}\right) \right\| \leq \mu \left\| w\left(\underline{z}^{(k)}\right) \right\|^{2}.$$

By step 8.5.8 of MAP, for $m = 1, \ldots, p^{(k)}$

$$w(\underline{z}^{(k,m+1)}) \leq w(\underline{v}^{(k,m)})$$

$$= w(\underline{K}_N(\underline{z}^{(k,m)}, \underline{F}'^{(k)}, A^{(k)})),$$

whence as before

$$\left\| w(\underline{z}^{(k,m+1)}) \right\| \leq \frac{\lambda}{2} \|M_{A^{(k)}}\| \left\| w(\underline{z}^{(k)}) \right\| \left\| w(\underline{z}^{(k,m)}) \right\|,$$

and so

$$\left\| w(\underline{z}^{(k+1)}) \right\| \leq \mu^{(k)} \left\| w(\underline{z}^{(k)}) \right\|^{p^{(k)}+2}.$$

Therefore $(\forall k \geq 0)$, 5.5 holds.

(c) Finally, if MAP terminates because of an empty intersection then for some $k \geq 0$, $\underline{z}^{(k)} = \emptyset$. But $(\exists z^* \in \underline{z}^{(0)} \wedge f(z^*) = 0) \Rightarrow (z^* \in \underline{z}^{(k)}\ (\forall k \geq 0))$. Therefore if MAP terminates because of an empty intersection then $\not\exists z^* \in \underline{z}^{(0)}$ such that $f(z^*) = 0$. □

Corollary 5.1 : If the hypotheses of the **Theorem 5.1** are valid save that instead of 5.4,

$$\underline{K}_N(\underline{z}^{(0)}, \underline{F}'(\underline{z}^{(0)}), B^{(0)}) \subset int(\underline{z}^{(0)}), \qquad 5.10$$

then $\exists z^* \in \underline{z}^{(0)}$ such that $F(z^*) = 0$, z^* is unique in $\underline{z}^{(0)}$, and conclusions of Theorem 5.1 hold.

Proof: Let $\underline{a} = \underline{K}_N(\underline{z}^{(0)}, \underline{F}'(\underline{z}^{(0)}), B^{(0)})$, and $\underline{b} = \underline{z}^{(0)}$. Then by 5.10,

$$\underline{a} \subset int(\underline{b})$$

$$\Rightarrow w(\underline{a}) < w(\underline{b})$$

$$\Rightarrow w(\underline{a}_i) < w(\underline{b}_i) \ (i = 1, \ldots, t).$$

Furthermore by 5.10, $w(\underline{b}_i) > 0$ $(i = 1, \ldots, t)$. Let

$$\alpha = \max_{1 \le i \le t} \{\frac{w(\underline{a}_i)}{w(\underline{b}_i)}\}.$$

Then $\alpha < 1$ and for $i = 1, \ldots, t$

$$\frac{w(\underline{a}_i)}{w(\underline{b}_i)} \le \max_{1 \le i \le t} \{\frac{w(\underline{a}_i)}{w(\underline{b}_i)}\} = \alpha.$$

So $w(\underline{a}_i) \le \alpha w(\underline{b}_i)$ $(i = 1, \ldots, t)$. Therefore $w(\underline{a}) \le \alpha w(\underline{b})$, whence

$$w(\underline{K}_N(\underline{z}^{(0)}, \underline{F}'(\underline{z}^{(0)}), B^{(0)})) \leq \alpha w(\underline{z}^{(0)})$$

where

$$\alpha = \max_{1 \leq i \leq t} \{ \frac{w((\underline{K}_N(\underline{z}^{(0)}, \underline{F}'(\underline{z}^{(0)}), B^{(0)}))_i)}{w(\underline{z}_i^{(0)})} \}.$$

Furthermore, as shown in [AleP-83a]

$$\underline{K}(\underline{z}^{(0)}, \underline{F}'^{(0)}, B^{(0)}) = m(\underline{z}^{(0)}) - \{B^{(0)}\}^{-1} F(m(\underline{z}^{(0)})) +$$

$$(I - \{B^{(0)}\}^{-1} \underline{F}'(\underline{z}^{(0)}))(\underline{z}^{(0)} - m(\underline{z}^{(0)}))$$

$$\subseteq \underline{K}_N(\underline{z}^{(0)}, \underline{F}'^{(0)}, B^{(0)}),$$

so by 5.10, $\underline{K}(\underline{z}^{(0)}, \underline{F}'^{(0)}, B^{(0)}) \subset int(\underline{z}^{(0)})$. Therefore [MooQ-82a] $\exists z^* \in \underline{z}^{(0)}$ such that $F(z^*) = 0$ and z^* is unique in $\underline{z}^{(0)}$. Therefore the hypotheses of **Theorem 5.1** are valid. □

Note 5.1 : If $b_i > 0$ $(i = 1, \ldots, t)$ and $a_i \geq 0$ $(i = 1, \ldots, t)$ then

$$\frac{a_i}{b_i} \leq max\{\frac{a_i}{b_i}\} \ (i = 1, \ldots, t)$$

$$\Rightarrow \; a_i \leq max\{\frac{a_i}{b_i}\}\, b_i$$

$$\leq max\{\frac{a_i}{b_i}\}\, max\{b_i\} \; (i = 1, \ldots, t)$$

$$\Rightarrow \; max\{a_i\} \leq max\{\frac{a_i}{b_i}\}\, max\{b_i\}$$

$$\Rightarrow \; \frac{max\{a_i\}}{max\{b_i\}} \leq max\{\frac{a_i}{b_i}\}$$

$$\Rightarrow \; \frac{\|a\|}{\|b\|} \leq max\{\frac{a_i}{b_i}\}.$$

Therefore $(\frac{\|a\|}{\|b\|} < 1) \not\Rightarrow \; (max\{\frac{a_i}{b_i}\} < 1)$. Therefore it is <u>not</u> sufficient to take

$$\alpha = \frac{\|w(\underline{K}_N(\underline{z}^{(0)}, \underline{F}'(\underline{z}^{(0)}), B^{(0)}))\|}{\|w(\underline{z}^{(0)})\|}$$

in **Corollary 5.1.** □

5.3 Methods for Determining the $p^{(k)}$

Methods which automatically select $p^{(k)}$, independently of the size and complexity of the system of equations [SheW-85b], are described in this section.

Assume that, in MAP, for some $k \geq 0$ and some $m \geq 1$ $\underline{z}^{(k,m)}$ has been computed, and that a reliable estimate of the relative efficiences of computing a new outer

iterate, $\underline{z}^{(k+1,1)}$ and that of computing a new inner iterate, $\underline{z}^{(k,m+1)}$ can be obtained. Then, whichever iterate is expected to be more efficient could be computed and the decision process could be repeated.

Efficiency indices ρ_I and ρ_O corresponding to the computation of $\underline{z}^{(k,m+1)}$ and $\underline{z}^{(k+1,1)}$ respectively are given by

$$\rho_I = -\ln\left(\left\|w(\underline{z}^{(k,m+1)})\right\| / \left\|w(\underline{z}^{(k,m)})\right\|\right)/T_I \qquad 5.11$$

and

$$\rho_O = -\ln\left(\left\|w(\underline{z}^{(k+1,1)})\right\| / \left\|w(\underline{z}^{(k,m)})\right\|\right)/T_O, \qquad 5.12$$

where T_I and T_O are the CPU times required to compute $\underline{z}^{(k,m+1)}$ and $\underline{z}^{(k+1,1)}$ respectively from $\underline{z}^{(k,m)}$. The CPU time for the next outer iteration could be estimated by the CPU time required for the previous outer iteration. Similarly the CPU time for the next inner iteration could be estimated by the CPU time for the previous inner iteration [SheW-85b].

If $\underline{z}^{(k+1,0)} = \underline{z}^{(k,m)}$ then **Theorem 5.1**(a) suggests that

$$\left\|w(\underline{z}^{(k+1,1)})\right\| \approx M^{(k)} \left\|w(\underline{z}^{(k,m)})\right\|^2 \qquad 5.13$$

where

$$M^{(k)} = \left\|w(\underline{z}^{(k,1)})\right\| / \left\|w(\underline{z}^{(k,0)})\right\|^2 \qquad 5.14$$

and

$$\left\|w(\underline{z}^{(k,m+1)})\right\| \approx N^{(k,m)} \left\|w(\underline{z}^{(k,m)})\right\| \qquad 5.15$$

where

$$N^{(k,m)} = \left\| w(\underline{z}^{(k,m)}) \right\| / \left\| w(\underline{z}^{(k,m-1)}) \right\| . \qquad 5.16$$

Two strategies for deciding whether or not to compute another inner iteration use some or all of the above approximations, and are as follows.

Strategy 1

1. Estimate the efficiency index ρ_I for an inner iteration from

$$\rho_I = -\ln\left(\left\| w(\underline{z}^{(k,m)}) \right\| / \left\| w(\underline{z}^{(k,m-1)}) \right\| \right) / T^{(k,m)} \qquad 5.17$$

where $T^{(k,m)}$ is the time required to compute $\underline{z}^{(k,m)}$ from $\underline{z}^{(k,m-1)}$.

2. Estimate the efficiency index ρ_O which would have been obtained if instead $\underline{z}^{(k+1,1)}$ had been computed with $\underline{z}^{(k+1,0)} = \underline{z}^{(k,m-1)}$ from

$$\rho_O = -\ln\left(\left(M^{(k)} \left\| w(\underline{z}^{(k,m-1)}) \right\|^2 \right) / \left\| w(\underline{z}^{(k,m-1)}) \right\| \right) / T^{(k,1)} . \qquad 5.18$$

3. If $\rho_O > \rho_I$ then recompute the Jacobian and compute $\underline{z}^{(k+1,1)}$. Otherwise re-use the Jacobian, and compute $\underline{z}^{(k,m+1)}$.

Strategy 2

1. Estimate the efficiency index ρ_I for the inner iteration from

$$\rho_I = -\ln\left(\left(N^{(k,m)} \left\| w(\underline{z}^{(k,m)}) \right\| \right) / \left\| w(\underline{z}^{(k,m)}) \right\| \right) / T^{(k,m)}. \qquad 5.19$$

2. Estimate the efficiency index ρ_O for computing the outer iterate $\underline{z}^{(k+1,1)}$ with $\underline{z}^{(k+1,0)} = \underline{z}^{(k,m)}$ from

$$\rho_O = -\ln\left(\left(M^{(k)} \left\| w(\underline{z}^{(k,m)}) \right\|^2 \right) / \left\| w(\underline{z}^{(k,m)}) \right\| \right) / T^{(k,1)}. \qquad 5.20$$

3. If $\rho_O > \rho_I$ then recompute the Jacobian and compute $\underline{z}^{(k+1,1)}$. Otherwise re-use the Jacobian, and compute $\underline{z}^{(k,m+1)}$.

These ideas are embodied in the procedure *re.use.Jacobian* which follows.

procedure *re.use.Jacobian*$(w^{(O)}, w^{(I)} \in R^t, M, T_O, T_I \in R, p, m, t \in N$;

re.use.jacobian $\in B$)

! This procedure determines whether or not an interval extension of the Jacobian F'
! of F and the inverse of its centre are re-used $p^{(k)}$ times as explained in [SheW-85b].
! On entry, $w^{(O)} = w(\underline{z}^{(k,m-1)})$, $w^{(I)} = w(\underline{z}^{(k,m)})$, M is computed from 5.14, T_O is
! the CPU time required to compute $\underline{z}^{(k,1)}$ from $\underline{z}^{(k,0)} = \underline{z}^{(k-1,p^{(k-1)})}$, T_I is the cpu

! time required to compute $\underline{z}^{(k,m)}$ from $\underline{z}^{(k,m-1)}$, and $p \in \{-2, -1, 0, 1, \ldots, \infty\}$

! where $p = -2$ corresponds to strategy 2, and $p = -1$ corresponds to strategy 1;

! otherwise p is the given fixed number of times that $\underline{F}'$ and $\{m(\underline{F}')\}^{-1}$ are re-used,

! $m \geq 0$ is the inner iteration index, and t is the number of components in $\underline{z}$.

! On return, if *re.use.jacobian* = *true* then the Jacobian matrix $\underline{F}'$ and the inverse

! of its centre are re-used; otherwise $\underline{F}'$ and the inverse of its centre are re-computed.

! It is assumed that initially, *re.use.jacobian* = *true* on entry.

1. *if* $p \geq 0$

   then

   ! $\underline{F}'$ *and* $\{m(\underline{F}')\}^{-1}$ *are re – used* p *times.*

   1.1. *if* $m \geq p$ *do*

      1.1.1. *re.use.jacobian* := *false*

   else

   1.2. *case true of*

      $p = -1$: ! *Strategy* 1

      1.2.1. $\rho_O := -\ln(M \left\| w^{(O)} \right\|)/T_O$! *Equation* 5.18

      1.2.2. $\rho_I := -\ln(\left\| w^{(I)} \right\| / \left\| w^{(O)} \right\|)/T_I$! *Equation* 5.17

      1.2.3. *if* $\rho_O > \rho_I$ *do*

         1.2.3.1. *re.use.jacobian* := *false*

      $p = -2$: ! *Strategy* 2

1.2.4. $\rho_O := -\ln(M \left\| w^{(I)} \right\|)/T_O$! *Equation* 5.20

1.2.5. $\rho_I := -\ln(\left\| w^{(I)} \right\| / \left\| w^{(O)} \right\|)/T_I$! *Equation* 5.19

1.2.6. *if* $\rho_O > \rho_I$ *do*

1.2.6.1. *re.use.jacobian* := *false*

default :

1.2.7. *write* "*Error in re.use.Jacobian.*"

1.2.8. *stop*

2. *return* . □

5.4 The Algorithm *KMSW*

Let $\hat{\underline{z}} \in I(\hat{D})$ be given where $\hat{D} \subseteq D$ is an open convex set, and let

$$\hat{A} = \{m(\underline{F}'(\hat{\underline{z}}))\}^{-1}$$

$$= (\hat{a}_{ij})_{t \times t}, \qquad 5.21$$

$$\hat{R} = I - \hat{A}\underline{F}'(\hat{\underline{z}})$$

$$= (\hat{r}_{ij})_{t \times t}, \qquad 5.22$$

$$\hat{z} = m(\underline{\hat{z}}), \tag{5.23}$$

$$\hat{b} = \hat{A}F(\hat{z}), \tag{5.24}$$

$$\underline{K}(\underline{\hat{z}}) = \hat{z} - \hat{b} + \underline{\hat{R}}(\underline{\hat{z}} - \hat{z}), \tag{5.25}$$

$$\underline{H}_i(\underline{\hat{z}}) = \hat{z}_i - \hat{b}_i + \sum_{j=1}^{i-1} \underline{\hat{r}}_{ij}(\underline{H}'_j(\underline{\hat{z}}) - \hat{z}_j) + \sum_{j=i}^{t} \underline{\hat{r}}_{ij}(\underline{\hat{z}}_j - \hat{z}_j), \tag{5.26}$$

$$\underline{H}'_i(\underline{\hat{z}}) = \underline{H}_i(\underline{\hat{z}}) \cap \underline{\hat{z}}_i \ (i = 1, \ldots, t) \tag{5.27}$$

$$\underline{S}_i(\underline{\hat{z}}) = \hat{z}_i - \hat{b}_i + \sum_{j=1}^{i} \underline{\hat{r}}_{ij}(\underline{H}'_j(\underline{\hat{z}}) - \hat{z}_j) + \sum_{j=i+1}^{t} \underline{\hat{r}}_{ij}(\underline{S}'_j(\underline{\hat{z}}) - \hat{z}_j), \tag{5.28}$$

$$\underline{S}'_i(\underline{\hat{z}}) = \underline{S}_i(\underline{\hat{z}}) \cap \underline{H}'_i(\underline{\hat{z}}) \ (i = t, \ldots, 1), \tag{5.29}$$

where $F : D \subset R^t \to R^1$, $\underline{F} : I(D) \to I(R^t)$, and $\underline{F}' : I(\hat{D}) \to I(M(R^t))$ are as in §5.1. The mappings $\underline{K}$, $\underline{H}$, and $\underline{S}$ are the Krawczyk [Kra–69a], Hansen [HanS-81a], and Symmetric [SheW-85a] operators respectively.

The algorithm $KMSW$ consists of generating the sequence $(\underline{z}^{(k)})$ with $\underline{z}^{(0)} \in I(R^t)$ given, as follows.

Algorithm KMSW

Data : $\hat{\underline{z}} \in I(R^t)$, t, $p^{(k)} \in N$ $(k \geq 0)$.

1. $\underline{z}^{(0)} := \hat{\underline{z}}$
2. $\underline{F}'^{(0)} := \underline{F}'(\underline{z}^{(0)})$
3. $B^{(0)} := \{m(\underline{F}'^{(0)})\}^{-1}$
4. $A^{(0)} := B^{(0)}$
5. $\underline{R}^{(0)} := I - A^{(0)}\underline{F}'^{(0)}$ $(= (\underline{r}_{ij}^{(0)})_{t\times t})$
6. $r^{(0)} := \left\|\underline{R}^{(0)}\right\|$
7. $k := 0$
8. *while true do*
   - 8.1. $\underline{z}^{(k,0)} := \underline{z}^{(k)}$
   - 8.2. *for* $m = 0$ *to* $p^{(k)}$ *do*
     - 8.2.1. $z^{(k,m)} := m(\underline{z}^{(k,m)})$
     - 8.2.2. $b^{(k,m)} := A^{(k)}F(z^{(k,m)})$
     - 8.2.3. *for* $i = 1$ *to* t *do*

$$8.2.3.1.\quad \underline{H}_i^{(k,m)} := z_i^{(k,m)} - b_i^{(k,m)} + \sum_{j=1}^{i-1} \underline{r}_{ij}^{(k)}(\underline{H}_j'^{(k,m)} - z_j^{(k,m)}) + \sum_{j=i}^{t} \underline{r}_{ij}^{(k)}(\underline{z}_j^{(k,m)} - z_j^{(k,m)})$$

8.2.3.2. $\underline{H}'^{(k,m)}_i := \underline{H}^{(k,m)}_i \cap \underline{z}^{(k,m)}_i$

8.2.4. *for* $i = t$ *to* 1 *by* -1 *do*

8.2.4.1. $$\underline{S}^{(k,m)}_i := z^{(k,m)}_i - b^{(k,m)}_i + \sum_{j=1}^{i} \underline{r}^{(k)}_{ij} (\underline{H}'^{(k,m)}_j - z^{(k,m)}_j) + \sum_{j=i+1}^{t} \underline{r}^{(k)}_{ij} (\underline{S}'^{(k,m)}_j - z^{(k,m)}_j)$$

8.2.4.2. $\underline{S}'^{(k,m)}_i := \underline{S}^{(k,m)}_i \cap \underline{H}'^{(k,m)}_i$

8.2.5. $\underline{z}^{(k,m+1)} := \underline{S}'^{(k,m)}$

8.3. $\underline{z}^{(k+1)} := \underline{z}^{(k,p^{(k)}+1)}$

8.4. $\underline{F}'^{(k+1)} := \underline{F}'(\underline{z}^{(k+1)})$

8.5. $B^{(k+1)} := \{m(\underline{F}'^{(k+1)})\}^{-1}$

8.6. $\underline{R}^{(k+1)} := I - B^{(k+1)} \underline{F}'^{(k+1)}$

8.7. $r^{(k+1)} := \left\| \underline{R}^{(k+1)} \right\|$

8.8. *if* $r^{(k+1)} \leq r^{(k)}$

then

8.8.1. $A^{(k+1)} := B^{(k+1)}$

else

8.8.2. $A^{(k+1)} := A^{(k)}$

8.8.3. $\underline{R}^{(k+1)} := I - A^{(k+1)} \underline{F}'^{(k+1)}$

8.8.4. $r^{(k+1)} := \left\| \underline{R}^{(k+1)} \right\|$

8.9. $k := k + 1$ □

The determination of $p^{(k)}$ is described in §5.3.

5.5 Theoretical Results for $KMSW$

This section contains some existence, uniqueness, and convergence results related to the algorithm $KMSW$ which are due to Shearer and Wolfe [SheW-85c][SheW-85a].

Lemma 5.4 : Let $\underline{K}^* : I(\hat{D}) \to I(R^t)$ and $\underline{\hat{K}} : I(\hat{D}) \to I(R^t)$ be defined by

$$\underline{K}^*(\underline{z}) = m(\underline{z}) - \hat{A}F(m(\underline{z})) + \hat{\underline{R}}(\underline{z} - m(\underline{z})) \tag{5.30}$$

and

$$\underline{\hat{K}}(\underline{z}) = m(\underline{z}) - \hat{A}F(m(\underline{z})) + (I - \hat{A}\underline{F}'(\underline{z}))(\underline{z} - m(\underline{z})) \tag{5.31}$$

where $\hat{A}$ and $\hat{\underline{R}}$ are given by 5.21 and 5.22 respectively. Then (a) $\underline{H}(\hat{\underline{z}}) \subseteq \underline{K}(\hat{\underline{z}})$; (b) $\underline{S}(\hat{\underline{z}}) \subseteq \underline{H}(\hat{\underline{z}})$; (c) $(\underline{z} \subseteq \hat{\underline{z}}) \Rightarrow (\underline{\hat{K}}(\underline{z}) \subseteq \underline{K}^*(\underline{z}))$; (d) $(\underline{S}(\hat{\underline{z}}) \subseteq \hat{\underline{z}}) \Rightarrow (\underline{S}(\hat{\underline{z}}) \subseteq \underline{H}'(\hat{\underline{z}}))$.

Proof : (a) By 5.27, for $i = 1, \ldots, t$, $\underline{H}_i'(\hat{\underline{z}}) \subseteq \hat{\underline{z}}_i$. So by 5.26, for $i = 1, \ldots, t$

$$\underline{H}_i(\hat{\underline{z}}) \subseteq \hat{z}_i - \hat{g}_i + \sum_{j=1}^{i-1} \hat{r}_{ij}(\hat{\underline{z}}_j - \hat{z}_j) + \sum_{j=i}^{t} \hat{\underline{r}}_{ij}(\hat{\underline{z}}_j - \hat{z}_j)$$

$$= \underline{K}_i(\hat{\underline{z}}).$$

(b) By 5.28, 5.26, 5.27, since $\underline{H}_t'(\hat{\underline{z}}) \subseteq \hat{\underline{z}}_t$,

$$\underline{S}_t(\hat{\underline{z}}) = \hat{z}_t - \hat{b}_t + \sum_{j=1}^{t} \hat{\underline{r}}_{tj}(\underline{H}'_j(\hat{\underline{z}}) - \hat{z}_j)$$

$$= \hat{z}_t - \hat{b}_t + \sum_{j=1}^{t-1} \hat{\underline{r}}_{tj}(\underline{H}'_j(\hat{\underline{z}}) - \hat{z}_j) + \hat{\underline{r}}_{tt}(\underline{H}'_t(\hat{\underline{z}}) - \hat{z}_t)$$

$$\subseteq \hat{z}_t - \hat{b}_t + \sum_{j=1}^{t-1} \hat{\underline{r}}_{tj}(\underline{H}'_j(\hat{\underline{z}}) - \hat{z}_j) + \hat{\underline{r}}_{tt}(\hat{\underline{z}}_t - \hat{z}_t)$$

$$= \underline{H}_t(\hat{\underline{z}}).$$

Suppose that for some $l \geq 0$, $\underline{S}_{t-k}(\hat{\underline{z}}) \subseteq \underline{H}_{t-k}(\hat{\underline{z}})$, $(k = 0, \ldots, l)$. Then $\underline{S}'_j(\hat{\underline{z}}) \subseteq \underline{H}'_j(\hat{\underline{z}}) \subseteq \hat{\underline{z}}_j$ $(j = t, \ldots, t-l)$, so

$$\underline{S}_{t-l-1}(\hat{\underline{z}}) = \hat{z}_{t-l-1} + \hat{b}_{t-l-1} + \sum_{j=1}^{t-l-1} \hat{\underline{r}}_{t-l-1\,j}(\underline{H}'_j(\hat{\underline{z}}) - \hat{z}_j)$$

$$+ \sum_{j=t-l}^{t} \hat{\underline{r}}_{t-l-1\,j}(\underline{S}'_j(\hat{\underline{z}}) - \hat{z}_j)$$

$$\subseteq \hat{z}_{t-l-1} + \hat{b}_{t-l-1} + \sum_{j=1}^{t-l-1} \hat{\underline{r}}_{t-l-1\,j}(\underline{H}'_j(\hat{\underline{z}}) - \hat{z}_j)$$

$$+ \sum_{j=t-l}^{t} \hat{\underline{r}}_{t-l-1\,j}(\hat{\underline{z}}_j - \hat{z}_j)$$

$$= \underline{H}_{t-l-1}(\hat{\underline{z}}).$$

So by finite induction on l $\underline{S}_{t-k}(\hat{\underline{z}}) \subseteq \underline{H}_{t-k}(\hat{\underline{z}})$ $(k = 0, \ldots, t-1)$, whence $\underline{S}(\hat{\underline{z}}) \subseteq \underline{H}(\hat{\underline{z}})$.

(c) By 5.30, 5.31, and the inclusion monotonicity of $\underline{F}'$,

$$\hat{\underline{K}}(\underline{z}) = m(\underline{z}) - \hat{A}F(m(\underline{z})) + (I - \hat{A}\underline{F}'(\underline{z}))(\underline{z} - m(\underline{z}))$$

$$\subseteq m(\underline{z}) - \hat{A}F(m(\underline{z})) + (I - \hat{A}\underline{F}'(\hat{\underline{z}}))(\underline{z} - m(\underline{z}))$$

$$= m(\underline{z}) - \hat{A}F(m(\underline{z})) + \hat{\underline{R}}(\underline{z} - m(\underline{z}))$$

$$= \underline{K}^*(\underline{z}).$$

(d) By (b) $\underline{S}(\hat{\underline{z}}) \subseteq \underline{H}(\hat{\underline{z}})$, so if also, $\underline{S}(\hat{\underline{z}}) \subseteq \hat{\underline{z}}$, then $\underline{S}(\underline{z}) \subseteq \hat{\underline{z}} \cap \underline{H}(\hat{\underline{z}}) = \underline{H}'(\hat{\underline{z}})$. □

Lemma 5.5 : If $F : D \subseteq R^t \to R^t$, $\underline{F}' : I(\hat{D}) \to I(M(R^t))$ and $\hat{\underline{z}} \in I(\hat{D})$ are as in **Lemma 5.4** and $P : \hat{D} \to R^t$ is defined by

$$P(z) = z - \hat{A}F(z) \qquad \textbf{5.32}$$

where $\hat{A}$ is given by 5.21, then

(a) $(z \in \underline{S}(\hat{\underline{z}}) \wedge \underline{S}(\hat{\underline{z}}) \subseteq \underline{H}'(\hat{\underline{z}})) \Rightarrow (P(z) \in \underline{S}(\hat{\underline{z}}))$; (b) $(z \in \underline{z} \subseteq \hat{\underline{z}}) \Rightarrow (P(z) \in \hat{\underline{K}}(\underline{z}))$;

(c) $(z^* \in \hat{\underline{z}} \wedge F(z^*) = 0) \Rightarrow (z^* \in \underline{S}'(\hat{\underline{z}}))$.

Proof: (a) Let $\underline{\hat{S}} = \underline{S}(\hat{z})$, $\underline{\hat{S}}' = \underline{S}'(\hat{z})$, and $\underline{\hat{H}}' = \underline{H}'(\hat{z})$. If $\underline{\hat{S}} \subseteq \underline{\hat{H}}'$ then $\underline{\hat{S}}' = \underline{\hat{S}} \cap \underline{\hat{H}}' = \underline{\hat{S}}$ so by 5.32 and 5.28, for $i = 1, \ldots, t$, if $z \in \underline{\hat{S}}$ then since $(\underline{\hat{S}} \subseteq \underline{\hat{H}}') \Rightarrow (\underline{\hat{S}} \subseteq \hat{z})$, we have

$$P_i(z) = z_i - (\hat{A}F(z))_i$$

$$= \hat{z}_i - (\hat{A}F(\hat{z}))_i + z_i - \hat{z}_i - (\hat{A}(F(z) - F(\hat{z})))_i$$

$$\in \hat{z}_i - \hat{b}_i + ((I - \hat{A}\underline{F}'(\hat{\underline{z}}))(\underline{\hat{S}} - \hat{z}))_i$$

$$= \hat{z}_i - \hat{b}_i + \sum_{j=1}^{i} \hat{\underline{r}}_{ij}(\underline{\hat{S}}_j - \hat{z}_j) + \sum_{j=i+1}^{t} \hat{\underline{r}}_{ij}(\underline{\hat{S}}_j - \hat{z}_j)$$

$$\subseteq \hat{z}_i - \hat{b}_i + \sum_{j=1}^{i} \hat{\underline{r}}_{ij}(\underline{\hat{H}}'_j - \hat{z}_j) + \sum_{j=i+1}^{t} \hat{\underline{r}}_{ij}(\underline{\hat{S}}'_j - \hat{z}_j)$$

$$= \underline{\hat{S}}_i.$$

(b) If $z \in \underline{z} \subseteq \hat{\underline{z}}$ then by 5.32, 5.31 and the inclusion monotonicity of $\underline{F}'$,

$$P(z) = z - \hat{A}F(z)$$

$$= m(\underline{z}) - \hat{A}F(m(\underline{z})) + z - m(\underline{z}) - \hat{A}(F(z) - F(m(\underline{z})))$$

$$\in m(\underline{z}) - \hat{A}F(m(\underline{z})) + (I - \hat{A}\underline{F}'(\underline{z}))(\underline{z} - m(\underline{z}))$$

$$= \underline{\hat{K}}(\underline{z}).$$

(c) If $\exists z^* \in \hat{\underline{z}}$ such that $F(z^*) = 0$, then $z^* = P(z^*)$. Also by **Lemma 2.10** $(z \in \hat{\underline{z}}) \Rightarrow (P(z) \in \underline{K}(\hat{\underline{z}}))$. Therefore $z^* \in \underline{K}(\hat{\underline{z}})$. By 5.26, $\underline{K}_1(\hat{\underline{z}}) = \underline{H}_1(\hat{\underline{z}})$. Therefore $z_1^* \in \underline{H}_1(\hat{\underline{z}})$. So $z_1^* \in \underline{H}_1(\hat{\underline{z}}) \cap \hat{\underline{z}}_1 = \underline{\hat{H}}_1'$. Suppose that for some $i \geq 2$, $z_j^* \in \underline{\hat{H}}_j'$ $(j = 1 \ldots, i-1)$. Then

$$z_i^* = P_i(z^*)$$

$$\in \hat{z}_i - \hat{b}_i + \sum_{j=1}^{t} \hat{\underline{r}}_{ij}(z_j^* - \hat{z}_j)$$

$$\subseteq \hat{z}_i - \hat{b}_i + \sum_{j=1}^{i-1} \hat{\underline{r}}_{ij}(\underline{\hat{H}}_j' - \hat{z}_j) + \sum_{j=i}^{t} \hat{\underline{r}}_{ij}(\hat{\underline{z}}_j - \hat{z}_j)$$

$$= \underline{H}_i(\hat{\underline{z}})$$

whence $z_i^* \in \underline{\hat{H}}_i' = \underline{H}_i(\hat{\underline{z}}) \cap \hat{\underline{z}}_i$. So by finite induction on i, $z^* \in \underline{\hat{H}}'$. Now

$$\underline{\hat{S}}_t = \hat{z}_t - \hat{b}_t + \sum_{j=1}^{t} \hat{\underline{r}}_{tj}(\underline{\hat{H}}_j' - \hat{z}_j)$$

$$\supseteq \hat{z}_t - \hat{b}_t + \sum_{j=1}^{t} \hat{\underline{r}}_{tj}(z_j^* - \hat{z}_j)$$

$$= \hat{z}_t - \hat{b}_t + \sum_{j=1}^{t} \{\delta_{tj} - \sum_{k=1}^{t} \hat{a}_{tk} \partial_j \underline{F}_k(\hat{\underline{z}})\}(z_j^* - \hat{z}_j).$$

Now $\exists \theta_k \in [0, 1]$ such that

$$F_k(z^*) - F_k(\hat{z}) = \sum_{j=1}^{t} \partial_j F_k(\hat{z} + \theta_k(z^* - \hat{z}))(z_j^* - \hat{z}_j)$$

and by convexity of $\hat{\underline{z}}$, $\partial_j F_k(\hat{z} + \theta_k(z^* - \hat{z})) \in \partial_j \underline{F}_k(\hat{\underline{z}})$. Therefore

$$\hat{\underline{S}}_t \ni \hat{z}_t - \hat{b}_t + \sum_{j=1}^{t} \{\delta_{tj} - \sum_{k=1}^{t} \hat{a}_{tk} \partial_j F_k(\hat{z} + \theta_k(z^* - \hat{z}))\}(z_j^* - \hat{z}_j)$$

$$= \hat{z}_t - \hat{b}_t + z_t^* - \hat{z}_t - \sum_{k=1}^{t} \hat{a}_{tk} \{\sum_{j=1}^{t} \partial_j F_k(\hat{z} + \theta_k(z^* - \hat{z}))(z_j^* - \hat{z}_j)\}$$

$$= z_t^* - \hat{b}_t - \sum_{k=1}^{t} \hat{a}_{tk}(F_k(z^*) - F_k(\hat{z}))$$

$$= z_t^* - \sum_{k=1}^{t} \hat{a}_{tk} F_k(\hat{z}) + \sum_{k=1}^{t} \hat{a}_{tk} F_k(\hat{z})$$

$$= z_t^*.$$

So $z_t^* \in \underline{\hat{S}}_t$. Therefore $z_t^* \in \underline{\hat{S}}_t \cap \underline{\hat{H}}_t' = \underline{\hat{S}}_t'$. Suppose that for some $i \leq t-1$ $z_k^* \in \underline{\hat{S}}_k'$ $(k = t, \ldots, i+1)$. Then

$$\underline{\hat{S}}_i = \hat{z}_i - \hat{b}_i + \sum_{j=1}^{i} \hat{r}_{ij}(\underline{\hat{H}}_j' - \hat{z}_j) + \sum_{j=i+1}^{t} \hat{r}_{ij}(\underline{\hat{S}}_j' - \hat{z}_j)$$

$$\supseteq \hat{z}_i - \hat{b}_i + \sum_{j=1}^{i} \hat{r}_{ij}(z_j^* - \hat{z}_j) + \sum_{j=i+1}^{t} \hat{r}_{ij}(z_j^* - \hat{z}_j)$$

$$\ni \hat{z}_i - \hat{b}_i + \sum_{j=1}^{t} \{\delta_{ij} - \sum_{k=1}^{t} \hat{a}_{ik} \partial_j F_k(\hat{z} + \theta_k(z^* - \hat{z}))\}(z_j^* - \hat{z}_j)$$

$$= \hat{z}_i - \hat{b}_i + z_i^* - \hat{z}_i - \sum_{k=1}^{t} \hat{a}_{ik} \sum_{j=1}^{t} \partial_j F_k(\hat{z} + \theta_k(z^* - \hat{z}))(z_j^* - \hat{z}_j)$$

$$= z_i^* - \hat{b}_i - \sum_{k=1}^{t} \hat{a}_{ik}(F_k(z^*) - F_k(\hat{z}))$$

$$= z_i^* - \hat{b}_i + \hat{b}_i = z_i^*.$$

So by finite induction on i, $z_i^* \in \underline{\hat{S}}_i$ $(i = t-1, \ldots, 0)$, whence $z^* \in \underline{\hat{S}}'$. □

Theorem 5.2 : Suppose that $F : D \subseteq R^t \to R^t$, $\underline{F}' : I(\hat{D}) \to I(M(R^t))$, and $\underline{S} : I(\hat{D}) \to I(R^t)$ are as in **Lemma 5.5**. If (1) $\hat{A} \in M(R^t)$ defined by 5.21 exists; (2) $\underline{S}(\hat{z})$ is given by 5.28 and 5.29 and $\underline{S}(\hat{z}) \neq \emptyset$; (3) $\underline{S}(\hat{z}) \subseteq \hat{z}$, then $\exists z^* \in \underline{S}(\hat{z})$ such that $F(z^*) = 0$.

Proof : By Lemma 5.4(d), $\underline{S}(\hat{\underline{z}}) \subseteq \underline{H}'(\hat{\underline{z}})$, so by Lemma 5.5(a), if $z \in \underline{S}(\hat{\underline{z}})$ then $P(z) \in \underline{S}(\hat{\underline{z}})$. Therefore by Lemma 2.5, $\exists z^* \in \underline{S}(\hat{\underline{z}})$ such that $F(z^*) = 0$. □

By Lemma 5.4(b) $\underline{S}(\hat{\underline{z}}) \subseteq \underline{H}(\hat{\underline{z}})$ whether or not $\underline{H}(\hat{\underline{z}}) \subseteq \hat{\underline{z}}$, so it is possible that $\underline{S}(\hat{\underline{z}}) \subseteq \hat{\underline{z}}$ and $\underline{H}(\hat{\underline{z}}) \not\subseteq \hat{\underline{z}}$. Therefore Theorem 5.2 contains an existence test which is weaker than the test $\underline{H}(\hat{\underline{z}}) \subseteq \hat{\underline{z}}$ of Moore and Qi [MooQ-82a].

Theorem 5.3 : If the hypotheses of **Theorem 5.2** are valid and if also $w(\underline{S}(\hat{\underline{z}})) < w(\underline{H}'(\hat{\underline{z}}))$ then $\exists z^* \in \underline{S}(\hat{\underline{z}})$ such that $F(z^*) = 0$ and z^* is unique in $\hat{\underline{z}}$.

Proof : Let the sequence $(\underline{z}^{(k)})$ be generated from

$$\underline{z}^{(0)} = \underline{S}(\hat{\underline{z}}) \tag{5.33}$$

$$\underline{z}^{(k+1)} = \underline{S}(\hat{\underline{z}}) \cap \underline{K}^*(\underline{z}^{(k)}) \ (k \geq 0). \tag{5.34}$$

By **Theorem 5.2**, $\exists z^* \in \underline{z}^{(0)}$ such that $F(z^*) = 0$. Suppose that for some $k \geq 0$, $z^* \in \underline{z}^{(k)}$. By **Lemma 5.5**(b) $z^* = P(z^*) \in \hat{\underline{K}}(\underline{z}^{(k)})$, so by **Lemma 5.4**(c), $z^* \in \underline{K}^*(\underline{z}^{(k)})$. Therefore $z^* \in \underline{z}^{(k+1)}$. Therefore by induction on k, $z^* \in \underline{z}^{(k)}$ $(\forall k \geq 0)$.

We shall show that $w(\underline{z}^{(k)}) \to 0$ $(k \to \infty)$ whence z^* is unique in $\underline{z}^{(0)} = \underline{S}(\hat{\underline{z}})$. But by **Lemma 5.5**(c) the zeros of F in $\hat{\underline{z}}$ are all in $\underline{S}(\hat{\underline{z}})$. Therefore if z^* is unique in $\underline{S}(\hat{\underline{z}})$ then z^* is also unique in $\hat{\underline{z}}$.

By 5.34, and the symmetry of $\hat{R}$

$$w(\underline{z}_i^{(k+1)}) \leq w(\underline{K}_i^*(\underline{z}^{(k)}))$$

$$= w\Big(\sum_{j=1}^{t} \hat{\underline{r}}_{ij}(\underline{z}_j^{(k)} - z_j^{(k)})\Big)$$

$$= \sum_{j=1}^{t} w(\hat{\underline{r}}_{ij}(\underline{z}_j^{(k)} - z_j^{(k)}))$$

$$= \sum_{j=1}^{t} w(\hat{\underline{r}}_{ij}) \mid \underline{z}_j^{(k)} - z_j^{(k)} \mid$$

$$= \sum_{j=1}^{t} w(\hat{\underline{r}}_{ij}) w(\underline{z}_j^{(k)})/2 \ (i = 1, \ldots, t). \qquad 5.35$$

Also, if $\hat{\underline{H}}' = \underline{H}'(\hat{\underline{z}})$ and $\hat{\underline{S}} = \underline{S}(\hat{\underline{z}})$ then by Lemma 5.4(d) and the fact that $\mid \underline{u} - z \mid \geq w(\underline{u})/2 \ (\forall z \in R)(\forall \underline{u} \in I(R))$, we have

$$\sum_{j=1}^{t} w(\hat{\underline{r}}_{ij}) w(\hat{\underline{S}}_j)/2 = \sum_{j=1}^{i} w(\hat{\underline{r}}_{ij}) w(\hat{\underline{S}}_j)/2 + \sum_{j=i+1}^{t} w(\hat{\underline{r}}_{ij}) w(\hat{\underline{S}}_j)/2$$

$$\leq \sum_{j=1}^{i} w(\hat{\underline{r}}_{ij}) w(\hat{\underline{H}}'_j)/2 + \sum_{j=i+1}^{t} w(\hat{\underline{r}}_{ij}) w(\hat{\underline{S}}_j)/2$$

$$\leq \sum_{j=1}^{i} w(\hat{\underline{r}}_{ij}) \mid \hat{\underline{H}}'_j - \hat{z}_j \mid + \sum_{j=i+1}^{t} w(\hat{\underline{r}}_{ij}) \mid \hat{\underline{S}}'_j - \hat{z}_j \mid$$

$$= w(\hat{\underline{S}}_i) \ (i = 1, \ldots, t).$$

So for $i = 1, \ldots, t$, we have

$$\sum_{j=1}^{i} w(\hat{\underline{r}}_{ij}) w(\hat{\underline{H}}'_j)/2 + \sum_{j=i+1}^{t} w(\hat{\underline{r}}_{ij}) w(\hat{\underline{S}}_j)/2 \leq w(\hat{\underline{S}}_i), \tag{5.36}$$

and

$$\sum_{j=1}^{t} w(\hat{\underline{r}}_{ij}) w(\hat{\underline{S}}_j)/2 \leq w(\hat{\underline{S}}_i). \tag{5.37}$$

Let

$$\beta = \max_{1 \leq i \leq t} \{w(\hat{\underline{S}}_i)/w(\hat{\underline{H}}'_i)\}. \tag{5.38}$$

Then $\beta \in [0, 1)$. We shall show that $(\forall k \geq 0)$

$$w(\underline{z}^{(kt)}) \leq \beta^k w(\hat{\underline{S}}). \tag{5.39}$$

In order to establish 5.39 we shall prove that for $l = 0, \ldots, t$,

$$w(\underline{z}_i^{(kt+l)}) \leq \beta^k w(\hat{\underline{S}}_i) \ (i = 1, \ldots, t - l) \tag{5.40}$$

and

$$w(\underline{z}_i^{(kt+l)}) \leq \beta^{k+1} w(\hat{\underline{S}}_i) \ (i = t - l + 1, \ldots, t). \tag{5.41}$$

Now 5.39 holds for $k = 0$ since $\underline{z}^{(0)} = \hat{\underline{S}}$. Suppose that 5.39 holds for some $k \geq 0$.

Then 5.40 holds for $l = 0$, and there is no 5.41 for $l = 0$. We shall show that 5.40 and 5.41 both hold for $l = 1$. By 5.35 and 5.39,

$$w(\underline{z}_i^{(kt+1)}) \leq \sum_{j=1}^{t} w(\underline{\hat{r}}_{ij}) w(\underline{z}_j^{(kt)})/2$$

$$\leq \sum_{j=1}^{t} w(\underline{\hat{r}}_{ij}) \beta^k w(\underline{\hat{S}}_j)/2. \qquad 5.42$$

Also by 5.36,

$$\sum_{j=1}^{t} w(\underline{\hat{r}}_{tj}) w(\underline{\hat{H}}'_j)/2 \leq w(\underline{\hat{S}}_t).$$

So by 5.42,

$$w(\underline{z}_t^{(kt+1)}) \leq \sum_{j=1}^{t} w(\underline{\hat{r}}_{tj}) \beta^{k+1} w(\underline{\hat{H}}'_j)/2$$

$$\leq \beta^{k+1} w(\underline{\hat{S}}_t).$$

Also by 5.39 and 5.37, for $i \leq t - 1$,

$$w(\underline{z}_i^{(kt+1)}) \leq \sum_{j=1}^{t} w(\underline{\hat{r}}_{ij}) \beta^k w(\underline{\hat{S}}_j)/2$$

$$\leq \beta^k w(\underline{\hat{S}}_i).$$

So 5.40 and 5.41 both hold for $l = 1$.

Suppose that 5.40 and 5.41 hold for some $l \geq 1$. Then by 5.35,

$$w(\underline{z}_i^{(kt+l+1)}) \leq \sum_{j=1}^{t} w(\hat{\underline{r}}_{ij}) w(\underline{z}_j^{(kt+l)})/2$$

$$= \sum_{j=1}^{i} w(\hat{\underline{r}}_{ij}) w(\underline{z}_j^{(kt+l)})/2 + \sum_{j=i+1}^{t} w(\hat{\underline{r}}_{ij}) w(\underline{z}_j^{(kt+l)})/2.$$

So by 5.40, 5.41, 5.36, for $i \leq t - l - 1$,

$$w(\underline{z}_i^{(kt+l+1)}) \leq \sum_{j=1}^{i} w(\hat{\underline{r}}_{ij}) \beta^k w(\hat{\underline{S}}_j)/2 + \sum_{j=i+1}^{t-l} w(\hat{\underline{r}}_{ij}) \beta^k w(\hat{\underline{S}}_j)/2$$

$$+ \sum_{j=t-l+1}^{t} w(\hat{\underline{r}}_{ij}) \beta^{k+1} w(\hat{\underline{S}}_j)/2$$

$$\leq \sum_{j=1}^{i} w(\hat{\underline{r}}_{ij}) \beta^{k+1} w(\hat{\underline{H}}'_j)/2 + \sum_{j=i+1}^{t} w(\hat{\underline{r}}_{ij}) \beta^k w(\hat{\underline{S}}_j)/2$$

$$\leq \beta^k \{ \sum_{j=1}^{i} w(\hat{\underline{r}}_{ij}) w(\hat{\underline{H}}'_j)/2 + \sum_{j=i+1}^{t} w(\hat{\underline{r}}_{ij}) w(\hat{\underline{S}}_j)/2 \}$$

$$\leq \beta^k w(\hat{\underline{S}}_i).$$

Also for $i \geq t - l$,

$$w(\underline{z}_i^{(kt+l+1)}) \leq \sum_{j=1}^{t-l} w(\hat{\underline{r}}_{ij})\beta^k w(\hat{\underline{S}}_j)/2 + \sum_{j=t-l+1}^{i} w(\hat{\underline{r}}_{ij})\beta^{k+1} w(\hat{\underline{S}}_j)/2$$

$$+ \sum_{j=i+1}^{t} w(\hat{\underline{r}}_{ij})\beta^{k+1} w(\hat{\underline{S}}_j)/2$$

$$\leq \sum_{j=1}^{i} w(\hat{\underline{r}}_{ij})\beta^k w(\hat{\underline{S}}_j)/2 + \sum_{j=i+1}^{t} w(\hat{\underline{r}}_{ij})\beta^{k+1} w(\hat{\underline{S}}_j)/2$$

$$\leq \beta^{k+1}\{\sum_{j=1}^{i} w(\hat{\underline{r}}_{ij}) w(\hat{\underline{H}}'_j)/2 + \sum_{j=i+1}^{t} w(\hat{\underline{r}}_{ij}) w(\hat{\underline{S}}_j)/2\}$$

$$\leq \beta^{k+1} w(\hat{\underline{S}}_i).$$

Therefore by finite induction on l, 5.40 and 5.41 hold for $l = 0, \ldots, t$. In 5.41 set $l = t$, whence 5.39 holds for $k+1$. So by induction on k, 5.39 holds $(\forall k \geq 0)$. Therefore since $0 \leq \beta < 1$, $w(\underline{z}^{(k)}) \to 0$ $(k \to \infty)$. □

Theorem 5.4 : If the hypotheses of **Theorem 5.2** are valid, then $\exists z^* \in \hat{\underline{z}}$ such that $F(z^*) = 0$ and the sequence $(y^{(k)})$ generated from

$$y^{(k+1)} = y^{(k)} - \hat{A}F(y^{(k)}) \ (k \geq 0) \qquad 5.43$$

with $y^{(0)} = \hat{z}$ converges to z^*.

Proof : Let $\hat{\underline{S}} = \underline{S}(\hat{\underline{z}})$ and let $\underline{S}^* \in I(R^t)$ be defined by

$$m(\underline{S}^*) = \hat{z}, \tag{5.44}$$

and

$$w(\underline{S}^*) = w(\hat{\underline{S}}) + 2 \mid \hat{b} \mid, \tag{5.45}$$

where $\hat{b} = \hat{A}F(\hat{z})$.

By 5.28 and the symmetry of $\underline{\hat{R}}$,

$$m(\hat{\underline{S}}) = \hat{z} - \hat{b} \tag{5.46}$$

whence for $i = 1, \ldots, t$,

$$\hat{S}_{iI} = \hat{z}_i - \hat{b}_i - w(\hat{\underline{S}}_i)/2, \tag{5.47}$$

and

$$\hat{S}_{iS} = \hat{z}_i - \hat{b}_i + w(\hat{\underline{S}}_i)/2. \tag{5.48}$$

So by 5.44, 5.45, for $i = 1, \ldots, t$,

$$S^*_{iI} = \hat{z}_i - (w(\hat{\underline{S}}_i) + 2 \mid \hat{b}_i \mid)/2$$

$$= \hat{z}_i - |\hat{b}_i| - w(\underline{\hat{S}}_i)/2$$

$$\leq \hat{S}_{iI},$$

and

$$S^*_{iS} = \hat{z}_i + (w(\underline{\hat{S}}_i) + 2|\hat{b}_i|)/2$$

$$= \hat{z}_i + |\hat{b}_i| + w(\underline{\hat{S}}_i)/2$$

$$\geq \hat{S}_{iS},$$

whence

$$\underline{\hat{S}} \subseteq \underline{S}^*. \qquad 5.49$$

Also, by 5.44, 5.45, for $i = 1, \ldots, t$,

$$|\underline{S}^*_i - \hat{z}_i| = w(\underline{S}^*_i)/2$$

$$= |\hat{b}_i| + w(\underline{\hat{S}}_i)/2,$$

and by 5.47, 5.48, for $i = 1, \ldots, t$,

$$| \underline{\hat{S}}_i - \hat{z}_i | = max\{| \hat{S}_{iI} - \hat{z}_i |, | \hat{S}_{iS} - \hat{z}_i |\}$$

$$= | \hat{b}_i | + w(\underline{\hat{S}}_i)/2,$$

whence

$$| \underline{S}^* - \hat{z} | = | \underline{\hat{S}} - \hat{z} | . \qquad 5.50$$

Now by hypotheses, $\underline{\hat{S}} \subseteq \underline{\hat{z}}$ so $| \underline{\hat{S}} - \hat{z} | \leq | \underline{\hat{z}} - \hat{z} | = w(\underline{\hat{z}})/2$. Therefore by 5.50, $| \underline{S}^* - \hat{z} | \leq w(\underline{\hat{z}})/2$ whence $\underline{S}^* \subseteq \underline{\hat{z}}$. Therefore by 5.49,

$$\underline{\hat{S}} \subseteq \underline{S}^* \subseteq \underline{\hat{z}}. \qquad 5.51$$

We shall use this result to show that $\underline{\hat{K}}(\underline{S}^*) \subseteq \underline{S}^*$. The result which it is required to prove then follows from **Theorem 2.1.**

By 5.31, 5.44,

$$\underline{\hat{K}}(\underline{S}^*) = m(\underline{S}^*) - \hat{A}F(m(\underline{S}^*)) + (I - \hat{A}\underline{F}'(\underline{S}^*))(\underline{S}^* - m(\underline{S}^*))$$

$$= \underline{K}(\underline{S}^*). \qquad 5.52$$

Let

$$\underline{R}^* = I - \hat{A}\underline{F}'(\underline{S}^*).$$

Then $\underline{R}^* \subseteq \hat{\underline{R}}$ because $\underline{S}^* \subseteq \hat{\underline{z}}$. Also by Lemma 5.4(b) and the fact that by hypothesis, $\hat{\underline{S}} \subseteq \hat{\underline{z}}$ we have $\hat{\underline{S}} \subseteq \hat{\underline{H}} \cap \hat{\underline{z}} = \hat{\underline{H}}'$. Therefore $\hat{\underline{S}}' = \hat{\underline{S}} \cap \hat{\underline{H}}' = \hat{\underline{S}}$.

So by symmetry of $\hat{\underline{R}}$, and 5.50,

$$w(\hat{\underline{K}}(\underline{S}^*)) = w(\underline{R}^*(\underline{S}^* - m(\underline{S}^*)))$$

$$= w(\underline{R}^*(\underline{S}^* - \hat{z}))$$

$$\leq w(\hat{\underline{R}}(\underline{S}^* - \hat{z}))$$

$$= w(\hat{\underline{R}}) \mid \underline{S}^* - \hat{z} \mid$$

$$= w(\hat{\underline{R}}) \mid \hat{\underline{S}} - \hat{z} \mid,$$

whence for $i = 1, \ldots, t$, by 5.36,

$$w(\hat{\underline{K}}_i(\underline{S}^*)) \leq \sum_{j=1}^{i} w(\hat{\underline{r}}_{ij}) \mid \hat{\underline{S}}_j - \hat{z}_j \mid$$

$$= \sum_{j=1}^{t} w(\hat{\underline{r}}_{ij}) \mid \hat{\underline{S}}_j - \hat{z}_j \mid + \sum_{j=i+1}^{t} w(\hat{\underline{r}}_{ij}) \mid \hat{\underline{S}}_j - \hat{z}_j \mid$$

$$\leq \sum_{j=1}^{i} w(\hat{r}_{ij}) \mid \underline{\hat{H}}'_j - \hat{z}_j \mid + \sum_{j=i+1}^{l} w(\hat{r}_{ij}) \mid \underline{\hat{S}}'_j - \hat{z}_j \mid$$

$$= w(\underline{\hat{S}}_i)$$

whence $w(\underline{\hat{K}}(\underline{S}^*)) \leq w(\underline{\hat{S}})$. But by 5.46

$$m(\underline{\hat{K}}(\underline{S}^*)) = m(\underline{S}^*) - \hat{A}F(m(\underline{S}^*))$$

$$= \hat{z} - \hat{A}F(\hat{z})$$

$$= \hat{z} - \hat{b}$$

$$= m(\underline{\hat{S}}).$$

Therefore $\underline{\hat{K}}(\underline{S}^*) \subseteq \underline{\hat{S}}$, whence by 5.51, $\underline{\hat{K}}(\underline{S}^*) \subseteq \underline{S}^*$. So by 5.52, $\underline{K}(\underline{S}^*) \subseteq \underline{S}^*$. So by **Theorem 2.1** $\exists z^* \in \underline{K}(\underline{S}^*) \subseteq \underline{S}^* \subseteq \underline{\hat{z}}$ such that $F(z^*) = 0$ and $y^{(k)} \to z^*$ $(k \to \infty)$. □

Theorem 5.5 : If the hypotheses of **Theorem 5.3** are valid and $(y^{(k)})$ is generated from 5.43 with $y^{(0)} \in \underline{S}(\underline{\hat{z}})$ arbitrary then $y^{(k)} \to z^*$ $(k \to \infty)$ where z^* is the unique zero of F in $\underline{\hat{z}}$.

Proof : Let $(\underline{z}^{(k)})$ be generated from 5.33, 5.34. Then as shown in the proof of **Theorem 5.3**, $\exists z^* \in \underline{S}(\hat{\underline{z}})$ such that $F(z^*) = 0$, $z^* \in \underline{z}^{(k)}$ $(\forall k \geq 0)$, and $\underline{z}^{(k)} \to z^*$ $(k \to \infty)$. By 5.33, $y^{(0)} \in \underline{z}^{(0)}$. Suppose that $y^{(k)} \in \underline{z}^{(k)}$ for some $k \geq 0$. Then by **Lemma 5.5**(a), $y^{(k+1)} = P(y^{(k)}) \in \underline{S}(\hat{\underline{z}})$ since $y^{(k)} \in \underline{S}(\hat{\underline{z}})$, and $(\underline{S}(\hat{\underline{z}}) \subseteq \hat{\underline{z}}) \Rightarrow (\underline{S}(\hat{\underline{z}}) \subseteq \underline{H}'(\hat{\underline{z}}))$ by **Lemma 5.4**(d). Also by **Lemma 5.5**(b) and **Lemma 5.4**(a), $y^{(k+1)} = P(y^{(k)}) \in \hat{\underline{K}}(\underline{z}^{(k)}) \subseteq \underline{K}^*(\underline{z}^{(k)})$. Therefore by 5.34, $y^{(k+1)} \in \underline{S}(\hat{\underline{z}}) \cap \underline{K}^*(\underline{z}^{(k)}) = \underline{z}^{(k+1)}$. Therefore by induction on k, $y^{(k)} \in \underline{z}^{(k)}$ $(\forall k \geq 0)$. Therefore $y^{(k)} \to z^*$ $(k \to \infty)$. □

Another existence and uniqueness result which is certainly weaker than the result in **Theorem 1** of [Wol–80a] is contained in the following theorem.

Theorem 5.6 : With the preceding hypotheses and notation, if (1) $\underline{S}(\hat{\underline{z}}) \neq \emptyset$; (2) $\underline{S}(\hat{\underline{z}}) \subseteq \hat{\underline{z}}$; (3) $\left\|\hat{\underline{R}}\right\| < 1$, then $\exists z^* \in \underline{S}(\hat{\underline{z}})$ such that $F(z^*) = 0$ and z^* is unique in $\hat{\underline{z}}$.

Proof : Let the sequence $(\underline{z}^{(k)})$ be generated from

$$\underline{z}^{(0)} = \underline{S}(\hat{\underline{z}}), \tag{5.53}$$

$$\underline{z}^{(k+1)} = \underline{S}(\hat{\underline{z}}) \cap \underline{K}^*(\underline{z}^{(k)}) \ (k \geq 0). \tag{5.54}$$

By **Theorem 5.2**, $\exists z^* \in \underline{z}^{(0)}$ such that $F(z^*) = 0$. Suppose that for some $k \geq 0$, $z^* \in \underline{z}^{(k)}$. Then $z^* \in \underline{K}(\underline{z}^{(k)}) \subseteq \underline{K}^*(\underline{z}^{(k)})$. Therefore $z^* \in \underline{z}^{(k+1)}$, whence by

induction on k, $z^* \in \underline{z}^{(k)}$ $(\forall k \geq 0)$.

Also,

$$\left\| w(\underline{z}^{(k+1)}) \right\| \leq \left\| w(\underline{K}^*(\underline{z}^{(k)})) \right\|$$

$$\leq \left\| \underline{\hat{R}} \right\| \left\| w(\underline{z}^{(k)}) \right\| \quad (k \geq 0)$$

whence $w(\underline{z}^{(k)}) \to 0$ $(k \to \infty)$ because $\left\| \underline{\hat{R}} \right\| < 1$. Therefore since $z^* \in \underline{z}^{(k)}$ $(\forall k \geq 0)$ it follows that $\underline{z}^{(k)} \to z^*$ $(k \to \infty)$. Therefore z^* is unique in $\hat{\underline{z}}$. □

Theorem 5.7 : If the hypotheses of **Theorem 5.6** are valid and $(y^{(k)})$ is generated from 5.43 with $y^{(0)} \in \underline{S}(\hat{\underline{z}})$ arbitrary then $y^{(k)} \to z^*$ $(k \to \infty)$ where z^* is the unique zero of F in $\hat{\underline{z}}$.

Proof : Let $(\underline{z}^{(k)})$ be generated from 5.53, 5.54. Then $y^{(0)} \in \underline{z}^{(0)}$. Suppose that for some $k \geq 0$, $y^{(k)} \in \underline{z}^{(k)}$. Then $y^{(k+1)} = P(y^{(k)}) \in \underline{S}(\hat{\underline{z}})$. Also $y^{(k+1)} \in \underline{K}(\underline{z}^{(k)}) \subseteq \underline{K}^*(\underline{z}^{(k)})$, so $y^{(k+1)} \in \underline{z}^{(k+1)}$, whence by induction on k, $y^{(k)} \in \underline{z}^{(k)}$ $(\forall k \geq 0)$. But by **Theorem 5.6**, $\underline{z}^{(k)} \to z^*$ $(k \to \infty)$. So $y^{(k)} \to z^*$ $(k \to \infty)$. □

The following results use the notation corresponding to $KMSW$ which is given in §5.4.

Theorem 5.8 : Let $F : D \subseteq R^t \to R^t$ be a given mapping, with $F \in C^1(\hat{D})$, where $\hat{D} \subseteq D$ is an open convex set. Let $\underline{F}' : I(\hat{D}) \to I(M(R^t))$ be a continuous inclusion monotonic interval extension of the derivative $F' : \hat{D} \to M(R^t)$ of F. Let $\underline{z}^{(0)} \in I(\hat{D})$ be given. If (1) $B^{(0)}$ exists; (2) $\underline{S}^{(0,0)} \subseteq \underline{z}^{(0)}$ and $\underline{S}^{(0,0)} \neq \emptyset$; (3)

$w(\underline{S}^{(0,0)}) < w(\underline{H}'^{(0,0)})$, then $\exists z^* \in \underline{S}^{(0,0)}$ such that $F(z^*) = 0$, z^* is unique in $\underline{z}^{(0)}$, and if $(\underline{z}^{(k)})$ is generated from $KMSW$ then $z^* \in \underline{z}^{(k+1)} \subseteq \underline{z}^{(k)}$ $(\forall k \geq 0)$. Furthermore if (4) $r^{(0)} < 1$, then $\underline{z}^{(k)} \to z^*$ $(k \to \infty)$ and

$$\left\| w(\underline{z}^{(k+1)}) \right\| \leq (r^{(0)})^{p^{(k)}+1} \left\| w(\underline{z}^{(k)}) \right\| \quad (\forall k \geq 0) \qquad 5.55$$

where $p^{(k)} \geq 0$ $(\forall k \geq 0)$.

Proof : The existence and uniqueness results follow from Lemmas 5.4, 5.5 and Theorems 5.2 and 5.3. It remains to show that $z^* \in \underline{z}^{(k+1)} \subseteq \underline{z}^{(k)}$ $(\forall k \geq 0)$, that $\underline{z}^{(k)} \to z^*$ $(k \to \infty)$, and that 5.55 holds.

By Lemmas 5.4, 5.5 and Theorems 5.2 and 5.3, $z^* \in \underline{S}^{(0,0)} \subseteq \underline{H}^{(0,0)}$. Also $z^* \in \underline{z}^{(0,0)}$. Suppose that for some $k \geq 0$ and some $m \geq 0$, $z^* \in \underline{z}^{(k,m)}$, and $z_j^* \in \underline{H}_j^{(k,m)}$ $(j = 1, \ldots, i-1)$ for some $i \geq 2$. Then $z_j^* \in \underline{H}'^{(k,m)}_j$ $(j = 1, \ldots, i-1)$ and so by step 8.2.3.1 of $KMSW$, $z_i^* \in \underline{H}_i^{(k,m)}$. A similar argument shows that $z_1^* \in \underline{H}_1^{(k,m)}$. Therefore by finite induction on i, $z^* \in \underline{H}^{(k,m)}$, whence $z^* \in \underline{H}'^{(k,m)}$.

Suppose that for some $i \leq t-1$, $z_j^* \in \underline{S}_j^{(k,m)}$ $(j = i+1, \ldots, t)$. Then $z_j^* \in \underline{S}'^{(k,m)}_j$ $(j = i+1, \ldots, t)$, whence by step 8.2.4.1 of $KMSW$, $z_i^* \in \underline{S}_i^{(k,m)}$. A similar argument shows that $z_t^* \in \underline{S}_t^{(k,m)}$. Therefore by finite induction on i, $z^* \in \underline{S}^{(k,m)}$, whence $z^* \in \underline{S}'^{(k,m)}$. Therefore by finite induction on m, $(z^* \in \underline{z}^{(k)} = \underline{z}^{(k,0)}) \Rightarrow (z^* \in \underline{z}^{(k+1)})$, whence by induction on k, $z^* \in \underline{z}^{(k)}$ $(\forall k \geq 0)$, and so, by inspection of $KMSW$, $z^* \in \underline{z}^{(k+1)} \subseteq \underline{z}^{(k)}$ $(\forall k \geq 0)$.

Now by steps 8.2.3.1 and 8.2.3.2 of $KMSW$, for $m = 0, \ldots, p^{(k)}$ $(\forall k \geq 0)$,

$$w(\underline{H}'^{(k,m)}) \leq | \underline{R}^{(k)} | \, w(\underline{z}^{(k,m)}),$$

and by step 8.2.4.1 of $KMSW$, for $m = 0, \ldots, p^{(k)}$ $(\forall k \geq 0)$

$$w(\underline{S}^{(k,m)}) \leq | \underline{R}^{(k)} | \, w(\underline{z}^{(k,m)}),$$

whence for $m = 0, \ldots, p^{(k)}$ $(\forall k \geq 0)$

$$w(\underline{z}^{(k,m)}) \leq | \underline{R}^{(k)} |^m \, w(\underline{z}^{(k,0)}).$$

Therefore $(\forall k \geq 0)$

$$w(\underline{z}^{(k+1)}) \leq | \underline{R}^{(k)} |^{p^{(k)}+1} \, w(\underline{z}^{(k)}),$$

whence $(\forall k \geq 0)$

$$\left\| w(\underline{z}^{(k+1)}) \right\| \leq \left\| \underline{R}^{(k)} \right\|^{p^{(k)}+1} \left\| w(\underline{z}^{(k)}) \right\|. \qquad 5.56$$

Now by step 8.8 of $KMSW$, $\left\| \underline{R}^{(k+1)} \right\| \leq \left\| \underline{R}^{(k)} \right\|$ $(\forall k \geq 0)$, so 5.55 holds. Therefore if $r^{(0)} < 1$ then $\underline{z}^{(k)} \to z^*$ $(\forall k \geq 0)$. □

The following Theorem shows that under very easily satisfied hypotheses, the sequence $(\underline{z}^{(k)})$ generated from $KMSW$ can converge very rapidly.

Theorem 5.9 : If hypotheses (1) - (4) of Theorem 5.8 are valid and if also (5) $\exists \lambda > 0$ such that $\|w(\underline{F}'(\underline{z}))\| \leq \lambda \|w(\underline{z})\|$ $(\forall \underline{z} \in I(\hat{D}))$; (6) $\exists \mu > 0$ such that $A^{(k)} = \{m(\underline{F}'(\underline{z}^{(k)}))\}^{-1} + E^{(k)}$, where $\|E^{(k)}\| \leq \mu \|w(\underline{z}^{(k)})\|$ $(\forall k \geq 0)$, then $(\forall k \geq 0)$, $\exists \alpha^{(k)} > 0$ such that

$$\left\|w(\underline{z}^{(k+1)})\right\| \leq \alpha^{(k)} \left\|w(\underline{z}^{(k)})\right\|^{p^{(k)}+2}.$$

Proof : By the definition of $r^{(k)}$ and hypotheses (5) and (6), $(\forall k \geq 0)$

$$r^{(k)} = \left\|I - A^{(k)} \underline{F}'(\underline{z}^{(k)})\right\|$$

$$\leq \left\|E^{(k)} \underline{F}'(\underline{z}^{(k)})\right\| + \left\|\{m(\underline{F}'(\underline{z}^{(k)}))\}^{-1}\right\| \left\|w(\underline{F}'(\underline{z}^{(k)}))\right\| /2$$

$$\leq \{\mu \left\|\underline{F}'(\underline{z}^{(0)})\right\| + \lambda\nu/2\} \left\|w(\underline{z}^{(k)})\right\|, \qquad 5.57$$

where

$$\nu = sup\{\|A^{-1}\| \mid A \in \underline{F}'(\underline{z}^{(0)})\}.$$

Let

$$\alpha^{(k)} = \{\mu \left\|\underline{F}'(\underline{z}^{(0)})\right\| + \lambda\nu/2\}^{p^{(k)}+1} \ (k \geq 0).$$

Then by 5.43, 5.44, $(\forall k \geq 0)$

$$\left\|w(\underline{z}^{(k+1)})\right\| \leq \alpha^{(k)} \left\|w(\underline{z}^{(k)})\right\|^{p^{(k)}+2}. \ \square$$

CHAPTER 6

The Algorithm MW

In [Han–80a], Hansen has described an algorithm for computing the global minimizer of a function $f : R^n \rightarrow R^1$ in a box $\underline{\hat{x}} \in I(R^n)$; this is described in Chapter 3. In [Rob–73a], Robinson has presented a method for obtaining computable bounds for the error in an approximate Kuhn-Tucker point of $NP.1$.

In this chapter, we solve Hansen's global optimization problem (*Problem P*) by expressing it as a system of nonlinear algebraic equations and inequalities (problem $NP.2$), in a manner similar to that which has been used by Robinson. The algorithm which is described in this chapter is referred to as MW.

We solve the system of nonlinear algebraic equations and inequalities by using the methods which are described in Chapter 5. We also use the deletion strategies which are described in Chapter 3 and in [IchF-79a].

6.1 The Constituent Parts of The Algorithm MW

In this section the constituent parts of the algorithm MW for bounding a solution of $NP.2$ in a box $\underline{\hat{x}} \in I(R^n)$ are described briefly.

A KT point $z^* = (x^{*T}, u^{*T})^T \in (\hat{x}^T, \hat{u}^T)^T$ for $NP.2$ satisfies $F(z) = 0$ where $F : R^{3n} \rightarrow R^{3n}$ is defined by 4.33. Now $x^* \in \hat{x}$ but a box $\hat{u} \in I(R^{2n})$ which contains $u^* \in R^{2n}$ must be determined. A method for determining $\hat{u}$ is described in §6.2.

After the box $\hat{z} = (\hat{x}^T, \hat{u}^T)^T \in I(R^{3n})$ has been determined, sub-boxes $\hat{z}^{(i)}$ $(0 \leq i \leq 2^n - 1)$ of $\hat{z}$ which might contain z^* are constructed. The determination of the sub-boxes $\hat{z}^{(i)}$ $(0 \leq i \leq 2^n - 1)$ is discussed in §6.3.

In order to delete sub-boxes z of $\hat{z}$ which do not contain a KT point $z^* = (x^{*T}, u^{*T})^T$ corresponding to a global minimizer the procedures which are described in §6.4 - §6.9 and §6.13 are used.

An interval $\overline{f} \in I(R)$ containing an upper bound on $f^* = f(x^*)$ is determined and is updated continually as explained in §6.4, §6.6, §6.8 and §6.13. The interval $\overline{f}$ is used to delete sub-boxes x of $\hat{x}$ such that $x^* \notin x$. If $f(x) > \overline{f}_S$ $(\forall x \in x)$ then $f(x) > f^*$ $(\forall x \in x)$ so $x^* \notin x$. Therefore $z^* \notin z = (x^T, u^T)^T$ and z may be deleted from $\hat{z}$.

If $z \subseteq \hat{z}$ and $0 \notin F(z)$ then $z^* \notin z$ and therefore z can be deleted from $\hat{z}$; a procedure for deleting sub-boxes of $\hat{z}$ based on this idea is described in §6.5.

A deletion procedure based upon the ideas which are explained in §3.2 is described in §6.6.

Hansen [Han–80a] has described a procedure for deleting sub-boxes of $\hat{x} \in I(R^n)$ which do not contain x^* by using a quadratic interval extension of $f : R^n \rightarrow R^1$. A modified form of Hansen's procedure is used in MW and is described in §6.7.

Shearer and Wolfe [SheW-85a] have described some computable existence, uniqueness, and convergence results for systems of nonlinear algebraic equations which involve the so-called Symmetric operator. Associated with the tests which are described by Shearer and Wolfe [SheW-85a] are certain non-existence tests whereby it might be possible to determine that $z^* \notin \underline{z} \subseteq \hat{\underline{z}}$, in which case $\underline{z}$ may be deleted from $\hat{\underline{z}}$. One of the convergence tests also permits the bounding interval $\underline{f}$ of f^* to be updated. The use of the Symmetric operator is considered in §6.8.

A deletion procedure based upon the non-convexity of f in $\underline{x}$ which is explained in §3.3 is used in *MW* as described in §6.9.

As explained in Chapter 4, if $\exists z^* \in \underline{z}$ such that $F(z^*) = 0$ and $\underline{z}$ satisfies the test 4.43 then strict complementary slackness holds at z^* and z^* satisfies the KT first order necessary conditions for a minimizer or a maximizer of f. The application of the test 4.43 for strict complementary slackness is described in §6.10.

Let $\underline{F} : I(R^{3n}) \rightarrow I(R^{3n})$ be a continuous inclusion monotonic interval extension of $F : R^{3n} \rightarrow R^{3n}$, where F is defined by 4.33. In order to locate the KT point $z^* = (x^{*^T}, u^{*^T})^T$, the system $F(z) = 0$ is solved by using one of the algorithms *MAP* and *KMSW* (Chapter 5) as described in §6.11.

If strict complementary slackness does not hold over $\underline{z} = (\underline{x}^T, \underline{u}^T) \subseteq \hat{\underline{z}}$ or if $m(\underline{F}'(\underline{z}))$ is singular where $\underline{F}' : I(R^{3n}) \rightarrow I(M(R^{3n}))$ is a continuous inclusion monotonic interval extension of $F' : R^{3n} \rightarrow M(R^{3n})$ defined by 4.35, or in solving $F(z) = 0$, $\underline{z}$ is not reduced to a small box with $\|w(\underline{z})\| \leq \varepsilon_0$ where $\varepsilon_0 > 0$ is given, then $\underline{z}$ is bisected as explained in [Jon–78a]; details are given in §6.12.

If $\underline{z} = (\underline{x}^T, \underline{u}^T)^T \subseteq \hat{\underline{z}}$ has been bisected into two sub-boxes $\underline{z}^{(1)} = (\underline{x}^{(1)^T}, \underline{u}^{(1)^T})^T$ and $\underline{z}^{(2)} = (\underline{x}^{(2)^T}, \underline{u}^{(2)^T})^T$ then we compute $\underline{f}^{(1)} = \underline{f}(\underline{x}^{(1)})$ and $\underline{f}^{(2)} = \underline{f}(\underline{x}^{(2)})$. By comparing $\bar{f}$, $\underline{f}^{(1)}$, and $\underline{f}^{(2)}$, either $\underline{z}^{(1)}$ or $\underline{z}^{(2)}$ or neither can be deleted. Also $\underline{z}$ may be deleted by using $\bar{f}$ if $\underline{z}$ is not bisected. The deletion strategies based upon these ideas are described in §6.13.

Most of the techniques which are mentioned in the preceding part of this section involve the reduction of the component $\underline{x}$ of $\underline{z}$ where $\underline{z} = (\underline{x}^T, \underline{u}^T)^T \subseteq \hat{\underline{z}}$. If $\underline{x}$ is reduced then the bounds on the Lagrange multipliers corresponding to $\underline{x}$ should be recomputed; the method which is used to do this is described in §6.14. The method which is used to terminate algorithm MW, is described in §6.15. The pseudo-code form for algorithm MW is given in §6.16.

6.2 Bounding the Lagrange Multipliers

Suppose that $\hat{\underline{z}} = (\hat{\underline{x}}^T, \hat{\underline{u}}^T)^T$ is a box which is assumed to contain a KT point $z^* = (x^{*^T}, u^{*^T})^T$ for *NP*.2 where $\hat{\underline{x}} = (\hat{\underline{x}}_1, \ldots, \hat{\underline{x}}_n)^T$ and $\hat{\underline{u}} = (\hat{\underline{u}}_1, \ldots, \hat{\underline{u}}_{2n})^T$. We are given $\hat{\underline{x}}$ but not $\hat{\underline{u}}$. However, by using the interval extension of the derivative of the objective function $f : R^n \rightarrow R^1$ we can determine intervals $\hat{\underline{u}}_1, \ldots, \hat{\underline{u}}_{2n}$ which contain the corresponding Lagrange multipliers.

For *NP*.2 we have two cases, namely (i) $x^* \in int(\hat{\underline{x}})$ and (ii) $x^* \in \partial(\hat{\underline{x}})$ where $\partial(\hat{\underline{x}})$ is the boundary of $\hat{\underline{x}}$. In (i) $\underline{u}_i = \underline{0}$ $(i = 1, \ldots, 2n)$ because no constraint is active at x^*. Therefore we need solve only

$$F(x) = 0 \tag{6.1}$$

where

$$F(x) = (\partial_i f(x))_{n\times 1} \tag{6.2}$$

and

$$F'(x) = (\partial_j \partial_i f(x))_{n\times n}. \tag{6.3}$$

In (ii), $x^* \in \partial(\hat{\underline{x}})$, so some of the Lagrange multipliers might be non-zero.

We shall derive an algorithm for obtaining the initial bounding intervals for the Lagrange multipliers.

Consider the first n equations of 4.33, namely

$$\partial_i f(z_1, \ldots, z_n) - z_{n+i} + z_{2n+i} = 0 \ (i = 1, \ldots, n) \tag{6.4}$$

which is equivalent to

$$u_i - u_{i+n} = \partial_i f(x) \ (i = 1, \ldots, n). \tag{6.5}$$

If $x \in \hat{\underline{x}}$ then by 6.5

$$u_i - u_{i+n} \in \partial_i \underline{f}(\hat{\underline{x}}) \ (i = 1, \ldots, n). \tag{6.6}$$

So if $\hat{\underline{d}}_i = \partial_i \underline{f}(\hat{\underline{x}}) \ (i = 1, \ldots, n)$ then by 6.6

$$\hat{d}_{iI} \leq u_i - u_{i+n} \leq \hat{d}_{iS} \ (i = 1, \ldots, n). \tag{6.7}$$

Furthermore, we know that for $i = 1, \ldots, n$

$$(u_i > 0) \Rightarrow (u_{i+n} = 0), \tag{6.8}$$

that

$$(u_{i+n} > 0) \Rightarrow (u_i = 0), \tag{6.9}$$

and that

$$u_i \geq 0 \ (i = 1, \ldots, 2n). \tag{6.10}$$

Suppose that

$$\hat{d}_{iS} \leq 0. \tag{6.11}$$

Then by 6.7,

$$u_i \leq u_{i+n}. \tag{6.12}$$

Suppose that $u_i > 0$. Then by 6.12 $u_{i+n} > 0$, contradicting 6.8. Therefore if 6.11 holds then by 6.10 $u_i = 0$. Therefore by 6.7, if 6.11 holds then

$$\hat{d}_{iI} \leq -u_{i+n} \leq \hat{d}_{iS} \leq 0,$$

whence by 6.10, $u_{i+n} \in [-\hat{d}_{iS}, -\hat{d}_{iI}] = -\underline{\hat{d}}_i$. Therefore

$$(\hat{d}_{iS} \leq 0) \Rightarrow (u_i = 0 \wedge u_{i+n} \in -\underline{\hat{d}}_i). \qquad 6.13$$

Suppose that

$$\hat{d}_{iI} < 0 < \hat{d}_{iS}.$$

If $u_i > 0$ then by 6.8, $u_{i+n} = 0$, so by 6.7,

$$\hat{d}_{iI} < 0 < u_i \leq \hat{d}_{iS}$$

whence $u_i \in (0, \hat{d}_{iS}]$. So by 6.10, $u_i \in \{0\} \cup (0, \hat{d}_{iS}] = [0, \hat{d}_{iS}]$.

If $u_{i+n} > 0$ then by 6.9, $u_i = 0$ so by 6.7,

$$\hat{d}_{iI} \leq -u_{i+n} < 0 < \hat{d}_{iS}$$

whence $u_{i+n} \in (0, -\hat{d}_{iI}]$. So by 6.10, $u_{i+n} \in \{0\} \cup (0, -\hat{d}_{iI}] = [0, -\hat{d}_{iI}]$. Therefore

$$(\hat{d}_{iI} < 0 < \hat{d}_{iS}) \Rightarrow (u_i \in [0, \hat{d}_{iS}] \wedge u_{i+n} \in [0, -\hat{d}_{iI}]). \qquad 6.14$$

Finally, suppose that

$$0 \leq \hat{d}_{iI}.$$

Then by 6.7,

$$0 \leq \hat{d}_{iI} \leq u_i - u_{i+n} \leq \hat{d}_{iS}. \qquad 6.15$$

If $u_i > 0$ then by 6.8, $u_{i+n} = 0$ so by 6.15,

$$0 \leq \hat{d}_{iI} \leq u_i \leq \hat{d}_{iS},$$

whence if $u_i > 0$ then $u_i \in \underline{\hat{d}}_i$.

If $u_{i+n} > 0$ then by 6.9, $u_i = 0$ so by 6.15, $0 \leq -u_{i+n}$ whence by 6.10 $u_{i+n} = 0$ contrary to the hypothesis that $u_{i+n} > 0$. So $u_{i+n} = 0$. Therefore

$$(0 \leq \hat{d}_{iI}) \Rightarrow (u_i \in \underline{\hat{d}}_i \wedge u_{i+n} = 0). \qquad 6.16$$

If some of the multipliers are zero then we can solve $NP.2$ with less variables so that some of the rows of $F(z)$ in 4.33 and of $F'(z)$ in 4.35 can be ignored. Since some

of the multipliers might be zero, we introduce the parameters l and q_i $(i = 1, \ldots, 2n)$ such that

$$q_i = \begin{cases} 1 & (\underline{u}_i \neq \underline{0}) \\ 0 & (\underline{u}_i = \underline{0}) \end{cases} \qquad 6.17$$

and l is the number of Lagrange multipliers which take the value zero, so that we know whether or not we can reduce the number of variables in $\underline{F}$ and $\underline{F}'$.

Suppose that $x^* \in int(\underline{\hat{x}})$. Then by 4.33 and 4.37 $u_i^* = 0$ $(i = 1, \ldots, 2n)$ because no constraint is active at x^*. Therefore if $x^* \in int(\underline{\hat{x}})$ then by 6.17 we can take $\underline{\hat{u}}_i = \underline{0}$, $\hat{q}_i = 0$ $(i = 1, \ldots, 2n)$ and $\hat{l} = 2n$.

Suppose that $x^* \in \partial(\underline{\hat{x}})$. Let $x_i^* = \hat{x}_{iI}$ $(i \in \{1, \ldots, n\})$. Then $\partial_i f(x^*) \geq 0$ and $\partial_i f(x^*) \in \partial_i \underline{f}(\underline{\hat{x}}) = \underline{\hat{d}}_i$. If $0 \leq \hat{d}_{iI}$ then by 6.16 we may take $\underline{\hat{u}}_i = \underline{\hat{d}}_i$ and $\underline{\hat{u}}_{i+n} = \underline{0}$, so by 6.17 we obtain $\hat{q}_i = 1$ and $\hat{q}_{i+n} = 0$. If $\hat{d}_{iI} < 0 < \hat{d}_{iS}$ then by 6.14 we may take $\underline{\hat{u}}_i = [0, \hat{d}_{iS}]$ and $\underline{\hat{u}}_{i+n} = [0, -\hat{d}_{iI}]$, so by 6.17 we obtain $\hat{q}_i = 1$ and $\hat{q}_{i+n} = 1$. If $\hat{d}_{iS} \leq 0$ then $\partial_i f(x) \leq 0$ $(\forall x \in \underline{\hat{x}})$ which occurs only in cases such as

$$f(x) = x_1^2, \ \underline{\hat{x}} = ([0,1],[0,1])^T \ with \ i = 2.$$

For this problem, $\partial_2 f(x) = 0$ $(\forall x \in \underline{\hat{x}})$, so

$$\underline{\hat{d}}_2 = \underline{0},$$

whence $\hat{d}_{2S} = 0$. So $\underline{\hat{u}}_2 = \underline{0} \wedge \underline{\hat{u}}_4 = -\underline{\hat{d}}_2 = \underline{0}$. However, if $x^* \in \partial_I^{(i)}(\underline{\hat{x}}) = \{x \in \underline{\hat{x}} \mid x_i = \hat{x}_{iI}\}$ is a strong local minimizer of f then $(\forall i \in \{1,\ldots,n\})$ $\partial_i f(x_1^*,\ldots,x_{i-1}^*,t,x_{i+1}^*,\ldots,x_n^*) > 0$ for at least one value of $t \in \underline{\hat{x}}_i$ for if not, then $\partial_i f(x_1^*,\ldots,x_{i-1}^*,t,x_{i+1}^*,\ldots,x_n^*) \leq 0$ $(\forall t \in \underline{\hat{x}}_i)$ whence

$$f(x_1^*,\ldots,x_{i-1}^*,x,x_{i+1}^*,\ldots,x_n^*) = f(x^*) + \int_{\hat{x}_{iI}}^{x} \partial_i f(x_1^*,\ldots,x_{i-1}^*,t,x_{i+1}^*,\ldots,x_n^*)dt$$

$$\leq f(x^*) \ (\forall x \in \underline{\hat{x}}_i)$$

so x^* is not a strong local minimizer of f in $\underline{\hat{x}}$. So the case $\hat{d}_{iS} \leq 0$ does <u>not</u> occur when x^* is a strong local minimizer of f. Thus in practice if x^* is a strong local minimizer of f and $x_i^* = \hat{x}_{iI}$ then we always obtain $\hat{q}_i = 1$. In other words, if $\hat{q}_i \neq 1$ then $x_i^* \neq \hat{x}_{iI}$. A similar argument is valid when $x_i^* = \hat{x}_{iS}$.

Suppose that $\underline{x} \subset int(\underline{\hat{x}})$. Then by 4.18 and 4.19 $(\forall x \in \underline{x})$ $c_i(x) > 0$ $(i = 1,\ldots,2n)$ whence by 4.33 and 4.37 $u_i = 0$ $(i = 1,\ldots,2n)$ so by 6.17 we obtain $q_j = 0$ $(i = 1,\ldots,2n)$.

Therefore

$$(x^* \in int(\underline{\hat{x}})) \Rightarrow (\hat{q}_i = 0 \ (i = 1,\ldots,2n));$$

$$(x_i^* = \hat{x}_{iI} \wedge 0 \leq \partial_i \underline{f}(\underline{\hat{x}})) \Rightarrow (\hat{q}_i = 1 \wedge \hat{q}_{i+n} = 0);$$

$$(x_i^* = \hat{x}_{iS} \wedge \partial_i \underline{f}(\underline{\hat{x}}) \leq 0) \Rightarrow (\hat{q}_i = 0 \wedge \hat{q}_{i+n} = 1);$$

$$(\underline{x} \subset int(\underline{\hat{x}})) \Rightarrow (q_i = 0 \ (i = 1, \ldots, 2n)).$$

The preceding ideas give rise to the procedure *lagrange.multipliers* for determining $\underline{\hat{u}}_i$, $\underline{\hat{u}}_{i+n}$ $(i = 1, \ldots, n)$, the corresponding values of $\hat{q}_1, \ldots, \hat{q}_{2n}$, and $\hat{l}$.

procedure *lagrange.multipliers*$(\underline{\hat{d}} \in I(R^n), n \in N; \underline{\hat{u}} \in I(R^{2n}), \hat{q} \in N^{2n}, \hat{l} \in N)$

! This procedure returns a set of intervals $\underline{\hat{u}}_i$ $(i = 1, \ldots, 2n)$ which contain the
! Lagrange multipliers $\hat{u}_i$ $(i = 1, \ldots, 2n)$ for $NP.2$ corresponding to $\underline{\hat{x}} \in I(R^n)$, and
! the corresponding values $\hat{q}_i$ $(i = 1, \ldots, 2n)$ and $\hat{l}$.
! On entry $\underline{\hat{d}} = \underline{f}'(\underline{\hat{x}})^T$, $\underline{\hat{u}}_i = \underline{0}$ $(i = 1, \ldots, 2n)$, $\hat{q}_i = 1$ $(i = 1, \ldots, 2n)$, and $\hat{l} = 0$.
! On return, if $\underline{\hat{u}}_i = \underline{0}$ for some i $(i = 1, \ldots, 2n)$ then $\hat{q}_i = 0$; otherwise $\hat{q}_i = 1$. Also
! $\hat{l}$ is the number of Lagrange multipliers which take the value zero.

1. *for* $i = 1$ *to* n *do*

   1.1. *case* *true* *of*

   $\hat{d}_{iS} \leq 0$:

   1.1.1. $\hat{q}_i := 0$! $\underline{\hat{u}}_i = \underline{0}$.

1.1.2. $\hat{l} := \hat{l} + 1$

1.1.3. $\underline{\hat{u}}_{i+n} := -\underline{\hat{d}}_i$

$\hat{d}_{iI} < 0$ *and* $0 < \hat{d}_{iS}$:

1.1.4. $\underline{\hat{u}}_i := [0, \hat{d}_{iS}]$

1.1.5. $\underline{\hat{u}}_{i+n} := [0, -\hat{d}_{iI}]$

default :

1.1.6. $\underline{\hat{u}}_i := \underline{\hat{d}}_i$

1.1.7. $\hat{q}_{i+n} := 0$! $\underline{\hat{u}}_{i+n} = \underline{0}$.

1.1.8. $\hat{l} := \hat{l} + 1$

2. *return* □

6.3 The Determination of Sub-boxes Which Might Contain Kuhn-Tucker Points for *NP*.2

We shall derive a method for obtaining sub-boxes $\underline{\hat{z}}^{(i)}$ $(0 \leq i \leq 2^n - 1)$ of $\underline{\hat{z}}$ which might contain the KT point $z^* = (x^{*T}, u^{*T})^T$ corresponding to a global minimizer x^* of f in $\underline{\hat{x}}$. Let us consider the cases corresponding to $n = 1, 2, 3$.

(a) $n = 1$

In Fig 6.1 $A - H$ are the corners of the box $\underline{\hat{z}} = (\underline{\hat{x}}_1, \underline{\hat{u}}_1, \underline{\hat{u}}_2)^T$, where $\underline{\hat{u}}_1$ and $\underline{\hat{u}}_2$ are determined as in §6.2. Now

(*i*) $$(x^* \in int(AB)) \Rightarrow (u_1 = 0 \wedge u_2 = 0)$$

$$\Rightarrow (\hat{\underline{z}} = (\underline{\hat{x}}_1, \underline{0}, \underline{0})^T),$$

(*ii*) $$(x^* = \hat{x}_{1I}) \Rightarrow (u_1 \geq 0 \wedge u_2 = 0)$$

$$\Rightarrow (\hat{\underline{z}} = ([\hat{x}_{1I}, \hat{x}_{1I}], \underline{\hat{u}}_1, \underline{0})^T),$$

(*iii*) $$(x^* = \hat{x}_{1S}) \Rightarrow (u_1 = 0 \wedge u_2 \geq 0)$$

$$\Rightarrow (\hat{\underline{z}} = ([\hat{x}_{1S}, \hat{x}_{1S}], \underline{0}, \underline{\hat{u}}_2)^T).$$

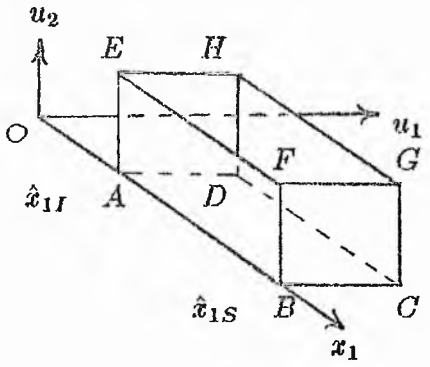


Fig 6.1

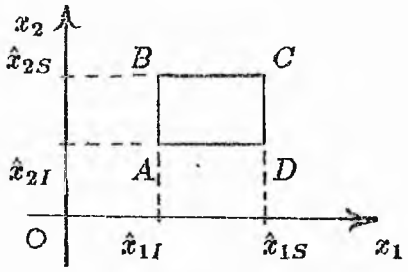


Fig 6.2

Therefore the KT point z^* might lie on AB, AD, or BF. So there exists 2^1 sub-boxes $\hat{\underline{z}}^{(0)}$, $\hat{\underline{z}}^{(1)}$ of $\hat{\underline{z}}$ which might contain the KT point z^*, and these are given by

$$\hat{\underline{z}}^{(0)} = ([m(\hat{\underline{x}}_1), \hat{x}_{1S}], \underline{0}, \underline{\hat{u}}_2)^T, \tag{6.18}$$

and

$$\hat{\underline{z}}^{(1)} = ([\hat{x}_{1I}, m(\hat{\underline{x}}_1)], \hat{\underline{u}}_1, \underline{0})^T. \tag{6.19}$$

(b) $n = 2$

For the case n=2, which is illustrated in Fig 6.2,

$$(i) \qquad (x^* \in int(ABCD)) \Rightarrow (u_1 = 0 \wedge u_2 = 0 \wedge u_3 = 0 \wedge u_4 = 0)$$

$$\Rightarrow (\underline{\hat{z}} = (\underline{\hat{x}}_1, \underline{\hat{x}}_2, \underline{0}, \underline{0}, \underline{0}, \underline{0})^T),$$

$$(ii) \qquad (x^* \in int(AB)) \Rightarrow (u_1 \geq 0 \wedge u_2 = 0 \wedge u_3 = 0 \wedge u_4 = 0)$$

$$\Rightarrow (\underline{\hat{z}} = ([\hat{x}_{1I}, \hat{x}_{1I}], \underline{\hat{x}}_2, \underline{\hat{u}}_1, \underline{0}, \underline{0}, \underline{0})^T),$$

$$(iii) \qquad (x^* \in int(AD)) \Rightarrow (u_1 = 0 \wedge u_2 \geq 0 \wedge u_3 = 0 \wedge u_4 = 0)$$

$$\Rightarrow (\underline{\hat{z}} = (\underline{\hat{x}}_1, [\hat{x}_{2I}, \hat{x}_{2I}], \underline{0}, \underline{\hat{u}}_2, \underline{0}, \underline{0})^T),$$

$$(iv) \qquad (x^* \in int(CD)) \Rightarrow (u_1 = 0 \wedge u_2 = 0 \wedge u_3 \geq 0 \wedge u_4 = 0)$$

$$\Rightarrow (\underline{\hat{z}} = ([\hat{x}_{1S}, \hat{x}_{1S}], \underline{\hat{x}}_2, \underline{0}, \underline{0}, \underline{\hat{u}}_3, \underline{0})^T),$$

$$(v) \qquad (x^* \in int(BC)) \Rightarrow (u_1 = 0 \wedge u_2 = 0 \wedge u_3 = 0 \wedge u_4 \geq 0)$$

$$\Rightarrow (\underline{\hat{z}} = (\underline{\hat{x}}_1, [\hat{x}_{2S}, \hat{x}_{2S}], \underline{0}, \underline{0}, \underline{0}, \underline{\hat{u}}_4)^T),$$

$$(vi) \qquad (x^* = A) \Rightarrow (u_1 \geq 0 \wedge u_2 \geq 0 \wedge u_3 = 0 \wedge u_4 = 0)$$

$$\Rightarrow (\underline{\hat{z}} = ([\hat{x}_{1I}, \hat{x}_{1I}], [\hat{x}_{2I}, \hat{x}_{2I}], \underline{\hat{u}}_1, \underline{\hat{u}}_2, \underline{0}, \underline{0})^T),$$

$$(vii) \qquad (x^* = B) \Rightarrow (u_1 \geq 0 \wedge u_2 = 0 \wedge u_3 = 0 \wedge u_4 \geq 0)$$

$$\Rightarrow (\underline{\hat{z}} = ([\hat{x}_{1I}, \hat{x}_{1I}], [\hat{x}_{2S}, \hat{x}_{2S}], \underline{\hat{u}}_1, \underline{0}, \underline{0}, \underline{\hat{u}}_4)^T),$$

$$(viii) \qquad (x^* = D) \Rightarrow (u_1 = 0 \wedge u_2 \geq 0 \wedge u_3 \geq 0 \wedge u_4 = 0)$$

$$\Rightarrow (\underline{\hat{z}} = ([\hat{x}_{1S}, \hat{x}_{1S}], [\hat{x}_{2I}, \hat{x}_{2I}], \underline{0}, \underline{\hat{u}}_2, \underline{\hat{u}}_3, \underline{0})^T),$$

and

$$(ix) \qquad (x^* = C) \Rightarrow (u_1 = 0 \wedge u_2 = 0 \wedge u_3 \geq 0 \wedge u_4 \geq 0)$$

$$\Rightarrow (\underline{\hat{z}} = ([\hat{x}_{1S}, \hat{x}_{1S}], [\hat{x}_{2S}, \hat{x}_{2S}], \underline{0}, \underline{0}, \underline{\hat{u}}_3, \underline{\hat{u}}_4)^T).$$

The cases $(i) - (ix)$ correspond to the 2^2 sub-boxes which are obtained by bisecting the box $\underline{\hat{x}}$ at the midpoints of AB and AD. These sub-boxes are

$$\underline{\hat{z}}^{(0)} = ([m(\underline{\hat{x}}_1), \hat{x}_{1S}], [m(\underline{\hat{x}}_2), \hat{x}_{2S}], \underline{0}, \underline{0}, \underline{\hat{u}}_3, \underline{\hat{u}}_4)^T, \qquad 6.20$$

$$\underline{\hat{z}}^{(1)} = ([m(\underline{\hat{x}}_1), \hat{x}_{1S}], [\hat{x}_{2I}, m(\underline{\hat{x}}_2)], \underline{0}, \underline{\hat{u}}_2, \underline{\hat{u}}_3, \underline{0})^T, \qquad 6.21$$

$$\underline{\hat{z}}^{(2)} = ([\hat{x}_{1I}, m(\underline{\hat{x}}_1)], [m(\underline{\hat{x}}_2), \hat{x}_{2S}], \underline{\hat{u}}_1, \underline{0}, \underline{0}, \underline{\hat{u}}_4)^T, \tag{6.22}$$

and

$$\underline{\hat{z}}^{(3)} = ([\hat{x}_{1I}, m(\underline{\hat{x}}_1)], [\hat{x}_{2I}, m(\underline{\hat{x}}_2)], \underline{\hat{u}}_1, \underline{\hat{u}}_2, \underline{0}, \underline{0})^T. \tag{6.23}$$

(c) $n = 3$

In this case there exist 2^3 sub-boxes $\underline{\hat{z}}^{(i)}$ $(i = 0, \ldots, 7)$ of $\underline{\hat{z}}$ which might contain the KT point z^* and they are obtained by bisecting the faces of the box $\underline{\hat{x}}$. These sub-boxes are as follows.

$$\underline{\hat{z}}^{(0)} = ([m(\underline{\hat{x}}_1), \hat{x}_{1S}], [m(\underline{\hat{x}}_2), \hat{x}_{2S}], [m(\underline{\hat{x}}_3), \hat{x}_{3S}], \underline{0}, \underline{0}, \underline{0}, \underline{\hat{u}}_4, \underline{\hat{u}}_5, \underline{\hat{u}}_6)^T, \tag{6.24}$$

$$\underline{\hat{z}}^{(1)} = ([m(\underline{\hat{x}}_1), \hat{x}_{1S}], [m(\underline{\hat{x}}_2), \hat{x}_{2S}], [\hat{x}_{3I}, m(\underline{\hat{x}}_3)], \underline{0}, \underline{0}, \underline{\hat{u}}_3, \underline{\hat{u}}_4, \underline{\hat{u}}_5, \underline{0})^T, \tag{6.25}$$

$$\underline{\hat{z}}^{(2)} = ([m(\underline{\hat{x}}_1), \hat{x}_{1S}], [\hat{x}_{2I}, m(\underline{\hat{x}}_2)], [m(\underline{\hat{x}}_3), \hat{x}_{3S}], \underline{0}, \underline{\hat{u}}_2, \underline{0}, \underline{\hat{u}}_4, \underline{0}, \underline{\hat{u}}_6)^T, \tag{6.26}$$

$$\underline{\hat{z}}^{(3)} = ([m(\underline{\hat{x}}_1), \hat{x}_{1S}], [\hat{x}_{2I}, m(\underline{\hat{x}}_2)], [\hat{x}_{3I}, m(\underline{\hat{x}}_3)], \underline{0}, \underline{\hat{u}}_2, \underline{\hat{u}}_3, \underline{\hat{u}}_4, \underline{0}, \underline{0})^T, \tag{6.27}$$

$$\hat{\underline{z}}^{(4)} = ([\hat{x}_{1I}, m(\hat{\underline{x}}_1)], [m(\hat{\underline{x}}_2), \hat{x}_{2S}], [m(\hat{\underline{x}}_3), \hat{x}_{3S}],$$
$$\hat{\underline{u}}_1, \underline{0}, \underline{0}, \underline{0}, \hat{\underline{u}}_5, \hat{\underline{u}}_6)^T, \qquad 6.28$$

$$\hat{\underline{z}}^{(5)} = ([\hat{x}_{1I}, m(\hat{\underline{x}}_1)], [m(\hat{\underline{x}}_2), \hat{x}_{2S}], [\hat{x}_{3I}, m(\hat{\underline{x}}_3)],$$
$$\hat{\underline{u}}_1, \underline{0}, \hat{\underline{u}}_3, \underline{0}, \hat{\underline{u}}_5, \underline{0})^T, \qquad 6.29$$

$$\hat{\underline{z}}^{(6)} = ([\hat{x}_{1I}, m(\hat{\underline{x}}_1)], [\hat{x}_{2I}, m(\hat{\underline{x}}_2)], [m(\hat{\underline{x}}_3), \hat{x}_{3S}],$$
$$\hat{\underline{u}}_1, \hat{\underline{u}}_2, \underline{0}, \underline{0}, \underline{0}, \hat{\underline{u}}_6)^T, \qquad 6.30$$

and

$$\hat{\underline{z}}^{(7)} = ([\hat{x}_{1I}, m(\hat{\underline{x}}_1)], [\hat{x}_{2I}, m(\hat{\underline{x}}_2)], [\hat{x}_{3I}, m(\hat{\underline{x}}_3)],$$
$$\hat{\underline{u}}_1, \hat{\underline{u}}_2, \hat{\underline{u}}_3, \underline{0}, \underline{0}, \underline{0})^T. \qquad 6.31$$

We shall derive a method for determining the sub-boxes when $n \geq 1$.

Arrange the Lagrange multiplier intervals given by 6.18 - 6.19, 6.20 - 6.23, 6.24 - 6.31 in the form

$$\begin{pmatrix} \underline{0} & \underline{u}_2 \\ \underline{u}_1 & \underline{0} \end{pmatrix}, \qquad 6.32$$

$$\begin{pmatrix} \underline{0} & \underline{0} & \underline{u}_3 & \underline{u}_4 \\ \underline{0} & \underline{u}_2 & \underline{u}_3 & \underline{0} \\ \underline{u}_1 & \underline{0} & \underline{0} & \underline{u}_4 \\ \underline{u}_1 & \underline{u}_2 & \underline{0} & \underline{0} \end{pmatrix}, \qquad 6.33$$

and

$$\begin{pmatrix} \underline{0} & \underline{0} & \underline{0} & \underline{u}_4 & \underline{u}_5 & \underline{u}_6 \\ \underline{0} & \underline{0} & \underline{u}_3 & \underline{u}_4 & \underline{u}_5 & \underline{0} \\ \underline{0} & \underline{u}_2 & \underline{0} & \underline{u}_4 & \underline{0} & \underline{u}_6 \\ \underline{0} & \underline{u}_2 & \underline{u}_3 & \underline{u}_4 & \underline{0} & \underline{0} \\ \underline{u}_1 & \underline{0} & \underline{0} & \underline{0} & \underline{u}_5 & \underline{u}_6 \\ \underline{u}_1 & \underline{0} & \underline{u}_3 & \underline{0} & \underline{u}_5 & \underline{0} \\ \underline{u}_1 & \underline{u}_2 & \underline{0} & \underline{0} & \underline{0} & \underline{u}_6 \\ \underline{u}_1 & \underline{u}_2 & \underline{u}_3 & \underline{0} & \underline{0} & \underline{0} \end{pmatrix} \qquad 6.34$$

with dimension $2^n \times 2n$ for n=1,2, and 3 respectively. For $i = 1, \ldots, n$, if $\underline{u}_i \neq \underline{0}$ then $\underline{u}_{i+n} = \underline{0}$ and *vice versa*. Therefore if the first n columns of the matrices are known then the last n columns can be determined and *vice versa*, where n is the number of components of the vector x. Also for $i = 1, \ldots, n$, if $\underline{u}_i \neq \underline{0}$ then the corresponding interval is $[\hat{x}_{iI}, m(\underline{\hat{x}}_i)]$ while if $\underline{u}_i = \underline{0}$ then the corresponding interval is $[m(\underline{\hat{x}}_i), \hat{x}_{iS}]$.

From the preceding remarks, we need use either the first n columns of the matrix or the last n columns of the matrix only in order to determine the 2^n sub-boxes of $\underline{\hat{z}}$ which might contain the KT point z^*. For this purpose, if we replace all the $\underline{u}_i \neq \underline{0}$ $(i = 1, \ldots, n)$ in the first n columns of 6.32, 6.33, and 6.34 with unity then we obtain the matrices

$$\begin{pmatrix} 0 \\ 1 \end{pmatrix}, \qquad 6.35$$

$$\begin{pmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{pmatrix}, \qquad 6.36$$

and

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix} \qquad 6.37$$

corresponding to n=1,2, and 3 respectively.

The rows of 6.35, 6.36, and 6.37 can be considered as binary numbers which are the components of the vectors

$$(0 \quad 1)^T, \qquad 6.38$$

$$(0 \quad 1 \quad 2 \quad 3)^T, \qquad 6.39$$

and

$$(0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 7)^T \qquad 6.40$$

corresponding to 6.35, 6.36, and 6.37 respectively. Therefore the box $\underset{\sim}{z}^{(4)}$ for $n = 4$ has corresponding binary digits (0 1 0 0), which give the Lagrange multiplier pattern (0 1 0 0 1 0 1 1), corresponding to the Lagrange multipliers $(0, u_2, 0, 0, u_5, 0, u_7, u_8)$. So

$$\hat{\underline{z}}^{(4)} = ([m(\hat{\underline{x}}_1), \hat{x}_{1S}], [\hat{x}_{2I}, m(\hat{\underline{x}}_2)], [m(\hat{\underline{x}}_3), \hat{x}_{3S}], [m(\hat{\underline{x}}_4), \hat{x}_{4S}],$$

$$\underline{0}, \hat{\underline{u}}_2, \underline{0}, \underline{0}, \hat{\underline{u}}_5, \underline{0}, \hat{\underline{u}}_7, \hat{\underline{u}}_8)^T. \qquad 6.41$$

It is, of course, possible that one or more of $\hat{\underline{u}}_2$, $\hat{\underline{u}}_5$, $\hat{\underline{u}}_7$, and $\hat{\underline{u}}_8$ are equal to $\underline{0}$. If, for example, $\hat{q}_5 = 0$, where $\hat{q}_5$ has been determined by the procedure *lagrange.multipliers*, then $\hat{\underline{u}}_5 = \underline{0}$.

A procedure for converting i $(i = 0, \ldots, 2^n - 1)$ into a binary number is as follows.

procedure decimal.to.binary$(n, i \in N : b \in N^n)$

! This procedure converts a decimal integer i $(i = 0, \ldots, 2^n - 1)$ into the

! corresponding n-digit binary number b, possibly with leading zeros.

1. $b := (0)_{n \times 1}$
2. $j := n$
3. *repeat*

   3.1. $b_j := i$ *rem* 2 ! i *rem* 2 = *remainder of* $i/2$.

3.2. $i := i$ *div* 2 ! i *div* $2 = quotient\ i/2.$

while $i \neq 0$ *do*

3.3. $j := j - 1$

4. *return* □

A procedure for obtaining the sub-boxes $\hat{\underline{z}}^{(i)}$ for one $i \in \{0, \ldots, 2^n - 1\}$ by using the procedure *decimal.to.binary*, is as follows.

procedure construct.z.i$(\hat{\underline{x}} \in I(R^n), \hat{\underline{u}} \in I(R^{2n}), \hat{q} \in N^{2n}, n, i \in N, x.star.in.x \in B :$

$$\hat{\underline{z}}^{(i)} \in I(R^{3n}), \hat{q}^{(i)} \in N^{2n}, \hat{l}^{(i)} \in N)$$

! This procedure determines the sub-box $\hat{\underline{z}}^{(i)}$ for $i \in \{0, \ldots, 2^n - 1\}$ and its

! corresponding values of $\hat{q}^{(i)}$ and $\hat{l}^{(i)}$.

! On entry $\hat{\underline{x}}$ is the initial box which is assumed to contain the global minimizer x^*

! of $f : R^n \to R^1$, $\hat{\underline{u}}$ is the $2n \times 1$ interval vector which bounds the $2n \times 1$ vector

! of Lagrange multipliers u^* corresponding to x^*, $\hat{\underline{u}}$ and $\hat{q}$ are computed from the

! procedure *lagrange.multipliers*, and *x.star.in.x* is such that if $x^* \in int(\hat{\underline{x}})$ then

! $x.star.in.x =$ *true* and if $x^* \in \hat{\underline{x}}$ then $x.star.in.x =$ *false*.

1. $\hat{\underline{z}}^{(i)} := (\underline{0})_{3n \times 1}$

2. $\hat{q}^{(i)} := (0)_{2n \times 1}$

3. $\hat{l}^{(i)} := 2n$

4. $decimal.to.binary(n, i : b)$

5. $\underline{for}\ j = 1\ \underline{to}\ n\ \underline{do}$

   5.1. $\underline{if}\ b_j = 0$

   $\underline{then}$! $\hat{\underline{z}}^{(i)}_{n+j} = \underline{0}$.

   5.1.1. $\hat{\underline{z}}^{(i)}_j := [m(\hat{\underline{x}}_j), \hat{x}_{jS}]$

   5.1.2. $\underline{if}\ \sim x.star.in.x\ \underline{and}\ \hat{q}_{n+j} \neq 0\ \underline{do}$

   5.1.2.1. $\hat{\underline{z}}^{(i)}_{2n+j} := \hat{\underline{u}}_{n+j}$

   5.1.2.2. $\hat{q}^{(i)}_{n+j} := 1$

   5.1.2.3. $\hat{l}^{(i)} := \hat{l}^{(i)} - 1$

   $\underline{else}$! $\hat{\underline{z}}^{(i)}_{2n+j} = \underline{0}$.

   5.1.3. $\hat{\underline{z}}^{(i)}_j := [\hat{x}_{jI}, m(\hat{\underline{x}}_j)]$

   5.1.4. $\underline{if}\ \sim x.star.in.x\ \underline{and}\ \hat{q}_j \neq 0\ \underline{do}$

   5.1.4.1. $\hat{\underline{z}}^{(i)}_{n+j} := \hat{\underline{u}}_j$

   5.1.4.2. $\hat{q}^{(i)}_j := 1$

   5.1.4.3. $\hat{l}^{(i)} := \hat{l}^{(i)} - 1$

6. $\underline{return}$ □

6.4 Bounding the Global Minimum f^* of f

In this section we consider how to bound the value f^* of f at a global minimizer $x^* \in \hat{\underline{x}}$. A degenerate interval $\underline{\bar{f}}$ such that $f^* \leq \bar{f}_S$ is determined and is updated at

several points in algorithm MW. Initially

$$\bar{\underline{f}} = \underline{f}(m(\hat{\underline{x}})) \qquad 6.42$$

and $m(\hat{\underline{x}})$ is pushed onto the stack S_7 of point boxes which might contain x^*.

The interval $\bar{\underline{f}}$ is updated by comparing it with an interval $\underline{f}$ which is known to contain f^* and which has been computed at various points in MW as described in subsequent sections. The updating procedure is as follows.

procedure update.f.bar$(\underline{x} \in I(R^n), \underline{f} \in I(R)\ ;\ S \in P, \bar{\underline{f}} \in I(R), n_S \in N)$

! This procedure updates $\bar{\underline{f}}$ and n_S where n_S is the number of boxes on the stack
! S which might contain a global minimizer x^*.
! On entry, $\underline{f}$ is such that $\{f(x) \mid x \in \underline{x}\} \subseteq \underline{f}$. If $\bar{f}_S < f_I$ then $\underline{x}$ cannot contain a
! global minimizer of f so $\underline{x}$ is not pushed onto S and $\bar{f}$ is not updated. If $f_S < \bar{f}_I$
! then $\bar{\underline{f}}$ can be updated and we push the corresponding box $\underline{x}$ onto the stack
! S and make $n_S = 1$ since all the previous boxes in S cannot contain a global
! minimizer. Therefore in order to delete the redundant boxes we set $n_S = 1$.
! Otherwise we push $\underline{x}$ onto S and set $n_S := n_S + 1$.

1. *if* $f_I \leq \bar{f}_S$ *do*

! *If* $\bar{f}_S < f_I$ *then* $\underline{x}$ *is deleted.*

1.1. *if* $f_S < \bar{f}_I$

then

1.1.1. $\bar{f} := [f_S, f_S]$

1.1.2. $n_S := 1$

else

1.1.3. $n_S := n_S + 1$

1.2. $\underline{x} \longrightarrow S$! $\underline{x}$ *is a box.*

2. *return* □

6.5 Deletion Test 1

Suppose that $\underline{z}$ is the current sub-box of $\hat{\underline{z}}^{(i)}$ $(0 \leq i \leq 2^n - 1)$ which is assumed to contain a minimizer of f either in its interior or on its boundary. By 6.8 and 6.9 at least n of the $2n$ Lagrange multiplier bounds are zero intervals. Therefore (§6.2), if we compute $\underline{F} = \underline{F}(\underline{z})$ from the procedure F (Appendix D) then the number of components in $\underline{F}$ is $3n - l$ where $n \leq l \leq 2n$.

If $0 \notin \underline{F}_i$ for some $i \in \{1, \ldots, 3n - l\}$ then we know that there is no zero z^* of F in $\underline{z}$. Therefore the whole of $\underline{z}$ can be deleted. The preceding ideas give rise to the following procedure.

procedure deletion.test.1($\underline{F} \in I(R^{3n-l}), n, l \in N$: *delete.z* $\in B$)

! This procedure deletes $\underline{z}$ if for at least one $i \in \{1, \ldots, 3n-l\}$ $0 < F_{iI}$ or $F_{iS} < 0$,

! where $n \leq l \leq 2n$.

! On entry, $\underline{F} = \underline{F}(\underline{z})$ (Appendix D), n is the number of components in the initial

! box $\underline{\hat{x}}$ and l is the number of Lagrange multipliers in $\underline{z}$ which take the value zero.

1. *delete.z* := *false*

2. $i := 1$

3. *repeat*

   3.1. *delete.z* := $0 < F_{iI}$ *or* $F_{iS} < 0$

   while $\sim$ *delete.z* *and* $i < 3n - l$ *do*

   3.2. $i := i + 1$

4. *return* □

6.6 A Gradient Test

Let $\underline{z} = (\underline{x}^T, \underline{u}^T)^T$ be a sub-box of $\underline{\hat{z}}^{(i)}$ $(0 \leq i \leq 2^n - 1)$. We shall use the monotonicity property of the objective function f which is described in §3.2 to delete some or all of $\underline{z}$.

If $x^* \in int(\underline{\hat{x}})$ or $l = 2n$ then $\underline{u} = \underline{0}$, so 4.33 and 4.35 become

$$F(x) = f'(x)^T \tag{6.43}$$

and

$$F'(x) = (\partial_j \partial_i f(x))_{n \times n} \tag{6.44}$$

respectively. The test which is described in §6.5 is made just before we apply the gradient test. Therefore, if $x^* \in int(\underline{\hat{x}})$ or $l = 2n$ then we do not need to apply the gradient test to decide whether or not to delete $\underline{z}$ because according to 6.43, $\underline{F}$ $(= \underline{F}(\underline{z}))$ in §6.5 is equal to $\underline{f}'^T (= \underline{f}'(\underline{x})^T)$ in the current test. Therefore, for this case, if $\underline{z}$ cannot be deleted by using the test in §6.5 then $\underline{z}$ also cannot be deleted by using the gradient test.

The parameters $q_1, \ldots, q_{2n}$ defined by 6.17 and l, the number of Lagrange multipliers which take the value zero can be used in order to determine the position of x^*, the global minimizer(s) and whether or not the current box $\underline{x}$ of $\underline{\hat{x}}$ contains the boundary points of $\underline{\hat{x}}$. q_i $(i = 1, \ldots, 2n)$ and l are determined at three points in MW as follows.

(a) At the first point, $\hat{q}_1, \ldots, \hat{q}_{2n}$ and $\hat{l}$ are determined for the initial box $\underline{\hat{x}}$ as follows.

(i) If $(\partial_j \underline{f}(\underline{\hat{x}}))_S \leq 0$ $(j \in \{1, \ldots, n\})$ then by 6.13 $\underline{\hat{u}}_j = \underline{0}$ and $\underline{\hat{u}}_{j+n} = -\partial_j \underline{f}(\underline{\hat{x}})$, so by 6.17 $\hat{q}_j = 0$ and $\hat{q}_{j+n} = 1$. Therefore $x_j^* = \hat{x}_{jS}$, and the value of $\hat{l}$ depends on how many of the $\underline{\hat{u}}_j$ $(j = 1, \ldots, 2n)$ take the value zero.

(ii) If $(\partial_j \underline{f}(\hat{\underline{x}}))_I \geq 0$ $(j \in \{1, \ldots, n\})$ then by 6.16 $\hat{\underline{u}}_j = \partial_j \underline{f}(\hat{\underline{x}})$ and $\hat{\underline{u}}_{j+n} = \underline{0}$, so by 6.17 $\hat{q}_j = 1$ and $\hat{q}_{j+n} = 0$. Therefore $x_j^* = \hat{x}_{jI}$, and $\hat{l}$ is as in (i).

(iii) If $(\partial_j \underline{f}(\hat{\underline{x}}))_I < 0 < (\partial_j \underline{f}(\hat{\underline{x}}))_S$ then by 6.14 $\hat{\underline{u}}_j = [0, (\partial_j \underline{f}(\hat{\underline{x}}))_S]$ and $\hat{\underline{u}}_{j+n} = [0, -(\partial_j \underline{f}(\hat{\underline{x}}))_I]$, so by 6.17 $\hat{q}_j = 1$ and $\hat{q}_{j+n} = 1$. Therefore $x_j^* \in \hat{\underline{x}}_j$, and $\hat{l}$ is as in (i).

(b) At the second point, $\hat{q}_1^{(i)}, \ldots, \hat{q}_{2n}^{(i)}$, and $\hat{l}^{(i)}$ are determined for the sub-box $\hat{\underline{x}}^{(i)}$ $(i = 0, \ldots, 2^n - 1)$ as follows. Suppose that $n = 1$. Then by the method described in §6.3 we have

$$\hat{\underline{z}}^{(0)} = ([m(\hat{\underline{x}}_1), \hat{x}_{1S}], \underline{0}, \hat{\underline{u}}_2)^T$$

and

$$\hat{\underline{z}}^{(1)} = ([\hat{x}_{1I}, m(\hat{\underline{x}}_1)], \hat{\underline{u}}_1, \underline{0})^T$$

given by 6.18 and 6.19 respectively. However the components $\hat{\underline{u}}_2$ of $\hat{\underline{z}}^{(0)}$ and $\hat{\underline{u}}_1$ of $\hat{\underline{z}}^{(1)}$ depend on the cases (i), (ii), and (iii) of (a). Therefore, for $\hat{\underline{z}}^{(0)}$, we obtain the following three possibilities.

(i)′ By (i), $\hat{\underline{z}}^{(0)} = ([m(\hat{\underline{x}}_1), \hat{x}_{1S}], \underline{0}, \hat{\underline{u}}_2)^T$, $\hat{q}_1^{(0)} = 0$, $\hat{q}_2^{(0)} = 1$, $\hat{l}^{(0)} = 1$ and $x_1^* = \hat{x}_{1S} \in \hat{\underline{x}}^{(0)} = [m(\hat{\underline{x}}_1), \hat{x}_{1S}]$.

(ii)′ By (ii), $\hat{\underline{z}}^{(0)} = ([m(\hat{\underline{x}}_1), \hat{x}_{1S}], \underline{0}, \underline{0})^T$, $\hat{q}_1^{(0)} = 0$, $\hat{q}_2^{(0)} = 0$, $\hat{l}^{(0)} = 2$, and $x_1^* = \hat{x}_{1I} \notin \hat{\underline{x}}^{(0)} = [m(\hat{\underline{x}}_1), \hat{x}_{1S}]$.

(iii)′ By (iii), $\hat{\underline{z}}^{(0)} = ([m(\hat{\underline{x}}_1), \hat{x}_{1S}], \underline{0}, \underline{\hat{u}}_2)^T$, $\hat{q}_1^{(0)} = 0$, $\hat{q}_2^{(0)} = 1$, $\hat{l}^{(0)} = 1$, and $\hat{\underline{x}}^{(0)} = [m(\hat{\underline{x}}_1), \hat{x}_{1S}]$ might contain the minimizer x_1^*.

Suppose that $x^* \in \hat{\underline{x}}$. Then *x.star.in.x* = *false* as explained in the procedure *construct.z.i.*

(1) If $\hat{\underline{z}}^{(0)}$ is given by (i)′ then deletion test 1 does not delete $\hat{\underline{z}}^{(0)}$. Therefore in *MW* we apply the gradient test to $\hat{\underline{z}}^{(0)}$ and since $(\partial_1 \underline{f}(\hat{\underline{x}}))_S \leq 0$ (See (i)) we obtain $\underline{z} = ([\hat{x}_{1S}, \hat{x}_{1S}], \underline{0}, \underline{\hat{u}}_2)^T$.

(2) If $\hat{\underline{z}}^{(0)}$ is given by (ii)′ then $\hat{\underline{z}}^{(0)}$ might be deleted by deletion test 1. If $\hat{\underline{z}}^{(0)}$ is not deleted by deletion test 1 then $\hat{\underline{z}}^{(0)}$ is also not deleted by the gradient test.

(3) If $\hat{\underline{z}}^{(0)}$ is given by (iii)′ and deletion test 1 cannot delete $\hat{\underline{z}}^{(0)}$ then the gradient test is applied to $\hat{\underline{z}}^{(0)}$. If $(\underline{g}([m(\hat{\underline{x}}_1), \hat{x}_{1S}]))_S < 0$ then we retain the box $\underline{z} = ([\hat{x}_{1S}, \hat{x}_{1S}], \underline{0}, \underline{\hat{u}}_2)^T$ because $\hat{q}_2^{(0)} = 1$ and $\underline{z}$ is used for updating $\underline{f}$ from which $\underline{z}$ is either deleted or inserted into stack S_7. If $(\underline{g}([m(\hat{\underline{x}}_1), \hat{x}_{1S}]))_S \leq 0$ then we retain the box $\underline{z} = ([\hat{x}_{1S}, \hat{x}_{1S}], \underline{0}, \underline{\hat{u}}_2)^T$ and proceed as before. If $0 \leq (\underline{g}([m(\hat{\underline{x}}_1), \hat{x}_{1S}]))_I$ then $\hat{\underline{z}}^{(0)}$ is deleted because $\hat{q}_1^{(0)} = 0$. If $(\underline{g}([m(\hat{\underline{x}}_1), \hat{x}_{1S}]))_I < 0 < (\underline{g}([m(\hat{\underline{x}}_1), \hat{x}_{1S}]))_S$ then $\hat{\underline{z}}^{(0)}$ is not changed so we proceed in *MW* with $\hat{\underline{z}}^{(0)} = ([m(\hat{\underline{x}}_1), \hat{x}_{1S}], \underline{0}, \underline{\hat{u}}_2)^T$, $\hat{q}_1^{(0)} = 0$, $\hat{q}_2^{(0)} = 1$ and $\hat{l}^{(0)} = 1$.

(c) The third point in *MW* at which q and l are computed is as follows. Suppose that at the end of the iteration $\hat{\underline{z}}^{(0)}$ is bisected into $\underline{z}^{(1)}$ and $\underline{z}^{(2)}$ where

$$\underline{z}^{(1)} = ([m(\underline{\hat{x}}_1), (m(\underline{\hat{x}}_1) + \hat{x}_{1S})/2], \underline{u}_1^{(1)}, \underline{u}_2^{(1)})^T$$

and

$$\underline{z}^{(2)} = ([(m(\underline{\hat{x}}_1) + \hat{x}_{1S})/2, \hat{x}_{1S}], \underline{u}_1^{(2)}, \underline{u}_2^{(2)})^T.$$

By determining whether or not $[m(\underline{\hat{x}}_1), (m(\underline{\hat{x}}_1)+\hat{x}_{1S})/2]$ and $\underline{\hat{x}}$ share common boundaries we can determine $\underline{u}_1^{(1)}$, $\underline{u}_2^{(1)}$, $q_1^{(1)}$, $q_2^{(1)}$, and $l^{(1)}$ for $\underline{z}^{(1)}$. Similar considerations apply to $\underline{z}^{(2)}$ (The third point in MW at which $q_1, \ldots, q_{2n}$ and l are computed.) Therefore we obtain

$$\underline{z}^{(1)} = ([m(\underline{\hat{x}}_1), (m(\underline{\hat{x}}_1) + \hat{x}_{1S})/2], \underline{0}, \underline{0})^T,\ q_1^{(1)} = 0,\ q_2^{(1)} = 0,\ l^{(1)} = 2 \textit{ and}$$

$$\underline{z}^{(2)} = ([(m(\underline{\hat{x}}_1) + \hat{x}_{1S})/2, \hat{x}_{1S}], \underline{0}, \underline{u}_2^{(2)} = \underline{\hat{u}}_2)^T,\ q_1^{(2)} = 0,\ q_2^{(2)} = 1,\ \textit{and}\ l^{(2)} = 1.$$

Therefore, from the preceding discussion for $n = 1$ we can guarantee that if $q_j = 1$ $(j \in \{1, \ldots, n\})$ then either $x_j^* = \hat{x}_{jI}$ or the component $\underline{x}$ of $\underline{z}$ is such that $x_{jI} = \hat{x}_{jI}$. A similar statement is valid for $q_{j+n} = 1$.

These ideas give rise to the gradient test for deleting $\underline{z}$ which is contained in the procedure *gradient.test*.

procedure *gradient.test*($\hat{\underline{x}} \in I(R^n), n, l \in N, q \in N^{2n}, x.star.in.x \in B$; $S_7 \in P$,

$\underline{z} \in I(R^{3n}), \bar{\underline{f}} \in I(R), n_7 \in N$: $k \in N, delete.z \in B$)

! In this procedure one of the following cases occurs :

! (i) $\underline{z}$ is deleted ;

! (ii) $\underline{z}$ is reduced to a point box ;

! (iii) $\underline{z}$ is changed;

! (iv) $\underline{z}$ is unchanged.

! If $\underline{x}$, where $\underline{z} = (\underline{x}^T, \underline{u}^T)^T$ is reduced to a point box then we update $\bar{\underline{f}}$ by using

! the procedure *update.f.bar* (§6.4). The number of components in $\underline{x}$ which

! become degenerate intervals is k and $k \leq n$.

! On entry, $\hat{\underline{x}}$ is the initial box, l is the number of Lagrange multipliers in $\underline{z}$ which

! take the value zero, and $q = (q_1, \ldots, q_{2n})^T$ where $q = \hat{q}^{(i)}$ is computed in the

! procedure *construct.z.i* if $\underline{z} = \hat{\underline{z}}^{(i)}$ and q is determined by the procedures

! *zero.multipliers* and *check.q* described in §6.14 otherwise. The Boolean

! $x.star.in.x$ is such that if $x^* \in int(\hat{\underline{x}})$ then $x.star.in.x = \underline{true}$ and if $x^* \in \hat{\underline{x}}$ then

! $x.star.in.x = \underline{false}$. The computation in this procedure involves the first n

! variables of $\underline{z}$ only, that is, $\underline{z}_j = \underline{x}_j$ $(j = 1, \ldots, n)$.

! On return, n_7 is the number of point boxes on the stack S_7 (§6.4) which might

! contain the global minimizer x^* of $f : R^n \to R^1$. The Boolean $delete.z = \underline{true}$ if

! either the box $\underline{z}$ is deleted or $\underline{x}$ is reduced to a point box which is used to

! update $\underline{f}$ and is either deleted or is pushed onto S_7.

1. $k := 0$

2. *delete.z* := *false*

3. *if* $\sim (x.star.in.x$ *or* $l = 2n)$ *do*

   ! *If* $x^* \in int(\underline{\hat{x}})$ *or* $l = 2n$ *then we do not use the gradient test.*

   3.1. $j := 1$

   3.2. $\underline{g} := \underline{g}(\underline{x})$

   ! $\underline{g}$ *is an interval extension of the gradient of* f. *Also* $\underline{z} = (\underline{x}^T, \underline{u}^T)^T$.

   3.3. *repeat*

      3.3.1. *case true of*

         $0 < g_{jI}$: ! $0 < (\underline{g}_j(\underline{x}))_I$.

         3.3.1.1. *if* $q_j = 0$

            then

            ! *As explained in* §6.2, $x_{jI} \neq x_j^*$, *so the whole of* $\underline{x}$

            ! *and hence* $\underline{z}$ *may be deleted because if* x^* *is the*

            ! *minimizer of* f *then* $g(x^*) = 0$.

            3.3.1.1.1. *delete.z* := *true*

            else

            ! *As explained in* §6.2, $x_{jI} = x_j^*$ *where* $x_j^* = \hat{x}_{jI}$.

! *All points in the* $\underline{z}_j$ *– direction save the lower boundary*

! *point may be deleted.*

3.3.1.1.2. $\underline{z}_j := [\hat{x}_{jI}, \hat{x}_{jI}]$

3.3.1.1.3. $k := k + 1$

$0 \leq g_{jI}$: ! $0 \leq (\underline{g}_j(\underline{x}))_I$.

! *The minimizer of* f *could lie in the face* $x = x_{jI}$ *of* $\underline{x}$.

3.3.1.2. $\underline{z}_j := [x_{jI}, x_{jI}]$

3.3.1.3. $k := k + 1$

$g_{jS} < 0$: ! $(\underline{g}_j(\underline{x}))_S < 0$.

3.3.1.4. *if* $q_{n+j} = 0$

then

! *As explained in* §6.2, $x_{jS} \neq x_j^*$, *so the whole of* $\underline{x}$

! *and hence* $\underline{z}$ *may be deleted because if* x^* *is the*

! *minimizer of* f *then* $g(x^*) = 0$.

3.3.1.4.1. *delete.z* := *true*

else

! *As explained in* §6.2, $x_{jS} = x_j^*$ *where* $x_j^* = \hat{x}_{jS}$.

! *All points in the* z_j *– direction save the upper boundary*

! *point may be deleted.*

3.3.1.4.2. $\underline{z}_j := [\hat{x}_{jS}, \hat{x}_{jS}]$

3.3.1.4.3. $k := k + 1$

$g_{jS} \leq 0$: ! $(\underline{g}_j(\underline{x}))_S \leq 0$.

! *The minimizer of* f *could lie in the face* $x = x_{jS}$ *of* $\underline{x}$.

3.3.1.5. $\underline{z}_j := [x_{jS}, x_{jS}]$

3.3.1.6. $k := k + 1$

default : ! $\underline{z}_j$ *is unchanged.*

3.3.1.7. { }

while $\sim$ *delete.z and* $j < n$ *do*

3.3.2. $j := j + 1$

3.4. *if* $\sim$ *delete.z and* $k = n$ *do*

! *If* $k = n$ *then* $\underline{x}$ *is completely reduced* (*See Chapter* 3.) *to a point*

! *box which is either deleted or is pushed onto* S_7.

3.4.1. *delete.z* := *true*

3.4.2. $\underline{f} := \underline{f}(\underline{x})$! $\underline{x}$ *is a point box, such that* $\underline{z} = (\underline{x}^T, \underline{u}^T)^T$.

3.4.3. *update.f.bar*$(\underline{x}, \underline{f}\ ;\ S_7, \overline{f}, n_7)$! §6.4.

4. *return* □

After the procedure *gradient.test* is invoked, if *delete.z* = *true* then we pop the next sub-box from the stack S_1 of boxes which are to be processed if $S_1 \neq \emptyset$. If $S_1 = \emptyset$ then we process the next sub-box $\hat{\underline{z}}^{(i+1)}$ of $\hat{\underline{z}}$ where $\hat{\underline{z}}^{(i+1)}$ is formed by using the procedure *construct.z.i* (§6.3).

6.7 Deletion Test 2

In this section we describe how to use the ideas which are described in §7 of [Han–80a], and in §3.4 and §3.6 to delete $\underline{x}$ and hence a sub-box $\underline{z} = (\underline{x}^T, \underline{u}^T)^T$ of $\hat{\underline{z}}^{(i)}$ $(0 \leq i \leq 2^n - 1)$. An interval extension $\underline{J}(\underline{x}, x)$ of the Jacobian matrix of $f'(x)^T$ is defined by $\underline{J}(\underline{x}, x) = (\underline{J}_{ij}(\underline{x}, x))_{n \times n}$ where $\underline{J}_{ij}(\underline{x}, x)$ is given by 3.5. We compute $\underline{J}(\underline{x}, x)$ using the procedure *F.prime* as given in Appendix D. Suppose also that an interval extension $\underline{G}(\underline{x}, x)$ of the Hessian matrix of f is defined by $\underline{G}(\underline{x}, x) = (\underline{G}_{ij}(\underline{x}, x))_{n \times n}$ where

$$\underline{G}_{ij}(\underline{x}, x) = \begin{cases} \underline{J}_{ii}(\underline{x}, x) & (j = i (i = 1, \ldots, n)) \\ 2\underline{J}_{ij}(\underline{x}, x) & (j < i (i = 1, \ldots, n;\ j = 1, \ldots, i-1)). \\ 0 & (j > i (j = 1, \ldots, n;\ i = 1, \ldots, j-1)) \end{cases} \qquad 6.45$$

As explained in [Han–80a] and in Chapter 3, $\underline{x}$ can be reduced in one dimension, say the kth, at a time by solving the quadratic relation 3.6. If $x = m(\underline{x})$, and 3.6 is satisfied for $t \in \underline{t} \in I(R)$, then the interval $\underline{x}_k$ is replaced with $\underline{y} = \underline{t} + x_k$. As explained in [Han–80a] and in Chapter 3, this procedure might reduce $\underline{x}_k$ $(k = 1, \ldots, n)$ to two intervals, one interval, or to no interval.

Suppose that $\underline{y}^{(1)}$ and $\underline{y}^{(2)}$ are the sub-intervals which are obtained by solving 3.6. If only $\underline{y}^{(1)}$ is non-empty then we replace $\underline{x}_k$ in $\underline{z}$ with $\underline{y}^{(1)}$. If both $\underline{y}^{(1)}$ and $\underline{y}^{(2)}$ are non-empty then we construct two sub-boxes $\underline{z}^{(1)}$ and $\underline{z}^{(2)}$ with $\underline{z}_k^{(1)} = \underline{y}^{(1)}$, $\underline{z}_k^{(2)} = \underline{y}^{(2)}$ and $\underline{z}_j^{(i)} = \underline{x}_j$ $(j \neq k, j = 1, \ldots, n;\ i = 1, 2)$, and save them for use in §6.13. However, it could be that $\underline{y}^{(1)} \neq \emptyset$ and $\underline{y}^{(2)} \neq \emptyset$ for more than one $k \in \{1, \ldots, n\}$; this could lead to the generation of a prohibitively large number of non-degenerate

sub-boxes $\underline{z}^{(i)}$ of $\underline{z}$. In order to simplify the problem, we choose one $k \in \{1, \ldots, n\}$ for which the largest sub-interval of $\underline{x}_k$ has been deleted, and we replace $\underline{x}_k$ with $\underline{y}^{(1)}$ and $\underline{y}^{(2)}$, leaving $\underline{x}_j$ $(j = 1, \ldots, n;\ j \neq k)$ unchanged. In this way we generate two sub-boxes with the smallest width for use in §6.13.

Note 6.1 : We use a different procedure from that which is used by Hansen[Han–80a] for solving 3.6. Hansen uses 3.6 for the cases $0 \leq (\underline{G}_{kk}(\underline{x}, x))_I$ (§3.4) and $(\underline{G}_{kk}(\underline{x}, x))_I < 0$ (§3.6) separately whereas we do not separate these cases. □

We use the procedure *quadratic.coefficients* to compute the coefficients $\underline{a}_k$, $\underline{b}_k$ and $\underline{c}_k$ in 3.6.

procedure quadratic.coefficients$(\underline{G} \in I(M(R^n)), g \in R^n, \underline{x}, \underline{w} \in I(R^n), \underline{E} \in I(R),$
$n, k \in N : \underline{a}, \underline{b}, \underline{c} \in I(R))$

! This procedure determines the coefficients $\underline{a}(= \underline{a}_k)$, $\underline{b}(= \underline{b}_k)$, and $\underline{c}(= \underline{c}_k)$ given by
! 3.7 - 3.9.
! On entry, $\underline{G} = (\underline{G}_{ij}(\underline{x}, x))_{n \times n}$ is given by 6.45, $g = f'(m(\underline{x}))^T$, $\underline{w} = [m(\underline{x}), m(\underline{x})]$,
! and $\underline{E} = \overline{f} - \underline{f}(m(\underline{x})) - \varepsilon_1$. The component in which $\underline{x}$ is reduced is k.

1. $\underline{a} := -\underline{E}$
2. $\underline{b} := \underline{0}$

3. $\underline{\tilde{x}} := \underline{x} - \underline{w}$

! *Steps* 4 – 8 *compute* $\underline{a}$ *from* 3.7.

4. *for* $j = 1$ *to* $k - 1$ *do*

   4.1. $\underline{a} := \underline{a} + g_j \underline{\tilde{x}}_j$

5. *for* $j = k + 1$ *to* n *do*

   5.1. $\underline{a} := \underline{a} + g_j \underline{\tilde{x}}_j$

6. $\underline{s} := \underline{0}$

7. *for* $j = 1$ *to* n *do*

   7.1. *for* $i = j$ *to* n *do*

      7.1.1. *if* $j \neq k$ *and* $i \neq k$ *do*

         7.1.1.1. *if* $i = j$

            then

            7.1.1.1.1. $\underline{s} := \underline{s} + \underline{G_{ii}} \underline{\tilde{x}}_i^2$

            else

            7.1.1.1.2. $\underline{s} := \underline{s} + \underline{G_{ij}} \underline{\tilde{x}}_i \underline{\tilde{x}}_j$

8. $\underline{a} := \underline{a} + \underline{s}/2$

! *Steps* 9 – 11 *compute* $\underline{b}$ *from* 3.8.

9. *for* $j = 1$ *to* $k - 1$ *do*

   9.1. $\underline{b} := \underline{b} + \underline{G_{kj}} \underline{\tilde{x}}_j$

10. *for* $j = k + 1$ *to* n *do*

   10.1. $\underline{b} := \underline{b} + \underline{G_{jk}} \underline{\tilde{x}}_j$

11. $\underline{b} := g_k + \underline{b}/2$

12. $\underline{c} := \underline{G_{kk}}/2$! *Compute* $\underline{c}$ *from* 3.9.

13. *return* □

After the procedure *quadratic.coefficients* is invoked, $\underline{a}$, $\underline{b}$, and $\underline{c}$ are $\underline{a_k}$, $\underline{b_k}$, and $\underline{c_k}$ respectively given by 3.7 - 3.9 and are used in the procedure *solve.quadratic* together with the following procedures.

function *positive.delta*$(\underline{d} \in I(R))$

! This function determines an interval which consists of the non-negative part of the
! discriminant $\underline{d}$ of the inequality in 3.6. Clearly $d_S > 0$ [Han–80a].

1. *if* $d_I < 0$

   then

   1.1. *positive.delta* $:= [0, d_S]$

   else

   1.2. *positive.delta* $:= \underline{d}$

2. *return* □

function *R.plus*$(\underline{b}, \underline{d}, \underline{c}, \underline{w} \in I(R))$

! This function computes an interval containing $(-\underline{b} + \sqrt{\underline{d}})/(2\underline{c}) + \underline{w}$ where

! $\underline{w} = [m(\underline{x}), m(\underline{x})]$, $\underline{b}$, $\underline{c}$ are as in the procedure *quadratic.coefficients*, and $\underline{d}$ is as

! in the function *positive.delta*.

1. $R.plus := (-\underline{b} + \sqrt{positive.delta(\underline{d})})/(2\underline{c}) + \underline{w}$
2. *return* □

function $R.minus(\underline{b}, \underline{d}, \underline{c}, \underline{w} \in I(R))$

! This function computes an interval containing $(-\underline{b} - \sqrt{\underline{d}})/(2\underline{c}) + \underline{w}$ where $\underline{b}$, $\underline{c}$, $\underline{d}$

! and $\underline{w}$ are as in the function *R.plus*.

1. $R.minus := (-\underline{b} - \sqrt{positive.delta(\underline{d})})/(2\underline{c}) + \underline{w}$
2. *return* □

function $S.plus(\underline{a}, \underline{b}, \underline{d}, \underline{w} \in I(R))$

! This function computes an interval containing $2\underline{a}/(-\underline{b} + \sqrt{\underline{d}}) + \underline{w}$ where $\underline{a}$ is as in

! the procedure *quadratic.coefficients* and $\underline{b}$, $\underline{d}$ and $\underline{w}$ are as in the function *R.minus*.

1. $S.plus := 2\underline{a}/(-\underline{b} + \sqrt{positive.delta(\underline{d})}) + \underline{w}$

2. *return* □

function $S.minus(\underline{a}, \underline{b}, \underline{d}, \underline{w} \in I(R))$

! This function computes an interval containing $2\underline{a}/(-\underline{b} - \sqrt{\underline{d}}) + \underline{w}$ where $\underline{a}$, $\underline{b}$, $\underline{d}$ and

! $\underline{w}$ are as in the function *S.plus*.

1. $S.minus := 2\underline{a}/(-\underline{b} - \sqrt{positive.delta(\underline{d})}) + \underline{w}$

2. *return* □

procedure $intersection.1(\underline{u}, \underline{x} \in I(R) : \underline{y} \in I(R), empty \in B)$

! This procedure determines $[-\infty, u_S] \cap \underline{x}$ to give the interval $\underline{y}$. If on return

! $empty = true$ then $\underline{y}$ is empty.

1. $\underline{y} := \underline{0}$

2. $empty :=$ *true*

3. *case true of*

$x_S \leq u_S$:

3.1. $\underline{y} := \underline{x}$

3.2. *empty* := *false*

$x_I \leq u_S$ *and* $u_S \leq x_S$:

3.3. $\underline{y} := [x_I, u_S]$

3.4. *empty* := *false*

default :

3.5. { }

4. *return* □

procedure intersection.2$(\underline{v}, \underline{x} \in I(R) : \underline{y} \in I(R), empty \in B)$

! This procedure determines $[v_I, +\infty] \cap \underline{x}$ to give the interval $\underline{y}$. If on return

! *empty* = *true* then $\underline{y}$ is empty.

1. $\underline{y} := \underline{0}$

2. *empty* := *true*

3. *case true of*

$v_I \leq x_I$:

3.1. $\underline{y} := \underline{x}$

3.2. *empty* := *false*

$x_I \leq v_I$ *and* $v_I \leq x_S$:

3.3. $\underline{y} := [v_I, x_S]$

3.4. $empty := \underline{false}$

default :

3.5. { }

4. *return* □

procedure $union.1(\underline{u}, \underline{v}, \underline{x} \in I(R) : \underline{y}^{(1)}, \underline{y}^{(2)} \in I(R), y1.empty, y2.empty \in B)$

! This procedure determines $\underline{y}^{(1)}$ and $\underline{y}^{(2)}$ when $c_I < 0$.

1. $intersection.1(\underline{u}, \underline{x} : \underline{y}^{(1)}, y1.empty)$
2. $intersection.2(\underline{v}, \underline{x} : \underline{y}^{(2)}, y2.empty)$
3. *return* □

function $max(x, y \in R)$

! This function determines $max\{x, y\}$.

1. *if* $x > y$

then

1.1. $max := x$

else

1.2. $max := y$

2. *return* □

function $min(x, y \in R)$

! This function determines $min\{x, y\}$.

1. *if* $x < y$

then

1.1. $min := x$

else

1.2. $min := y$

2. *return* □

procedure $intersection.3(\underline{u}, \underline{v}, \underline{x} \in I(R) : \underline{y} \in I(R), empty \in B)$

! This procedure determines $[u_I, v_S] \cap \underline{x}$ where $\underline{u}$ and $\underline{v}$ are computed from the

! procedures *R.plus*, *R.minus*, *S.plus*, or *S.minus*. If the intersection is empty then

! $empty = \underline{true}$, and $\underline{y} = \emptyset$; otherwise $\underline{y} \neq \emptyset$ and $empty = \underline{false}$.

1. $\underline{y} := \underline{0}$
2. $empty := \underline{false}$
3. $\underline{if}\ v_S < x_I\ \underline{or}\ x_S < u_I$

   $\underline{then}$

   3.1. $empty := \underline{true}$

   $\underline{else}$

   3.2. $\underline{y} := [max(u_I, x_I), min(v_S, x_S)]$
4. $\underline{return}$ □

$\underline{procedure}$ *union.2*$(\underline{s}, \underline{t}, \underline{u}, \underline{v}, \underline{x} \in I(R), i \in N : \underline{y}^{(1)}, \underline{y}^{(2)} \in I(R),$

$y1.empty, y2.empty \in B)$

! This procedure determines $\underline{y}^{(1)}$ and $\underline{y}^{(2)}$ when $c_I > 0$.

! If $i = 1$ then $\underline{y}^{(1)}$ and $y1.empty$ are determined. If $i = 2$ then $\underline{y}^{(1)}$, $\underline{y}^{(2)}$, $y1.empty$,

! and $y2.empty$ are determined.

1. $\underline{case}\ i\ \underline{of}$

   1 :

1.1. $intersection.3(\underline{u}, \underline{v}, \underline{x} : \underline{y}^{(1)}, y1.empty)$

1.2. $\underline{y}^{(2)} := \underline{0}$

1.3. $y2.empty :=$ *true*

2 :

1.4. $intersection.3(\underline{u}, \underline{v}, \underline{x} : \underline{y}^{(1)}, y1.empty)$

1.5. $intersection.3(\underline{s}, \underline{t}, \underline{x} : \underline{y}^{(2)}, y2.empty)$

default

1.6. *write* "*No such case in union.2*"

1.7. *stop*

2. *return* □

procedure $solve.quadratic(\underline{a}, \underline{b}, \underline{c}, \underline{x} \in I(R) : \underline{y}^{(1)}, \underline{y}^{(2)} \in I(R),$

$y1.empty, y2.empty \in B)$

! This procedure determines $\underline{y}^{(1)}$, $\underline{y}^{(2)}$, $y1.empty$, and $y2.empty$ respectively from
! the current box $\underline{x}$ by solving the quadratic inequality $\underline{a} + \underline{b}t + \underline{c}t^2 \leq 0$ [Han–80a]
! to obtain $\underline{u}$ and $\underline{v}$ or $\underline{s}$ and $\underline{t}$ and then forming $\underline{x} \cap (\underline{u} \cup \underline{v})$ or $\underline{x} \cap (\underline{s} \cup \underline{t})$ to give
! $\underline{y}^{(1)}$ and $\underline{y}^{(2)}$.

1. $\underline{y}^{(1)} := \underline{0}$
2. $\underline{y}^{(2)} := \underline{0}$

3. $y1.empty := \underline{true}$

4. $y2.empty := \underline{true}$

5. $\underline{d}^{(i)} := ([b_I, b_I])^2 - 4[a_I, a_I][c_I, c_I]$

6. $\underline{d}^{(s)} := ([b_S, b_S])^2 - 4[a_I, a_I][c_I, c_I]$

7. $\underline{s} := \underline{0}$

8. $\underline{t} := \underline{0}$

9. $\underline{u} := \underline{0}$

10. $\underline{v} := \underline{0}$

11. $\underline{w} := [m(\underline{x}), m(\underline{x})]$

12. *case true of*

    $c_I < 0$:

    12.1. *case true of*

        $0 \leq b_I$:

        12.1.1. *case true of*

            $0 < a_I$:

            12.1.1.1. $\underline{u} := S.minus([a_I, a_I], [b_s, b_s], \underline{d}^{(s)}, \underline{w})$

            12.1.1.2. $\underline{v} := R.minus([b_I, b_I], \underline{d}^{(i)}, [c_I, c_I], \underline{w})$

            12.1.1.3. $union.1(\underline{u}, \underline{v}, \underline{x} : \underline{y}^{(1)}, \underline{y}^{(2)}, y1.empty, y2.empty)$

            $0 \leq d_I^{(i)}$:

            12.1.1.4. $\underline{u} := S.minus([a_I, a_I], [b_I, b_I], \underline{d}^{(i)}, \underline{w})$

            12.1.1.5. $\underline{v} := R.minus([b_I, b_I], \underline{d}^{(i)}, [c_I, c_I], \underline{w})$

12.1.1.6. $union.1(\underline{u}, \underline{v}, \underline{x} : \underline{y}^{(1)}, \underline{y}^{(2)}, y1.empty, y2.empty)$

default :

12.1.1.7. $\underline{y}^{(1)} := \underline{x}$

12.1.1.8. $y1.empty :=$ *false*

$b_S \leq 0$:

12.1.2. *case true of*

$0 < a_I$:

12.1.2.1. $\underline{u} := R.plus([b_S, b_S], \underline{d}^{(s)}, [c_I, c_I], \underline{w})$

12.1.2.2. $\underline{v} := S.plus([a_I, a_I], [b_I, b_I], \underline{d}^{(i)}, \underline{w})$

12.1.2.3. $union.1(\underline{u}, \underline{v}, \underline{x} : \underline{y}^{(1)}, \underline{y}^{(2)}, y1.empty, y2.empty)$

$0 \leq d_I^{(s)}$:

12.1.2.4. $\underline{u} := R.plus([b_S, b_S], \underline{d}^{(s)}, [c_I, c_I], \underline{w})$

12.1.2.5. $\underline{v} := S.plus([a_I, a_I], [b_S, b_S], \underline{d}^{(s)}, \underline{w})$

12.1.2.6. $union.1(\underline{u}, \underline{v}, \underline{x} : \underline{y}^{(1)}, \underline{y}^{(2)}, y1.empty, y2.empty)$

default :

12.1.2.7. $\underline{y}^{(1)} := \underline{x}$

12.1.2.8. $y1.empty :=$ *false*

default

12.1.3. *if* $0 \leq a_I$

then

12.1.3.1. $\underline{u} := S.minus([a_I, a_I], [b_S, b_S], \underline{d}^{(s)}, \underline{w})$

12.1.3.2. $\underline{v} := S.plus([a_I, a_I], [b_I, b_I], \underline{d}^{(i)}, \underline{w})$

12.1.3.3. $union.1(\underline{u}, \underline{v}, \underline{x} : \underline{y}^{(1)}, \underline{y}^{(2)}, y1.empty, y2.empty)$

else

12.1.3.4. $\underline{y}^{(1)} := \underline{x}$

12.1.3.5. $y1.empty :=$ *false*

$0 < c_I$:

12.2. *case true of*

$0 \leq b_I$:

12.2.1. *case true of*

$a_I \leq 0$:

12.2.1.1. $\underline{u} := R.minus([b_S, b_S], \underline{d}^{(s)}, [c_I, c_I], \underline{w})$

12.2.1.2. $\underline{v} := S.minus([a_I, a_I], [b_I, b_I], \underline{d}^{(i)}, \underline{w})$

12.2.1.3. $union.2(\underline{s}, \underline{t}, \underline{u}, \underline{v}, \underline{x}, 1 : \underline{y}^{(1)}, \underline{y}^{(2)}, y1.empty, y2.empty)$

$0 \leq d_S^{(s)}$:

12.2.1.4. $\underline{u} := R.minus([b_S, b_S], \underline{d}^{(s)}, [c_I, c_I], \underline{w})$

12.2.1.5. $\underline{v} := S.minus([a_I, a_I], [b_S, b_S], \underline{d}^{(s)}, \underline{w})$

12.2.1.6. $union.2(\underline{s}, \underline{t}, \underline{u}, \underline{v}, \underline{x}, 1 : \underline{y}^{(1)}, \underline{y}^{(2)}, y1.empty, y2.empty)$

default :

12.2.1.7. { }

$b_S \leq 0$:

12.2.2. *case true of*

$a_I \leq 0$:

12.2.2.1. $\underline{u} := S.plus([a_I, a_I], [b_S, b_S], \underline{d}^{(s)}, \underline{w})$

12.2.2.2. $\underline{v} := R.plus([b_I, b_I], \underline{d}^{(i)}, [c_I, c_I], \underline{w})$

12.2.2.3. $union.2(\underline{s}, \underline{t}, \underline{u}, \underline{v}, \underline{x}, 1 : \underline{y}^{(1)}, \underline{y}^{(2)}, y1.empty, y2.empty)$

$0 \leq d_S^{(i)}$:

12.2.2.4. $\underline{u} := S.plus([a_I, a_I], [b_I, b_I], \underline{d}^{(i)}, \underline{w})$

12.2.2.5. $\underline{v} := R.plus([b_I, b_I], \underline{d}^{(i)}, [c_I, c_I], \underline{w})$

12.2.2.6. $union.2(\underline{s}, \underline{t}, \underline{u}, \underline{v}, \underline{x}, 1 : \underline{y}^{(1)}, \underline{y}^{(2)}, y1.empty, y2.empty)$

default :

12.2.2.7. { }

default :

12.2.3. *case true of*

$a_I \leq 0$:

12.2.3.1. $\underline{u} := R.minus([b_S, b_S], \underline{d}^{(s)}, [c_I, c_I], \underline{w})$

12.2.3.2. $\underline{v} := R.plus([b_I, b_I], \underline{d}^{(i)}, [c_I, c_I], \underline{w})$

12.2.3.3. $union.2(\underline{s}, \underline{t}, \underline{u}, \underline{v}, \underline{x}, 1 : \underline{y}^{(1)}, \underline{y}^{(2)}, y1.empty, y2.empty)$

$0 < min(d_S^{(i)}, d_S^{(s)})$:

12.2.3.4. $\underline{u} := R.minus([b_S, b_S], \underline{d}^{(s)}, [c_I, c_I], \underline{w})$

12.2.3.5. $\underline{v} := S.minus([a_I, a_I], [b_S, b_S], \underline{d}^{(s)}, \underline{w})$

12.2.3.6. $\underline{s} := S.plus([a_I, a_I], [b_I, b_I], \underline{d}^{(i)}, \underline{w})$

12.2.3.7. $\underline{t} := R.plus([b_I, b_I], \underline{d}^{(i)}, [c_I, c_I], \underline{w})$

12.2.3.8. $union.2(\underline{s}, \underline{t}, \underline{u}, \underline{v}, \underline{x}, 2 : \underline{y}^{(1)}, \underline{y}^{(2)}, y1.empty, y2.empty)$

$b_S <\mid b_I \mid$ *and* $0 \leq max(d_S^{(i)}, d_S^{(s)})$:

12.2.3.9. $\underline{u} := S.plus([a_I, a_I], [b_I, b_I], \underline{d}^{(i)}, \underline{w})$

12.2.3.10. $\underline{v} := R.plus([b_I, b_I], \underline{d}^{(i)}, [c_I, c_I], \underline{w})$

12.2.3.11. $union.2(\underline{s}, \underline{t}, \underline{u}, \underline{v}, \underline{x}, 1 : \underline{y}^{(1)}, \underline{y}^{(2)}, y1.empty, y2.empty)$

$\mid b_I \mid < b_S$ *and* $0 \leq max(d_S^{(i)}, d_S^{(s)})$:

12.2.3.12. $\underline{u} := R.minus([b_S, b_S], \underline{d}^{(s)}, [c_I, c_I], \underline{w})$

12.2.3.13. $\underline{v} := S.minus([a_I, a_I], [b_S, b_S], \underline{d}^{(s)}, \underline{w})$

12.2.3.14. $union.2(\underline{s}, \underline{t}, \underline{u}, \underline{v}, \underline{x}, 1 : \underline{y}^{(1)}, \underline{y}^{(2)}, y1.empty, y2.empty)$

default :

12.2.3.15. { }

default : $! \, c_I = 0$

12.3. *case true of*

$a_I \leq 0$ *and* $b_S < 0$:

12.3.1. $intersection.2(-[a_I, a_I]/[b_S, b_S] + \underline{w}, \underline{x} : \underline{y}^{(1)}, y1.empty)$

$0 < a_I$ *and* $b_I < 0$ *and* $b_S \leq 0$:

12.3.2. $intersection.2(-[a_I, a_I]/[b_I, b_I] + \underline{w}, \underline{x} : \underline{y}^{(1)}, y1.empty)$

$a_I \leq 0$ *and* $b_I \leq 0$ *and* $0 \leq b_S$:

12.3.3. $\underline{y}^{(1)} := \underline{x}$

12.3.4. $y1.empty :=$ *false*

$0 < a_I$ *and* $b_I < 0$ *and* $0 < b_S$:

12.3.5. $union.1(-[a_I, a_I]/[b_S, b_S] + \underline{w}, -[a_I, a_I]/[b_I, b_I] + \underline{w}, \underline{x}:$

$\underline{y}^{(1)}, \underline{y}^{(2)}, y1.empty, y2.empty)$

$a_I \leq 0$ *and* $0 < b_I$:

12.3.6. $intersection.1(-[a_I, a_I]/[b_I, b_I] + \underline{w}, \underline{x}: \underline{y}^{(1)}, y1.empty)$

$0 < a_I$ *and* $0 \leq b_I$ *and* $0 < b_S$:

12.3.7. $intersection.1(-[a_I, a_I]/[b_S, b_S] + \underline{w}, \underline{x}: y^{(1)}, y1.empty)$

default :

12.3.8. { }

13. *return* □

After the procedure *solve.quadratic* is invoked $\underline{y}^{(1)}$, $\underline{y}^{(2)}$ are the intervals which are obtained by reducing the interval $\underline{x}_k$, $k \in \{1, \ldots, n\}$ and are used together with the procedure *quadratic.coefficients* in the procedure *deletion.test.2* which is as follows.

procedure $deletion.test.2(\underline{f}, \underline{f} \in I(R), \varepsilon_1 \in R, n, l, k \in N, q \in N^{2n},$

$x.star.in.x \in B$; $\underline{z} \in I(R^{3n})$: $\underline{J} \in I(M(R^{3n-l}))$,

$\underline{z}^{(1)}, \underline{z}^{(2)} \in I(R^{3n}), delete.z, z.is.split \in B)$

! This procedure implements Deletion Test 2 to reduce or to delete $\underline{z}$. If more than

! one $\underline{x}_i$ $(i = 1, \ldots, n)$ in $\underline{z}$ has been reduced to two sub-intervals then this

! procedure will determine which i $(i = 1, \ldots, n)$ gives the largest reduction of $\underline{x}_i$.

! On entry, l and q are used to compute the Jacobian $\underline{J} = (\underline{J}_{ij})_{3n-l \times 3n-l}$ where

! $\underline{J}_{ij}$ is given by 3.5 $(i, j = 1, \ldots, n)$, and then $\underline{J}$ is used to calculate the Hessian

! $\underline{G} = (\underline{G}_{ij})_{n \times n}$ given by 6.45. The integer k is the number of components of $\underline{z}$

! which become degenerate intervals as a result of the test in §6.6. Also $\varepsilon_1 \geq 0$,

! $\bar{f}$ is such that $\bar{f}_S > f^*$ and $\underline{f} = \underline{f}(m(\underline{x}))$.

! On return, if *delete.z* = *true* then $\underline{z}$ is deleted; otherwise $\underline{z}$ is not deleted. If

! *z.is.split* = *true* then $\underline{z}$ is reduced to two sub-boxes $\underline{z}^{(1)}$ and $\underline{z}^{(2)}$ and these are

! saved for use in §6.13.

1. $t := 0$
2. $w_c := \|w(\underline{x})\|$! $\underline{z} = (\underline{x}^T, \underline{u}^T)^T$.
3. *z.is.split* := *false*
4. *delete.z* := *false*
5. $\underline{G} := (\underline{0})_{n \times n}$
6. $\underline{y} := (\underline{0})_{n \times 2}$
7. $\underline{z}^{(1)} := (\underline{0})_{3n \times 1}$
8. $\underline{z}^{(2)} := (\underline{0})_{3n \times 1}$
9. $\underline{J} := \underline{F}'(\underline{z})$! *The procedure F.prime(Appendix D) is used.*
10. *for* $i = 1$ *to* n *do*

10.1. $\underline{G_{ii}} := \underline{J_{ii}}$

10.2. *for* $j = 1$ *to* $i - 1$ *do*

10.2.1. $\underline{G_{ij}} := 2\underline{J_{ij}}$

11. $g := f'(m(\underline{x}))^T$! $\underline{z} = (\underline{x}^T, \underline{u}^T)^T$.

12. *if* $\sim (x.star.in.x$ *or* $k = 0)$ *do*

! *If* $x.star.in.x =$ *true or* $k = 0$ *then the current box* $\underline{z}$ *has not been changed*

! *since Deletion Test* 1 *was last used. Therefore we do not need to compute*

! $\underline{f} := \underline{f}(m(\underline{x}))$ *because* $\underline{f}$ *has already been computed as described in* §6.15

! *and* §6.16.

12.1. $\underline{f} := \underline{f}(m(\underline{x}))$

13. $\underline{E} := \overline{\underline{f}} - \underline{f} - \varepsilon_1$

14. $i := 1$

15. *repeat*

! *We consider* $\underline{x}_i = \underline{z}_i$ $(i = 1, \ldots, n)$ *only.*

15.1. $quadratic.coefficients(\underline{G}, g, \underline{x}, [m(\underline{x}), m(\underline{x})], \underline{E}, n, i : \underline{a}, \underline{b}, \underline{c})$

15.2. $solve.quadratic(\underline{a}, \underline{b}, \underline{c}, \underline{z}_i : \underline{y}^{(1)}, \underline{y}^{(2)}, y1.empty, y2.empty)$

15.3. *case true of*

$\sim y1.empty$ *and* $y2.empty$:

15.3.1. $\underline{z}_i := \underline{y}^{(1)}$

$y1.empty$ *and* $\sim y2.empty$:

15.3.2. $\underline{z}_i := \underline{y}^{(2)}$

$\sim y1.empty$ *and* $\sim y2.empty$:

15.3.3. $z.is.split :=$ *true*

15.3.4. $\underline{y}_{i1} := \underline{y}^{(1)}$

15.3.5. $\underline{y}_{i2} := \underline{y}^{(2)}$

15.3.6. $w := w(\underline{y}_{i1}) + w(\underline{y}_{i2})$

15.3.7. *if* $w \leq w_c$ *do*

15.3.7.1. $t := i$

15.3.7.2. $w_c := w$

default :

15.3.8. $delete.z :=$ *true*

while $\sim delete.z$ *and* $i < n$ *do*

15.4. $i := i + 1$

16. *if* $\sim delete.z$ *and* $z.is.split$ *do*

16.1. *for* $i = 1$ *to* $3n$ *do*

16.1.1. $\underline{z}_i^{(1)} := \underline{z}_i$

16.1.2. $\underline{z}_i^{(2)} := \underline{z}_i$

16.2. $\underline{z}_t^{(1)} := \underline{y}_{t1}$

16.3. $\underline{z}_t^{(2)} := \underline{y}_{t2}$

17. *return* □

6.8 The Symmetric Operator Test

In this section we consider how to bound the solution z^* of $F(z) = 0$ and how to update the value of $\underline{f}$ by using the Symmetric operator which is described in §5.4 and §5.5.

Suppose that $\underline{z}$ is the current sub-box of $\underline{\tilde{z}}^{(i)}$ $(0 \leq i \leq 2^n - 1)$. Suppose that $\underline{\tilde{z}} = (\tilde{z}_i)_{t\times 1}$ is a box which is obtained from the box $\underline{z}$ by deleting the zero intervals corresponding to the zero Lagrange multipliers from the last $2n$ components of $\underline{z}$. This is done by using the procedure *new.z* which is described in this section. Since l is the number of Lagrange multipliers which take the value zero where $n \leq l \leq 2n$ (§6.5) then the number of Lagrange multipliers which are not deleted from $\underline{z}$ is $2n-l$. Therefore the number of components in $\underline{\tilde{z}}$ is $t = 3n - l$.

We compute $F = F(m(\underline{\tilde{z}}))$ and $\underline{F'} = \underline{F'}(\underline{\tilde{z}})$. If $m(\underline{F'})$ is non-singular then we can compute A, $\underline{R}$, g, $\underline{H_i} = \underline{H_i}(\underline{\tilde{z}})$ and $\underline{H_i'} = \underline{H_i'}(\underline{\tilde{z}})$ $(i = 1, \ldots, 3n - l)$, and $\underline{S_i} = \underline{S_i}(\underline{\tilde{z}})$ and $\underline{S_i'} = \underline{S_i'}(\underline{\tilde{z}})$ $(i = 3n - l, \ldots, 1)$ from 5.21 - 5.29 respectively. If $\underline{H_i'} = \emptyset$ for at least one $i \in \{1, \ldots, 3n - l\}$ then $\underline{S} = \emptyset$ and there is no zero of F in $\underline{\tilde{z}}$ and hence in $\underline{z}$. Therefore $\underline{z}$ can be deleted.

We shall use the Symmetric operator $\underline{S}$ according to the following six cases to form the Symmetric Operator Test which is implemented in the procedure *symmetric.operator.test* in this section.

(i) If $\underline{S} = \emptyset$ then there is no zero z^* of F in $\underline{\tilde{z}}$ and hence in $\underline{z}$, so $\underline{x}$ does not contain a minimizer of f. Therefore $\underline{z}$ can be deleted.

(ii) If $\underline{S} \cap \tilde{\underline{z}} = \emptyset$ then there is no zero z^* of F in $\tilde{\underline{z}}$ and hence in $\underline{z}$, so $\underline{x}$ does not contain a minimizer of f. Therefore $\underline{z}$ can be deleted.

(iii) If $\tilde{\underline{z}} \subset \underline{S}$ where $\underline{S}$ is computed at $\tilde{\underline{z}}$ then we retain the boxes $\tilde{\underline{z}}$ and $\underline{z}$ unchanged. Furthermore, suppose that $\underline{Q}$ is one of the operators which are described in Chapter 5 namely $\underline{K}_N$ and $\underline{S}$. In order to solve $F(z) = 0$ we compute $\underline{Q} = \underline{Q}(\tilde{\underline{z}})$. If $\underline{Q} = \underline{S}$ then $\tilde{\underline{z}} \subset \underline{S}$. As proved by Alefeld and Platzöder [AleP-83a] $\underline{K}(\underline{z}) \subseteq \underline{K}_N(\underline{z})$. So by **Lemma 5.4** if $\tilde{\underline{z}} \subseteq \underline{S}$ then $\tilde{\underline{z}} \subseteq \underline{S} \subseteq \underline{H} \subseteq \underline{K} \subseteq \underline{K}_N$. Therefore we retain the boxes $\tilde{\underline{z}}$ and $\underline{z}$ unchanged. Therefore, in case (iii) we bisect $\underline{z}$ into $\underline{z}^{(1)}$ and $\underline{z}^{(2)}$ for use in §6.13.

(iv) If $\underline{S} \subseteq \tilde{\underline{z}}$ and $w(\underline{S}) < w(\underline{H}')$ then we replace $\tilde{\underline{z}}$ with $\underline{S}$ because [SheW-85a] $\underline{S}$ contains the unique zero of F in $\tilde{\underline{z}}$. Also if $\underline{S} \subseteq \tilde{\underline{z}}$ and $w(\underline{S}) < w(\underline{H}')$ then the sequence $(s^{(k)})$ generated from

$$s^{(k+1)} = s^{(k)} - AF(s^{(k)}) \ (k \geq 0) \qquad 6.46$$

with $s_i^{(0)} = m(\underline{S}_i)$ $(i = 1, \ldots, 3n - l)$, is guaranteed to remain in $\underline{S}$ and to converge to the unique zero z^* of F in $\tilde{\underline{z}}$ **(Theorem 5.4)**.

If $\underline{S} \subset int(\tilde{\underline{z}})$ then $\underline{u}_i = \underline{0}$ $(i = 1, \ldots, 2n)$ where the $\underline{u}_i$ are the Lagrange multiplier intervals for sub-box $\underline{S}$. Therefore if $\underline{S} \subset int(\tilde{\underline{z}})$ and $w(\underline{S}) < w(\underline{H}')$ then 6.46 can be simplified to

$$\tilde{x}^{(k+1)} = \tilde{x}^{(k)} - \tilde{A}f'(\tilde{x}^{(k)})^T \ (k \geq 0) \qquad 6.47$$

where $\tilde{x}_i^{(0)} = m(\underline{S}_i)$ $(i = 1, \ldots, n)$, $\tilde{A} = (\tilde{a}_{ij})_{n\times n} = (a_{ij})_{n\times n}$ and from 4.33 $F(s^{(k)}) = f'(\tilde{x}^{(k)})^T$, and we compute $\tilde{x}^{(k)}$ $(k \geq 0)$ from 6.47 until for some $\hat{k} \geq 1$,

$$|\underline{\tilde{f}}^{(\hat{k})} - \underline{\tilde{f}}^{(\hat{k}-1)}| \leq \varepsilon_2 \, max\{|\underline{\tilde{f}}^{(\hat{k})}|, 1\},$$

where $\underline{\tilde{f}}^{(\hat{k})} = \underline{f}(\tilde{x}^{(\hat{k})})$, $\underline{\tilde{f}}^{(\hat{k}-1)} = \underline{f}(\tilde{x}^{(\hat{k}-1)})$ and $\varepsilon_2 > 0$. We could also stop iterating 6.47 if for some $\bar{k} \geq 2$, $\tilde{f}_I^{(\bar{k})} > \tilde{f}_S^{(\bar{k}-1)}$ and $\tilde{f}_I^{(\bar{k}-1)} > \tilde{f}_S^{(\bar{k}-2)}$ in which case $\tilde{z}$ might contain the maximizer rather than minimizer of f. We can then use $\tilde{f}^{(\bar{k})}$ to update $\underline{\tilde{f}}$. The procedure *simplified.Newton* which implements the preceding ideas is given in this section.

The test $\tilde{f}_I^{(\bar{k})} > \tilde{f}_S^{(\bar{k}-1)}$ and $\tilde{f}_I^{(\bar{k}-1)} > \tilde{f}_S^{(\bar{k}-2)}$ has been implemented for Example 10 which has 6 minimizers, 2 maximizers, 7 saddle points and was found to be unnecessary.

In case (iv) we use the methods which are described in Chapter 5 to get the sharpest bound possible for the solution $F(z) = 0$. But this solution might or might not be a minimizer of $f : R^n \to R^1$. Therefore we use the test which is described in §6.9 to examine the box $\tilde{z}$ which has been replaced with $\underline{S}$, before using the methods which are described in Chapter 5.

(v) If $\underline{S} \subseteq \tilde{z}$ and $w(\underline{S}) \not< w(\underline{H}')$ then we replace $\tilde{z}$ with $\underline{S}$ because [SheW-85a] $\underline{S}$ contains all the solutions of $F(z) = 0$ in $\tilde{z}$.

(vi) If $\tilde{\underline{z}}' = \underline{S} \cap \tilde{\underline{z}}$ and $\tilde{\underline{z}}' \neq \emptyset$ then [SheW-85a] $\tilde{\underline{z}}'$ contains a zero of F if $\tilde{\underline{z}}$ does. Therefore $\tilde{\underline{z}}$ is replaced with $\tilde{\underline{z}}'$ if $\underline{S} \not\subseteq \tilde{\underline{z}}$, $\tilde{\underline{z}} \not\subseteq \underline{S}$, and $\tilde{\underline{z}}' \neq \emptyset$.

Since the Symmetric Operator Test is made just after the test in §6.7 we can use the interval matrix $\underline{J}$ which is obtained from the procedure *deletion.test*.2. We have $\underline{F}' = \underline{J}$.

If $m(\underline{F}')$ is singular then we cannot use the Symmetric Operator Test. Neither can we apply the Newton Operator $\underline{N}$ which is described in §4.3 and the other methods for solving $F(z) = 0$ which are described in Chapter 5. Therefore, if $m(\underline{F}')$ is singular then we bisect $\underline{z}$ into two sub-boxes $\underline{z}^{(1)}$ and $\underline{z}^{(2)}$ for use in §6.13.

The procedures *new.z*, *old.z*, *symmetric.hansen*, and *simplified.newton* which are required by the procedure *symmetric.operator.test*, which embodies the ideas which have been described in this section, are as follows.

procedure $new.z(\underline{z} \in I(R^{3n}), q \in N^{2n}, n, l \in N, x.star.in.x \in B : \tilde{\underline{z}} \in I(R^{3n-l}))$

! This procedure determines a new box $\tilde{\underline{z}}$ from the box $\underline{z}$ by deleting l zero

! components corresponding to the zero Lagrange multipliers from the last 2n

! components of $\underline{z}$.

! On entry, $\underline{z} = (\underline{z}_1, \ldots, \underline{z}_{3n})^T$, q and l are as in §6.2, and $x.star.in.x$ is as in §6.6.

! On return, $\tilde{\underline{z}} = (\underline{z}_1, \ldots, \underline{z}_n, \tilde{\underline{z}}_{n+1}, \ldots, \tilde{\underline{z}}_t)^T$ where if $x.star.in.x = \underline{true}$ then $t = n$

! because $l = 2n$; otherwise $t = 3n - l$.

1. *for* $i = 1$ *to* n *do*

   1.1. $\tilde{\underline{z}}_i := \underline{z}_i$

2. *if* $\sim x.star.in.x$ *and* $l \neq 2n$ *do*

   2.1. $j := n$

   2.2. *for* $i = 1$ *to* $2n$ *do*

      2.2.1. *if* $q_i = 1$ *do*

         2.2.1.1. $j := j + 1$

         2.2.1.2. $\tilde{\underline{z}}_j := \underline{z}_{n+i}$

3. *return* □

procedure $old.z(\tilde{\underline{z}} \in I(R^{3n-l}), q \in N^{(2n)}, n, l \in N, x.star.in.x \in B : \underline{z} \in I(R^{3n}))$

! This procedure determines a box $\underline{z} = (\underline{z}_1, \ldots, \underline{z}_{3n})^T$ from a box

! $\tilde{\underline{z}} = (\tilde{\underline{z}}_1, \ldots, \tilde{\underline{z}}_n, \tilde{\underline{z}}_{n+1}, \ldots, \tilde{\underline{z}}_{3n-l})^T$ by inserting l zero interval components into

! the positions of l zero Lagrange multipliers in $\tilde{\underline{z}}$.

! On entry, $\tilde{\underline{z}}$, q, n, l, and $x.star.in.x$ are as in the procedure *new.z*.

! On return, $\underline{z} = (\underline{z}_1, \ldots, \underline{z}_{3n})^T$ where l of the last 2n components are zero intervals.

1. $\underline{z} := (\underline{0})_{3n\times 1}$

2. *for* $i = 1$ *to* n *do*

   2.1. $\underline{z}_i := \underline{\tilde{z}}_i$

3. *if* $\sim x.star.in.x$ *and* $l \neq 2n$ *do*

   3.1. $j := n$

   3.2. *for* $i = 1$ *to* $2n$ *do*

      3.2.1. *if* $q_j = 1$ *do*

         3.2.1.1. $j := j + 1$

         3.2.1.2. $\underline{z}_{n+i} := \underline{\tilde{z}}_j$

4. *return* □

procedure $symmetric.hansen(\underline{R} \in I(M(R^{3n-l})), b, \tilde{z} \in R^{3n-l}, n, l \in N, \underline{\tilde{z}} \in I(R^{3n-l})$

$: \underline{S}, \underline{S}', \underline{H}' \in I(R^{3n-l}), H'.empty, S'.empty \in B)$

! This procedure determines the interval vectors $\underline{S}$, $\underline{S}'$, and $\underline{H}'$ which are described

! in Chapter 5.

! On entry, $\underline{R}$ has been computed from 5.22, $\tilde{z} = m(\underline{\tilde{z}})$, b has been computed from

! 5.24, n is the number of components in the initial box $\underline{\hat{z}}$, l is the number of

! Lagrange multipliers which take the value zero for the current box $\underline{z}$, and $\underline{\tilde{z}}$ has

! been obtained from the box $\underline{z}$ by using the procedure *new.z*.

! On return, $\underline{S}$ and $\underline{S}'$ are as computed from 5.28 and 5.29 respectively, $\underline{H}'$ is

! computed from 5.27, $H'.empty = \underline{true}$ if for at least one $i \in \{1, \ldots, 3n-l\}$,

! $\underline{H'_i} = \emptyset$, and $S'.empty = \underline{true}$ if, for at least one $i \in \{3n-l, \ldots, 1\}$ $\underline{S'_i} = \emptyset$.

! Furthermore if $\underline{H'_i} = \emptyset$ for at least one $i \in \{1, \ldots, 3n-l\}$ then $\underline{S} = \emptyset$ and if $\underline{S'_i} = \emptyset$

! for at least one $i \in \{3n-l, \ldots, 1\}$ then $\underline{S} \cap \underline{\tilde{z}} = \emptyset$.

1. $\underline{S} := (\underline{0})_{3n-l \times 1}$
2. $\underline{S'} := (\underline{0})_{3n-l \times 1}$
3. $\underline{H'} := (\underline{0})_{3n-l \times 1}$
4. $H'.empty :=$ *false*
5. $S'.empty :=$ *false*
6. $i := 1$
7. $\underline{v} := (\underline{0})_{3n-l \times 1}$
8. *while* $i \leq 3n-l$ *and* $\sim H'.empty$ *do*

   8.1. $\underline{v_i} := \sum_{j=1}^{i-1} \underline{r_{ij}}(\underline{H'_j} - \tilde{z}_j)$

   8.2. $\underline{H_i} := \tilde{z}_i - b_i + \underline{v_i} + \sum_{j=i}^{3n-l} \underline{r_{ij}}(\underline{\tilde{z}_j} - \tilde{z}_j)$! 5.26.

   8.3. *if* $\underline{H_i} \cap \underline{\tilde{z}_i} = \emptyset$

   then

   8.3.1. $H'.empty :=$ *true*

   else

   8.3.2. $\underline{H'_i} := \underline{H_i} \cap \underline{\tilde{z}_i}$! 5.27.

   8.3.3. $i := i + 1$

9. *if* $\sim H'.empty$ *do*

9.1. $i := 3n - l$

9.2. *while* $i \geq 1$ *and* $\sim S'.empty$ *do*

9.2.1. $\underline{S}_i := \tilde{z}_i - b_i + \underline{v}_i + \underline{r}_{ii}(\underline{H}'_i - \tilde{z}_i) + \sum_{j=i+1}^{3n-l} \underline{r}_{ij}(\underline{S}'_j - \tilde{z}_j)$! 5.28.

9.2.2. *if* $\underline{S}_i \cap \underline{H}'_i = \emptyset$

then

9.2.2.1. $S'.empty := true$

else

9.2.2.2. $\underline{S}'_i := \underline{S}_i \cap \underline{H}'_i$! 5.29.

9.2.2.3. $i := i - 1$

10. *return* □

procedure $simplified.newton(\underline{S} \in I(R^{3n-l}), A \in M(R^{3n-l}), \varepsilon_2 \in R, n, l \in N :$

$$\tilde{x} \in R^n, \underline{f} \in I(R))$$

! This procedure implements case (iv) of the Symmetric Operator Test.

! On entry, $\underline{S}$ has been computed in the procedure *symmetric.hansen*, A has been

! computed from 5.21, $\varepsilon_2 > 0$, n is the number of variables of $f : R^n \to R^1$ and l

! is the number of Lagrange multipliers which take the value zero in sub-box $\underline{z}$.

! On return, $\underline{f} = \underline{f}([\tilde{x}, \tilde{x}])$ and $\tilde{x}$ is computed from 6.47.

1. *continue* := *true*

2. *for* $i = 1$ *to* n *do*

   2.1. $\tilde{x}_i := m(S_i)$

   2.2. *for* $j = 1$ *to* n *do*

      2.2.1. $\tilde{a}_{ij} := a_{ij}$! $A = (a_{ij})_{3n-l \times 3n-l}$.

3. $f := f(\tilde{x})$

4. $\tilde{f} := f$

5. $y := \tilde{x}$

6. *while continue do*

   6.1. $y := y - \tilde{A} f'(y)^T$! $\tilde{A} = (\tilde{a}_{ij})_{n \times n}$.

   6.2. $f_y := f(y)$

   6.3. *if* $f_{yS} < f_I$ *do*

      6.3.1. $f := f_y$

      6.3.2. $\tilde{x} := y$

   6.4. *if* $| f_y - \tilde{f} | \leq \varepsilon_2 \max(| f_y |, 1)$

      then ! *Stop iterating* 6.47.

      6.4.1. *continue* := *false*

      else ! *Iterate* 6.47.

      6.4.2. $\tilde{f} := f_y$

7. *return* □

procedure $symmetric.operator.test(\hat{\underline{x}} \in I(R^n), \underline{J} \in I(M(R^{3n-l})), \varepsilon_2 \in R, n \in N,$
$x.star.in.x \in B$; $S_7 \in P, \underline{z} \in I(R^{3n}), \bar{\underline{f}} \in I(R),$
$q \in N^{2n}, l, n_7 \in N : \underline{S} \in I(R^{3n-l}), bisect.z,$
$solve.F.z, delete.z, singular \in B)$

! This procedure implements the Symmetric Operator Test.

! On entry, $\hat{\underline{x}}$ is the initial box, $\underline{J}$ is the Jacobian matrix which has been

! computed in the procedure *deletion.test*.2 (§6.7), $\varepsilon_2 > 0$, and n, and $x.star.in.x$

! are as in the procedure *new.z*.

! The input-output parameters S_7, $\bar{\underline{f}}$, n_7 are as in the procedure *gradient.test*,

! $\underline{z}$ is the current box, and q and l are as in the procedure *new.z*.

! If cases (i) or (ii) occur then $delete.z = \underline{true}$; if case (iii) occurs or $m(\underline{J})$ is singular

! then $bisect.z = \underline{true}$; if case (iv) occurs then $solve.F.z = \underline{true}$, and $singular = \underline{true}$

! if $m(\underline{J})$ is singular.

1. $singular := \underline{false}$
2. $bisect.z := \underline{false}$
3. $solve.F.z := \underline{false}$
4. $delete.z := \underline{false}$
5. $\underline{S} := (\underline{0})_{3n-l \times 1}$
6. $new.z(\underline{z}, q, n, l, x.star.in.x : \check{\underline{z}})$

7. $F := F(m(\tilde{z}))$

8. $J := m(\underline{J})$

9. *if* J *is singular*

   then

   9.1. *singular* := *true*

   9.2. *bisect.z* := *true*

   else

   9.3. $A := J^{-1}$

   9.4. $b := AF$

   9.5. $\underline{R} := I - A\underline{J}$

   ! I *is the unit matrix of order* $t = 3n - l$.

   9.6. $\tilde{z} := m(\tilde{z})$

   9.7. $symmetric.hansen(\underline{R}, b, \tilde{z}, n, l, \tilde{z} : \underline{S}, \underline{S}', \underline{H}', H'.empty, S'.empty)$

   9.8. *case true of*

      $H'.empty$: ! $\underline{S} = \emptyset$, *case* (*i*).

      9.8.1. *delete.z* := *true*

      $S'.empty$: ! $\underline{S} \cap \tilde{z} = \emptyset$, *case* (*ii*).

      9.8.2. *delete.z* := *true*

      $\tilde{z} \subseteq \underline{S}$:

      ! $\tilde{z} \subset \underline{S}$ *for case* (*iii*) *and* $\tilde{z} = \underline{S}$ *for case* (*iv*) *and* (*v*).

      9.8.3. *bisect.z* := *true*

$\underline{S} \subset int(\underline{\tilde{z}})$: ! *case* (*iv*) *or* (*v*) *but not* $\underline{\tilde{z}} = \underline{S}$.

! *Since* $\underline{S} \subset int(\underline{\tilde{z}})$, $u_i = 0$, $q_i = 0$ $(i = 1, \ldots, 2n)$, *and* $l = 2n$.

9.8.4. *if* $w(\underline{S}) < w(\underline{H'})$ *do*

9.8.4.1. $simplified.newton(\underline{S}, A, \varepsilon_2, n, l \ : \ \tilde{x}, \underline{f})$

9.8.4.2. $update.f.bar([\tilde{x}, \tilde{x}], \underline{f} \ ; \ S_7, \underline{\tilde{f}}, n_7)$! §6.4.

9.8.4.3. $solve.F.z := \underline{true}$

9.8.5. $\underline{\tilde{z}} := \underline{S}$

9.8.6. *if* $\sim (x.star.in.x \ \underline{or} \ l = 2n)$ *do*

9.8.6.1. $q := (0)_{2n \times 1}$

9.8.6.2. $l := 2n$

9.8.7. $old.z(\underline{\tilde{z}}, q, n, l, x.star.in.x \ : \ \underline{z})$

default : ! *case* (*vi*).

! *Since* $\underline{\tilde{z}'} = \underline{S} \cap \underline{\tilde{z}}$ *some of the Lagrange multiplier bounds in* $\underline{\tilde{z}'}$

! *might be zero intervals. Therefore we need to re – compute* q *and* l

! *by using the procedures zero.multipliers and check.q described in* §6.14.

9.8.8. $\underline{\tilde{z}'} := \underline{S'}$

9.8.9. $zero.multipliers(\underline{\hat{x}}, \underline{\tilde{x}'}, n \ : \ \tilde{q}', \tilde{l}')$! §6.14.

9.8.10. $check.q(q, n \ ; \ \tilde{q}', \tilde{l}')$! §6.14.

9.8.11. $q := \tilde{q}'$

9.8.12. $l := \tilde{l}'$

9.8.13. $old.z(\underline{\tilde{z}'}, q, n, l, x.star.in.x \ : \ \underline{z})$

10. *return* □

6.9 A Non-Convexity Test

Let $\underline{z} = (\underline{x}^T, \underline{u}^T)^T$ be the current sub-box of $\hat{\underline{z}}^{(i)}$ $(0 \leq i \leq 2^n - 1)$. We shall use the non-convexity test for the objective function f in $\underline{x}$ which is described in §3.3 to delete all of $\underline{x}$ save the points of $\underline{x}$ which lie in $\partial(\hat{\underline{x}})$ where $\hat{\underline{x}}$ is the initial box.

According to §3.3 we need to compute the diagonal of $\underline{G}(\underline{x}, x)$ which is given by 3.4, with the argument $(\underline{x}_1, \ldots, \underline{x}_n)$ separately, because all $\underline{G}_{ij}(\underline{x}, x)$ for $i = j$ and $i < n$ have argument different from $(\underline{x}_1, \ldots, \underline{x}_n)$ save $\underline{G}_{nn}(\underline{x}, x)$. Since the non-convexity test is made just after the test in §6.8 in which $\underline{J}(\underline{x}, x)$ which is computed in §6.7, has been used, $\underline{G}_{nn}(\underline{x}, \underline{x})$ is already known. So we need compute only $\underline{G}_{ii}(\underline{x}, \underline{x})$ $(i = 1, \ldots, n-1)$.

Suppose that $\underline{x}'$ is the smallest box containing the boundary points of $\hat{\underline{x}}$ which lie in $\underline{x}$. Then

(i) $\underline{x}' = \underline{x}$ if $\hat{\underline{x}}$ and $\underline{x}$ share at least two common faces;

(ii) $\underline{x}'$ is a degenerate box of dimension less than that of $\underline{x}$ if $\hat{\underline{x}}$ and $\underline{x}$ share exactly one common face;

(iii) $\underline{x}' = \emptyset$ if $\underline{x} \subset int(\hat{\underline{x}})$.

In §6.2 it is shown how to compute the number l of Lagrange multipliers which take the value zero,. By using l we can determine $\underline{x}'$ as follows.

(i)′ If $l < 2n-1$ where n is the number of components in $\hat{\underline{x}}$ then the number of nonzero Lagrange multipliers is at least 2. Therefore, the number of active constraints is at

least 2. Therefore, $\underline{\hat{x}}$ and $\underline{x}$ share at least two common faces. So $\underline{x}' = \underline{x}$.

(ii)' If $l = 2n - 1$ then the number of nonzero Lagrange multipliers is exactly 1. Therefore, $\underline{\hat{x}}$ and $\underline{x}$ share exactly one common face. So $\underline{x}'$ is a degenerate box of dimension less than that of $\underline{x}$.

(iii)' If $l = 2n$ then all the Lagrange multipliers take the value zero. Therefore by 6.17, $q_i = 0$ $(i = 1, \ldots, 2n)$ where q corresponds to $\underline{x}$. Therefore as explained in §6.2 no global minimizer which lies in $\partial(\underline{\hat{x}})$ lies in $\partial(\underline{x})$. Therefore we may take $\underline{x}' = \emptyset$.

Therefore, according to (i)' - (iii)', if $l < 2n - 1$ then we must retain the current box $\underline{z}$; if $l = 2n - 1$, and $(\underline{G}_{ii}(\underline{x}, \underline{x}))_S < 0$ for at least one $i \in \{1, \ldots, n\}$ then we replace $\underline{x}$ in $\underline{z}$ with $\underline{x}'$; if $l = 2n$ then we delete the whole of $\underline{z}$ when $(\underline{G}_{ii}(\underline{x}, \underline{x}))_S < 0$ for at least one $i \in \{1, \ldots, n\}$.

Note 6.2 : In MW, if $l < 2n - 1$ then the non-convexity test is not used because whether or not $(\underline{G}_{ii}(\underline{x}, \underline{x}))_S < 0$ we still retain the current box $\underline{z}$ since by (i)' $\underline{x}' = \underline{x}$. □

The preceding ideas give rise to the procedure *n.c.test* (non-convexity test).

procedure *n.c.test*($\underline{J} \in I(M(R^{3n-l}))$, $\underline{\hat{x}} \in I(R^n)$, $q \in N^{2n}$, $n, l \in N$, *x.star.in.x* $\in B$;

$\underline{z} \in I(R^{3n})$: *delete.z* $\in B$)

! This procedure determines whether f is non-convex over $\underline{x}$, where $\underline{x}$ and $\underline{\hat{x}}$ share at

! most one common face. If f is not convex over $\underline{x}$ then $\underline{x}$ and hence $\underline{z}$ can be

! deleted save the points of $\underline{x}$ which belong to the boundary of the initial box $\underline{\hat{x}}$.

! On entry, $\underline{J}$ is the Jacobian computed in §6.7, q, n, l are as in §6.2, and *x.star.in.x*

! is as in §6.6.

! On return, $\underline{z}$ might contain the solution of $F(z) = 0$, and *delete.z* is the Boolean

! which allows us to decide whether or not to delete $\underline{z}$.

1. *delete.z* := *false*

2. $i := 1$

3. *if* *x.star.in.x* *or* $l = 2n$

   then

   ! *If* *x.star.in.x* = *true* *then* $x^* \in int(\hat{\underline{x}})$. *If* $l = 2n$ *is* *true* *then by* 6.17

   ! $q_i = 0$ $(i = 1, \ldots, 2n)$ *where* q *corresponds to* $\underline{x}$. *Therefore if* $l = 2n$ *then*

   ! *as explained in* §6.2 *no global minimizer which lies in* $\partial(\hat{\underline{x}})$ *lies in* $\partial(\underline{x})$.

   ! *Therefore if* $l = 2n$ *we can use the non – convexity test on the current box*

   ! $\underline{z} = (\underline{x}^T, \underline{u}^T)^T$ *knowing that* $\underline{x}$ *cannot contain the boundary points of* $\hat{\underline{x}}$.

   3.1. *repeat*

      3.1.1. *if* $i \neq n$

         then ! *We need to compute* $\underline{G_{ii}}(\underline{x}, \underline{x})$.

         3.1.1.1. *delete.z* := $(\underline{G_{ii}}(\underline{x}, \underline{x}))_S < 0$! $\underline{z} = (\underline{x}^T, \underline{u}^T)^T$.

         else ! *We use* $\underline{J_{ii}}$.

         3.1.1.2. *delete.z* := $J_{iiS} < 0$

   while ~ *delete.z* *and* $i < n$ *do*

3.1.2. $i := i + 1$

else ! *The minimizer of* f *is in* $\underline{\hat{x}}$ *and* $\underline{x}' \neq \emptyset$.

3.2. $k := 0$! k *is such that either the face* $x = x_{kI}$ *or* $x = x_{kS}$ *lies in* $\underline{\hat{x}}$.

! *We examine which face is common to* $\underline{\hat{x}}$ *and* $\underline{x}$ *where* $\underline{z} = (\underline{x}^T, \underline{u}^T)^T$.

3.3. *for* $j = 1$ *to* n *do*

3.3.1. *if* $q_j = 1$ *do*

3.3.1.1. $k := j$

3.3.2. *if* $q_{n+j} = 1$ *do*

3.3.2.1. $k := n + j$

3.4. *repeat*

3.4.1. *if* $i \neq n$

then ! *We need to compute* $\underline{G_{ii}}(\underline{x}, \underline{x})$.

3.4.1.1. *delete.z* $:= (\underline{G_{ii}}(\underline{x}, \underline{x}))_S < 0$! $\underline{z} = (\underline{x}^T, \underline{u}^T)^T$.

else ! *We use* $\underline{J_{ii}}$.

3.4.1.2. *delete.z* $:= J_{iiS} < 0$

while $\sim$ *delete.z and* $i < n$ *do*

3.4.2. $i := i + 1$

3.5. *if delete.z do*

! $\underline{z}$ *cannot be deleted since (step* 3.3*)* $\underline{x}$ *and* $\underline{\hat{x}}$ *share a common face.*

3.5.1. *delete.z* := *false*

3.5.2. *if* $k \leq n$

then

3.5.2.1. $\underline{z}_k := [\hat{x}_{kI}, \hat{x}_{kI}]$

else

3.5.2.2. $\underline{z}_k := [\hat{x}_{k-nS}, \hat{x}_{k-nS}]$

4. *return* □

6.10 A Bisection Test Which is Based on Strict Complementary Slackness

Suppose that $\underline{z} = (\underline{x}^T, \underline{u}^T)^T$ is a sub-box of $\hat{\underline{z}}^{(i)}$ $(0 \leq i \leq 2^n - 1)$. If by the test in §6.8 $\exists z^* \in \underline{z}$ such that $F(z^*) = 0$ and $0 \notin \underline{c}_i^0 \cap \underline{u}_i^0$ $(i = 1, \ldots, 2n)$ where $\underline{c}_i^0 = \underline{c}_i(\underline{z})^0$ then (Chapter 4) strict complementary slackness holds at z^*, and z^* is a KT point which might correspond to a maximizer or to a minimizer of f over $\underline{x}$.

On the other hand, $0 \notin \underline{c}_i^0 \cap \underline{u}_i^0$ $(i = 1, \ldots, 2n)$ is not necessary for strict complementary slackness. Therefore if for at least one $i \in \{1, \ldots, 2n\}$, $0 \in \underline{c}_i^0 \cap \underline{u}_i^0$ then strict complementary slackness might still hold at z^*. If, however, it is known that $x^* \in int(\underline{x})$ where $z^* = (x^{*T}, u^{*T})^T$, and $\underline{z} \subset int(\hat{\underline{z}})$ then $c_i(x) > 0$ $(\forall x \in \underline{x})$ $(i = 1, \ldots, 2n)$ whence $0 \notin \underline{c}_i^0 \cap \underline{u}_i^0$ $(i = 1, \ldots, 2n)$. This suggests that it might beneficial to bisect $\underline{z}$ if for some $i \in \{1, \ldots, 2n\}$ $0 \in \underline{c}_i^0 \cap \underline{u}_i^0$ when it is known that $x^* \in int(\hat{\underline{x}})$. If $x^* \in \partial(\hat{\underline{x}})$ so that for at least one $i \in \{1, \ldots, 2n\}$, $x_i^* = \hat{x}_{iI}$ or $x_i^* = \hat{x}_{iS}$ then $c_i(x^*) = 0$ or $c_{i+n}(x^*) = 0$ so if $z^* \in \partial(\hat{\underline{z}})$ then $0 \in \underline{c}_i^0$ or $0 \in \underline{c}_{i+n}^0$. In this case, bisection might not be beneficial because it could occur an indefinite number of times and produce an excessively large number of boxes. In practice however, it is necessary to check whether $0 \notin \underline{c}_i^0 \cap \underline{u}_i^0$ $(i = 1, \ldots, 2n)$ even when it is not known

that $x^* \in int(\hat{\underline{x}})$. Measures must be taken to ensure that an excessively large number of bisections do not take place when $x^* \in \partial(\hat{\underline{x}})$ and strict complementary slackness does not hold at x^*; the strategy which is used in this case is described in §6.12.

The vectors $\underline{c}_i^0$ and $\underline{u}_i^0$ are computed from the procedure $x.0$ which is as follows.

procedure $x.0(\underline{x} \in I(R^n), n \in N : \underline{x}^0 \in I(R^n))$

! This procedure determines an interval vector $(\underline{x}_1^0, \ldots, \underline{x}_n^0)^T$ such that $\underline{x}_i^0 = [x_{iI}, 0]$
! if $x_{iS} \leq 0$, and $\underline{x}_i^0 = \underline{x}_i$ otherwise.

1. *for* $i = 1$ *to* n *do*
   1.1. *if* $x_{iS} \leq 0$
   then
   1.1.1. $\underline{x}_i^0 := [x_{iI}, 0]$
   else
   1.1.2. $\underline{x}_i^0 := \underline{x}_i$
2. *return* □

The procedure *s.c.s.test* (strict complementary slackness test) which implements the test $0 \notin \underline{c}_i^0 \cap \underline{u}_i^0$ $(i = 1, \ldots, 2n)$ is as follows.

procedure $s.c.s.test(\underline{z} \in I(R^{3n}), n \in N : bisect.z \in B)$

! This procedure determines whether or not to bisect $\underline{z}$.

! On entry, $\underline{z}$ is a sub-box of $\hat{\underline{z}}^{(i)}$ $(0 \leq i \leq 2^n - 1)$.

! On return, $bisect.z$ is a Boolean which allows us to decide whether or not to bisect

! the current sub-box $\underline{z}$.

1. $bisect.z :=$ *false*
2. *for* $i = 1$ *to* n *do*
   2.1. $\underline{x}_i := \underline{z}_i$
3. *for* $i = 1$ *to* $2n$ *do*
   3.1. $\underline{u}_i := z_{n+i}$
4. $\underline{c} := \underline{c}(\underline{x})$! *The procedure constraint (Appendix D) is used.*
5. $x.0(\underline{c}, 2n : \underline{c}^0)$
6. $x.0(\underline{u}, 2n : \underline{u}^0)$
7. $i := 1$
8. *repeat*
   8.1. *if* $\underline{c}_i^0 \cap \underline{u}_i^0 \neq \emptyset$ *do*
      8.1.1. $bisect.z := 0 \in \underline{c}_i^0 \cap \underline{u}_i^0$

   while $\sim bisect.z$ *and* $i < 2n$ *do*

8.2. $i := i + 1$

9. *return* □

Note 6.3 : In MW, if $x.star.in.x = \underline{true}$ and strict complementary slackness does not hold at a KT point z^* then we do not apply the methods which are mentioned in Chapter 5. Instead the current box $\underline{z}$ is bisected into $\underline{z}^{(1)}$ and $\underline{z}^{(2)}$ as described in §6.12. One of the boxes $\underline{z}^{(1)}$ and $\underline{z}^{(2)}$ is kept on the stack S_1 and the other is processed in the next iteration of MW. If strict complementary slackness holds at a KT point z^* then we apply the methods which are mentioned in Chapter 5 to the current box $\underline{z}$. The procedure *s.c.s.test* is used in order to decide whether or not to apply the methods which are described in Chapter 5. □

6.11 Methods for Solving $F(z) = 0$

In the preceding sections non-existence, existence, uniqueness and convergence tests for a solution z^* of $F(z) = 0$ in the current sub-box $\underline{z}$ of $\hat{\underline{z}}^{(i)}$ $(0 \leq i \leq 2^n - 1)$ are described. The point z^* is a KT point which might correspond to a minimizer of f in $\hat{\underline{x}}$.

In this section we describe how to apply the methods which are described in Chapter 5 and which are used in the algorithm MW for solving $F(z) = 0$, where F is defined by 4.33.

Since some of the Lagrange multipliers are zero as described in §6.2, §6.3, §6.5, and §6.8 then some of the rows of $F(z)$ in 4.33 and the corresponding rows in 4.35

as well as the columns can be ignored. Therefore the number of variables in $\underline{z}$ is reduced. Suppose that $\tilde{\underline{z}}$ is obtained from $\underline{z}$ by using the procedure *new.z* (§6.8).

6.11.1 The Application of *MAP*

The algorithm *MAP* which is described in Chapter 5 is used in Algorithm *MW* as follows.

(i) If $B^{(0)}$ which is computed in step 3 of *MAP* is singular then we cannot proceed with the algorithm *MAP*. Therefore in *MW*, if $B^{(0)}$ is singular then $\tilde{\underline{z}}$ and hence $\underline{z}$ is bisected into $\underline{z}^{(1)}$ and $\underline{z}^{(2)}$ for use in §6.13.

(ii) If the algorithm *MAP* terminates because an empty intersection occurs then there is no zero of F in $\tilde{\underline{z}}$ and hence in $\underline{z}$. Therefore $\underline{z}$ can be deleted. Therefore if $\underline{K}_N \cap \tilde{\underline{z}} = \emptyset$ where $\underline{K}_N$ is given by 5.3 then we delete $\tilde{\underline{z}}$ and hence $\underline{z}$.

(iii) If $\underline{K}_N(\tilde{\underline{z}}^{(0,0)}, \underline{F}'^{(0)}, B^{(0)}) \not\subseteq \tilde{\underline{z}}^{(0,0)}$ where $\underline{K}_N$ is computed in step 5 of *MAP* then we do not proceed with the algorithm *MAP*. Instead we compute

$$\tilde{\underline{z}}' = \underline{K}_N(\tilde{\underline{z}}^{(0,0)}, \underline{F}'^{(0)}, B^{(0)}) \cap \tilde{\underline{z}}^{(0,0)},$$

and

$$a = \{B^{(0)}\}^{-1} F(m(\tilde{\underline{z}}^{(0,0)}))$$

for use in §6.12 in which we bisect $\underline{z}'$ as in [Jon-78a] where $\underline{z}'$ is the box which is obtained from $\tilde{\underline{z}}'$ by using the procedure *old.z* (§6.8).

(iv) According to 5.4, if $\alpha \geq 1$ then we do not proceed with the algorithm *MAP*. In this case we compute $\underline{z}'$ as in (iii). In *MW* α is computed from

$$\alpha = \max_{1 \leq i \leq 3n-l} \{w(\underline{K}_N(\tilde{\underline{z}}^{(0,0)}, \underline{F}'^{(0)}, B^{(0)})_i)/w(\tilde{\underline{z}}_i^{(0,0)})\} \qquad 6.48$$

in step 5 of *MAP* for $m = 0$.

(v) Suppose that $(\tilde{\underline{z}}^{(k)})$ is generated from the algorithm *MAP* and the stopping criterion $\left\| w(\tilde{\underline{z}}^{(k)}) \right\| \leq \varepsilon_0$ is used, where $\varepsilon_0 > 0$ is given. If, for some $\hat{k} \geq 0$, $\tilde{\underline{z}}^{(\hat{k})}$ satisfies the stopping criterion then $\underline{z}^{(\hat{k})}$ is pushed onto the stack S_4, which contains the boxes $\underline{z}$ which satisfy the stopping criterion where $\underline{z}^{(\hat{k})}$ is obtained from $\tilde{\underline{z}}^{(\hat{k})}$ by using the procedure *old.z* (§6.8.)

(vi) Suppose that $(\tilde{\underline{z}}^{(k)})$ is as in (v). If a certain number of iterations, say *maxit*, have been performed where in *MW*, *maxit* is the maximum number of iterations, and $\left\| w(\tilde{\underline{z}}^{(maxit)}) \right\| > \varepsilon_0$ then we say that $\tilde{\underline{z}}^{(k)}$ does not converge to z^* within a prescribed number *maxit* of iterations or $\tilde{\underline{z}}^{(k)} \not\to z^*$ $(k \leq maxit)$. However $\tilde{\underline{z}}^{(maxit)}$ possibly contains z^*. Therefore if $\tilde{\underline{z}}^{(k)} \not\to z^*$ $(k \leq maxit)$ then $\underline{z}^{(maxit)}$ is pushed onto S_5, where S_5 is a stack of boxes which might contain the minimizer, and $\underline{z}^{(maxit)}$ is obtained from $\tilde{\underline{z}}^{(maxit)}$ by using the procedure *old.z* (§6.8).

(vii) If in MAP $\underline{\tilde{z}}$ is reduced to give $\underline{\tilde{z}}'$ with $\|w(\underline{\tilde{z}}')\| \leq \varepsilon_0$ then $\underline{z}'$, where $\underline{z}'$ is obtained from $\underline{\tilde{z}}'$ by using the procedure *old.z* (§6.8), is pushed onto S_4 whether or not $\underline{z}'$ contains the solution z^* of $F(z) = 0$ because the width of the box $\underline{\tilde{z}}'$ is small and satisfies the condition $\|w(\underline{\tilde{z}}')\| \leq \varepsilon_0$.

The three procedures $MAP.1$, $MAP.2$, and $MAP.3$ are invoked by the procedure MAP in MW as follows.

procedure $MAP.1(\underline{F}' \in I(M(R^{3n-l})), B^{(0)} \in M(R^{3n-l}), \bar{w} \in R^{3n-l}, T_S, \varepsilon_0 \in R,$
$q \in N^{2n}, n, l \in N, x.star.in.x \in B\ ;\ \underline{z} \in I(R^{3n}), \underline{\tilde{z}} \in I(R^{3n-l}),$
$no.zero, too.big, not.less, converged \in B : \underline{Q} \in I(R^{3n-l}),$
$a, \tilde{w} \in R^{3n-l}, M, T_O, \alpha \in R)$

! This procedure implements step 5 of MAP for $m = 0$ in order to obtain $\underline{z}^{(0,1)}$
! if $B^{(0)}$ which is given at step 3 of MAP is non-singular. Also, this procedure
! determines which of the cases (ii), (iii), and (iv) is valid and whether or not
! $\|w(\underline{z}^{(0,1)})\| \leq \varepsilon_0$.
! On entry, $\underline{F}' := \underline{F}'(\underline{\tilde{z}})$, $B^{(0)} = m(\underline{F}')$ is non-singular, $\bar{w} := w(\underline{\tilde{z}})$, T_S is the starting
! time for MAP, $\varepsilon_0 > 0$, q, n, l are as in §6.2, and $x.star.in.x$ is as in §6.6, $\underline{z}$ is the
! current box, $\underline{\tilde{z}}$ is obtained from $\underline{z}$ by using the procedure *new.z* (§6.8),
! $no.zero = \underline{true}$ if $z^* \notin \underline{z}$, $too.big = \underline{true}$ if $\underline{Q} = \underline{K}_N(\underline{\tilde{z}}, \underline{F}', B^{(0)}) \not\subseteq \underline{\tilde{z}}$,

! *not.less* = *true* if $\alpha \geq 1$ and *converged* = *true* if $\|w(\tilde{\underline{z}})\| \leq \varepsilon_0$.

! On return, $\underline{Q} = \underline{K}_N(\tilde{\underline{z}}, \underline{F}', B^{(0)})$, a is as in case (iii), $\tilde{w} = w(\tilde{\underline{z}})$, M is as in

! the procedure *re.use.Jacobian* (§5.3), T_O is the required CPU time for this

! procedure, and α is computed from 6.48.

1. $a := (0)_{3n-l\times 1}$
2. $\tilde{w} := (0)_{3n-l\times 1}$
3. $M := 0$
4. $T_O := 0$
5. $\alpha := 0$
6. $\underline{y} := \tilde{\underline{z}}$
7. $\underline{Q} := \underline{K}_N(\underline{y}, \underline{F}', B^{(0)})$! $B^{(0)}$ *is non – singular.*
8. $\tilde{\underline{z}} := \underline{Q} \cap \underline{y}$! *Step* 5.1 *of MAP for* $m = 0$.
9. *if* $\tilde{\underline{z}} = \emptyset$

   then ! *Case* (*ii*).

   9.1. *no.zero* := *true*

   else

   9.2. $\tilde{w} := w(\tilde{\underline{z}})$

   9.3. $M := \|\tilde{w}\| / \|\bar{w}\|^2$

   9.4. *if* $\underline{Q} \not\subseteq \underline{y}$

   then ! *Case* (*iii*).

9.4.1. $too.big := \underline{true}$

9.4.2. $a := \{B^{(0)}\}^{-1} F(m(\underline{y}))$

9.4.3. $old.z(\tilde{\underline{z}}, q, n, l, x.star.in.x \ : \ \underline{z})$! §6.8.

else

9.4.4. $\alpha := \max_{1 \le i \le 3n-l} \{w(\underline{Q}_i)/w(\underline{y}_i)\}$

9.4.5. *if* $\alpha \ge 1$

then ! *Case* (*iv*).

9.4.5.1. $not.less := \underline{true}$

9.4.5.2. $a := \{B^{(0)}\}^{-1} F(m(\underline{y}))$

9.4.5.3. $old.z(\tilde{\underline{z}}, q, n, l, x.star.in.x \ : \ \underline{z})$! §6.8.

else

9.4.5.4. $converged := \|\tilde{w}\| \le \varepsilon_0$

9.4.5.5. $T_F := time$! *Appendix D.*

9.4.5.6. $T_O := T_F - T_S$! *The required CPU time for this procedure.*

9.4.5.7. $old.z(\tilde{\underline{z}}, q, n, l, x.star.in.x \ : \ \underline{z})$! §6.8.

10. *return* □

procedure $MAP.2(\underline{F'} \in I(M(R^{3n-l})), A \in M(R^{3n-l}), \varepsilon_0 \in R, n, l \in N$;

$\tilde{\underline{z}} \in I(R^{3n-l}), converged \in B \ : \ \underline{Q} \in I(R^{3n-l}), \tilde{w} \in R^{3n-l})$

! This procedure implements steps 8.3 - 8.4 or 8.5.6 - 8.5.7 of *MAP* (Chapter 5).

! Also, this procedure determines whether or not $\underline{x}^{(k)}$ obtained at step 8.4 of *MAP*

! and $\underline{z}^{(k,1)}$ obtained at step 8.5.7 of *MAP* satisfy $\|w(\underline{x}^{(k)})\| \leq \varepsilon_0$ and

! $\|w(\underline{z}^{(k,1)})\| \leq \varepsilon_0$ respectively.

! The input, input-output, and output parameters are as in the procedure *MAP*.1

! save that $A = m(\underline{F}')$.

1. *if* A *is singular* *do*

   ! *According to step* 9.4 *of the procedure* *MAP*.1 *if* $\underline{Q} \subseteq \underline{y}$ *and* $\alpha < 1$ *then*

   ! *we have a unique zero* z^* *in* $\underline{z}$. *Therefore,* A *should not be singular.*

   1.1. *write* "*A is singular in MAP.2.*"

   1.2. *stop*

2. $\underline{y} := \tilde{\underline{z}}$

3. $\underline{Q} := \underline{K}_N(\underline{y}, \underline{F}', A)$! *Step* 8.3 (*or step* 8.5.6) *of* *MAP*.

4. $\tilde{\underline{z}} := \underline{Q} \cap \underline{y}$! *Step* 8.4 (*or step* 8.5.7) *of* *MAP*.

5. *if* $\tilde{\underline{z}} = \emptyset$ *do*

   ! *At this stage we have guaranteed that* $\tilde{\underline{z}}$ *contains the zero* z^*.

   ! *Therefore the empty intersection at step* 4 *should not occur.*

   5.1. *write* "*Empty intersection in MAP.2.*"

   5.2. *stop*

6. $\tilde{w} := w(\tilde{\underline{z}})$

7. *converged* $:= \|\tilde{w}\| \leq \varepsilon_0$

8. *return* □

procedure $MAP.3(\underline{F}' \in I(M(R^{3n-l})), B \in M(R^{3n-l}), \bar{w} \in R^{3n-l}, M, T_O, \varepsilon_0 \in R,$
$q \in N^{2n}, n, l, p, m \in N, x.star.in.x \in B$; $\underline{z} \in I(R^{3n}),$
$\underline{\tilde{z}} \in I(R^{3n-l})$: $\underline{Q} \in I(R^{3n-l}), \tilde{w} \in R^{3n-l}, converged,$
$re.use.jacobian \in B)$

! This procedure implements steps 5, 8.5.3 and 8.5.8 of MAP for $m \geq 1$.

! Also, this procedure determines whether or not $\underline{z}^{(0,m+1)}$ obtained at step 5.1

! of MAP and $\underline{z}^{(k,m+1)}$ obtained at step 8.5.3.2 (or 8.5.8.2) of MAP satisfy

! $\|w(\underline{z}^{(0,m+1)})\| \leq \varepsilon_0$ and $\|w(\underline{z}^{(k,m+1)})\| \leq \varepsilon_0$ respectively and computes the

! value of $p^{(k)}$ (§5.3).

! All the parameters are as in the procedures $MAP.1$ and $MAP.2$ except the

! parameters M, p, m and *re.use.jacobian* which are explained in the procedure

! *re.use.Jacobian* in Chapter 5.

1. $T_S :=$ *time* ! *Appendix D.*

2. *if* B *is singular* *do*

   ! *See procedure* $MAP.2$ *at step* 1.

   2.1. *write* "B *is singular in* $MAP.3$."

   2.2. *stop*

3. $\underline{y} := \underline{\tilde{z}}$

4. $\underline{Q} := \underline{K_N}(\underline{y}, \underline{F'}, B)$

5. $\underline{\tilde{z}} := \underline{Q} \cap \underline{y}$

6. *if* $\underline{\tilde{z}} = \emptyset$ *do*

   ! *See procedure MAP.2 at step* 5.

   6.1. *write* "*Empty intersection in MAP.3.*"

   6.2. *stop*

7. $\tilde{w} := w(\underline{\tilde{z}})$

8. $converged := \|\tilde{w}\| \leq \varepsilon_0$

9. $T_F := time$! *Appendix D.*

10. $T_I := T_F - T_S$! *The required CPU time for this procedure.*

11. $re.use.Jacobian(\bar{w}, \tilde{w}, M, T_O, T_I, p, m, 3n-l$: $re.use.jacobian)$

12. $old.z(\underline{\tilde{z}}, q, n, l, x.star.in.x$: $\underline{z})$! §6.8.

13. *return* □

The algorithm MAP is used in the algorithm MW as follows.

procedure $MAP(q \in N^{2n}, n, l, p, maxit \in N, \varepsilon_0 \in R, x.star.in.x \in B$;

$\underline{z} \in I(R^{3n})$: $\underline{Q} \in I(R^{3n-l}), a \in R^{3n-l}, singular,$

$not.less, no.zero, converged, maxit.reached \in B)$

! This procedure implements the algorithm MAP for use in the algorithm MW.

! All the parameters are as in the procedures $MAP.i$ $(i = 1, 2, 3)$ except

! $maxit.reached$ where $maxit.reached = \underline{true}$ if $k \geq maxit$; $maxit$ is described

! in case (vi).

1. $new.z(\underline{z}, q, n, l, x.star.in.x : \underline{\tilde{z}})$! §6.8.
2. $singular := \underline{false}$
3. $no.zero := \underline{false}$
4. $too.big := \underline{false}$
5. $not.less := \underline{false}$
6. $maxit.reached := \underline{false}$
7. $re.use.jacobian := (p \neq 0)$! *If* $p = 0$ *then* $p^{(k)} = 0$ $(k \geq 0)$ *in* MAP.
8. $converged := \underline{false}$
9. $\underline{Q} := (\underline{0})_{3n-l \times 1}$
10. $a := (0)_{3n-l \times 1}$
11. $\bar{w} := w(\underline{\tilde{z}})$
12. $\tilde{w} := (0)_{3n-l \times 1}$
13. $M := 0$
14. $n_I := 1$
15. $T_S := time$! *Appendix D*.

16. $\underline{F'} := \underline{F'}(\underline{\tilde{z}})$

17. $B := m(\underline{F'})$

18. *if* B *is singular*

then

18.1. $singular := \underline{true}$

else

18.2. $MAP.1(\underline{F'}, B, \bar{w}, T_S, \varepsilon_0, q, n, l, x.star.in.x\ ;\ \underline{z}, \underline{\tilde{z}}, no.zero,$

$too.big, not.less, converged\ :\ \underline{Q}, a, \tilde{w}, M, T_O, \alpha)$

18.3. $\bar{w} := \tilde{w}$

18.4. $m := 0$

18.5. *if* $\sim no.zero$ *and* $\sim too.big$ *and* $\sim not.less$ *do*

18.5.1. *while* $re.use.jacobian$ *and* $\sim converged$ *do*

18.5.1.1. $m := m + 1$

18.5.1.2. $MAP.3(\underline{F'}, B, \bar{w}, M, T_O, \varepsilon_0, q, n, l, p, m, x.star.in.x\ ;$

$\underline{z}, \underline{\tilde{z}}\ :\ \underline{Q}, \tilde{w}, converged, re.use.jacobian)$

18.5.1.3. $\bar{w} := \tilde{w}$

18.5.2. *if* $n_I \geq maxit$ *do*

18.5.2.1. $maxit.reached := \underline{true}$

18.5.3. *while* $\sim converged$ *and* $\sim maxit.reached$ *do*

18.5.3.1. $n_I := n_I + 1$

18.5.3.2. $m := 0$

18.5.3.3. $re.use.jacobian := (p \neq 0)$

18.5.3.4. *if* $n_I \geq maxit$ *do*

18.5.3.4.1. $maxit.reached :=$ *true*

18.5.3.5. $T_S :=$ *time* ! (*Appendix D*).

18.5.3.6. $\underline{F'} := \underline{F'}(\tilde{\underline{z}})$

18.5.3.7. $A := m(\underline{F'})$

18.5.3.8. $\underline{y} := \tilde{\underline{z}}$

18.5.3.9. $MAP.2(\underline{F'}, A, \varepsilon_0, n, l\ ;\ \tilde{\underline{z}}, converged\ :\ \underline{Q}, \tilde{w})$

18.5.3.10. $M := \|\tilde{w}\| \,/\, \|w\|^2$

18.5.3.11. *if* $w(\underline{Q}) \leq \alpha w(\underline{y})$

then

18.5.3.11.1. $B := A$

18.5.3.11.2. $T_F :=$ *time* ! (*Appendix D*).

18.5.3.11.3. $T_O := T_F - T_S$

18.5.3.11.4. $old.z(\tilde{\underline{z}}, q, n, l, x.star.in.x\ :\ \underline{z})$! §6.8.

18.5.3.11.5. $\bar{w} := \tilde{w}$

18.5.3.11.6. *while* $re.use.jacobian$ *and*

$\sim converged$ *do*

18.5.3.11.6.1. $m := m + 1$

18.5.3.11.6.2. $MAP.3(\underline{F'}, A, \bar{w}, M, T_O,$

$\varepsilon_0, q, n, l, p, m$

$x.star.in.x$; $\underline{z}, \tilde{\underline{z}}$:

$\underline{Q}, \tilde{w}, converged,$

$re.use.jacobian)$

18.5.3.11.6.3. $\bar{w} := \tilde{w}$

else

18.5.3.11.7. $MAP.2(\underline{F}', B, \varepsilon_0, n, l$; $\tilde{\underline{z}}, converged$:

$\underline{Q}, \tilde{w})$

18.5.3.11.8. $T_F := time$! (*Appendix D*)

18.5.3.11.9. $T_O := T_F - T_S$

18.5.3.11.10. $old.z(\tilde{\underline{z}}, q, n, l, x.star.in.x$: $\underline{z})$! §6.8

18.5.3.11.11. $\bar{w} := \tilde{w}$

18.5.3.11.12. *while* $re.use.jacobian$ *and*

$\sim converged$ *do*

18.5.3.11.12.1. $m := m + 1$

18.5.3.11.12.2. $MAP.3(\underline{F}', A, \bar{w}, M, T_O,$

$\varepsilon_0, q, n, l, p, m,$

$x.star.in.x$; $\underline{z}$,

$\tilde{\underline{z}}$: $\underline{Q}, \tilde{w},$

$converged,$

$re.use.jacobian)$

18.5.3.11.12.3. $\bar{w} := \tilde{w}$

19. *return* □

6.11.2 The Application of $KMSW$

The algorithm $KMSW$ which is described in Chapter 5 for solving the system of equations $F(z) = 0$ given by 4.33 is supported by **Theorem 5.8.**

The algorithm $KMSW$ which is described in Chapter 5, with $\underline{z}$ and $\tilde{\underline{z}}$ as in §6.11.1, is used in the algorithm MW as follows.

(i)′ If $m(\underline{F}'^{(0)})$ in step 3 of $KMSW$ is singular then we cannot proceed with $KMSW$ so $\underline{z}$ is bisected into $\underline{z}^{(1)}$ and $\underline{z}^{(2)}$ by using the Moore-Jones rules which are described in §6.12 for use in §6.13.

(ii)′ If $KMSW$ terminates because an empty intersection occurs then there is no zero of F in $\tilde{\underline{z}}$. Therefore $\tilde{\underline{z}}$ and hence $\underline{z}$ can be deleted.

(iii)′ If $\underline{S}(\tilde{\underline{z}}) \not\subseteq \tilde{\underline{z}}$ where $\underline{S}(\tilde{\underline{z}})$ is computed in step 8.2.4 for $k = 0$ and $m = 0$ then we do not proceed with $KMSW$. Instead we compute

$$\tilde{\underline{z}}' = \underline{S}(\tilde{\underline{z}}) \cap \tilde{\underline{z}},$$

and

$$a = B^{(u)} F(m(\tilde{\underline{z}}))$$

where $B^{(0)}$ is computed in step 3 of $KMSW$ for use in §6.12 in which we bisect $\underline{z}'$ as in [Jon–78a] and $\underline{z}'$ is obtained from $\tilde{\underline{z}}'$ by using the procedure *old.z*.

(iv)′ If $\underline{S}(\tilde{\underline{z}}) \subseteq \tilde{\underline{z}}$, $w(\underline{S}(\tilde{\underline{z}})) < w(\underline{H}'(\tilde{\underline{z}}))$ and $\|\underline{R}\| < 1$ then we proceed with $KMSW$ to obtain the sequence $(\tilde{\underline{z}}^{(k)})$.

Suppose that for some $k \leq maxit$ where *maxit* is the maximum number of iterations which is permitted in $KMSW$ we have $\left\|w(\tilde{\underline{z}}^{(k)})\right\| \leq \varepsilon_0$ for $\varepsilon_0 > 0$. Then we obtain a box, say $\underline{z}^*$, with $\|w(\underline{z}^*)\| \leq \varepsilon_0$ which bounds the KT point z^* and $\underline{z}^*$ is pushed onto S_4.

(v)′ If $\underline{S}(\tilde{\underline{z}}) \subseteq \tilde{\underline{z}}$ and $w(\underline{S}(\tilde{\underline{z}})) \not< w(\underline{H}'(\tilde{\underline{z}}))$ then we replace $\tilde{\underline{z}}$ with $\underline{S}(\tilde{\underline{z}})$ but we do not proceed with $KMSW$. Instead we compute $\underline{z}$ from $\tilde{\underline{z}}$ by using the procedure *old.z* for use in §6.12 where $\tilde{\underline{z}}$ has already been replaced with $\underline{S}(\tilde{\underline{z}})$.

(vi)′ Suppose that $(\tilde{\underline{z}}^{(k)})$ is as in (iv)′. Suppose also that $\tilde{\underline{z}}^{(k)} \not\to z^*$ $(k \leq maxit)$. Then $\underline{z}^{(maxit)}$ is pushed onto S_5 where $\underline{z}^{(maxit)}$ is obtained from $\tilde{\underline{z}}^{(maxit)}$ by using the procedure *old.z*.

(vii)′ If $\tilde{\underline{z}}$ is reduced to $\tilde{\underline{z}}'$ with $\left\|w(\tilde{\underline{z}}')\right\| \leq \varepsilon_0$ then $\underline{z}'$, where $\underline{z}'$ is obtained from the box $\tilde{\underline{z}}'$ by using the procedure *old.z*, is pushed onto S_4 whether or not $\underline{z}'$ contains the solution z^* of $F(z) = 0$.

The pseudo-code for $KMSW$ which is used in MW contains the procedures *symmetric.hansen*, *new.z*, *old.z* which are described in §6.8, and the procedure

re.use.Jacobian which is described in Chapter 5. The procedures *KMSW.1*, *KMSW.2*, and *KMSW.3* which are described in this section, are also used.

The procedure *KMSW.1* is used if $m(\underline{F}'(\tilde{z}))$ is non-singular, and the procedures *KMSW.2* and *KMSW.3* are used if we have guaranteed that the current box $\underline{z}$ contains the zero z^* of $F(z) = 0$. These procedures are as follows.

procedure $KMSW.1(\underline{F}' \in I(M(R^{3n-l})), B \in M(R^{3n-l}), T_S, \varepsilon_0 \in R, q \in N^{\hat{}n}, n,$
$l \in N, x.star.in.x \in B$; $\underline{z} \in I(R^{3n}), \tilde{\underline{z}} \in I(R^{3n-l}),$
$\tilde{z}, \tilde{w} \in R^{3n-l}, no.zero, too.big, not.less, converged \in B$:
$\underline{R} \in I(M(R^{3n-l})), \underline{Q} \in I(R^{3n-l}), A \in M(R^{3n-l}), a,$
$M, T_O, r \in R)$

! This procedure implements steps 8.1 - 8.2 of *KMSW* for $(k = 0 \wedge m = 0)$ if
! $m(\underline{F}'^{(0)})$ which is given at step 3 of *KMSW* is non-singular. Also, this
! procedure determines which case of the cases (ii)', (iii)', (iv)', and (v)' is valid.
! On entry, $\underline{F}' = \underline{F}'(\tilde{\underline{z}})$ is computed from the procedure *F.prime* (Appendix D),
! $B = \{m(\underline{F}')\}^{-1}$ where $m(\underline{F}')$ is non-singular, T_S is the starting time
! (Appendix D) for *KMSW*, $\varepsilon_0 > 0$, q, n, l are as in §6.2, and *x.star.in.x* is as in
! §6.6, $\underline{z}$ is the current box, $\tilde{\underline{z}}$ is obtained from $\underline{z}$ by using the procedure *new.z*
! which is described in §6.8, $\tilde{z} = m(\tilde{\underline{z}})$, $\tilde{w} = w(\tilde{\underline{z}})$, $no.zero = \underline{true}$ if $z^* \notin \underline{z}$,

! *too.big* = *true* if $\underline{S}(\tilde{z}) \not\subseteq \tilde{z}$, *not.less* = *true* if $w(\underline{S}(\tilde{z})) \not< w(\underline{H}'(\tilde{z}))$, and

! *converged* = *true* if $\|w(\tilde{z})\| \leq \varepsilon_0$.

! On return, $\underline{R} = I - A\underline{F}'$ where $A = B$, $\underline{Q} = \underline{S}(\tilde{z})$, $a = AF(m(\tilde{z}))$, M is as in

! the procedure *re.use.Jacobian* (§5.3), T_O is the required CPU time for this

! procedure, and $r = \|\underline{R}\|$.

1. $T_O := 0$
2. $\underline{Q} := (\underline{0})_{3n-l\times 1}$
3. $M := 0$
4. $r := 0$
5. $\underline{y} := \tilde{z}$
6. $A := B$
7. $\underline{R} := I - A\underline{F}'$! *Step 5 of KMSW.*
8. $F := F(m(\underline{y}))$
9. $b := AF$
10. $a := b$! *For use in* §6.12.

! *We implement steps* 8.2.3 – 8.2.5 *of KMSW for* $(k = 0 \wedge m = 0)$

! *by using the procedure symmetric.hansen.*

11. *symmetric.hansen*$(\underline{R}, b, \tilde{z}, n, l, \underline{z}$: $\underline{S}, \underline{S}', \underline{H}', H'.empty, S'.empty)$! §6.8.
12. *if* $H'.empty$ *or* $S'.empty$

    then ! *Case* $(ii)'$.

12.1. $no.zero := \underline{true}$

else

12.2. $\underline{Q} := \underline{S}$

12.3. $\underline{\tilde{z}} := \underline{S'}$

12.4. $\tilde{z} := m(\underline{\tilde{z}})$

12.5. *if* $\underline{S} \not\subseteq \underline{y}$

then

! *Case* $(iii)'$. *If* $\underline{y} \subset \underline{S}$ *then* $\underline{\tilde{z}} = \underline{S'} = \underline{y}$. *If* $\underline{S} \cap \underline{y} \neq \emptyset$ *then*

! $\underline{\tilde{z}} = \underline{S'} = \underline{S} \cap \underline{y}$.

12.5.1. $too.big := \underline{true}$

12.5.2. $old.z(\underline{\tilde{z}}, q, n, l, x.star.in.x \ : \ \underline{z})$! §6.8.

else ! $\underline{S} \subseteq \underline{y}$

12.5.3. $r := \|\underline{R}\|$

12.5.4. *if* $w(\underline{S}) \not< w(\underline{H'})$ *or* $1 \leq r$

then ! *Case* $(v)'$.

12.5.4.1. $not.less := \underline{true}$

12.5.4.2. $old.z(\underline{\tilde{z}}, q, n, l, x.star.in.x \ : \ \underline{z})$! §6.8.

else ! *Case* $(iv)'$. $\underline{\tilde{z}}$ *must contain a unique zero of* F.

12.5.4.3. $T_F := time$! *Appendix D.*

12.5.4.4. $T_O := T_F - T_S$

! T_O *is the required CPU time for this procedure.*

12.5.4.5. $old.z(\tilde{\underline{z}}, q, n, l, x.star.in.x \ : \ \underline{z})$! §6.8.

12.5.4.6. $w := \tilde{w}$

12.5.4.7. $\tilde{w} := w(\tilde{\underline{z}})$

12.5.4.8. $M := \|\tilde{w}\| \, / \, \|w\|^2$

12.5.4.9. $converged := \|\tilde{w}\| \leq \varepsilon_0$! *Case* $(vii)'$.

13. *return* □

procedure $KMSW.2(\varepsilon_0 \in R, q \in N^{2n}, n, l \in N, x.star.in.x \in B \ ; \ A \in M(R^{3n-l}),$

$\underline{z} \in I(R^{3n}), \tilde{\underline{z}} \in I(R^{3n-l}), \tilde{z}, \tilde{w} \in R^{3n-l}, r \in R,$

$converged \in B \ : \ \underline{R} \in I(M(R^{3n-l})), M, T_O \in R)$

! This procedure implements steps 8.1 - 8.2 of $KMSW$ for $(k \geq 1 \wedge m = 0)$ by using

! the results from steps 8.3 - 8.8 of $KMSW$. In this procedure the value of M is

! computed for use in the procedure *re.use.Jacobian* (§5.3).

! All the parameters are as in the procedure $KMSW.1$.

1. $T_S := time$! *Appendix D.*

2. $\underline{F'} := \underline{F'}(\tilde{\underline{z}})$! *Step 8.4 of* $KMSW$.

3. $B := m(\underline{F'})$

4. *if* B *is singular* *do*

   ! *This procedure is used after we have guaranteed that* $\underline{z}$ *contains the*

! *unique zero of* $F(z) = 0$ *in* $\underline{z}$ *as described in the procedure* $KMSW.1$.

! *Therefore* B *should not be singular.*

4.1. *write* "B *is singular in* $KMSW.2$."

4.2. *stop*

5. $\underline{R} := I - B^{-1}\underline{F'}$! *Step* 8.6 *of* $KMSW$.

6. $s := \|\underline{R}\|$! *Step* 8.7 *of* $KMSW$.

7. *if* $s \leq r$! *Step* 8.8 *of* $KMSW$.

then

7.1. $A := B^{-1}$! *Step* 8.8.1 *of* $KMSW$.

7.2. $r := s$

else

7.3. $\underline{R} := I - A\underline{F'}$! *Step* 8.8.3 *of* $KMSW$.

7.4. $r := \|\underline{R}\|$! *Step* 8.8.4 *of* $KMSW$.

8. $F := F(\tilde{z})$

9. $b := AF$

! *Now, we implement steps* 8.2.3 – 8.2.5 *of* $KMSW$ *for* $(k \geq 1 \wedge m = 0)$.

10. $symmetric.hansen(\underline{R}, b, \tilde{z}, n, l, \underline{\tilde{z}} : \underline{S}, \underline{S'}, \underline{H'}, H'.empty, S'.empty)$! §6.8.

11. *if* $H'.empty$ *or* $S'.empty$ *do*

! *Since we have guaranteed that the box* $\underline{z}$ *contains the unique zero of*

! $F(z) = 0$ *in* $\underline{z}$ *as described in the procedure* $KMSW.1$ *then this possibility*

! *should not occur.*

11.1. *write* "H' *is empty or* S' *is empty in* $KMSW.2$."

11.2. *stop*

12. $T_F := time$! *Appendix D.*

13. $T_O := T_F - T_S$! *The required CPU time for this procedure.*

14. $\underline{\hat{z}} := \underline{S'}$

15. $\tilde{z} := m(\underline{\hat{z}})$

16. $old.z(\underline{\tilde{z}}, q, n, l, x.star.in.x \ : \ \underline{z})$! §6.8.

17. $\bar{w} := \tilde{w}$

18. $\tilde{w} := w(\underline{\tilde{z}})$

19. $M := \|\tilde{w}\| \, / \, \|\bar{w}\|^2$

20. $converged := \|\tilde{w}\| \leq \varepsilon_0$! *Case* $(vii)'$.

21. *return* □

procedure $KMSW.3(\underline{R} \in I(M(R^{3n-l})), A \in M(R^{3n-l}), M, T_O, \varepsilon_0 \in R, q \in N^{2n},$

$n, l, p, m \in N, x.star.in.x \in B \ ; \ \underline{z} \in I(R^{3n}), \underline{\tilde{z}} \in I(R^{3n-l}),$

$\tilde{z}, \tilde{w} \in R^{3n-l}, converged, re.use.jacobian \in B)$

! This procedure implements steps 8.1 - 8.2 of $KMSW$ for $(k \geq 0 \wedge m \geq 1)$.

! Also in this procedure, the procedure *re.use.Jacobian* (§5.3) is used to

! determine $p^{(k)}$ (§5.3).

! All the parameters are as in the procedures $KMSW.1$ and $KMSW.2$ except M, p,

! m, and *re.use.jacobian* which are explained in the proccdure *re.use.Jacobian*

! in Chapter 5.

1. $T_S := time$! *Appendix D.*

2. $F := F(\tilde{z})$

3. $b := AF$

4. $symmetric.hansen(\underline{R}, b, \tilde{z}, n, l, \underline{z} \ : \ \underline{S}, \underline{S'}, \underline{H'}, H'.empty, S'.empty)$! §6.8.

5. *if* $H'.empty$ *or* $S'.empty$ *do*

   ! *As step* 11 *of KMSW.2.*

   5.1. *write* "H' *is empty or* S' *is empty in KMSW.3.*"

   5.2. *stop*

6. $T_F := time$! *Appendix D.*

7. $T_I := T_F - T_S$! *The required CPU time for this procedure.*

8. $\underline{z} := \underline{S'}$

9. $\tilde{z} := m(\underline{z})$

10. $old.z(\tilde{z}, q, n, l, x.star.in.x \ : \ \underline{z})$! §6.8.

11. $\bar{w} := \tilde{w}$

12. $\tilde{w} := w(\underline{z})$

13. $converged := \|\tilde{w}\| \leq \varepsilon_0$! *Case (vii)'.*

14. $re.use.Jacobian(\bar{w}, \tilde{w}, M, T_O, T_I, p, m, 3n - l \ : \ re.use.jacobian)$! *Chapter* 5.

15. *return* □

The pseudo-code for the algorithm $KMSW$ as used in the algorithm MW is as follows.

procedure $KMSW(q \in N^{2n}, n, l, p, maxit \in N, \varepsilon_0 \in R, x.star.in.x \in B$;

$z \in I(R^{5n})$: $Q \in I(R^{3n-l}), a \in R^{3n-l}, singular,$

$not.less, no.zero, converged, maxit.reached \in B)$

! This procedure implements the algorithm $KMSW$ (Chapter 5) for use in the

! algorithm MW.

! All the parameters are explained in the procedures $KMSW.i$ $(i = 1, 2, 3)$ except

! $maxit$ and $maxit.reached$ where $maxit$ is the maximum number of iteration as

! explained in the case (iv)′ and $maxit.reached = true$ if $k \geq maxit$ where k is the

! number of times that step 8 in $KMSW$ is executed.

1. $new.z(z, q, n, l, x.star.in.x$: $\tilde{z})$! §6.8.
2. $\tilde{w} := w(\tilde{z})$
3. $\bar{w} := (0)_{3n-l \times 1}$
4. $M := 0$
5. $re.use.jacobian := (p \neq 0)$
6. $n_I := 1$
7. $singular :=$ *false*

8. $no.zero := \underline{false}$

9. $maxit.reached := \underline{false}$

10. $converged := \underline{false}$

11. $too.big := \underline{false}$

12. $not.less := \underline{false}$

13. $\underline{Q} := (\underline{0})_{3n-l\times 1}$

14. $a := (0)_{3n-l\times 1}$

15. $\tilde{z} := m(\underline{\tilde{z}})$

16. $T_S :=$ *time* ! *Appendix D.*

17. $\underline{F'} := \underline{F'}(\underline{\tilde{z}})$! *Appendix D.*

18. $B := m(\underline{F'})$

19. *if* B *is singular*

    then ! *Case* $(i)'$.

    19.1. $singular := \underline{true}$

    else

    19.2. $KMSW.1(\underline{F'}, B^{-1}, T_S, \varepsilon_0, q, n, l, x.star.in.x$; $\underline{z}, \underline{\tilde{z}}, \tilde{z}, \tilde{w}, no.zero,$

    $too.big, not.less, converged$: $\underline{R}, \underline{Q}, A, a, M, T_O, r)$

    19.3. $m := 1$

    19.4. *if* $\sim no.zero$ *and* $\sim too.big$ *and* $\sim not.less$ *do*

    19.4.1. *while* $re.use.jacobian$ *and* $\sim converged$ *do*

    19.4.1.1. $KMSW.3(\underline{R}, A, M, T_O, \varepsilon_0, q, n, l, p, m, x.star.in.x$;

$\underline{z}, \tilde{\underline{z}}, \tilde{z}, \tilde{w}, converged, re.use.jacobian)$

19.4.1.2. $m := m + 1$

19.4.2. $\underline{if}\ n_I = maxit\ \underline{do}$! *Case (vi)'*

19.4.2.1. $maxit.reached :=\ \underline{true}$

19.4.3. $\underline{while} \sim maxit.reached\ \underline{and} \sim converged\ \underline{do}$

19.4.3.1. $n_I := n_I + 1$

19.4.3.2. $re.use.jacobian := (p \neq 0)$

19.4.3.3. $\underline{if}\ n_I = maxit\ \underline{do}$

19.4.3.3.1. $maxit.reached :=\ \underline{true}$

19.4.3.4. $KMSW.2(\varepsilon_0, q, n, l, x.star.in.x\ ;\ A, \underline{z}, \tilde{\underline{z}}, \tilde{z}, \tilde{w}, r,$

$converged\ :\ \underline{R}, M, T_O)$

19.4.3.5. $m := 1$

19.4.3.6. $\underline{while}\ re.use.jacobian\ \underline{and} \sim converged\ \underline{do}$

19.4.3.6.1. $KMSW.3(\underline{R}, A, M, T_O, \varepsilon_0, q, n, l, p, m,$

$x.star.in.x\ ;\ \underline{z}, \tilde{\underline{z}}, \tilde{z}, \tilde{w}, converged,$

$re.use.jacobian)$

19.4.3.6.2. $m := m + 1$

20. $\underline{return}$ □

6.12 Bisection and Selection Rules for Use With The Moore-Jones Search

If we cannot guarantee either the existence or non-existence of a solution of $F(z) = 0$ in the current box $\underline{z}$ then for some $j \in \{1, \ldots, n\}$ we bisect $\underline{z}$ into the sub-boxes $\underline{z}^{(1)}$ and $\underline{z}^{(2)}$ along the jth co-ordinate direction $(1 \leq j \leq n)$ so that

$$\underline{z}^{(1)} = (\underline{x}_1, \ldots, \underline{x}_{j-1}, [x_{jI}, m(\underline{x}_j)], \underline{x}_{j+1}, \ldots, \underline{x}_n,$$

$$\underline{u}_1, \ldots, \underline{u}_{j-1}, \underline{u}_j^{(1)}, \underline{u}_{j+1}, \ldots, \underline{u}_n,$$

$$\underline{u}_{n+1}, \ldots, \underline{u}_{n+j-1}, \underline{u}_{n+j}^{(1)}, \underline{u}_{n+j+1}, \ldots, \underline{u}_{2n})^T \qquad 6.49$$

and

$$\underline{z}^{(2)} = (\underline{x}_1, \ldots, \underline{x}_{j-1}, [m(\underline{x}_j), x_{jS}], \underline{x}_{j+1}, \ldots, \underline{x}_n,$$

$$\underline{u}_1, \ldots, \underline{u}_{j-1}, \underline{u}_j^{(2)}, \underline{u}_{j+1}, \ldots, \underline{u}_n,$$

$$\underline{u}_{n+1}, \ldots, \underline{u}_{n+j-1}, \underline{u}_{n+j}^{(2)}, \underline{u}_{n+j+1}, \ldots, \underline{u}_{2n})^T. \qquad 6.50$$

Now $m(\underline{x}_j) \in int(\underline{x}_j)$, so in 6.49 $\underline{u}_j^{(1)} = \underline{u}_j$, $\underline{u}_{n+j}^{(1)} = \underline{0}$, and in 6.50 $\underline{u}_j^{(2)} = \underline{0}$ and $\underline{u}_{n+j}^{(2)} = \underline{u}_{n+j}$. Therefore we do not need to bisect $\underline{z}$ along any of the directions $\underline{u}_1, \ldots, \underline{u}_{2n}$.

The bisection procedure is determined by the rules which specify in which co-ordinate direction to bisect $\underline{x}$ and which half region to select for processing next. The following rules have been suggested by Jones [Jon–78a].

Bisection Rule 1 (Cyclic Direction Rule) : If initially $\underline{x} = \hat{\underline{x}}^{(i)}$ $(i \in \{0, \ldots, 2^n - 1\})$ then bisect $\underline{x}$ in co-ordinate direction 1. If $\underline{x}$ was the result of bisection in co-ordinate direction j, for $1 \leq j \leq n-1$, then bisect $\underline{x}$ in co-ordinate direction $j+1$. If $\underline{x}$ was the result of bisection in co-ordinate direction n, then bisect $\underline{x}$ in co-ordinate direction 1. □

Bisection Rule 1 (BR1) is contained in the procedure *cyclic.direction* which follows.

procedure cyclic.direction$(i, n \in N \ : \ j \in N)$

! This procedure determines the co-ordinate direction j in order to bisect the box $\underline{x}$

! and hence $\underline{z}$ according to $BR1$.

! On entry, i is the last co-ordinate direction in which bisection took place, and n is

! the number of variables in $\underline{x}$.

! On return, j is the co-ordinate direction in which bisection is next to occur.

1. $j := i + 1$

2. *if* $j > n$ *do*

   2.1. $j := 1$

3. *return* □

Bisection Rule 2 (Maximum Width Rule) : Bisect $\underline{x}$ in the co-ordinate direction j where $j \in \{1, \ldots, n\}$ is the least value of j such that

$$w(\underline{x}_j) = \max_{1 \le i \le n} \{w(\underline{x}_i)\}. \quad \square \qquad 6.51$$

The procedure *maximum.width.direction* which implements **Bisection Rule 2** ($BR2$) is as follows.

procedure maximum.width.direction$(\underline{z} \in I(R^{3n}), n \in N \ : \ j \in N)$

! This procedure computes j, where $j \in \{1, \ldots, n\}$ is the least value of j such that

! $$w(\underline{x}_j) = \max_{1 \le i \le n} \{w(\underline{x}_i)\},$$

! where $\underline{x}_i = \underline{z}_i \ (i = 1, \ldots, n)$.

! On entry, $\underline{z}$ is the current box to be bisected, and n is the number of variables in

! the initial box $\hat{\underline{x}}$.

1. $j := 1$

2. $w_{max} := w(\underline{z}_1)$

3. *for* $i = 2$ *to* n *do*

   3.1. $w := w(\underline{z}_i)$

   3.2. *if* $w > w_{max}$ *do*

      3.2.1. $j := i$

      3.2.2. $w_{max} := w$

4. *return* □

Bisection Rule 3 (Newton-Step Direction Rule) : Bisect $\underline{z}$ in the co-ordinate direction j where $j \in \{1, \ldots, n\}$ is the least value of j such that

$$| a_j | = \max_{1 \le i \le n} \{ | a_i | \} \qquad 6.52$$

in which the first n components of $a = \{m(\underline{F}'(\underline{z}))\}^{-1} F(m(\underline{z}))$ only are used and $\underline{F}'(\underline{z})$ and $F(m(\underline{z}))$ are computed from the procedures *F.prime* and F in Appendix D respectively. □

Procedure *newton.step.direction* which implements **Bisection Rule 3** ($BR3$) is as follows.

procedure *newton.step.direction*$(a \in R^{3n-l}, n \in N \ : \ j \in N)$

! This procedure computes j, where $j \in \{1, \ldots, n\}$ is the least value of j such that

! $$| a_j | = \max_{1 \le i \le n} \{ | a_i | \}.$$

! On entry, a is as in §6.11, and n is the number of components in the initial box $\underline{\hat{x}}$.

! On return, j is the co-ordinate direction in which $\underline{x}_j$ is to be bisected.

1. $j := 1$
2. $b_{max} := | a_1 |$
3. *for* $i = 2$ *to* n *do*
   - 3.1. $b := | a_i |$
   - 3.2. *if* $b > b_{max}$ *do*
     - 3.2.1. $j := i$
     - 3.2.2. $b_{max} := b$
4. *return* □

Selection Rule 1 (Static Region Selection) : Let $\underline{z}^{(1)}$ and $\underline{z}^{(2)}$ be the half-regions resulting from the bisection of $\underline{z}$. Always choose $\underline{z}^{(1)}$ as the next region for analysis and keep $\underline{z}^{(2)}$. □

The procedure *static.region.selection* which implements Selection Rule 1 ($SR1$) is as follows.

procedure *static.region.selection*(: *region.swap* $\in B$)

! This procedure determines that $\underline{z}^{(1)}$ is always to be chosen as the next region
! for analysis and that $\underline{z}^{(2)}$ should be kept, where $\underline{z}^{(1)}$ and $\underline{z}^{(2)}$ are the half-regions
! resulting from the bisection of $\underline{z}$.

1. *region.swap* := *true*
2. *return* □

Selection Rule 2 (Random Region Selection) : Randomly select one of the resulting half-regions as the next region to be analyzed. □

The procedure *random.region.selection* which contains **Selection Rule 2** ($SR2$) is as follows.

procedure *random.region.selection*($r \in N$: *region.swap* $\in B$)

! This procedure returns *region.swap* = *true* if $r = 0$ and *region.swap* = *false*
! otherwise.

! On entry, r has been determined from the random number generator which is

! described in §6.16, and takes the equi-probable values 0 and 1.

1. $region.swap := r = 0$

2. *return* □

Selection Rule 3 (Newton-Step Region Selection) : Choose the resulting half-region toward which $(-\{m(\underline{F}'(\underline{z}))\}^{-1}F(m(\underline{z})))_j$ points where j is as in $BR3$. □

The procedure *newton.step.region.selection* which implements Selection Rule 3 ($SR3$) is as follows.

procedure $newton.step.region.selection(a \in R^{3n-l}, n, r, j \in N \ : \ region.swap \in B)$

! This procedure implements $SR3$ where a is as in $BR3$, n is the number of

! components in the initial box $\underline{\hat{x}}$, r is as in the procedure

! *random.region.selection*, j is as in $BR3$ and *region.swap* is as in the

! previous procedures.

1. *case true of*

   $a_j < 0$:

1.1. *region.swap* := *false*

$0 < a_j$:

1.2. *region.swap* := *true*

default : ! $a_j = 0$

1.3. *random.region.selection*(*r* : *region.swap*)

2. *return* □

Note 6.4 : If *Deletion Test* 2 in §6.7 has produced two sub-boxes $\underline{z}^{(1)} (= \underline{y}^{(1)})$ and $\underline{z}^{(2)} (= \underline{y}^{(2)})$ of $\underline{z}$ then we do not need to apply the Moore-Jones technique to bisect $\underline{z}$. □

Suppose that $\underline{Q} = \underline{Q}(\tilde{\underline{z}})$ where $\underline{Q}$ is one of the operators which are used in MAP and $KMSW$ and $\tilde{\underline{z}}$ is obtained from the box $\underline{z}$ by using the procedure *new.z*. We bisect $\tilde{\underline{z}}$ as follows.

(a) If $\tilde{\underline{z}} \subset \underline{Q}$ and $m(\underline{Q}) \notin \tilde{\underline{z}}$ then we use $BR2$ and $SR2$.

(b) If $\tilde{\underline{z}} \subseteq \underline{Q}$ and $m(\underline{Q}) \in \tilde{\underline{z}}$ then we use $BR2$ and $SR3$.

(c) Let $\tilde{\underline{z}}' = \tilde{\underline{z}} \cap \underline{Q}$. If $m(\tilde{\underline{z}}) \notin \tilde{\underline{z}}'$ then we do not bisect $\tilde{\underline{z}}'$.

(d) Let $\tilde{\underline{z}}'$ be as defined in (c). If $m(\tilde{\underline{z}}) \in \tilde{\underline{z}}'$ and $m(\underline{Q}) \in \tilde{\underline{z}}'$ then we use $BR3$ and $SR3$.

(e) If $m(\tilde{\underline{z}}) \in \tilde{\underline{z}}'$ and $m(\underline{Q}) \notin \tilde{\underline{z}}'$ where $\tilde{\underline{z}}'$ is as defined in (c) then we use $BR2$ and $SR3$.

An explanation of (a) - (e) can be found in [Jon–78a].

In order to describe the procedure *bisect* which contains the preceding ideas we need the following subsidiary procedure.

procedure automatic.rule$(\underline{\tilde{z}}, \underline{Q} \in I(R^{3n-l}), \tilde{z}, Q \in R^{3n-l}, n, l \in N \ : \ c, s \in N)$

! This procedure determines which rule is to be used to bisect the current box $\underline{\tilde{z}}$ and
! which region is to be selected for processing next.
! On entry, $\underline{\tilde{z}}$ has been obtained from $\underline{z}$ by using the procedure *new.z*, $\underline{Q} = \underline{Q}(\underline{\tilde{z}})$,
! $\tilde{z} = m(\underline{\tilde{z}})$, $Q = m(\underline{Q})$, n is the number of components in the initial box $\underline{\hat{x}}$, and l
! is the number of Lagrange multipliers which take the value zero for the box $\underline{z}$.
! On return, $c \in \{1, 2, 3\}$ where 1,2, and 3 correspond to $BR1$, $BR2$, and $BR3$
! respectively, and $s \in \{1, 2, 3\}$ where 1,2, and 3 correspond to $SR1$, $SR2$, and
! $SR3$ respectively.

1. $c := 0$
2. $s := 0$
3. *case true of*

   $\underline{\tilde{z}} \subseteq \underline{Q}$ *and* $Q \in \tilde{z}$: ! (b).

   3.1. $c := 2$! $BR2$

   3.2. $s := 3$! $SR3$

   $\underline{\tilde{z}} \subset \underline{Q}$ *and* $Q \notin \tilde{z}$: ! (a)

3.3. $c := 2$! $BR2$

3.4. $s := 2$! $SR2$

$\tilde{z} \in \underline{\tilde{z}} \cap \underline{Q}$ *and* $Q \in \underline{\tilde{z}} \cap \underline{Q}$: ! (d)

3.5. $c := 3$! $BR3$

3.6. $s := 3$! $SR3$

$\tilde{z} \in \underline{\tilde{z}} \cap \underline{Q}$ *and* $Q \notin \underline{\tilde{z}} \cap \underline{Q}$: ! (e)

3.7. $c := 2$! $BR2$

3.8. $s := 3$! $SR3$

default :

3.9. *write* "*No such case in automatic.rule.*"

3.10. *stop* ! *Choices mutually exclusive.*

4. *return* □

Note 6.5 : The procedure *automatic.rule* is used when case (c) does not occur. Therefore case (c) does not appear in this procedure. □

The procedure *bisect* which contains the implementation of the strategies (a) - (e) is as follows.

procedure $bisect(\underline{z} \in I(R^{3n}), \underline{Q} \in I(R^{3n-l}), a \in R^{3n-l}, \varepsilon_0 \in R, q \in N^{2n}, l, r, n, c,$

$s \in N, singular, x.star.in.x, automatic, not.less \in B$; $i \in N$:

$\underline{z}^{(1)}, \underline{z}^{(2)} \in I(R^{3n}), bisect1, bisect2, bisect3 \in B)$

! This procedure determines the sub-boxes $\underline{z}^{(1)}$ and $\underline{z}^{(2)}$ from the box $\underline{z}$ by using

! BRi and SRi $(i = 1, 2, 3)$ according to the strategies (a) - (e). However, we bisect

! only the first n components $\underline{x}_i$ $(i = 1, \ldots, n)$ in $\underline{z}$.

! On entry, $\underline{Q} = \underline{Q}(\tilde{\underline{z}})$ where $\underline{Q}$ is one of the operators which are described in Chapter

! 5 and $\tilde{\underline{z}}$ is obtained from $\underline{z}$ by using the procedure *new.z*, $e_0 > 0$ (See §6.11.),

! $a = \{m(\underline{F}'(\tilde{\underline{z}}))\}^{-1} F(m(\tilde{\underline{z}}))$, q and l are as in §6.2, $r \in \{0, 1\}$ has been computed

! from the procedure *generate.random.number* in §6.16, c, $s \in \{0, 1, 2, 3\}$ where

! $c = 1, 2, 3$ correspond to BRi $(i = 1, 2, 3)$ respectively and $s = 1, 2, 3$ correspond

! to SRi $(i = 1, 2, 3)$ respectively, if $c = 0$ and $s = 0$ then the procedure

! *automatic.rule*, is used, *singular* is as in §6.8, *x.star.in.x* is as in §6.6, and

! *not.less* is as in §6.11.

! The input-output parameter i is as in the procedure *cyclic.direction*.

! On return, if $bisect1 = \underline{true}$ and $bisect2 = \underline{true}$ then $\underline{z}^{(1)} \neq \emptyset$ and $\underline{z}^{(2)} \neq \emptyset$ and if

! $\underline{z}$ is too small to bisect and might contain a global minimizer then $bisect3 = \underline{true}$.

1. $\underline{z}^{(1)} := (\underline{0})_{3n \times 1}$
2. $\underline{z}^{(2)} := (\underline{0})_{3n \times 1}$
3. $bisect1 := \underline{true}$
4. $bisect2 := \underline{true}$
5. $bisect3 := \underline{false}$

6. $new.z(\underline{z}, q, n, l, x.star.in.x \ : \ \underline{\tilde{z}})$! §6.8.

7. $\underline{\tilde{z}}' := \underline{Q} \cap \underline{\tilde{z}}$

8. $\tilde{z} := m(\underline{\tilde{z}})$

9. $no.bisection := (\tilde{z} \notin \underline{\tilde{z}}' \ \underline{or} \ not.less)$

10. *if automatic* <u>and</u> $\sim$ *no.bisection* <u>do</u>

    10.1. $Q := m(\underline{Q})$

    10.2. *if singular*

    <u>then</u>

    ! *BR2 and SR2 are used to bisect* $\underline{\tilde{z}}$ *and to select the half – region*

    ! *for processing next, respectively.*

    10.2.1. $c := 2$! *BR2.*

    10.2.2. $s := 2$! *SR2.*

    <u>else</u> ! *Apply one of the cases* $(a), (b), (d)$, *and* (e) *whichever is valid.*

    10.2.3. $automatic.rule(\underline{\tilde{z}}, \underline{Q}, \tilde{z}, Q, n, l \ : \ c, s)$

11. *if no.bisection*

    <u>then</u> ! *Case* (c).

    11.1. $\underline{\tilde{z}}^{(1)} := \underline{\tilde{z}}'$

    11.2. $\underline{\tilde{z}}^{(2)} := \emptyset$

    11.3. $old.z(\underline{\tilde{z}}^{(1)}, q, n, l, x.star.in.x \ : \ \underline{z}^{(1)})$! §6.8.

    11.4. $bisect2 :=$ <u>*false*</u>

    <u>else</u>

11.5. *case* c *of*

1 : ! $BR1$.

11.5.1. $cyclic.direction(i, n : j)$

2 : ! $BR2$.

11.5.2. $old.z(\tilde{z}', q, n, l, x.star.in.x : \underline{z}')$! §6.8.

11.5.3. $maximum.width.direction(\underline{z}', n : j)$

3 : ! $BR3$.

11.5.4. $newton.step.direction(a, n : j)$

default :

11.5.5. *write* "*No such bisection choice in bisect.*"

11.5.6. *stop*

11.6. *if* $\tilde{z}_j < \tilde{z}'_{jI}$ *or* $\tilde{z}'_{jS} < \tilde{z}_j$

then

! $\underline{z}$ *is too small to bisect.*

11.6.1. $bisect3 :=$ *true*

else

! *Steps* 11.6.2 *and* 11.6.3 *implement the strategy which is required*

! *when* $x^* \in \partial(\underline{\hat{x}})$ *and for at least one* $i \in \{1, \ldots, 2n\}$ $0 \in \underline{c}_i^0 \cap \underline{u}_i^0$.

! *See* §6.10. *This strategy prevents an excessively large number*

! *of bisections from taking place. This strategy also prevents an*

! *excessively large number of bisections from taking place due to*

! *the singularity of J or to the condition* $\underline{z} \subseteq \underline{S}$ *in the symmetric*

! *operator test.*

11.6.2. *maximum.width.direction*$(\underline{z}, n : k)$

11.6.3. *if* $k = j$ *and* $w(\underline{x}_j) \leq \varepsilon_0$! $\underline{z} = (\underline{x}^T, \underline{u}^T)^T$.

then

! *If the direction in which* $\underline{x}$ *is to be bisected is the direction*

! k *of maximum width and* $w(\underline{x}_j) \leq \varepsilon_0$ *then no further*

! *bisections are needed because* $\|w(\underline{x})\| \leq \varepsilon_0$.

11.6.3.1. *bisect3* := *true*

else

! *If* j *is not the direction of maximum width and* $w(\underline{x}_j) \leq \varepsilon_0$

! *and* $\|w(\underline{x})\| > \varepsilon_0$ *then* $\underline{x}$ *should not be bisected in the*

! *direction* j *in order to avoid producing long narrow boxes.*

! *Yet* $\underline{x}$ *should be bisected since* $\|w(\underline{x})\| > \varepsilon_0$. *So* $\underline{x}$ *is bisected*

! *in the direction* k *of maximum width instead of in the*

! *direction* j.

11.6.3.2. *if* $k \neq j$ *and* $w(\underline{x}_j) \leq \varepsilon_0$ *and* $\|w(\underline{x})\| > \varepsilon_0$ *do*

11.6.3.2.1. $j := k$

11.6.3.3. $i := j$! *For use in the next iteration.*

11.6.3.4. $\underline{y}^{(1)} := (\underline{\tilde{z}}'_1, \ldots, \underline{\tilde{z}}'_{j-1}, [\tilde{z}'_{jI}, m(\tilde{z}'_j)], \underline{\tilde{z}}'_{j+1}, \ldots, \underline{\tilde{z}}'_{3n-l})^T$

11.6.3.5. $\underline{y}^{(2)} := (\underline{\tilde{z}}'_1, \ldots, \underline{\tilde{z}}'_{j-1}, [m(\tilde{z}'_j), \tilde{z}'_{jS}], \underline{\tilde{z}}'_{j+1}, \ldots, \underline{\tilde{z}}'_{3n-l})^T$

11.6.3.6. *case* s *of*

1 : ! $SR1$.

11.6.3.6.1. *static.region.selection*(: *region.swap*)

2 : ! $SR2$.

11.6.3.6.2. *random.region.selection*(r : *region.swap*)

3 : ! $SR3$.

11.6.3.6.3. *newton.step.region.selection*(a, n, r, j : *region.swap*)

default :

11.6.3.6.4. *write* "*No such region selection in bisect.*"

11.6.3.6.5. *stop*

11.6.3.7. *if region.swap*

then

11.6.3.7.1. $\underset{\sim}{z}^{(1)} := \underset{\sim}{y}^{(2)}$

11.6.3.7.2. $\underset{\sim}{z}^{(2)} := \underset{\sim}{y}^{(1)}$

else

11.6.3.7.3. $\underset{\sim}{z}^{(1)} := \underset{\sim}{y}^{(1)}$

11.6.3.7.4. $\underset{\sim}{z}^{(2)} := \underset{\sim}{y}^{(2)}$

11.6.3.8. *old.z*($\underset{\sim}{z}^{(1)}, q, n, l, x.star.in.x$: $\underset{\sim}{z}^{(1)}$) ! §6.8.

11.6.3.9. *old.z*($\underset{\sim}{z}^{(2)}, q, n, l, x.star.in.x$: $\underset{\sim}{z}^{(2)}$) ! §6.8.

12. *return* □

6.13 Deletion Test 3

In §6.7 it is shown that the box $\underline{z}$ can be split into two disjoint sub-boxes $\underline{z}^{(1)}$ and $\underline{z}^{(2)}$. In §6.12 it is shown that the box $\underline{z}$ can also be split into two sub-boxes $\underline{z}^{(1)}$ and $\underline{z}^{(2)}$ which are not disjoint. If two sub-boxes are obtained then one of them must be pushed onto the stack S_1 of boxes to be processed and we process the other in the next iteration.

In order to avoid too many sub-boxes in S_1 we introduce two deletion tests which are called **Deletion Test 3.1** and **Deletion Test 3.2**. **Deletion Test 3.1** is used if $\underline{z}$ is split into two sub-boxes, which are produced by the tests in §6.7 or in §6.12. **Deletion Test 3.2** is used if the technique which is explained in §6.12 does not bisect $\underline{z}$.

6.13.1 Deletion Test 3.1

Suppose that $\underline{z}$ is split into two sub-boxes $\underline{z}^{(1)}$ and $\underline{z}^{(2)}$ which might be disjoint. If we compute $\underline{f}^{(1)} = \underline{f}(\underline{x}^{(1)})$ and $\underline{f}^{(2)} = \underline{f}(\underline{x}^{(2)})$ where $\underline{z}^{(1)} = (\underline{x}^{(1)^T}, \underline{u}^{(1)^T})^T$ and $\underline{z}^{(2)} = (\underline{x}^{(2)^T}, \underline{u}^{(2)^T})^T$, and we make a comparison between $\underline{\bar{f}}$, $\underline{f}^{(1)}$, and $\underline{f}^{(2)}$ then $\underline{z}^{(1)}$, $\underline{z}^{(2)}$ or both can be deleted as follows.

(i) $\bar{f}_S < f_I^{(1)}$ *or* $\bar{f}_S < f_I^{(2)}$: $\underline{z}^{(1)}$ or $\underline{z}^{(2)}$ can be deleted, whichever is appropriate.

(ii) $f_S^{(2)} < f_I^{(1)}$: $\underline{z}^{(1)}$ can be deleted.

(iii) $f_S^{(1)} < f_I^{(2)}$: $\underline{z}^{(2)}$ can be deleted.

(iv) $f_I^{(1)} \leq f_I^{(2)} \wedge f_S^{(1)} \leq f_S^{(2)}$: If $f(m(\underline{x}^{(1)})) < f_I^{(2)}$ then $\underline{z}^{(2)}$ can be deleted.

(v) $f_I^{(2)} \leq f_I^{(1)} \wedge f_S^{(2)} \leq f_S^{(1)}$: If $f(m(\underline{x}^{(2)})) < f_I^{(1)}$ then $\underline{z}^{(1)}$ can be deleted.

(vi) $\underline{f}^{(1)} \subseteq \underline{f}^{(2)} \vee \underline{f}^{(2)} \subseteq \underline{f}^{(1)}$: $\underline{z}^{(1)}$ and $\underline{z}^{(2)}$ cannot be deleted.

In the algorithm *MW* case (i) is used together with cases (ii) - (vi). Therefore it is possible that both $\underline{z}^{(1)}$ and $\underline{z}^{(2)}$ can be deleted.

6.13.2 Deletion Test 3.2

Suppose that $\underline{z}$ is the sub-box which is obtained at the end of any one iteration. Before we start the next iteration we make a comparison between $\underline{f}(\underline{x})$ and $\bar{f}$ where $\underline{z} = (\underline{x}^T, \underline{u}^T)^T$. If $\bar{f}_S < f_I$ then $\underline{z}$ can be deleted.

The pseudo-code for **Deletion Test 3.1** and **Deletion Test 3.2** is given in §6.16 in the procedures *delete.or.keep.z.*1 and *delete.or.keep.z.*2 respectively.

6.14 The Construction of New Lagrange Multiplier Bounds

Suppose that $\underline{z}'$ is a sub-box of $\underline{z} \subseteq \hat{\underline{z}}^{(i)}$ $(0 \leq i \leq 2^n - 1)$ which is obtained at the end of one of the iterations of *MW* and which will be submitted for the next iteration, and let $\underline{z}$ be the sub-box at the beginning of the current iteration. The box $\underline{z}'$ might be given by 6.49, 6.50 or case (c) of the bisection and selection strategy which is described in §6.12. Let

$$\underline{z}' = (\underline{x}'_1, \ldots, \underline{x}'_n, \underline{u}'_1, \ldots, \underline{u}'_n, \underline{u}'_{n+1}, \ldots, \underline{u}'_{2n})^T. \qquad 6.53$$

It is clear that one of the following holds : For $(i = 1, \ldots, 2n)$,

(i) $\underline{u}'_i = \underline{0}$;

(ii) $\underline{u}'_i = \underline{u}_i$;

(iii) $\underline{u}'_i$ is obtained as explained in §6.1.1 from the application of MAP or of $KMSW$;

(iv) $\underline{u}'_i$ is obtained from a combination of (i), (ii), or (iii).

Therefore we need to recalculate the $\underline{u}'_i$. The simplest way of computing the $\underline{u}'_i$ is by determining which of the u'_i are zero; this is done by checking the relationship between $\underline{x}'_i$ and $\hat{\underline{x}}_i$ as follows where $\hat{\underline{x}}$ is the initial box for *Problem P*.

(a) $(\hat{x}_{iI} = x'_{iI} \wedge x'_{iS} < \hat{x}_{iS}) \Rightarrow (\underline{u}'_i = \underline{u}_i \wedge \underline{u}'_{n+i} = \underline{0})$;

(b) $(\hat{x}_{iI} < x'_{iI} \wedge x'_{iS} < \hat{x}_{iS}) \Rightarrow (\underline{u}'_i = \underline{0} \wedge \underline{u}'_{n+i} = \underline{0})$;

(c) $(\hat{x}_{iI} < x'_{iI} \wedge x'_{iS} = \hat{x}_{iS}) \Rightarrow (\underline{u}'_i = \underline{0} \wedge \underline{u}'_{n+i} = \underline{u}_{n+i})$;

(d) otherwise $\underline{u}'_i$ is unchanged.

The preceding ideas are contained in the procedure *zero.multipliers* as follows.

procedure *zero.multipliers*$(\hat{\underline{x}}, \underline{x} \in I(R^n), n \in N \; : \; q \in N^{2n}, l \in N)$

! This procedure determines $q \in N^{2n}$ and $l \in N$.

! If $q_i = 0$ then $u_i = 0$; otherwise $u_i \neq 0$.

! On entry, $\hat{\underline{z}}$ is the initial box for *Problem P* and $\underline{z}$ is the current box where

! $\underline{z} = (\underline{x}^T, \underline{u}^T)^T$.

1. $l := 0$

2. $q := (1)_{2n \times 1}$

3. *for* $i = 1$ *to* n *do*

   3.1. *case true of*

   $\hat{x}_{iI} = x_{iI}$ *and* $x_{iS} < \hat{x}_{iS}$: ! (*a*).

   3.1.1. $q_{n+i} := 0$

   3.1.2. $l := l + 1$

   $\hat{x}_{iI} < x_{iI}$ *and* $x_{iS} < \hat{x}_{iS}$: ! (*b*).

   3.1.3. $q_i := 0$

   3.1.4. $q_{n+i} := 0$

   3.1.5. $l := l + 2$

   $\hat{x}_{iI} < x_{iI}$ *and* $x_{iS} = \hat{x}_{iS}$: ! (*c*).

   3.1.6. $q_i := 0$

   3.1.7. $l := l + 1$

   default : ! (*d*).

3.1.8. { }

4. *return* □

If we use only the procedure *zero.multipliers* to determine which $\underline{u}'_i = \underline{0}$ $(i = 1, \ldots, 2n)$ then we might obtain $\underline{u}'_i \neq \underline{0}$ even though $\underline{u}_i = \underline{0}$ $(i = 1, \ldots, 2n)$. Therefore, if $\underline{u}_i = \underline{0}$ then we must set $\underline{u}'_i = \underline{0}$ $(i = 1, \ldots, 2n)$. So we check whether or not both $\underline{u}_i = \underline{0}$ and $\underline{u}'_i = \underline{0}$ by using the procedure *check.q* which follows.

procedure *check.q*$(q \in N^{2n}, n \in N \; ; \; q' \in N^{2n}, l' \in N)$

! This procedure ensures that if $q_i = 0$ then $q'_i = 0$, and therefore that if

! $u_i = 0$ then $u'_i = 0$.

! On entry, q and q' correspond to $\underline{z}$ and $\underline{z}'$ respectively.

1. *for* $i = 1$ *to* $2n$ *do*

   1.1. *if* $q_i = 0$ *and* $q'_i = 1$ *do*

      1.1.1. $q'_i := 0$

      1.1.2. $l' := l' + 1$

2. *return* □

Note 6.6 : The procedures *zero.multipliers* and *check.q* must be used together in order to obtain the values of q and l for the box $\underline{z}' \subseteq \hat{\underline{z}}^{(i)}$ $(0 \leq i \leq 2^n - 1)$. □

After the sub-box $\underline{z}'$ which is given by 6.53 has been determined we compute $\underline{F}(\tilde{\underline{z}}')$ where $\tilde{\underline{z}}'$ is obtained from $\underline{z}'$ by using the procedure *new.z* (§6.8) before we push $\underline{z}'$ onto the stack S_1. This is done as follows.

procedure push$(\hat{\underline{x}} \in I(R^n), \underline{z}' \in I(R^{3n}), q \in N^{2n}, l, n \in N, x.star.in.x \in B$;

$S_1, S_2, S_3, S_6 \in P)$

! This procedure performs the following operations :

! (i) $\underline{z}' = (\underline{x}'^T, \underline{u}'^T)^T \longrightarrow S_1$;

! (ii) $q' \longrightarrow S_2$ where q' is computed by using the procedures *zero.multipliers*

! and *check.q*;

! (iii) $l' \longrightarrow S_3$ where l' is the number of Lagrange multiplier bounds in $\underline{z}'$ which

! take the value zero;

! (iv) $\underline{F} = \underline{F}(\tilde{\underline{z}}') \longrightarrow S_6$ where $\tilde{\underline{z}}'$ is obtained from $\underline{z}'$ by using the procedure

! *new.z* (§6.8).

! On entry, $\hat{\underline{x}}$ is the initial box for *Problem P*, $\underline{z}'$ is the current box which is obtained

! from the box $\underline{z}$, q corresponds to $\underline{z}$, l is the number of Lagrange multipliers

! bounds in $\underline{z}$ which take the value zero and *x.star.in.x* is as in §6.6.

1. *if* $x.star.in.x$ *or* $l = 2n$

   then ! $q = (0)_{2n\times 1}$ *and* $l = 2n$.

   1.1. $q \longrightarrow S_2$

   1.2. $l \longrightarrow S_3$

   1.3. $new.z(\underline{z}', q, n, l, x.star.in.x : \tilde{\underline{z}}')$! §6.8.

   else

   ! *We need to compute the Lagrange multiplier bounds for* $\underline{z}' = (\underline{x}'^T, \underline{u}'^T)^T$.

   1.4. $zero.multipliers(\hat{\underline{x}}, \underline{z}', n \ : \ q', l')$

   1.5. $check.q(q, n \ ; \ q', l')$

   1.6. *if* $0 < l'$ *do*

      1.6.1. *for* $i = 1$ *to* $2n$ *do*

         1.6.1.1. *if* $q'_i = 0$ *do*

            1.6.1.1.1. $\underline{z}'_{n+i} := \underline{0}$

   1.7. $q' \longrightarrow S_2$

   1.8. $l' \longrightarrow S_3$

   1.9. $new.z(\underline{z}', q', n, l', x.star.in.x : \tilde{\underline{z}}')$! §6.8.

2. $\underline{F} := \underline{F}(\tilde{\underline{z}}')$

3. $\underline{F} \longrightarrow S_6$

4. $\underline{z}' \longrightarrow S_1$

5. *return* □

6.15 The Stopping Criteria

In this section we describe how to terminate Algorithm *MW*. Suppose that we seek a solution z^* of $F(z) = 0$ in $\underline{\hat{z}}$ as explained in Chapter 4. We have determined sub-boxes $\underline{\hat{z}}^{(i)}$ $(0 \leq i \leq 2^n - 1)$ of $\underline{\hat{z}}$ which might contain zeros of F and we process these sub-boxes one by one begining with $\underline{\hat{z}}^{(0)}$ and finishing with $\underline{\hat{z}}^{(2^n-1)}$. The procedure *construct.z.i*, which is given in §6.3, is used to determine sub-box $\underline{\hat{z}}^{(i)}$ $i \in \{0, \ldots, 2^n - 1\}$ and to push it onto the stack S_1.

If $S_1 = \emptyset$ and $i = 2^n - 1$, so that the last sub-box $\underline{\hat{z}}^{(2^n-1)}$ has been processed, then we pop the box(es) containing the solution(s) of $F(z) = 0$ from the stacks S_4 and S_5. At least one of these boxes should contain the minimizer(s) of f over $\underline{\hat{x}}$.

If $S_4 = \emptyset$ and $S_5 = \emptyset$ then we pop the point box(es) from the stack S_7. At least one of these boxes might contain the minimizer(s) of f over $\underline{\hat{x}}$. Next we write $[\bar{f}_S - \epsilon_1, \bar{f}_S]$, which contains the least value of f over $\underline{\hat{x}}$, and stop.

If $S_4 = \emptyset$, $S_5 = \emptyset$, $S_7 = \emptyset$ and $\epsilon_1 > 0$ then we bound f^* but not x^* [Han–80a], so we write $[\bar{f}_S - \epsilon_1, \bar{f}_S]$, which contains the least value of f over $\underline{\hat{x}}$, and stop.

If $S_4 \neq \emptyset$ or $S_5 \neq \emptyset$ then we pop all the box(es) from the stack S_4 or S_5. If the number of boxes is more than one, $\underline{z}^{*1}, \ldots, \underline{z}^{*s}$, say, then we use the procedure *update.f.bar* to delete the unwanted boxes, write $[f_M, \bar{f}_S]$ where

$$f_M = \min_{1 \leq i \leq s} \{(\underline{f}(\underline{x}^{*i}))_I \mid \underline{z}^{*i} = (\underline{x}^{*i^T}, \underline{u}^{*i^T})^T \ (i = 1, \ldots, s)\}, \qquad 6.54$$

and stop. The procedure *compute.f.m* which computes f_M given by 6.54 is as follows.

procedure $compute.f.m(\underline{f} \in I(R^s), s \in N \ : \ f_M)$

! This procedure determines $f_M = min\{f_{1I}, \ldots, f_{sI}\}$ where $\underline{f} = (\underline{f}_1, \ldots, \underline{f}_s)^T$.

1. $f_M := f_{1I}$
2. *for* $i = 2$ *to* s *do*
   2.1. *if* $f_{iI} < f_M$ *do*
      2.1.1. $f_M := f_{iI}$
3. *return* □

If $S_1 = \emptyset$ and $i < 2^n - 1$ then we process the next sub-box $\underline{z}^{(i+1)}$. If $S_1 \neq \emptyset$ then we pop the sub-box $\underline{z}$ from the stack S_1 and its corresponding q, l, and $\underline{F} = \underline{F}(\underline{z})$ from the stacks S_2, S_3, and S_6 respectively for processing next.

The preceding ideas are contained in the following procedure.

procedure $termination(\varepsilon_1 \in R, n, i, n_4, n_5 \in N, f.bar.at.mid.x \in B, S_4, S_5 \in P$;
$S_1, S_2, S_3, S_6, S_7 \in P, n_7 \in N, \underline{\bar{f}}, \underline{\hat{f}} \in I(R)$:
$\underline{z} \in I(R^{3n}), \underline{F} \in I(R^{3n-l}), q \in N^{2n}, l \in N, next \in B)$

! This procedure determines how to terminate the algorithm *MW*.

! On entry, $\varepsilon_1 \geq 0$, n is the number of components in the initial box $\hat{\underline{x}}$,

! $i \in \{0, \ldots, 2^n - 1\}$, n_4 is the number of boxes in S_4, n_5 is the number of boxes

! in S_5 and *f.bar.at.mid.x* = *true* if $\bar{f}$ is computed at $m(\underline{x})$ where $\underline{z} = (\underline{x}^T, \underline{u}^T)^T$.

! The input-output parameters $S_1 - S_3$, S_6 and S_7 are stacks, n_7 is the number

! of point minimizers in S_7, $\underline{\bar{f}}$ is such that $f^* < \bar{f}_S$ where f^* is the global minimum

! of $f : R^n \to R^1$ and $\underline{f} = \underline{f}(m(\underline{x}))$.

! On return, $\underline{z}$, q, l, and $\underline{F}$ have been popped from the stacks S_1, S_2, S_3, and S_6

! respectively, and the Boolean *next* indicates whether or not to process the next

! sub-box $\hat{\underline{z}}^{(i+1)}$.

1. $\underline{z} := (\underline{0})_{3n \times 1}$
2. $\underline{F} := (\underline{0})_{3n-l \times 1}$
3. $q := (1)_{2n \times 1}$
4. $l := 0$
5. $next :=$ *false*
6. $n_S := n_4 + n_5$
7. *if* $S_1 = \emptyset$ *and* $i = 2^n - 1$

   then ! *All the sub – boxes* $\hat{\underline{z}}^{(i)}$ $(i = 0, \ldots, 2^n - 1)$ *have been processed.*

   7.1. *if* $n_S = 0$

then ! $S_4 = \emptyset$ *and* $S_5 = \emptyset$.

7.1.1. *if* $n_7 = 0$

then ! $S_7 = \emptyset$.

7.1.1.1. *if* $\varepsilon_1 \neq 0$

then

7.1.1.1.1. *write* $[\bar{f}_S - \varepsilon_1, \bar{f}_S]$

7.1.1.1.2. *stop*

else

7.1.1.1.3. *write* "*Epsilon 1 is zero in termination.*"

7.1.1.1.4. *stop*

else ! $S_7 \neq \emptyset$.

7.1.1.2. *for* $i = 1$ *to* n_7 *do*

7.1.1.2.1. $S_7 \longrightarrow \underline{z}^{*(i)}$! *Pop* $\underline{z}^{*(i)}$ *from* S_7.

7.1.1.2.2. *write* $\underline{z}^{*(i)}$

7.1.1.3. *write* $[\bar{f}_S - \varepsilon_1, \bar{f}_S]$

7.1.1.4. *stop*

else ! $S_4 \neq \emptyset$ *or* $S_5 \neq \emptyset$.

7.1.2. $\underline{\tilde{f}} := (\underline{0})_{n_S \times 1}$

7.1.3. $n_M := 0$! *The number of minimizers.*

7.1.4. *for* $i = 1$ *to* n_4 *do*

7.1.4.1. $S_4 \longrightarrow \underline{z}^{(i)}$

7.1.5. *for* $i = 1$ *to* n_5 *do*

7.1.5.1. $S_5 \longrightarrow \underline{z}^{(n_4+i)}$

7.1.6. *for* $i = 1$ *to* n_S *do*

! *Delete the boxes which do not contain minimizer(s)*

! *by using the procedure update.f.bar.*

7.1.6.1. $\tilde{\underline{f}}_i := \underline{f}(\underline{x}^{(i)})$! $\underline{z}^{(i)} = (\underline{x}^{(i)T}, \underline{u}^{(i)T})^T$.

7.1.6.2. *update.f.bar*$(\underline{x}^{(i)}, \tilde{\underline{f}}_i \; ; \; S_4, \bar{\underline{f}}, n_M)$! §6.4.

7.1.7. *compute.f.m*$(\tilde{\underline{f}}, n_S \; : \; f_M)$! §6.15.

7.1.8. *for* $i = 1$ *to* n_M *do*

7.1.8.1. $S_4 \longrightarrow \underline{z}^{*(i)}$! *Pop* $\underline{z}^{*(i)}$ *from* S_4.

7.1.8.2. *write* $\underline{z}^{*(i)}$

7.1.9. *write* $[f_M, \bar{f}_S]$

7.1.10. *stop*

else ! $S_1 \neq \emptyset$ *or* $i \neq 2^n - 1$.

7.2. *if* $S_1 = \emptyset$

then ! $i \neq 2^n - 1$.

7.2.1. *next* := *true*

else ! $S_1 \neq \emptyset$.

7.2.2. $S_1 \longrightarrow \underline{z}$! *Pop* $\underline{z}$ *from* S_1.

7.2.3. *if* $\sim$ *f.bar.at.mid.x do*

7.2.3.1. $x := m(\underline{x})$! $\underline{z} = (\underline{x}^T, \underline{u}^T)^T$.

7.2.3.2. $\underline{f} := \underline{f}(x)$! *x is a point box.*

7.2.3.3. *update.f.bar*$(x, \underline{f}\ ;\ S_7, \bar{\underline{f}}, n_7)$! §6.4.

7.2.4. $S_2 \longrightarrow q$! *Pop q from* S_2.

7.2.5. $S_3 \longrightarrow l$! *Pop l from* S_3.

7.2.6. $S_6 \longrightarrow \underline{F}$! *Pop* $\underline{F}$ *from* S_6.

8. *return* □

6.16 The Algorithm *MW*

In order to describe the algorithm *MW*, we need the following subsidiary procedures.

procedure delete.or.keep.z.1$(\underline{\hat{x}} \in I(R^n), \underline{z}^{(1)}, \underline{z}^{(2)} \in I(R^{3n}), \underline{f}^{(1)}, \underline{f}^{(2)} \in I(R),$

$q \in N^{2n}, n, l \in N, x.star.in.x \in B\ ;\ S_1, S_2, S_3, S_6 \in P,$

$\bar{\underline{f}} \in I(R), n_7 \in N\ :\ \underline{f} \in I(R), f.bar.at.mid.x \in B)$

! This procedure determines how to delete the boxes $\underline{z}^{(1)}$ and $\underline{z}^{(2)}$ according to

! deletion test 3.1 in §6.13. If $\underline{z}^{(1)}$ or $\underline{z}^{(2)}$ cannot be deleted then $\underline{z}^{(1)}$ or $\underline{z}^{(2)}$ and

! the corresponding q, l and $\underline{F}$ are pushed onto the stacks S_1, S_2, S_3 and S_6

! respectively where l and q are as in §6.2 and $\underline{F}$ is either $\underline{F}(\tilde{\underline{z}}^{(1)})$ or $\underline{F}(\tilde{\underline{z}}^{(2)})$

! whichever is appropriate and $\tilde{\underline{z}}^{(1)}$ and $\tilde{\underline{z}}^{(2)}$ are obtained from $\underline{z}^{(1)}$ and $\underline{z}^{(2)}$

! respectively by the procedure *new.z*.

! On entry, $\underline{\hat{x}}$ is the initial box, $\underline{z}^{(1)}$ and $\underline{z}^{(2)}$ are the boxes which are obtained either

! by using the test in §6.7 or by using the technique in §6.12, $\underline{f}^{(1)} = \underline{f}(\underline{x}^{(1)})$ and

! $\underline{f}^{(2)} = \underline{f}(\underline{x}^{(2)})$ where $\underline{z}^{(1)} = (\underline{x}^{(1)^T}, \underline{u}^{(1)^T})^T$ and $\underline{z}^{(2)} = (\underline{x}^{(2)^T}, \underline{u}^{(2)^T})^T$ respectively,

! q corresponds to $\underline{z}$, $\underline{z}$ is the initial box for each iteration, l is the number of

! Lagrange multiplier bounds in $\underline{z}$, and *x.star.in.x* is as in §6.6.

! The input-output parameters S_1, S_2, S_3, and S_6 are stacks and $\underline{\bar{f}}$ and n_7 are

! as in §6.6.

! On return, $\underline{f}$ is the function value at the mid-point of whichever of $\underline{x}^{(1)}$ or $\underline{x}^{(2)}$

! is not deleted and which is selected to be processed next. If both $\underline{z}^{(1)}$ and $\underline{z}^{(2)}$

! are deleted then $\underline{f}$ is ignored by making *f.bar.at.mid.x* = *false*.

1. $\underline{f} := \underline{0}$

2. *f.bar.at.mid.x* := *false*

3. *case true of*

   $f_S^{(2)} < f_I^{(1)}$: ! *(ii) in* §6.13, $\underline{z}^{(1)}$ *is deleted.*

   3.1. *if* $f_I^{(2)} \leq \bar{f}_S$ *do*

   ! *If* $\bar{f}_S < f_I^{(2)}$ *((i) in* §6.13*) then* $\underline{z}^{(2)}$ *is deleted.*

   3.1.1. $\underline{f} := \underline{f}(m(\underline{x}^{(2)}))$! $\underline{z}^{(2)} = (\underline{x}^{(2)^T}, \underline{u}^{(2)^T})^T$.

   3.1.2. *f.bar.at.mid.x* := *true*

   3.1.3. *if* $f_S < \bar{f}_I$ *do*

3.1.3.1. $\underline{\bar{f}} := [f_S, f_S]$

3.1.3.2. $n_7 := 0$! $\underline{\bar{f}}$ *can be updated at* $m(\underline{x}^{(2)})$ *but* $m(\underline{x}^{(2)}) \not\longrightarrow S_7$.

3.1.4. $push(\hat{\underline{x}}, \underline{z}^{(2)}, q, l, n, x.star.in.x\ ;\ S_1, S_2, S_3, S_6)$! §6.14.

$f_S^{(1)} < f_I^{(2)}$: ! *(iii) in* §6.13, $\underline{z}^{(2)}$ *is deleted.*

3.2. *if* $f_I^{(1)} \leq \bar{f}_S$ *do*

! *If* $\bar{f}_S < f_I^{(1)}$ *((i) in* §6.13*) then* $\underline{z}^{(1)}$ *is deleted.*

3.2.1. $\underline{f} := \underline{f}(m(\underline{x}^{(1)}))$! $\underline{z}^{(1)} = (\underline{x}^{(1)^T}, \underline{u}^{(1)^T})^T$.

3.2.2. $f.bar.at.mid.x :=$ *true*

3.2.3. *if* $f_S < \bar{f}_I$ *do*

3.2.3.1. $\underline{\bar{f}} := [f_S, f_S]$

3.2.3.2. $n_7 := 0$! $\underline{\bar{f}}$ *can be updated at* $m(\underline{x}^{(1)})$ *but* $m(\underline{x}^{(1)}) \not\longrightarrow S_7$.

3.2.4. $push(\hat{\underline{x}}, \underline{z}^{(1)}, q, l, n, x.star.in.x\ ;\ S_1, S_2, S_3, S_6)$! §6.14.

$f_I^{(1)} \leq f_I^{(2)}$ *and* $f_S^{(1)} \leq f_S^{(2)}$: ! *(iv) in* §6.13.

3.3. *if* $f_I^{(1)} \leq \bar{f}_S$ *do*

! *If* $\bar{f}_S < f_I^{(1)}$ *then* $\underline{z}^{(1)}$ *and* $\underline{z}^{(2)}$ *can be deleted.*

3.3.1. $\underline{f} := \underline{f}(m(\underline{x}^{(1)}))$! $\underline{z}^{(1)} = (\underline{x}^{(1)^T}, \underline{u}^{(1)^T})^T$.

3.3.2. $f.bar.at.mid.x :=$ *true*

3.3.3. *if* $f_I^{(2)} \leq f_S$ *do*

! *If* $f_S < f_I^{(2)}$ *then* $\underline{z}^{(2)}$ *can be deleted.*

3.3.3.1. $push(\hat{\underline{x}}, \underline{z}^{(2)}, q, l, n, x.star.in.x\ ;$

$S_1, S_2, S_3, S_6)$! §6.14.

3.3.4. *if* $f_S < \bar{f}_I$ *do*

3.3.4.1. $\underline{\bar{f}} := [f_S, f_S]$

3.3.4.2. $n_7 = 0$! $\underline{\bar{f}}$ *can be updated at* $m(\underline{x}^{(1)})$ *but* $m(\underline{x}^{(1)}) \not\longmapsto S_7$.

3.3.5. $push(\underline{\hat{x}}, \underline{z}^{(1)}, q, l, n, x.star.in.x\ ;\ S_1, S_2, S_3, S_6)$! §6.14.

$f_I^{(2)} \leq f_I^{(1)}$ *and* $f_S^{(2)} \leq f_S^{(1)}$: ! (v) *in* §6.13.

3.4. *if* $f_I^{(2)} \leq \bar{f}_S$ *do*

! *If* $\bar{f}_S < f_I^{(2)}$ *then* $\underline{z}^{(2)}$ *and* $\underline{z}^{(1)}$ *can be deleted.*

3.4.1. $\underline{f} := \underline{f}(m(\underline{x}^{(2)}))$! $\underline{z}^{(2)} = (\underline{x}^{(2)^T}, \underline{u}^{(2)^T})^T$.

3.4.2. $f.bar.at.mid.x :=$ *true*

3.4.3. *if* $f_I^{(1)} \leq f_S$ *do*

! *If* $f_S < f_I^{(1)}$ *then* $\underline{z}^{(1)}$ *can be deleted.*

3.4.3.1. $push(\underline{\hat{x}}, \underline{z}^{(1)}, q, l, n, x.star.in.x\ ;$

$S_1, S_2, S_3, S_6)$! §6.14.

3.4.4. *if* $f_S < \bar{f}_I$ *do*

3.4.4.1. $\underline{\bar{f}} := [f_S, f_S]$

3.4.4.2. $n_7 = 0$! $\underline{\bar{f}}$ *can be updated at* $m(\underline{x}^{(2)})$ *but* $m(\underline{x}^{(2)}) \not\longmapsto S_7$.

3.4.5. $push(\underline{\hat{x}}, \underline{z}^{(2)}, q, l, n, x.star.in.x\ ;\ S_1, S_2, S_3, S_6)$! §6.14.

default : ! (vi) *in* §6.13.

3.5. *if* $f_I^{(1)} \leq \bar{f}_S$ *do*

! *If* $\bar{f}_S < f_I^{(1)}$ *then* $\underline{z}^{(1)}$ *can be deleted. Case* (i) *in* §6.13.

3.5.1. *if* $f_S^{(1)} < \bar{f}_I$ *do*

3.5.1.1. $\underline{\bar{f}} := [f_S^{(1)}, f_S^{(1)}]$

3.5.1.2. $n_7 := 0$! $\underline{\bar{f}}$ *can be updated in* $\underline{x}^{(1)}$ *but* $\underline{x}^{(1)} \not\longrightarrow S_7$.

3.5.2. $push(\underline{\hat{x}}, \underline{z}^{(1)}, q, l, n, x.star.in.x\ ;\ S_1, S_2, S_3, S_6)$! §6.14.

3.6. *if* $f_I^{(2)} \leq \bar{f}_S$ *do*

! *If* $\bar{f}_S < f_I^{(2)}$ *then* $\underline{z}^{(2)}$ *can be deleted. Case* (*i*) *in* §6.13.

3.6.1. *if* $f_S^{(2)} < \bar{f}_I$ *do*

3.6.1.1. $\underline{\bar{f}} := [f_S^{(2)}, f_S^{(2)}]$

3.6.1.2. $n_7 := 0$! $\underline{\bar{f}}$ *can be updated in* $\underline{x}^{(2)}$ *but* $\underline{x}^{(2)} \not\longrightarrow S_7$.

3.6.2. $push(\underline{\hat{x}}, \underline{z}^{(2)}, q, l, n, x.star.in.x\ ;\ S_1, S_2, S_3, S_6)$! §6.14.

4. *return* □

procedure $delete.or.keep.z.2(\underline{\hat{x}} \in I(R^n), \underline{z}' \in I(R^{3n}), q \in N^{2n}, n, l \in N,$

$x.star.in.x \in B\ ;\ S_1, S_2, S_3, S_6 \in P, \underline{\bar{f}}, \underline{f} \in I(R),$

$n_7 \in N\ :\ f.bar.at.mid.x \in B)$

! This procedure determines how to delete the sub-box $\underline{z}'$ according to deletion

! test 3.2 in §6.13. If $\underline{z}'$ cannot be deleted then $\underline{z}'$, and its corresponding q, l, and F

! are pushed onto the stacks S_1, S_2, S_3, and S_6 respectively where q and l are as in

! §6.2 and $\underline{F} = \underline{F}(\underline{\tilde{z}}')$ in which $\underline{\tilde{z}}'$ is obtained from $\underline{z}'$ by using the procedure *new.z*.

! All the parameters are as in the procedure *delete.or.keep.z.1* except $\underline{z}'$ where $\underline{z}'$ is

! the sub-box which is obtained by processing the box $\underline{z}$ in one of the iterations

! of MW.

1. $f.bar.at.mid.x := \underline{false}$

2. $\underline{if}\ f_I \leq \bar{f}_S\ \underline{do}$

   ! *If* $\bar{f}_S < f_I$ *then* $\underline{z}'$ *can be deleted.*

   2.1. $\underline{f} := \underline{f}(m(\underline{x}'))$! $\underline{z}' = (\underline{x}'^T, \underline{u}'^T)^T$.

   2.2. $f.bar.at.mid.x := \underline{true}$

   2.3. $\underline{if}\ f_S < \bar{f}_I\ \underline{do}$

      2.3.1. $\underline{\bar{f}} := [f_S, f_S]$

      2.3.2. $n_7 := 0$! $\underline{\bar{f}}$ *can be updated at* $m(\underline{x}')$ *but* $m(\underline{x}') \not\longrightarrow S_7$.

   2.4. $push(\hat{\underline{x}}, \underline{z}', q, l, n, x.star.in.x\ ;\ S_1, S_2, S_3, S_6)$! §6.14.

3. $\underline{return}$ □

$\underline{procedure}\ solve(\hat{\underline{x}} \in I(R^n), \hat{q}, q \in N^{2n}, m, n, l, p, maxit \in N, \varepsilon_0, \varepsilon_1 \in R,$

$x.star.in.x \in B\ ;\ S_1, S_2, S_3, S_4, S_5, S_6 \in P, \underline{z} \in I(R^{3n}), \underline{\bar{f}} \in I(R),$

$n_4, n_5, n_7 \in N\ :\ \underline{Q} \in I(R^{3n-l}), a \in R^{3n-l}, not.less,$

$push.or.delete.z, singular, go.to.deletion.test.2 \in B)$

! This procedure implements algorithms MAP and $KMSW$ which are described in

! Chapter 5.

! On entry, all the parameters are as explained in the previous sections except

! $m \in \{1, 2\}$ where if $m = 1$ then algorithm MAP is used and if $m = 2$ then

! algorithm $KMSW$ is used.

! The input-output parameters are as explained in previous sections.

! On return, $\underline{Q} = \underline{Q}(\tilde{z})$ where $\underline{Q}$ is one of the operators which are described in §6.11,

! a is as in §6.11, *push.or.delete.z* = *true* if $\underline{z} \longrightarrow S_1$, $\underline{z} \longrightarrow S_4$, $\underline{z} \longrightarrow S_5$

! or $\underline{z}$ is deleted, *singular*, and *not.less* are as in §6.11.

! If by the procedures MAP or $KMSW$ (See §6.11.) *converged* = *true* (or

! *maxit.reached* = *true*) then $\underline{z}$ is pushed onto S_4 (or S_5). However, if

! $\varepsilon_1 > 0$ then f^* is to be bounded but not x^*. Therefore if *converged* = *true*

! (or *maxit.reached* = *true*) and $\varepsilon_1 > 0$ then we do not push $\underline{z}$ onto S_4 (or S_5)

! because we wish to process $\underline{z}$ until $\underline{z}$ is deleted. Therefore $\underline{z}$ is pushed onto S_1

! for processing next. Since $\underline{z}$ is small and might contain a global minimizer, we

! start to process $\underline{z}$ using the procedure *deletion.test.2* so we set

! *go.to.deletion.test.2* := *true* to denote this fact.

1. $\underline{Q} := (\underline{0})_{3n-t\times 1}$
2. $a := (0)_{3n-t\times 1}$
3. *not.less* := *false*
4. *singular* := *false*
5. *push.or.delete.z* := *false*

6. *go.to.deletion.test.2* := *false*

7. *case m of*

1 :

7.1. $MAP(q, n, l, p, maxit, \varepsilon_0, x.star.in.x\ ;\ \underline{z}\ :\ \underline{Q}, a, singular,$

$not.less, no.zero, converged, maxit.reached)$! §6.11.1.

2 :

7.2. $KMSW(q, n, l, p, maxit, \varepsilon_0, x.star.in.x\ ;\ \underline{z}\ :\ \underline{Q}, a, singular,$

$not.less, no.zero, converged, maxit.reached)$! §6.11.2.

default : ! *No such algorithm.*

7.3. *write* "*Inappropriate value for m in solve.*"

7.4. *stop*

! *After one of the algorithms MAP and KMSW is used the following*

! *cases might occur.*

8. *case true of*

converged : ! $\|w(\underline{z})\| \leq \varepsilon_0$, *cases* (v) *for MAP and* $(iv)'$ *for KMSW.*

8.1. $\underline{f} := \underline{f}(\underline{x})$! $\underline{z} = (\underline{x}^T, \underline{u}^T)^T$.

8.2. *if* $f_I \leq \bar{f}_S$ *do*

8.2.1. *if* $f_S < \bar{f}_I$ *do*

8.2.1.1. $\bar{\underline{f}} := [f_S, f_S]$

8.2.1.2. $n_7 := 0$! $\underline{x} \not\longrightarrow S_7$.

8.2.2. *if* $0 < \varepsilon_1$

then

! f^* *is bounded but not* x^*. *Therefore we*

! *push* $\underline{z}$ *onto* S_1 *for processing next.*

8.2.2.1. *go.to.deletion.test.2* := *true*

8.2.2.2. *push*$(\underline{\hat{x}}, \underline{z}, q, l, n, x.star.in.x$;

$S_1, S_2, S_3, S_6)$! §6.14.

else

8.2.2.3. $\underline{z} \longrightarrow S_4$! $\underline{z}$ *is pushed onto* S_4.

8.2.2.4. $n_4 := n_4 + 1$

8.3. *push.or.delete.z* := *true*

no.zero : ! *The cases* (ii) *and*$(ii)'$ *for MAP and KMSW respectively.*

8.4. *push.or.delete.z* := *true*

maxit.reached : ! $\underline{z}^{(k)} \not\to z^*$ $(k \leq maxit)$ (§6.11).

8.5. $\underline{f} := \underline{f}(\underline{x})$! $\underline{z} = (\underline{x}^T, \underline{u}^T)^T$.

8.6. *if* $f_I \leq \bar{f}_S$ *do*

8.6.1. *if* $f_S < \bar{f}_I$ *do*

8.6.1.1. $\bar{\underline{f}} := [f_S, f_S]$

8.6.1.2. $n_7 := 0$! $\underline{x} \not\longrightarrow S_7$.

8.6.2. *if* $0 < \varepsilon_1$

then ! *See step* 8.2.2. *of this procedure.*

8.6.2.1. *go.to.deletion.test.2* := *true*

8.6.2.2. $push(\underline{\hat{x}}, \underline{z}, q, l, n, x.star.in.x$;

$S_1, S_2, S_3, S_6)$! §6.14.

else

8.6.2.3. $\underline{z} \longrightarrow\triangleright S_5$! $\underline{z}$ *might or might not contain* z^*.

8.6.2.4. $n_5 := n_5 + 1$

8.7. *push.or.delete.z* := *true*

default :

! *Cases* (i) *and* $(i)'$ *for MAP and KMSW respectively are included here.*

8.8. *if* $\|w(\underline{z})\| \leq \varepsilon_0$ *do*

8.8.1. $\underline{f} := \underline{f}(\underline{x})$! $\underline{z} = (\underline{x}^T, \underline{u}^T)^T$.

8.8.2. *if* $f_I \leq \bar{f}_S$ *do*

8.8.2.1. *if* $f_S < \bar{f}_I$ *do*

8.8.2.1.1. $\bar{\underline{f}} := [f_S, f_S]$

8.8.2.1.2. $n_7 := 0$! $\underline{x} \not\longrightarrow\triangleright S_7$.

8.8.2.2. *if* $0 < \varepsilon_1$

then ! *See step* 8.2.2 *of this procedure.*

8.8.2.2.1. *go.to.deletion.test.2* := *true*

8.8.2.2.2. $push(\underline{\hat{x}}, \underline{z}, q, l, n, x.star.in.x$;

$S_1, S_2, S_3, S_6)$! §6.14.

else

8.8.2.2.3. $\underline{z} \longrightarrow\triangleright S_4$! $\underline{z}$ *is pushed onto* S_4.

8.8.2.2.4. $n_4 := n_4 + 1$

8.8.3. *push.or.delete.z* := *true*

9. *return* □

procedure time.to.real$(s \in S \ : \ t \in R)$

! The CPU (central processing unit) time since logging in is expressed by the system

! procedure *time* as a s string in the form $n1, n2 : n3, n4 : n5, n6 : n7, n8$ where the

! decimal digits ni $(i = 1, \ldots, 8)$ represent the time in hours, minutes, seconds,

! tenths of a second and hundredths of a second from midnight.

1. *for* $i = 1$ *to* 2 *do*

   1.1. $n_i := decode(s(i \mid 1)) - decode("0")$! *Appendix D.*

   ! $s(i \mid 1)$ *is the sub – string of s consisting of the ith character*

   ! *of s, and* "0" *is zero expressed as a string.*

   ! *The procedure decode is such that* $decode(t)$ *is the ASCII*

   ! *code corresponding to the character t.*

2. *for* $i = 3$ *to* 4 *do*

   2.1. $n_i := decode(s(i + 1 \mid 1)) - decode("0")$! *Appendix D.*

3. *for* $i = 5$ *to* 6 *do*

   3.1. $n_i := decode(s(i + 2 \mid 1)) - decode("0")$! *Appendix D.*

4. *for* $i = 7$ *to* 8 *do*

   4.1. $n_i := decode(s(i+3 \mid 1)) - decode("0")$! *Appendix D.*

5. $t := 3600(10n_1 + n_2) + 60(10n_3 + n_4) + (10n_5 + n_6) + n_7/10 + n_8/100$

6. *return* □

procedure generate.random.number(: $r \in N$)

! This procedure generates an integer $r \in \{0, 1\}$ from the procedure *time*

! (Appendix D) which returns the CPU time used since logging in.

1. $r := truncate(time.to.real(time : t))$ *rem* 1023

! *The procedure truncate [ColM – 82a] is such that truncate(x) returns the*

! *integer i such that* $|i| \leq |x| < |i| + 1$ *and* $i * x \geq 0$.

2. $r := (29r + 217)$ *rem* 1024

3. $r := truncate(r/1023 + 0.5)$

4. *return* □

procedure bisect.or.delete.z($\underline{\hat{x}} \in I(R^n), \underline{z} \in I(R^{3n}), \underline{Q} \in I(R^{3n-l}), \varepsilon_0, \varepsilon_1 \in R, \underline{z}^{(1)}$,

$\underline{z}^{(2)} \in I(R^{3n}), a \in R^{3n-l}, q \in N^{2n}, n, l, c, s \in N$,

$x.star.in.x, singular, z.is.split, automatic, not.less \in B$;

$S_1, S_2, S_3, S_5, S_6 \in P, \underline{\bar{j}} \in I(R), n_5, n_7, j \in N$:

$\underline{f} \in I(R), go.to.deletion.test.2, f.bar.at.mid.x \in B)$

! This procedure implements the ideas which are described in §6.12, §6.13, and §6.14.

! On entry, $\underline{\hat{x}}$ is the initial box, $\underline{z}$ is the current box, $\underline{Q}$ and a are as in §6.11, ε_0 and

! ε_1 are as in the procedure *solve*, $\underline{z}^{(1)} \neq \emptyset$ and $\underline{z}^{(2)} \neq \emptyset$ if $z.is.split = \underline{true}$ which is

! obtained from the procedure *deletion.test.2* in §6.7, q and l are as in the procedure

! *delete.or.keep.z.1*, *x.star.in.x* is as in §6.6, *singular* is either a result of the

! procedures *symmetric.operator.test* or *solve*, *z.is.split* is a result of the

! procedure *deletion.test.2* (§6.7), c, s, *automatic* is as in the procedure *bisect*

! (§6.12) and *not.less* is as in §6.11.

! The input-output parameters are as explained in previous sections.

! On return, all the parameters are as in the procedure *delete.or.keep.z.1* except

! *go.to.deletion.test.2* where *go.to.deletion.test.2* is as in the procedure *solve*.

1. *generate.random.number*(: r)
2. *f.bar.at.mid.x* := *false*
3. *go.to.deletion.test.2* := *false*
4. $\underline{f} := \underline{0}$
5. *if z.is.split*

   then

   ! $\underline{z}$ *is split into* $\underline{z}^{(1)}$ *and* $\underline{z}^{(2)}$ *as described in* §6.7.

5.1. $\underline{f}^{(1)} := \underline{f}(\underline{x}^{(1)})$! $\underline{z}^{(1)} = (\underline{x}^{(1)^T}, \underline{u}^{(1)^T})^T$

5.2. $\underline{f}^{(2)} := \underline{f}(\underline{x}^{(2)})$! $\underline{z}^{(2)} = (\underline{x}^{(2)^T}, \underline{u}^{(2)^T})^T$

5.3. *delete.or.keep.z.1*$(\hat{\underline{x}}, \underline{z}^{(1)}, \underline{z}^{(2)}, \underline{f}^{(1)}, \underline{f}^{(2)}, q, n, l,$ *x.star.in.x* ;

$S_1, S_2, S_3, S_6, \bar{\underline{f}}, n_7$: $\underline{f}$, *f.bar.at.mid.x*) ! §6.16

else

5.4. *bisect*$(\underline{z}, \underline{Q}, a, \varepsilon_0, q, l, r, n, c, s,$ *singular*, *x.star.in.x*, *automatic*, *not.less* ;

j : $\underline{z}^{(1)}, \underline{z}^{(2)}$, *bisect1*, *bisect2*, *bisect3*) ! §6.12.

5.5. *if* *bisect3* = *true*

then

! $\underline{z}$ *is too small to bisect.*

5.5.1. $\underline{f} := \underline{f}(\underline{x})$! $\underline{z} = (\underline{x}^T, \underline{u}^T)^T$.

5.5.2. *if* $f_I \leq \bar{f}_S$ *do*

! *If* $\bar{f}_S < f_I$ *then* $\underline{z}$ *can be deleted.*

5.5.2.1. *if* $f_S < \bar{f}_I$ *do*

5.5.2.1.1. $\bar{\underline{f}} := [f_S, f_S]$

5.5.2.1.2. $n_7 := 0$! $\underline{x} \not\longrightarrow S_7$.

5.5.2.2. *if* $0 < \varepsilon_1$! *See the comment in the procedure solve.*

then

5.5.2.2.1. *go.to.deletion.test.2* := *true*

5.5.2.2.2. *push*$(\hat{\underline{x}}, \underline{z}, q, l, n,$ *x.star.in.x* ;

S_1, S_2, S_3, S_6) ! §6.14.

else

5.5.2.2.3. $\underline{z} \longrightarrow S_5$! $\underline{z}$ *might contain a global minimizer.*

5.5.2.2.4. $n_5 := n_5 + 1$

else

5.5.3. *case true of*

bisect1 and $\sim$ *bisect2* :

5.5.3.1. $\underline{f} := \underline{f}(\underline{x}^{(1)})$! $\underline{z}^{(1)} = (\underline{x}^{(1)^T}, \underline{u}^{(1)^T})^T$

5.5.3.2. *delete.or.keep.z.2*$(\underline{\hat{x}}, \underline{z}^{(1)}, q, n, l, x.star.in.x$; $S_1, S_2, S_3, S_6,$

$\underline{\bar{f}}, \underline{f}, n_7$: *f.bar.at.mid.x*) ! §6.16.

bisect1 and bisect2 :

5.5.3.3. $\underline{f}^{(1)} := \underline{f}(\underline{x}^{(1)})$! $\underline{z}^{(1)} = (\underline{x}^{(1)^T}, \underline{u}^{(1)^T})^T$

5.5.3.4. $\underline{f}^{(2)} := \underline{f}(\underline{x}^{(2)})$! $\underline{z}^{(2)} = (\underline{x}^{(2)^T}, \underline{u}^{(2)^T})^T$

5.5.3.5. *delete.or.keep.z.1*$(\underline{\hat{x}}, \underline{z}^{(1)}, \underline{z}^{(2)}, \underline{f}^{(1)}, \underline{f}^{(2)}, q, n, l, x.star.in.x$;

$S_1, S_2, S_3, S_6, \underline{\bar{f}}, n_7$:

$\underline{f}, f.bar.at.mid.x$) ! §6.16.

default : ! *No such case.*

5.5.3.6. *write* "*No such case in bisect.or.delete.z.*"

5.5.3.7. *stop*

6. *return* □

procedure process.z.i$(\underline{\hat{x}} \in I(R^n), \hat{q} \in N^{2n}, m, n, p, maxit, i, c, s \in N, \varepsilon_0, \varepsilon_1,$

$\varepsilon_2 \in R, x.star.in.x, automatic \in B$; $S_1, S_2, S_3,$

$S_4, S_5, S_6, S_7 \in P, \underline{\bar{f}}, \underline{f} \in I(R), n_4, n_5, n_7, j \in N,$

$f.bar.at.mid.x, go.to.deletion.test.2 \in B$:

$next \in B$)

! This procedure combines all the techniques which are described in §6.4 - §6.15 in

! order to obtain the solution(s) of $F(z) = 0$ for $NP.2$ and hence the minimizer(s)

! of $f : R^n \to R^1$ for *Problem P*.

! All the parameters are as explained in the previous procedures.

1. $k := 0$

2. *termination*$(\varepsilon_1, n, i, n_4, n_5, f.bar.at.mid.x, S_4, S_5$;

$S_1, S_2, S_3, S_6, S_7, n_7, \underline{\bar{f}}, \underline{f}$:

$\underline{z}, \underline{F}, q, l, next)$! §6.15.

3. *if* $\sim next$ *do*

! *If go.to.deletion.test.2* = *true then the box which is popped from* S_1 *can be*

! *pushed onto* S_4 *or* S_5 *as described in* §6.11, *since if we wish only to bound*

! f^* *but not* x^* *then we must push that box onto* S_1 *for processing next.*

! *If go.to.deletion.test.2* = *true then we by – pass the procedures deletion.test.1*

! *and gradient.test because these procedures cannot delete any part of* $\underline{z}$.

3.1. *if* $\sim$ *go.to.deletion.test.2 do*

3.1.1. *deletion.test.1*($\underline{F}, n, l$: *delete.z*) ! §6.5.

3.1.2. *if* ~ *delete.z* *do*

3.1.2.1. *gradient.test*($\hat{\underline{x}}, n, l, q, x.star.in.x$; $S_7, \underline{z}, \bar{\underline{f}}, n_7$: k, *delete.z*) ! §6.6.

3.2. *if* ~ *delete.z* *do*

3.2.1. *deletion.test.2*($\bar{\underline{f}}, \underline{f}, \varepsilon_1, n, l, k, q, x.star.in.x$; $\underline{z}$: $\underline{J}, \underline{z}^{(1)}, \underline{z}^{(2)}$, *delete.z*, *z.is.split*) ! §6.7.

3.2.2. *not.less* := *false* ! §6.11.

3.2.3. *singular* := *false*

3.2.4. *push.or.delete.z* := *false*

3.2.5. $\underline{Q} := (\underline{0})_{3n-l\times 1}$

3.2.6. $a := (0)_{3n-l\times 1}$

! $\underline{Q}$ *is either* $\underline{K}_N$ (MAP) *or* $\underline{S}$ $(KMSW)$ *as explained in Chapter* 5.

3.2.7. *if* ~ *delete.z* *and* ~ *z.is.split* *do*

3.2.7.1. *symmetric.operator.test*($\hat{\underline{x}}, \underline{J}, \varepsilon_2, n, x.star.in.x$; $S_7, \underline{z}, \bar{\underline{f}}$, q, l, n_7 : $\underline{S}$, *bisect.z*, *solve.F.z*, *delete.z*, *singular*) ! §6.8.

3.2.7.2. $\underline{Q} := \underline{S}$

3.2.7.3. *if* ~ *delete.z* *and* ~ *bisect.z* *and* $l \geq 2n-1$ *do*

3.2.7.3.1. *n.c.test*($\underline{J}, \hat{\underline{x}}, q, n, l, x.star.in.x$; $\underline{z}$: *delete.z*) ! §6.9.

3.2.7.3.2. *if* ~ *delete.z* *do*

3.2.7.3.2.1. *if* ~ *solve.F.z* *do*

! §6.10.

3.2.7.3.2.1.1. *s.c.s.test*($\underline{z}, n$:

bisect.z)

3.2.7.3.2.2. *if* ~ *bisect.z* *do*

3.2.7.3.2.2.1. *solve*($\underline{\hat{x}}, \hat{q}, q, m, n, l,$

p, *maxit*, $\varepsilon_0, \varepsilon_1,$

x.star.in.x ;

$S_1, S_2, S_3, S_4,$

$S_5, S_6, \underline{z}, \underline{f},$

n_4, n_5, n_7 : $\underline{Q}$,

a, *not.less*,

push.or.

delete.z,

singular,

go.to.deletion.

test.2) ! §6.16.

3.2.8. *if* ~ *delete.z* *and* ~ *push.or.delete.z* *do*

3.2.8.1. *bisect.or.delete.z*($\underline{\hat{x}}, \underline{z}, \underline{Q}, \varepsilon_0, \varepsilon_1, \underline{z}^{(1)}, \underline{z}^{(2)}, a, q, n, l, c, s,$

x.star.in.x, *singular*, *z.is.split*, *automatic*,

$not.less$; $S_1, S_2, S_3, S_5, S_6, \underline{f}, n_5, w_7, j$: $\underline{f}$

$go.to.deletion.test.2$, $f.bar.zt.mid.x$) ! §6.16.

4. *return* □

Using the preceding subsidiary procedures, the algorithm MW may be described as follows.

Algorithm MW

Data : $S_i = \emptyset$ $(i = 1, \ldots, 7) \in P, n, m, p, c, s, maxit \in N, \varepsilon_0, \varepsilon_1, \varepsilon_2 \in R$,

$automatic, x.star.in.x \in B, \underline{\hat{x}} \in I(R^n)$

! 7 stacks are used in MW as follows :

! S_1 contains the boxes to be processed.

! S_2 contains the integer vectors $q = (q_i)_{2n \times 1}$ where q_i is computed from 6.17.

! S_3 contains the integers l where l is the number of Lagrange multipliers which take

! the value zero as described in §6.2.

! S_4 contains the boxes which contain the solutions of $F(z) = 0$ or the boxes $\underline{z}$

! which satisfy $\|w(\underline{z})\| \leq \varepsilon_0$ where $\varepsilon_0 > 0$.

! S_5 contains the boxes which satisfy the condition $\underline{z}^{(k)} \not\to z^*$ $(k \leq maxit)$ as

! described in §6.11, where z^* is the solution of $F(z) = 0$ and $maxit$ is the

! maximum number of iterations.

! S_6 contains $\underline{F}$ where $\underline{F} = \underline{F}(\underline{z})$ for the box $\underline{z}$ which is pushed onto the stack S_1.

! S_7 contains the point boxes $\underline{x}$ such that $\bar{f} = \underline{f}(\underline{x})$ where $\bar{f}_S$ is an upper bound on

! the minimum value of f over $\hat{\underline{x}}$.

! n is the number of variables of $f : R^n \to R^1$.

! $m \in \{1, 2\}$ where if $m = 1$ then MAP is used and if $m = 2$ then $KMSW$ is used.

! $p \in \{-2, -1, 0, \ldots, \infty\}$ where if $p = -2$ then strategy 2 (Chapter 5) is used, if

! $p = -1$ then strategy 1 (Chapter 5) is used and if $p \geq 0$ then p is the given fixed

! number of times that $\underline{F}'$ and $\{m(\underline{F}')\}^{-1}$ are reused as described in §5.3.

! $c, s \in \{0, 1, 2, 3\}$ where $c = 1, 2, 3$ correspond to the BRi $(i = 1, 2, 3)$ (§6.12)

! respectively and $s = 1, 2, 3$ correspond to the SRi $(i = 1, 2, 3)$ (§6.12)

! respectively. If $c = 0$ and $s = 0$ then the procedure *automatic.rule* is used.

! *maxit* is the maximum number of iterations.

! $\varepsilon_0 > 0$ such that the box $\underline{z}$ with $\|w(\underline{z})\| \leq \varepsilon_0$ is pushed onto S_4.

! $\varepsilon_1 \geq 0$ is used in §6.7 for bounding the global minimum value f^*.

! $\varepsilon_2 > 0$ is used with 6.47 which is decribed in the case (iv) of §6.8.

! The Boolean *automatic* is such that if *automatic* = *true* then the bisection and

! selection rules which are described in [Jon–78a] are used else BR2 and SR3

! (§6.12) are used.

! The Boolean *x.star.in.x* is such that if *x.star.in.x* = *true* then $x^* \in int(\hat{\underline{x}})$ else

! $x^* \in \underline{\hat{x}}$ where x^* is the global minimizer of $f : R^n \to R^1$, and $\underline{\hat{x}}$ is the initial

! box which is assumed to contain x^*.

1. $\underline{\hat{u}} := (\underline{0})_{2n\times 1}$! *Box containing the Lagrange Multipliers.*

2. $\hat{q} := (1)_{2n\times 1}$! *If* $\underline{\hat{u}}_i \neq \underline{0}$ *then* $\hat{q}_i = 1$ *else* $\hat{q}_i = 0$ $(i = 1, \ldots, 2n)$.

3. $\hat{l} := 0$! $\hat{l}$ *is the number of Lagrange multipliers which take the value zero.*

4. *if* $x.star.in.x$

   then

   ! $x^* \in int(\underline{\hat{x}})$. *Therefore only the first n components*

   ! *of* F *need to be considered.*

   4.1. $\hat{q} := (0)_{2n\times 1}$

   4.2. $\hat{l} := 2n$

   else

   ! $x^* \in \underline{\hat{x}}$. *Therefore we need to determine* $\underline{\hat{u}}$, $\hat{q}$ *and* $\hat{l}$

   ! *for the initial box* $\underline{\hat{x}}$ *as explained in* §6.2.

   4.3. $\underline{\hat{d}} := \underline{g}(\underline{\hat{x}})$! $\underline{g}(\underline{\hat{x}}) = \underline{f}'(\underline{\hat{x}})^T$.

   4.4. $lagrange.multipliers(\underline{\hat{d}}, n\ ;\ \underline{\hat{u}}, \hat{q}, \hat{l})$! §6.2.

5. $n_4 := 0$! *The number of boxes on* S_4.

6. $n_5 := 0$! *The number of boxes on* S_5.

7. $\bar{f} := \underline{f}(m(\underline{\hat{x}}))$! $\bar{f}$ *is such that* $f^* \leq \bar{f}_S$ (6.42).

8. $f.bar.at.mid.x :=$ *true* ! $\bar{f}$ *is computed at* $m(\underline{\hat{x}})$.

9. $m(\hat{\underline{x}}) \longrightarrow S_7$! *$m(\hat{\underline{x}})$ is pushed onto S_7.*

10. $n_7 := 1$! *n_7 is the number of point boxes on S_7.*

11. $j := 0$! *The co – ordinate direction for use in* §6.12.

12. *next* := *false*

13. *go.to.deletion.test.2* := *false*

! *This Boolean is used for bounding f^* but not x^*. See the comment in the*

! *procedure solve.*

14. *for* $i = 0$ *to* $2^n - 1$ *do*

! *Process the box $\hat{\underline{z}}^{(i)}$ where $i \in \{0, \ldots, 2^n - 1\}$. The box $\hat{\underline{z}}^{(i)}$ is determined*

! *in* §6.3. *If $\hat{\underline{z}}^{(i)}$ has been processed then $\hat{\underline{z}}^{(i+1)}$ is processed, so*

! *next* = *true* (*see* §6.15).

14.1. *construct.z.i*$(\hat{\underline{x}}, \hat{\underline{u}}, \hat{q}, n, i, x.star.in.x \ : \ \hat{\underline{z}}^{(i)}, \hat{q}^{(i)}, \hat{l}^{(i)})$! §6.3.

14.2. *for* $j = 1$ *to* n *do*

! *We assign the first n components of $\hat{\underline{z}}^{(i)}$ to $\underline{y}$ in order either to*

! *update $\overline{f}$ if possible or to delete $\underline{y}$ and hence $\hat{\underline{z}}^{(i)}$ if possible.*

14.2.1. $\underline{y}_j := \hat{\underline{z}}_j^{(i)}$

! *We wish to process $\hat{\underline{z}}^{(i)}$ $(0 \leq i \leq 2^n - 1)$ where $\hat{\underline{z}}^{(i)} = (\hat{\underline{x}}^{(i)T}, \hat{\underline{u}}^{(i)T})^T$.*

! *Initially we compute $\overline{f} = \underline{f}(m(\hat{\underline{x}}))$. If $i = 0$ then we update $\overline{f}$ at $m(\hat{\underline{x}}^{(0)})$*

! *because $m(\hat{\underline{x}}) \in \hat{\underline{x}}^{(0)}$. If $0 < i$ then the current $\overline{f}$ has been updated at*

! *several points during the processing of the sub – boxes $\hat{\underline{z}}^j$*

! *$(j = 0, \ldots, i - 1)$. Therefore the sub – box at which $\overline{f}$ is updated does not*

! *belong to* $\hat{\underline{z}}^{(i)}$. *Therefore if* $0 < i$ *then we can update* $\underline{\bar{f}}$ *by comparing it*

! *with* $\underline{f}(\hat{\underline{x}}^{(i)})$.

14.3. *if* $i = 0$

then

14.3.1. $\underline{f} := \underline{f}(m(\underline{y}))$! *f.bar.at.mid.x* = *true.*

14.3.2. *if* $f_S < \bar{f}_I$ *do*

14.3.2.1. $\underline{\bar{f}} := [f_S, f_S]$

14.3.2.2. $n_7 := 0$! $m(\underline{y}) \not\longrightarrow S_7$

else

14.3.3. *f.bar.at.mid.x* := *false*

14.3.4. $\underline{f} := \underline{f}(\underline{y})$! $\underline{f} = \underline{f}(\hat{\underline{x}}^{(i)})$.

14.3.5. *if* $f_I \leq \bar{f}_S$

then ! *update* $\underline{\bar{f}}$ *if possible.*

14.3.5.1. *if* $f_S < \bar{f}_I$ *do*

14.3.5.1.1. $\underline{\bar{f}} := [f_S, f_S]$

14.3.5.1.2. $n_7 := 0$! $\underline{y} \not\longrightarrow S_7$.

else ! $\hat{\underline{z}}^{(i)}$ *is deleted. Therefore the box* $\hat{\underline{z}}^{(i+1)}$ *is processed.*

14.3.5.2. *next* := *true*

14.4. *if* $\sim$ *next do*

14.4.1. $\hat{\underline{z}}^{(i)} \longrightarrow S_1$! $\hat{\underline{z}}^{(i)}$ *is pushed onto* S_1.

14.4.2. $\hat{q}^{(i)} \longrightarrow S_2$! $\hat{q}^{(i)}$ *is pushed onto* S_2.

14.4.3. $\hat{l}^{(i)} \longrightarrow\triangleright S_3$! $\hat{l}^{(i)}$ *is pushed onto* S_3.

14.4.4. $\underline{F} := \underline{F}(\hat{\underline{z}}^{(i)})$! *Appendix D.*

14.4.5. $\underline{F} \longrightarrow\triangleright S_6$! $\underline{F}(\hat{\underline{z}}^{(i)})$ *is pushed onto* S_6.

14.5. *while* $\sim$ *next or* $i = 2^n - 1$ *do*

! *The box* $\hat{\underline{z}}^{(2^n-1)}$ *is the last box to be processed by the algorithm*

! *MW. If* $i = 2^n - 1$ *then all the results which are kept on the*

! *stacks* S_4, S_5, *and* S_7 *are popped in the procedure*

! *termination* (*See* §6.15).

14.5.1. *process.z.i*($\hat{\underline{z}}, \hat{q}, m, n, p, maxit, i, c, s, \varepsilon_0, \varepsilon_1, \varepsilon_2,$

$x.star.in.x, automatic$; $S_1, S_2, S_3, S_4, S_5,$

$S_6, S_7, \underline{\bar{f}}, \underline{f}, n_4, n_5, n_7, j, f.bar.at.mid.x,$

go.to.deletion.test.2 : *next*) ! §6.16.

14.6. *next* := *false*

15. *stop* □

CHAPTER 7

Numerical Results

This chapter contains some numerical results which are obtained by using Hansen's algorithm (H) [Han–80a] which is described in Chapter 3 and the algorithm MW which is described in Chapter 6. The effectiveness of the both algorithms is exhibited using the examples which are given in Appendix C. For each example it is assumed that the global minimizers are to be found in the box $\underline{\hat{x}} \in I(R^n)$.

7.1 Computations and Tables

The algorithms H, and MW have been implemented in Triplex S-algol [BaCM-82a] [MorC-83a] on a VAX-11/785 computer.

In the algorithm H, if f^* the global minimum and x^* the global minimizer(s) are to be bounded within an error ε_0 then we choose $\varepsilon_0 = 10^{-6}$, $\varepsilon_1 = 0$, and $\varepsilon_2 = 10^{-6}$ [Han–80a], and if f^* is to be bounded but not x^* within an error ε_1 then we choose $\varepsilon_0 = 10^{-6}$, $\varepsilon_1 = 10^{-6}$, and $\varepsilon_2 = 0$ [Han–80a].

In the algorithm MW, if f^* and x^* are to be bounded within an error ε_0 then we choose $\varepsilon_0 = 10^{-6}$, $\varepsilon_1 = 0$, and $\varepsilon_2 = 10^{-2}$, and if f^* is to be bounded but not x^* within an error ε_1 then we choose $\varepsilon_0 = 10^{-6}$, $\varepsilon_1 = 10^{-6}$, and $\varepsilon_2 = 10^{-2}$.

In the algorithm H, $\varepsilon_2 = 10^{-6}$ is such that if $||w(\underline{x})|| > \varepsilon_2$ where $\underline{x}$ is a sub-box of $\hat{\underline{x}}$ then $\underline{x}$ is inserted into the queue L_2 of boxes to be processed. In contrast, in the algorithm MW, $\varepsilon_2 = 10^{-2}$ is used in the procedure *simplified.newton* (§6.8) only.

Since we do not know whether $x^* \in int(\hat{\underline{x}})$ or $x^* \in \partial(\hat{\underline{x}})$ we test both algorithms in the two cases (a) $x^* \in int(\hat{\underline{x}})$ and (b) $x^* \in \hat{\underline{x}}$ separately.

The results which are obtained by using the algorithm MW depend on the choice of methods for solving $F(z) = 0$ (§6.11), for determining the $p^{(k)}$ (§5.3), and on the bisection and selection rules (§6.12). With both MAP and $KMSW$ (§6.11), strategy 2 for determining $p^{(k)}$ (§5.3) and the bisection and selection rules corresponding to cases $(a) - (e)$ (§6.12) have produced the best results.

Table 7.6 of CPU times shows how Algorithm MAP compares with Algorithm $KMSW$ for 3 examples. These results indicate that MAP and $KMSW$ are almost equally effective.

The results which are obtained by applying H and MW to the examples listed in Appendix C, and which consist of the number of evaluations of various functions, the CPU times, the global minimizers and the global minima which correspond to the four cases (i) $x^* \in int(\hat{\underline{x}})$ and $\varepsilon_1 = 0$, (ii) $x^* \in \hat{\underline{x}}$ and $\varepsilon_1 = 0$, (iii) $x^* \in int(\hat{\underline{x}})$ and $\varepsilon_1 = 10^{-6}$, and (iv) $x^* \in \hat{\underline{x}}$ and $\varepsilon_1 = 10^{-6}$ are given in tables 7.1.1 - 7.4.3. For the algorithm H, n_f is the number of objective function evaluations, n_g is the number of gradient evaluations, n_{Hd} is the number of Hessian diagonal element evaluations, and n_J is the number of Jacobian evaluations. For the algorithm MW, n_f, n_g, and n_{Hd} have the same meanings. In addition, n_c is the number of constraint evaluations, n_F is the number of F evaluations, and $n_{F'}$ is the number of F' evaluations, where F and

F' are given in Chapter 4. Since the Hessian of the objective function f is computed from the Jacobian of f'^T and from F' for algorithms H and MW respectively the number of Hessian evaluations is not included in the tables. In all of the tables n_s is the total number of function evaluations.

If $x^* \in \partial(\underline{\hat{x}})$ then for the cases (i) and (iii) the algorithms H and MW might not be able to bound the global minimizers because boundary points of $\underline{\hat{x}}$, including x^*, might be deleted. Furthermore, if the global minimizers cannot be bounded then the global minimum bound cannot be obtained. Therefore in tables 7.1.3, 7.1.4, and 7.3.3 for example 14, for which $x^* \in \partial(\underline{\hat{x}})$, we write $x^* \in \partial(\underline{\hat{x}})$ to denote that the global minimizers cannot be bounded and the correct global minimum values cannot be obtained.

Tables 7.1.1 - 7.1.4, 7.2.1 - 7.2.4, 7.3.1 - 7.3.3, and 7.4.1 - 7.4.3 contain the results corresponding to the cases (i) $x^* \in int(\underline{\hat{x}})$ and $\varepsilon_1 = 0$, (ii) $x^* \in \underline{\hat{x}}$ and $\varepsilon_1 = 0$, (iii) $x^* \in int(\underline{\hat{x}})$ and $\varepsilon_1 = 10^{-6}$, and (iv) $x^* \in \underline{\hat{x}}$ and $\varepsilon_1 = 10^{-6}$ respectively.

For both algorithms, n_f, n_g, n_J, n_{Hd}, n_c, n_F and $n_{F'}$ are given in tables 7.1.1, 7.2.1, 7.3.1, and 7.4.1; the CPU times are given in tables 7.1.2, 7.2.2, 7.3.2, and 7.4.2; the global minimizer bounds are given in tables 7.1.3, and 7.2.3 only, because for the cases (iii) and (iv) f^* is to be bounded but not x^*. The global minimum bounds are given in tables 7.1.4, 7.2.4, 7.3.3, and 7.4.3. The abbreviation Ex. and Alg. which are appear in the tables are abbreviations of Example and Algorithm respectively. In tables 7.1.1, 7.2.1, 7.3.1, and 7.4.1, $n_J/n_{F'}$ means that n_J and $n_{F'}$ correspond to H and MW respectively, and n_s is the total number of function evaluations for H and MW. The notations $*$ and $**$ mean that the computations need more than 600

and 3600 seconds respectively of CPU time. If $n = 1$ then the code for algorithm H cannot be used and this is indicated by the character '#' in the tables.

We have applied the algorithm MW to the whole box $\hat{\underline{z}}$ without constructing the sub-boxes $\hat{\underline{z}}^{(i)}$ $(i = 0, \ldots, 2^n - 1)$ from $\hat{\underline{z}}$. Numerical results are shown in tables 7.7 and 7.8. In the second columns of tables 7.7 and 7.8, *"MW with"* means that the algorithm MW has been tested with sub-boxes $\hat{\underline{z}}^{(i)}$ $(i = 0, \ldots, 2^n - 1)$ and with $\hat{\underline{z}}$, the whole box, separately. These results indicate that the method in which the sub-boxes $\hat{\underline{z}}^{(i)}$ $(i = 0, \ldots, 2^n - 1)$ are constructed from $\hat{\underline{z}}$ is more efficient than the method in which the sub-boxes $\hat{\underline{z}}^{(i)}$ $(i = 0, \ldots, 2^n - 1)$ are not constructed.

7.2 Discussion

The results which are obtained from the algorithms H and MW are given in tables 7.1.1 - 7.4.3 for the cases (i) $x^* \in int(\hat{\underline{x}})$ and $\varepsilon_1 = 0$, (ii) $x^* \in \hat{\underline{x}}$ and $\varepsilon_1 = 0$, (iii) $x^* \in int(\hat{\underline{x}})$ and $\varepsilon_1 = 10^{-6}$, and (iv) $x^* \in \hat{\underline{x}}$ and $\varepsilon_1 = 10^{-6}$.

(i) $x^* \in int(\hat{\underline{x}})$ and $\varepsilon_1 = 0$:

From Table 7.1.1, the number of evaluations n_s (n_f, n_g, n_{Hd}, n_c, n_F, n_J, $n_{F'}$) which are needed by MW is less than that needed by H.

For examples 4 - 16 the CPU times (Table 7.1.2) which are needed by the algorithm MW are less than those needed by the algorithm H especially for examples 5, 9, 12, 13, 14, 15 and 16. For examples 6, 7, and 11, however, H requires less CPU time than does MW.

The global minimizers and the global minima for both algorithms which are bounded within an error 10^{-6} are given in tables 7.1.3 and 7.1.4 respectively correct to 16 decimal digits.

(ii) $x^* \in \hat{\underline{x}}$ and $\varepsilon_1 = 0$:

From Table 7.2.1 the number of evaluations n_s $(n_f, n_g, n_{Hd}, n_c, n_F, n_J, n_{F'})$ which are needed by MW is less than that needed by H.

From Table 7.2.2, we observe that for examples 4 - 16, the CPU times which are needed by the algorithm MW are less than those needed by the algorithm H, save for example 10.

The global minimizers and the global minima for both algorithms which are bounded within an error 10^{-6} and which are recorded to 16 decimal digits are given in tables 7.2.3 and 7.2.4 respectively.

(iii) $x^* \in int(\hat{\underline{x}})$ and $\varepsilon_1 = 10^{-6}$:

From Table 7.3.1 the number of evaluations n_s $(n_f, n_g, n_{Hd}, n_c, n_F, n_J, n_{F'})$ which are needed by MW is less than that needed by H.

From Table 7.3.2, the CPU times which are needed by MW for examples 4 - 16 are less than those needed by H save for 6, 7, and 11.

For this case the global minima only are be bounded. See Table 7.3.3.

(iv) $x^* \in \hat{\underline{x}}$ and $\varepsilon_1 = 10^{-6}$:

The number of evaluations n_s (n_f, n_g, n_{Hd}, n_c, n_F, n_J, $n_{F'}$) for both algorithms are given in Table 7.4.1 where it can be seen that the algorithm MW needs fewer evaluations than does H.

From Table 7.4.2, it is seen that the CPU times which are required by MW to solve problems 4 - 16 are less than those required by H save for examples 6 and 10.

Table 7.4.3 contains the global minima for both algorithms.

For Example 17, $x^* \in \partial(\hat{\underline{x}})$ and strict complementary slackness does not hold at x^*. The results for this example are contained in tables 7.9.1 - 7.9.5. The results in tables 7.9.1 - 7.9.4 indicate that MW tends to be increasing less computationally expensive than H as n increases. The results in Table 7.9.5 indicate that it is better to process the sub-boxes $\hat{\underline{z}}^{(i)}$ ($i = 0, \ldots, 2^n - 1$) than to process the whole box $\hat{\underline{z}}$. The results in tables 7.9.1 - 7.9.5 also suggest that as n increases the proportion of CPU time which ie required for function evaluation in the algorithm H increases to a greater extent than it does in the algorithm MW. This might account for the increasing superiority of MW over H as n increases, which is frequently observed, and is illustrated by Example 17. This might also account for the often-observed superiority of MW over H when f and its derivatives are expensive to evaluate.

7.3 Conclusion

From the numerical results which have been presented in Chapter 7 it is clear that, with few exceptions, MW requires fewer evaluations of function, gradient, Hessian diagonal elements, constraints, F and F' where F and F' are given in Chapter 4, and less CPU time than does H, at least for the implementations which have been used.

Ex.	Alg.	n_f	n_g	n_{Hd}	n_c	n_F	$n_J/n_{F'}$	n_s
1	H	#	#	#	-	-	#	#
	MW	28	11	0	2	18	12	71
2	H	#	#	#	-	-	#	#
	MW	12	4	0	1	7	5	29
3	H	#	#	#	-	-	#	#
	MW	102	38	0	1	46	36	223
4	H	128	198	84	-	-	42	452
	MW	80	31	5	5	71	36	228
5	H	1353	2121	900	-	-	450	4824
	MW	348	138	14	13	299	150	962
6	H	80	120	51	-	-	25	276
	MW	70	28	5	5	50	32	190
7	H	36	56	22	-	-	11	125
	MW	30	12	9	8	31	19	109
8	H	215	331	146	-	-	73	765
	MW	139	58	12	11	123	67	410
9	H	190	290	114	-	-	57	651
	MW	134	48	8	6	104	54	354
10	H	171	257	98	-	-	49	575
	MW	110	41	8	8	93	50	310
11	H	41	61	39	-	-	13	154
	MW	44	15	20	10	39	22	150
12	H	1553	2386	1383	-	-	461	5783
	MW	704	253	70	34	590	286	1937
13	H	478	738	616	-	-	154	1986
	MW	115	40	48	16	85	41	345
14	H	539	789	684	-	-	171	2183
	MW	121	37	6	2	62	36	264
15	H	321	493	600	-	-	100	1514
	MW	106	16	70	14	36	18	260
16	H	**	**	**	-	-	**	**
	MW	369	46	217	31	103	48	814

$x^* \in int(\hat{\underline{x}})$ *and* $\varepsilon_1 = 0$: The number of evaluations of various functions

Table 7.1.1

Example	H	MW
1	#	18.2
2	#	17.2
3	#	365.0
4	54.3	46.2
5	584.0	204.0
6	31.5	32.6
7	13.1	15.9
8	130.0	99.7
9	821.0	459.0
10	64.8	52.9
11	114.0	133.0
12	2170.0	1020.0
13	573.0	157.0
14	381.0	72.3
15	699.0	135.0
16	**	640.0

$x^* \in int(\hat{\underline{x}})$ *and* $\epsilon_1 = 0$: CPU times in seconds

Table 7.1.2

Ex.	Alg.	$\underline{x}^*$
1	H	#
	MW	$[-0.1000000000000001, 0.1000000000000001] \times 10^{-15}$
2	H	#
	MW	$[-4.493409457991092, -4.493409457827031]$
3	H	#
	MW	$[-6.774576200365678, -6.774576093965520]$
	MW	$[-0.4913908362593289, -0.4913908362593003]$
	MW	$[5.791794464674531, 5.791794478216515]$
4	H	$\begin{pmatrix} [0.9999996425698224, 1.000000362277280] \\ [0.9999998833269862, 1.000000118492277] \end{pmatrix}$
	MW	$\begin{pmatrix} [0.9999999999848898, 1.000000000015111] \\ [0.9999999999663685, 1.000000000033632] \end{pmatrix}$
5	H	$\begin{pmatrix} [0.9999999916587334, 1.000000008604749] \\ [0.9999999937156958, 1.000000006597782] \end{pmatrix}$
	MW	$\begin{pmatrix} [0.9999999302856068, 1.000000069714394] \\ [0.9999998535085455, 1.000000146491455] \end{pmatrix}$
6	H	$\begin{pmatrix} [-0.1 \times 10^{-5}, 0.1 \times 10^{-5}] \\ [-0.1 \times 10^{-5}, 0.1 \times 10^{-5}] \end{pmatrix}$
	MW	$\begin{pmatrix} [-0.1 \times 10^{-5}, 0.1 \times 10^{-5}] \\ [-0.1 \times 10^{-5}, 0.1 \times 10^{-5}] \end{pmatrix}$
7	H	$\begin{pmatrix} [0.6958843861143607, 0.6958843861194298] \\ [-1.347942193059716, -1.347942193057180] \end{pmatrix}$
	MW	$\begin{pmatrix} [0.6958843854741107, 0.6958843866076324] \\ [-1.347942193303817, -1.347942192736555] \end{pmatrix}$

$x^* \in int(\hat{\underline{x}})$ *and* $\varepsilon_1 = 0$: The global minimizers

Table 7.1.3

Ex.	Alg.	$\underline{x}^*$
8	H	$\begin{pmatrix} [-0.2152506901732498, -0.2152501840448925] \\ [0.2152504206610418, 0.2152504538768566] \end{pmatrix}$
	H	$\begin{pmatrix} [1.548583764242279, 1.548583776466430] \\ [-1.548583770761696, -1.548583769942562] \end{pmatrix}$
	MW	$\begin{pmatrix} [-0.2152504749635659, -0.2152503990794942] \\ [0.2152504076267709, 0.2152504664162894] \end{pmatrix}$
	MW	$\begin{pmatrix} [1.548583770354756, 1.548583770354971] \\ [-1.548583770355123, -1.548583770354604] \end{pmatrix}$
9	H	$\begin{pmatrix} [-0.1554372358601676, -0.1554372358589545] \\ [0.6945637753028883, 0.6945637753029206] \end{pmatrix}$
	H	$\begin{pmatrix} [0.1554372142111183, 0.1554372575241228] \\ [-0.6945637770584706, -0.6945637735231850] \end{pmatrix}$
	MW	$\begin{pmatrix} [-0.1554373160732023, -0.1554371556427049] \\ [0.6945635663239460, 0.6945639842858986] \end{pmatrix}$
	MW	$\begin{pmatrix} [0.1554372358595596, 0.1554372358595625] \\ [-0.6945637753029066, -0.6945637753029024] \end{pmatrix}$
10	H	$\begin{pmatrix} [-0.08984201310035930, -0.08984201310028215] \\ [0.7126564030207366, 0.7126564030207429] \end{pmatrix}$
	H	$\begin{pmatrix} [0.08984201310029249, 0.08984201310034506] \\ [-0.7126564030207431, -0.7126564030207359] \end{pmatrix}$
	MW	$\begin{pmatrix} [-0.08984205017177143, -0.08984197868282227] \\ [0.7126561261392704, 0.7126567010366590] \end{pmatrix}$
	MW	$\begin{pmatrix} [0.08984200752332696, 0.08984201761298961] \\ [-0.7126564590207734, -0.7126563385572218] \end{pmatrix}$

$x^* \in int(\hat{\underline{x}})$ *and* $\varepsilon_1 = 0$: The global minimizers

Table 7.1.3 contd.

Ex.	Alg.	$\underline{x}^*$
11	H	$\begin{pmatrix} [0.9999999997850837, 1.000000000141812] \\ [-0.1 \times 10^{-9}, 0.3377 \times 10^{-8}] \\ [-0.4179 \times 10^{-8}, 0.4276 \times 10^{-8}] \end{pmatrix}$
	MW	$\begin{pmatrix} [0.9999999999999990, 1.000000000000002] \\ [-0.1 \times 10^{-9}, 0.1 \times 10^{-9}] \\ [-0.1 \times 10^{-9}, 0.1 \times 10^{-9}] \end{pmatrix}$
12	H	$\begin{pmatrix} [-0.1833 \times 10^{-9}, 0.1835 \times 10^{-9}] \\ [-0.1895 \times 10^{-9}, 0.1874 \times 10^{-9}] \\ [0.9999999999984805, 1.000000000001518] \end{pmatrix}$
	MW	$\begin{pmatrix} [-0.6709 \times 10^{-9}, 0.6896 \times 10^{-9}] \\ [-0.6996 \times 10^{-9}, 0.6711 \times 10^{-9}] \\ [0.9999999998600570, 1.000000000143692] \end{pmatrix}$
13	H	$\begin{pmatrix} [0.9999999992855064, 1.000000000716914] \\ [0.9999999993278663, 1.000000000673811] \\ [0.9999999998831391, 1.000000000115989] \\ [0.9999999998567569, 1.000000000141365] \end{pmatrix}$
	MW	$\begin{pmatrix} [0.9999999057464136, 1.000000094186385] \\ [0.9999997880316363, 1.000000211833639] \\ [0.9999998558124518, 1.000000144258459] \\ [0.9999997073029567, 1.000000292837894] \end{pmatrix}$
14	H	$x^* \in \partial(\underline{\hat{x}})$
	MW	$x^* \in \partial(\underline{\hat{x}})$

$x^* \in int(\underline{\hat{x}})$ *and* $\varepsilon_1 = 0$: The global minimizers

Table 7.1.3 contd.

Ex.	Alg.	$\underline{x}^*$
15	H	$\begin{pmatrix} [0.9999999995742102, 1.000000000425778] \\ [0.9999999997333121, 1.000000000266735] \\ [0.9999999999122131, 1.000000000087809] \\ [0.9999999999800504, 1.000000000019960] \\ [0.9999999999954827, 1.000000000004525] \\ [0.9999999999973683, 1.000000000002639] \end{pmatrix}$
	MW	$\begin{pmatrix} [0.9999999999072405, 1.000000000027338] \\ [0.9999999998656809, 1.000000000066065] \\ [0.9999999998561317, 1.000000000087643] \\ [0.9999999998699426, 1.000000000100879] \\ [0.9999999998987722, 1.000000000103273] \\ [0.9999999998974739, 1.000000000104571] \end{pmatrix}$
16	H	**
	MW	$\begin{pmatrix} [0.9999999963177311, 1.000000003989016] \\ [0.9999999901753535, 1.000000010591764] \\ [0.9999999769773503, 1.000000024636345] \\ [0.9999999461650608, 1.000000057115426] \\ [0.9999998795131941, 1.000000127137331] \\ [0.9999997685832237, 1.000000245045555] \\ [0.9999997130987849, 1.000000319080082] \\ [0.9999996899929608, 1.000000334485296] \end{pmatrix}$

$x^* \in int(\underline{\hat{x}})$ *and* $\varepsilon_1 = 0$: The global minimizers

Table 7.1.3 contd.

Ex.	Alg.	f^*
1	H	#
	MW	$[-100.0000000000001, -99.99999999999998]$
2	H	#
	MW	$[-0.2172336282191532, -0.2172336282032902]$
3	H	#
	MW	$[-12.03125091859858, -12.03124944216674]$
4	H	$[0.0, 0.2138 \times 10^{-16}]$
	MW	$[0.0, 0.4305 \times 10^{-20}]$
5	H	$[0.0, 0.1157 \times 10^{-17}]$
	MW	$[0.0, 0.8785 \times 10^{-27}]$
6	H	$[-0.1 \times 10^{-11}, 0.3 \times 10^{-11}]$
	MW	$[-0.1 \times 10^{-11}, 0.4 \times 10^{-11}]$
7	H	$[-0.5824451744522318, -0.5824451744436349]$
	MW	$[-0.5824451763663049, -0.5824451743368871]$
8	H	$[0.0, 0.7939 \times 10^{-22}]$
	MW	$[0.0, 0.9808 \times 10^{-23}]$
9	H	$[0.7731990285939301, 0.7731990564929243]$
	MW	$[0.7731985047798560, 0.7731990564929242]$
10	H	$[-1.031628453489955, -1.031628453489877]$
	MW	$[-1.031630504890371, -1.031628027141163]$

$x^* \in int(\hat{\underline{x}})$ *and* $\varepsilon_1 = 0$: The global minima

Table 7.1.4

Ex.	Alg.	f^*
11	H	$[0.0, 0.6803 \times 10^{-15}]$
	MW	$[0.0, 0.6726 \times 10^{-17}]$
12	H	$[0.0, 0.2753 \times 10^{-23}]$
	MW	$[0.0, 0.1038 \times 10^{-27}]$
13	H	$[-0.1911 \times 10^{-17}, 0.2525 \times 10^{-21}]$
	MW	$[-0.1229 \times 10^{-11}, 0.4901 \times 10^{-10}]$
14	H	$x^* \in \partial(\underline{\hat{x}})$
	MW	$x^* \in \partial(\underline{\hat{x}})$
15	H	$[0.0, 0.2929 \times 10^{-26}]$
	MW	$[0.0, 0.5884 \times 10^{-18}]$
16	H	**
	MW	$[0.0, 0.1927 \times 10^{-11}]$

$x^* \in int(\underline{\hat{x}})$ *and* $\epsilon_1 = 0$: The global minima

Table 7.1.4 contd.

Ex.	Alg.	n_f	n_g	n_{Hd}	n_c	n_F	$n_J/n_{F'}$	n_s
1	*H*	#	#	#	-	-	#	#
	MW	26	15	0	4	24	16	85
2	*H*	#	#	#	-	-	#	#
	MW	12	5	0	1	7	5	30
3	*H*	#	#	#	-	-	#	#
	MW	102	46	0	1	46	36	231
4	*H*	129	199	68	-	-	42	438
	MW	78	35	11	11	70	37	242
5	*H*	1353	2121	880	-	-	450	4804
	MW	416	179	23	23	384	193	1218
6	*H*	*	*	*	-	-	*	*
	MW	91	49	8	7	69	41	265
7	*H*	53	82	26	-	-	16	177
	MW	40	17	8	7	30	15	117
8	*H*	218	337	132	-	-	74	761
	MW	138	73	17	16	121	67	432
9	*H*	192	293	94	-	-	58	637
	MW	137	69	18	16	109	56	405
10	*H*	181	270	94	-	-	52	597
	MW	125	67	20	20	110	60	402
11	*H*	175	267	120	-	-	54	616
	MW	60	31	24	12	53	30	210
12	*H*	1684	2565	1437	-	-	502	6188
	MW	793	331	134	66	649	311	2284
13	*H*	**	**	**	-	-	**	**
	MW	332	226	84	27	269	127	1063
14	*H*	150	211	0	-	-	55	416
	MW	32	12	0	0	11	5	60
15	*H*	**	**	**	-	-	**	**
	MW	283	141	10	1	175	74	684
16	*H*	**	**	**	-	-	**	**
	MW	676	262	21	2	335	133	1429

$x^* \in \underline{\hat{x}}$ *and* $\varepsilon_1 = 0$: The number of evaluations of various functions

Table 7.2.1

Example	H	MW
1	#	22.6
2	#	17.3
3	#	375.0
4	54.4	47.9
5	584.0	261.0
6	*	47.9
7	19.6	15.2
8	133.0	106.0
9	811.0	524.0
10	67.2	72.4
11	519.0	196.0
12	2340.0	1180.0
13	**	646.0
14	94.9	26.9
15	**	909.0
16	**	2880.0

$x^* \in \underline{\hat{x}}$ *and* $\varepsilon_1 = 0$: CPU times in seconds

Table 7.2.2

Ex.	Alg.	$\underline{x}^*$
1	H	#
	MW	$[-0.1000000000000001 \times 10^{-15}, 0.1000000000000001 \times 10^{-15}$
2	H	#
	MW	$[-4.493409457991092, -4.493409457827031]$
3	H	#
	MW	$[-6.774576200365678, -6.774576093965520]$
	MW	$[-0.4913908362593289, -0.4913908362593003]$
	MW	$[5.791794464674531, 5.791794478246515]$
4	H	$\begin{pmatrix} [0.9999996425698224, 1.000000362277280] \\ [0.9999998833269862, 1.000000118492277] \end{pmatrix}$
	MW	$\begin{pmatrix} [0.9999999997998508, 1.000000000200195] \\ [0.9999999995762231, 1.000000000423770] \end{pmatrix}$
5	H	$\begin{pmatrix} [0.9999999916587334, 1.000000008604749] \\ [0.9999999937156958, 1.000000006597782] \end{pmatrix}$
	MW	$\begin{pmatrix} [0.9999999834021490, 1.000000016537236] \\ [0.9999999652607698, 1.000000034617997] \end{pmatrix}$
6	H	*
	MW	$\begin{pmatrix} [-0.1 \times 10^{-5}, 0.1 \times 10^{-5}] \\ [-0.1 \times 10^{-5}, 0.1 \times 10^{-5}] \end{pmatrix}$
7	H	$\begin{pmatrix} [0.6958843861176124, 0.6958843861183095] \\ [-1.347942193059155, -1.347942193058806] \end{pmatrix}$
	MW	$\begin{pmatrix} [0.6958843860079745, 0.6958843862612586] \\ [-1.347942193130630, -1.347942193003487] \end{pmatrix}$

$x^* \in \hat{\underline{x}}$ *and* $\varepsilon_1 = 0$: The global minimizers

Table 7.2.3

Ex.	Alg.	$\underline{x}^*$
8	H	$\begin{pmatrix} [-0.2152506901732498, -0.2152501840448925] \\ [0.2152504206610418, 0.2152504538768566] \end{pmatrix}$
	H	$\begin{pmatrix} [1.548583764242279, 1.548583776466430] \\ [-1.548583770761696, -1.548583769942562] \end{pmatrix}$
	MW	$\begin{pmatrix} [-0.2152504370215312, -0.2152504370215291] \\ [0.2152504370215293, 0.2152504370215311] \end{pmatrix}$
	MW	$\begin{pmatrix} [1.548583770354670, 1.548583770355057] \\ [-1.548583770355345, -1.548583770354383] \end{pmatrix}$
9	H	$\begin{pmatrix} [-0.1554372358601683, -0.1554372358589538] \\ [0.6945637753028883, 0.6945637753029206] \end{pmatrix}$
	H	$\begin{pmatrix} [0.1554372142111180, 0.1554372575241232] \\ [-0.6945637770584702, -0.6945637735231855] \end{pmatrix}$
	MW	$\begin{pmatrix} [-0.1554373133591897, -0.1554371583571956] \\ [0.6945635724957992, 0.6945639781134846] \end{pmatrix}$
	MW	$\begin{pmatrix} [0.1554372358595429, 0.1554372358595792] \\ [-0.6945637753029317, -0.6945637753028773] \end{pmatrix}$
10	H	$\begin{pmatrix} [-0.08984201310033582, -0.08984201310030332] \\ [0.7126564030207352, 0.7126564030207445] \end{pmatrix}$
	H	$\begin{pmatrix} [0.08984201310029255, 0.08984201310034499] \\ [-0.7126564030207431, -0.7126564030207359] \end{pmatrix}$
	MW	$\begin{pmatrix} [-0.08984205017087017, -0.08984197868364899] \\ [0.7126561261459840, 0.7126567010293519] \end{pmatrix}$
	MW	$\begin{pmatrix} [0.08984201306827619, 0.08984201312718163] \\ [-0.7126564032579741, -0.7126564027423868] \end{pmatrix}$

$x^* \in \hat{\underline{x}}$ *and* $\varepsilon_1 = 0$: The global minimizers

Table 7.2.3 contd.

Ex.	Alg.	$\underline{x}^*$
11	H	$\begin{pmatrix} [0.9999999999933536, 1.000000000004419] \\ [-0.1 \times 10^{-9}, 0.1 \times 10^{-9}] \\ [-0.1 \times 10^{-9}, 0.1 \times 10^{-9}] \end{pmatrix}$
	MW	$\begin{pmatrix} [0.9999999999999997, 1.000000000000001] \\ [-0.1 \times 10^{-9}, 0.1 \times 10^{-9}] \\ [-0.1 \times 10^{-9}, 0.1 \times 10^{-9}] \end{pmatrix}$
12	H	$\begin{pmatrix} [-0.1833 \times 10^{-9}, 0.1835 \times 10^{-9}] \\ [-0.1895 \times 10^{-9}, 0.1874 \times 10^{-9}] \\ [0.9999999999984805, 1.000000000001518] \end{pmatrix}$
	MW	$\begin{pmatrix} [-0.6709 \times 10^{-9}, 0.6896 \times 10^{-9}] \\ [-0.6996 \times 10^{-9}, 0.6711 \times 10^{-9}] \\ [0.9999999998600566, 1.000000000143693] \end{pmatrix}$
13	H	**
	MW	$\begin{pmatrix} [0.9999999705336638, 1.000000029466344] \\ [0.9999999324878824, 1.000000067512133] \\ [0.9999999512835931, 1.000000048716400] \\ [0.9999999009056188, 1.000000099094367] \end{pmatrix}$
14	H	$\begin{pmatrix} [2, 2] \\ [2, 2] \\ [2, 2] \\ [-1, -1] \end{pmatrix}$
	MW	$\begin{pmatrix} [2, 2.000000000000001] \\ [2, 2.000000000000001] \\ [2, 2.000000000000001] \\ [-1.000000000000001, -1] \end{pmatrix}$

$x^* \in \underline{\hat{x}}$ *and* $\varepsilon_1 = 0$: The global minimizers

Table 7.2.3 contd.

Ex.	Alg.	$\underline{x}^*$
15	H	**
	MW	$\begin{pmatrix} [0.9999999988362505, 1.000000001175125] \\ [0.9999999971237866, 1.000000002904477] \\ [0.9999999941970247, 1.000000005860447] \\ [0.9999999893595722, 1.000000010747300] \\ [0.9999999829497547, 1.000000017219284] \\ [0.9999999811318012, 1.000000018984584] \end{pmatrix}$
16	H	**
	MW	$\begin{pmatrix} [0.9999999999217082, 1.000000000120453] \\ [9.9999999907926775, 1.000000000217338] \\ [0.9999999995604018, 1.000000000372349] \\ [0.9999999990560754, 1.000000000868659] \\ [0.9999999975712629, 1.000000002377980] \\ [0.9999999931778598, 1.000000006804451] \\ [0.9999999811346974, 1.000000018865302] \\ [0.9999999789624369, 1.000000021037511] \end{pmatrix}$

$x^* \in \underline{\hat{x}}$ *and* $\varepsilon_1 = 0$: The global minimizers

Table 7.2.3 contd.

Ex.	Alg.	f^*
1	H	#
	MW	$[-100.0000000000001, -99.99999999999998]$
2	H	#
	MW	$[-0.2172336282191532, -0.2172336282032902]$
3	H	#
	MW	$[-12.03125091859858, -12.03124944216674]$
4	H	$[0.0, 0.2138 \times 10^{-16}]$
	MW	$[0.0, 0.7193 \times 10^{-13}]$
5	H	$[0.0, 0.1157 \times 10^{-17}]$
	MW	$[0.0, 0.4601 \times 10^{-12}]$
6	H	*
	MW	$[-0.1 \times 10^{-11}, 0.4 \times 10^{-11}]$
7	H	$[-0.5824451744448174, -0.5824451744436349]$
	MW	$[-0.5824451748735235, -0.5824451740137467]$
8	H	$[0.0, 0.7939 \times 10^{-22}]$
	MW	$[0.0, 0.3310 \times 10^{-22}]$
9	H	$[0.7731990285939308, 0.7731990564929243]$
	MW	$[0.7731985214566105, 0.7731990564929242]$
10	H	$[-1.031628453489940, -1.031628453489877]$
	MW	$[-1.031630504840352, -1.031628451653841]$

$x^* \in \hat{\underline{x}}$ *and* $\varepsilon_1 = 0$: The global minima

Table 7.2.4

Ex.	Alg.	$\underline{f}^*$
11	H	$[0.0, 0.2543 \times 10^{-17}]$
	MW	$[0.0, 0.6726 \times 10^{-17}]$
12	H	$[0.0, 0.2753 \times 10^{-23}]$
	MW	$[0.0, 0.1037 \times 10^{-27}]$
13	H	**
	MW	$[-0.1325 \times 10^{-12}, 0.1302 \times 10^{-20}]$
14	H	$[-24.00000000000002, -23.99999999999999]$
	MW	$[-24.00000000000001, -24.0]$
15	H	**
	MW	$[0.0, 0.5351 \times 10^{-14}]$
16	H	**
	MW	$[0.0, 0.5111 \times 10^{-14}]$

$x^* \in \underline{\hat{x}}$ *and* $\varepsilon_1 = 0$: The global minima

Table 7.2.4 contd.

Ex.	Alg.	n_f	n_g	n_{Hd}	n_c	n_F	$n_J/n_{F'}$	n_s
1	H	#	#	#	-	-	#	#
	MW	20	9	0	2	14	10	55
2	H	#	#	#	-	-	#	#
	MW	12	5	0	1	8	6	32
3	H	#	#	#	-	-	#	#
	MW	102	42	0	1	49	40	234
4	H	125	195	84	-	-	42	446
	MW	68	25	6	6	58	31	194
5	H	1350	2118	900	-	-	450	4818
	MW	411	171	18	18	376	188	1182
6	H	80	122	53	-	-	26	281
	MW	74	30	5	5	53	34	201
7	H	33	53	22	-	-	11	119
	MW	30	13	9	8	32	20	112
8	H	209	325	146	-	-	73	753
	MW	120	54	11	11	115	65	376
9	H	181	279	112	-	-	56	628
	MW	138	51	8	6	109	57	369
10	H	165	251	98	-	-	49	563
	MW	113	47	8	8	99	56	331
11	H	35	55	39	-	-	13	142
	MW	36	15	16	- 8	34	20	129
12	H	1547	2378	1380	-	-	460	5765
	MW	730	266	78	39	625	304	2042
13	H	475	735	616	-	-	154	1980
	MW	115	41	48	16	86	42	348
14	H	539	789	684	-	-	171	2183
	MW	121	37	6	2	62	36	264
15	H	318	490	600	-	-	100	1508
	MW	106	17	70	14	37	19	263
16	H	**	**	**	-	-	**	**
	MW	369	47	217	31	104	49	817

$x^* \in int(\hat{x})$ *and* $\varepsilon_1 = 10^{-6}$: The number of evaluations of various functions

Table 7.3.1

Example	H	MW
1	#	15.1
2	#	19.0
3	#	385.0
4	52.0	36.5
5	585.0	250.0
6	31.3	34.6
7	12.0	16.4
8	126.0	92.9
9	793.0	474.0
10	61.5	57.1
11	107.0	110.0
12	2230.0	1070.0
13	581.0	157.0
14	381.0	73.3
15	692.0	136.0
16	**	651.0

$x^* \in int(\underline{\hat{x}})$ *and* $\varepsilon_1 = 10^{-6}$: CPU times in seconds

Table 7.3.2

Ex.	Alg.	f^*
1	H	#
	MW	$[-100.0000010000000, -99.99999999999998]$
2	H	#
	MW	$[-0.2172346282112217, -0.2172336282112216]$
3	H	#
	MW	$[-12.03125044216714, -12.03124944216713]$
4	H	$[-0.9987 \times 10^{-6}, 0.1273 \times 10^{-8}]$
	MW	$[-0.1 \times 10^{-5}, 0.2995 \times 10^{-26}]$
5	H	$[-0.9997 \times 10^{-6}, 0.2900 \times 10^{-9}]$
	MW	$[-0.1 \times 10^{-5}, 0.1727 \times 10^{-20}]$
6	H	$[-0.1 \times 10^{-5}, 0.3 \times 10^{-11}]$
	MW	$[-0.1 \times 10^{-5}, 0.4 \times 10^{-11}]$
7	H	$[-0.5824461744390637, -0.5824451744390636]$
	MW	$[-0.5824461744436350, -0.5824451744436349]$
8	H	$[-0.1 \times 10^{-5}, 0.5561 \times 10^{-10}]$
	MW	$[-0.1 \times 10^{-5}, 0.5458 \times 10^{-29}]$
9	H	$[0.7731980565150145, 0.7731990565150146]$
	MW	$[0.7731980564929241, 0.7731990564929242]$
10	H	$[-1.031629453489845, -1.031628453489844]$
	MW	$[-1.031629453489878, -1.031628453489877]$

$x^* \in int(\underline{\hat{x}})$ *and* $\varepsilon_1 = 10^{-6}$: The global minima

Table 7.3.3

Ex.	Alg.	f^*
11	H	$[-0.9956 \times 10^{-6}, 0.4379 \times 10^{-8}]$
	MW	$[-0.1 \times 10^{-5}, 0.3249 \times 10^{-11}]$
12	H	$[-0.9822 \times 10^{-6}, 0.1784 \times 10^{-7}]$
	MW	$[-0.1 \times 10^{-5}, 0.2179 \times 10^{-15}]$
13	H	$[-0.1 \times 10^{-5}, 0.3096 \times 10^{-11}]$
	MW	$[-0.1 \times 10^{-5}, 0.4403 \times 10^{-20}]$
14	H	$x^* \in \partial(\underline{\hat{x}})$
	MW	$x^* \in \partial(\underline{\hat{x}})$
15	H	$[-0.999999974 \times 10^{-6}, 0.2601 \times 10^{-13}]$
	MW	$[-0.9999999999999937 \times 10^{-6}, 0.6308 \times 10^{-20}]$
16	H	**
	MW	$[-0.9999999994464952 \times 10^{-6}, -0.5535 \times 10^{-15}]$

$x^* \in int(\underline{\hat{x}})$ *and* $\epsilon_1 = 10^{-6}$: The global minima

Table 7.3.3 contd.

Ex.	Alg.	n_f	n_g	n_{Hd}	n_c	n_F	$n_J/n_{F'}$	n_s
1	H	#	#	#	-	-	#	#
	MW	18	13	0	4	20	14	69
2	H	#	#	#	-	-	#	#
	MW	12	6	0	1	8	6	33
3	H	#	#	#	-	-	#	#
	MW	102	50	0	1	49	40	242
4	H	126	196	68	-	-	42	432
	MW	68	30	10	10	58	31	207
5	H	1350	2118	880	-	-	450	4798
	MW	274	112	19	19	239	122	785
6	H	107	161	46	-	-	32	346
	MW	86	46	6	6	62	37	243
7	H	59	93	32	-	-	19	203
	MW	40	20	8	7	34	19	128
8	H	212	331	132	-	-	74	749
	MW	138	75	17	16	124	70	440
9	H	183	282	92	-	-	57	614
	MW	137	70	18	16	111	58	410
10	H	175	264	94	-	-	52	585
	MW	124	71	21	21	115	66	418
11	H	169	261	120	-	-	54	604
	MW	52	29	20	10	48	28	187
12	H	1678	2557	1434	-	-	501	6170
	MW	877	367	158	79	748	357	2586
13	H	**	**	**	-	-	**	**
	MW	326	221	78	25	261	123	1034
14	H	150	211	0	-	-	55	416
	MW	32	12	0	0	11	5	60
15	H	**	**	**	-	-	**	**
	MW	263	132	15	2	160	71	643
16	H	**	**	**	-	-	**	**
	MW	612	244	21	2	307	125	1311

$x^* \in \underline{\hat{x}}$ *and* $\varepsilon_1 = 10^{-6}$: The number of evaluations of various functions

Table 7.4.1

Example	H	MW
1	#	19.7
2	#	19.9
3	#	398.0
4	52.5	39.8
5	582.0	164.0
6	42.1	42.4
7	21.7	19.3
8	130.0	107.0
9	770.0	534.0
10	66.1	77.5
11	531.0	175.0
12	2330.0	1330.0
13	**	621.0
14	94.1	27.0
15	**	849.0
16	**	2680.0

$x^* \in \hat{\underline{x}}$ *and* $\varepsilon_1 = 10^{-6}$: CPU times in seconds

Table 7.4.2

Ex.	Alg.	f^*
1	H	#
	MW	$[-100.0000010000000, -99.99999999999998]$
2	H	#
	MW	$[-0.2172346282112217, -0.2172336282112216]$
3	H	#
	MW	$[-12.03125044216714, -12.03124944216713]$
4	H	$[-0.9987 \times 10^{-6}, 0.1273 \times 10^{-8}]$
	MW	$[-0.1 \times 10^{-5}, 0.2995 \times 10^{-26}]$
5	H	$[-0.9997 \times 10^{-6}, 0.2900 \times 10^{-9}]$
	MW	$[-0.1 \times 10^{-5}, 0.2307 \times 10^{-10}]$
6	H	$[-0.1 \times 10^{-5}, 0.3 \times 10^{-11}]$
	MW	$[-0.1 \times 10^{-5}, 0.4 \times 10^{-11}]$
7	H	$[-0.5824461744430856, -0.5824451744430854]$
	MW	$[-0.5824461744436350, -0.5824451744436349]$
8	H	$[-0.1 \times 10^{-5}, 0.5561 \times 10^{-10}]$
	MW	$[-0.1 \times 10^{-5}, 0.5412 \times 10^{-29}]$
9	H	$[0.7731980565150145, 0.7731990565150146]$
	MW	$[0.7731980564929241, 0.7731990564929242]$
10	H	$[-1.031629453489845, -1.031628453489844]$
	MW	$[-1.031629453489878, -1.031628453489877]$

$x^* \in \hat{\underline{x}}$ *and* $\epsilon_1 = 10^{-6}$: The global minima

Table 7.4.3

Ex.	Alg.	f^*
11	H	$[-0.1 \times 10^{-5}, 0.1154 \times 10^{-9}]$
	MW	$[-0.1 \times 10^{-5}, 0.6727 \times 10^{-17}]$
12	H	$[-0.9822 \times 10^{-6}, 0.1784 \times 10^{-7}]$
	MW	$[-0.1 \times 10^{-5}, 0.2126 \times 10^{-15}]$
13	H	**
	MW	$[-0.1 \times 10^{-5}, 0.1041 \times 10^{-27}]$
14	H	$[-24.00000100000001, -23.99999999999999]$
	MW	$[-24.00000100000001, -24.0]$
15	H	**
	MW	$[-0.1 \times 10^{-5}, 0.1623 \times 10^{-19}]$
16	H	**
	MW	$[-0.9999999999999909 \times 10^{-6}, 0.9165 \times 10^{-20}]$

$x^* \in \hat{\underline{x}}$ *and* $\varepsilon_1 = 10^{-6}$: The global minima

Table 7.4.3 contd.

Ex. : case	Alg.	initial box	n_s	CPU time
15 : (i)	H	$([0.99, 1.045])_{6\times1}$	62	29.9
	MW	$([0.99, 1.045])_{6\times1}$	275	178.0
	H	$([0.5, 1.045])_{6\times1}$	1514	699.0
	MW	$([0.5, 1.045])_{6\times1}$	260	135.0
15 : (ii)	H	$([0.99, 1.045])_{6\times1}$	**	**
	MW	$([0.99, 1.045])_{6\times1}$	1154	1570.0
	H	$([0.5, 1.045])_{6\times1}$	**	**
	MW	$([0.5, 1.045])_{6\times1}$	684	909.0

Comparison between different values of $\hat{\underline{x}}$

Table 7.5

Example	Algorithm	(i)	(ii)	(iii)	(iv)
4	*KMSW*	46.2	47.9	36.5	39.8
	MAP	46.1	42.2	37.4	50.6
6	*KMSW*	32.6	47.9	34.6	42.4
	MAP	45.2	41.6	33.7	53.2
14	*KMSW*	72.3	26.9	73.3	27.0
	MAP	72.0	26.5	71.6	26.1

CPU times in seconds

Table 7.6

Example	*MW* with	(i)	(ii)	(iii)	(iv)
6	$\hat{\underline{z}}^{(i)}$ $(i = 0, \ldots, 2^n - 1)$	32.6	47.9	34.6	42.4
6	$\hat{\underline{z}}$	48.9	60.9	48.0	64.0

CPU times in seconds

Table 7.7

Example	MW with	(i)	(ii)	(iii)	(iv)
14	$\underline{\hat{z}}^{(i)}$ $(i = 0, \ldots, 2^n - 1)$	72.3	26.9	73.3	27.0
14	$\underline{\hat{z}}$	317.0	*	307.0	*

CPU times in seconds

Table 7.8

n	Alg.	n_f	n_g	n_{Hd}	n_c	n_F	n_J/n_F	$Time$
2	H	14	20	8	-	-	4	5.98
	MW	19	8	4	4	11	7	7.72
3	H	26	39	24	-	-	8	19.9
	MW	33	10	10	5	15	9	22.5
4	H	42	66	52	-	-	13	47.1
	MW	39	12	18	6	22	11	42.1
5	H	80	125	125	-	-	25	125.0
	MW	61	14	32	8	26	13	73.0
6	H	194	304	366	-	-	61	426.0
	MW	103	20	50	10	38	19	153.0
7	H	*	*	*	-	-	*	*
	MW	172	20	72	12	39	19	212.0
8	H	*	*	*	-	-	*	*
	MW	310	25	98	14	50	24	361.0

$x^* \in int(\underline{\hat{z}})$ *and* $\varepsilon_1 = 0$:

The number of evaluations of various functions and CPU times in seconds

Table 7.9.1

n	Alg.	n_f	n_g	n_{Hd}	n_c	n_F	n_J/n_F	*Time*
2	*H*	137	217	36	-	-	48	66.6
	MW	104	79	4	4	76	40	66.2
3	*H*	426	644	54	-	-	115	298.0
	MW	244	157	8	4	161	78	308.0
4	*H*	787	1162	32	-	-	205	748.0
	MW	414	261	0	0	308	130	893.0
5	*H*	2077	3053	360	-	-	603	2820.0
	MW	582	389	0	0	443	194	2080.0
6	*H*	**	**	**	-	-	**	**
	MW	**	**	**	**	**	**	**

$x^* \in \hat{\underline{x}}$ *and* $\varepsilon_1 = 0$:

The number of evaluations of various functions and CPU times in seconds

Table 7.9.2

n	Alg.	n_f	n_g	n_{Hd}	n_c	n_F	$n_J/n_{F'}$	*Time*
2	*H*	11	17	8	-	-	4	4.59
	MW	17	8	3	3	12	7	8.34
3	*H*	23	36	24	-	-	8	17.7
	MW	25	10	8	4	15	9	17.8
4	*H*	36	58	48	-	-	12	41.3
	MW	39	12	15	5	18	11	34.6
5	*H*	77	122	125	-	-	25	120.0
	MW	59	14	28	7	24	13	65.1
6	*H*	191	301	366	-	-	61	419.0
	MW	99	19	45	9	34	18	131.0
7	*H*	*	*	*	-	-	*	*
	MW	170	20	66	11	37	19	198.0
8	*H*	*	*	*	-	-	*	*
	MW	307	25	91	13	47	24	337.0

$x^* \in int(\hat{x})$ *and* $\varepsilon_1 = 10^{-6}$:

The number of evaluations of various functions and CPU times in seconds

Table 7.9.3

n	Alg.	n_f	n_g	n_{Hd}	n_c	n_F	$n_J/n_{F'}$	*Time*
2	*H*	73	114	16	-	-	24	33.0
	MW	44	39	0	0	30	19	26.6
3	*H*	202	307	24	-	-	57	135.0
	MW	118	77	8	4	77	38	142.0
4	*H*	439	656	40	-	-	121	422.0
	MW	214	133	0	0	149	66	415.0
5	*H*	1069	1581	175	-	-	319	1440.0
	MW	310	197	0	0	220	98	970.0
6	*H*	**	**	**	-	-	**	**
	MW	453	271	0	0	315	135	2080.0

$x^* \in \hat{\underline{x}}$ *and* $\varepsilon_1 = 10^{-6}$:

The number of evaluations of various functions and CPU times in seconds

Table 7.9.4

$n(case)$	with	n_f	n_g	n_{Hd}	n_c	n_F	$n_J/n_{F'}$	$Time$
2(i)	$\hat{\underline{z}}^{(i)}$	19	8	4	4	11	7	7.72
	$\hat{\underline{z}}$	19	10	5	5	15	9	10.4
2(ii)	$\hat{\underline{z}}^{(i)}$	104	79	4	4	76	40	66.2
	$\hat{\underline{z}}$	120	88	7	7	92	45	79.1
2(iii)	$\hat{\underline{z}}^{(i)}$	17	8	3	3	12	7	8.34
	$\hat{\underline{z}}$	17	10	4	4	15	9	9.96
2(iv)	$\hat{\underline{z}}^{(i)}$	44	39	0	0	30	19	26.6
	$\hat{\underline{z}}$	49	45	1	1	39	23	34.7
3(i)	$\hat{\underline{z}}^{(i)}$	33	10	10	5	15	9	22.5
	$\hat{\underline{z}}$	33	13	10	5	22	12	28.9
3(ii)	$\hat{\underline{z}}^{(i)}$	244	157	8	4	161	78	308.0
	$\hat{\underline{z}}$	277	183	12	6	200	91	363.0
3(iii)	$\hat{\underline{z}}^{(i)}$	25	10	8	4	15	9	17.8
	$\hat{\underline{z}}$	27	13	8	4	21	12	24.6
3(iv)	$\hat{\underline{z}}^{(i)}$	118	77	8	4	77	38	142.0
	$\hat{\underline{z}}$	129	89	6	3	91	44	169.0

The number of evaluations of various functions and CPU times in seconds

Table 7.9.5

APPENDIX A

Notation

In this thesis, the sets R of real numbers, R^n of real $n \times 1$ vectors, $M(R^m, R^n)$ of real $m \times n$ matrices, $M(R^n)$ of real $n \times n$ matrices, $I(R)$ of real intervals, $I(R^n)$ of real $n \times 1$ interval vectors, $I(M(R^m, R^n))$ of real $m \times n$ interval matrices, and $I(M(R^n))$ of real $n \times n$ interval matrices are defined as follows.

$$R = \{x \mid x \text{ is a real number}\}, \tag{A.1}$$

$$R^n = \{x = (x_i)_{n\times 1} \mid x_i \in R \ (i = 1, \ldots, n)\}, \tag{A.2}$$

$$M(R^m, R^n) = \{A = (a_{ij})_{m\times n} \mid a_{ij} \in R \ (i = 1, \ldots, m, \ j = 1, \ldots, n)\}, \tag{A.3}$$

$$M(R^n) = \{A = (a_{ij})_{n\times n} \mid a_{ij} \in R \ (i, j = 1, \ldots, n)\}, \tag{A.4}$$

$$I(R) = \{\underline{x} = [x_I, x_S] \mid x_I, x_S \in R \wedge x_I \leq x_S\}, \tag{A.5}$$

$$I(R^n) = \{\underline{x} = (\underline{x}_i)_{n\times 1} \mid \underline{x}_i \in I(R) \ (i = 1, \ldots, n)\}, \tag{A.6}$$

$$I(M(R^m, R^n)) = \{\underline{A} = (\underline{a}_{ij})_{m\times n} \mid \underline{a}_{ij} \in I(R)\ (i = 1, \ldots, m,\ j = 1, \ldots, n)\},\quad A.7$$

and

$$I(M(R^n)) = \{\underline{A} = (\underline{a}_{ij})_{n\times n} \mid \underline{a}_{ij} \in I(R)\ (i, j = 1, \ldots, n)\}.\quad A.8$$

Although we normally use lower-case italic letters to denote real numbers and real vectors, upper-case italic letters to denote real matrices, the underlined lower-case italic letters to denote intervals and interval vectors, and the underlined upper-case italic letters to denote interval matrices we use also, for example n, N, $\underline{S}$, $\underline{K}_N$, $\underline{H}$, $\underline{H}'$, $\underline{Q}$, B and P which are defined in the text and which appear in deference to common usage.

The mapping $|\cdot| : R^n \to R^n$ is defined by $| x | = (| x_i |)_{n\times 1}$. The sets R^n and $M(R^n)$ are partially ordered through $(x \leq y) \Leftrightarrow (x_i \leq y_i\ (i = 1, \ldots, n))$ and $(A \leq B) \Leftrightarrow (a_{ij} \leq b_{ij}\ (i, j = 1, \ldots, n))$ respectively.

The width $w : I(R) \to R$, the magnitude $|\cdot| : I(R) \to R$, and the midpoint $m : I(R) \to R$ are defined by

$$w(\underline{x}) = x_S - x_I,\quad A.9$$

$$| \underline{x} | = max\{| x_I |, | x_S |\},\quad A.10$$

and

$$m(\underline{x}) = (x_I + x_S)/2.\quad A.11$$

The width $w : I(R^n) \to R^n$, the magnitude $|\cdot| : I(R^n) \to R^n$, and the midpoint $m : I(R^n) \to R^n$ are defined by

$$w(\underline{x}) = (w(\underline{x}_i))_{n\times 1}, \tag{A.12}$$

$$|\underline{x}| = (|\underline{x}_i|)_{n\times 1}, \tag{A.13}$$

and

$$m(\underline{x}) = (m(\underline{x}_i))_{n\times 1}. \tag{A.14}$$

The width $w : I(M(R^n)) \to M(R^n)$, the magnitude $|\cdot| : I(M(R^n)) \to M(R^n)$, and the midpoint $m : I(M(R^n)) \to M(R^n)$ are defined by

$$w(\underline{A}) = (w(\underline{a}_{ij}))_{n\times n}, \tag{A.15}$$

$$|\underline{A}| = (|\underline{a}_{ij}|)_{n\times n}, \tag{A.16}$$

and

$$m(\underline{A}) = (m(\underline{a}_{ij}))_{n\times n}. \tag{A.17}$$

The norms $\|\cdot\| : R^n \to R$, $\|\cdot\| : I(R^n) \to R$, and $\|\cdot\| : I(M(R^n)) \to R$ are defined by

$$\|x\| = \max_{1 \le i \le n} \{| x_i |\}, \tag{A.18}$$

$$\|\underline{x}\| = \max_{1 \le i \le n} \{| \underline{x}_i |\}, \tag{A.19}$$

and

$$\|\underline{A}\| = \max_{1 \le i \le n} \{\sum_{j=1}^{n} | \underline{a}_{ij} |\} \tag{A.20}$$

respectively.

The mappings $\sigma : R^n \times R^n \to R$, $\sigma_1 : I(R) \times I(R) \to R$ and $\sigma_n : I(R^n) \times I(R^n) \to R$ which are defined by

$$\sigma(x, y) = \|x - y\|, \tag{A.21}$$

$$\sigma_1(\underline{x}, \underline{y}) = max\{| x_I - y_I |, | x_S - y_S |\}, \tag{A.22}$$

and

$$\sigma_n(\underline{x}, \underline{y}) = \max_{1 \le i \le n} \{\sigma_1(\underline{x}_i, \underline{y}_i)\}, \tag{A.23}$$

are metrics for the sets R^n, $I(R)$, and $I(R^n)$ respectively. Furthermore (R^n, σ) and $(I(R^n), \sigma_n)$ are complete metric spaces.

An interval $\underline{x} \in I(R)$ is *degenerate* if and only if $x_I = x_S$. Thus degenerate intervals are of the form $[x, x]$ $(x \in R)$.

The function $f : R^n \to R^1$ which is m times continuously differentiable in an open convex set $\hat{D} \subseteq D$ is denoted by $f \in C^m(\hat{D})$.

If $f : D \subseteq R^n \to R^1$ is a given mapping with $f \in C^2(D)$ then the gradient $g : D \to R^n$ and the Jacobian $J : D \to M(R^n)$ of f are defined by

$$g = (g_i)_{n\times 1} = (\partial_1 f, \ldots, \partial_n f)^T = (\frac{\partial}{\partial x_1} f, \ldots, \frac{\partial}{\partial x_n} f)^T \qquad A.24$$

and

$$J = (J_{ij})_{n\times n} = (\partial_j \partial_i f)_{n\times n} = (\frac{\partial^2}{\partial x_j \partial x_i} f)_{n\times n}. \qquad A.25$$

Let $f : D \subseteq R^n \to R^1$, $c_i : R^n \to R^1$ $(i = 1, \ldots, m)$, and $h_j : R^n \to R^1$ $(j = 1, \ldots, r)$ be given mappings with f, c_i $(i = 1, \ldots, m)$, h_j $(j = 1, \ldots, r) \in C^1(\hat{D})$ where $\hat{D} \subseteq D$ is an open convex set. Then the derivative of the Lagrangian function $L : R^n \times R^m \times R^r \to R^1$ defined by

$$L(x, u, w) = f(x) - \sum_{i=1}^{m} u_i c_i(x) + \sum_{j=1}^{r} w_j h_j(x)$$

with respect to x is denoted by $\nabla_x L(x, u, w)$.

If $f : D \subseteq R^n \to R^1$ is a given mapping and $f(x)$ can be written in the form

$$f(x) = f(c) + g(x - c) \qquad A.26$$

where

$$g(x - c) = f(x) - f(c)$$

and $c = m(\underline{x})$ $(\underline{x} \in I(D))$ then the expression on the right-hand side of A.26 is called the *centred form* of $f(x)$ [RatR-84a].

An n-dimensional interval vector

$$\underline{x} = (\underline{x}_1, \ldots, \underline{x}_n)^T \qquad A.27$$

is called a *box* if and only if $\underline{x}$ is a parallelepiped with sides parallel to the co-ordinate axes.

APPENDIX B

Pseudo-code

This appendix contains a brief description of the pseudo-code which is used to express algorithms in this thesis. The pseudo-code is intended to be self-explanatory.

B.1 Control Structures

This section contains a description of the control structures for the pseudo-code. Statements are denoted by $s_1, \ldots, s_n$, and c is a boolean expression which might contain the operators *and*, *or*, $\sim$ (= *not*).

The control structures are as follows.

Assignment : An assignment statement (:=) is written as follows.

$$< variable >:=< expression >$$

Meaning : The value of $< expression >$ is assigned to $< variable >$.

if-do : An *if-do* statement is written as follows.

1. *if* c *do*

1.1 s_1

⋮

1.n s_n

2. s

Meaning : If c is true, then execute $s_1, \ldots, s_n$ and then execute s; if c is false then execute s only.

if-then-else : An *if-then-else* statement is written as follows.

1. *if* c

   then

   1.1 s_1

   ⋮

   1.n s_n

   else

   1.$n+1$ s_{n+1}

   ⋮

   1.$n+m$ s_{n+m}

2. s

Meaning : If c is true, then execute $s_1, \ldots, s_n$; otherwise execute $s_{n+1}, \ldots, s_{n+m}$. Then execute s.

case-true-of : A *case-true-of* statement is written as follows.

1. *case true of*

   c_1 :

   1.1 s_1

   .
   .
   .

   $1.n_1$ s_{n_1}

   c_2 :

   $1.n_1 + 1$ s_{n_1+1}

   .
   .
   .

   $1.n_2$ s_{n_2}

   .
   .
   .

   c_m :

   $1.n_{m-1} + 1$ $s_{n_{m-1}+1}$

   .
   .
   .

   $1.n_m$ s_{n_m}

default :

1.$n_m + 1$ s_{n_m+1}

⋮

1.n_{m+1} $s_{n_{m+1}}$

2. s

Meaning : For $i = 1, \ldots, m$ if c_i is true then execute the statements $s_{n_{i-1}+1}, \ldots, s_{n_i}$ only, where $n_0 = 0$; otherwise execute $s_{n_m+1}, \ldots, s_{n_{m+1}}$. Then execute s.

case-< *integer* >-*of* : A *case*-< *integer* >-*of* statement is written as follows.

1. *case* < *integer* > *of*

i_1 :

1.1 s_1

⋮

1.n_1 s_{n_1}

i_2 :

1.$n_1 + 1$ s_{n_1+1}

⋮

1.n_2 s_{n_2}

$\vdots$

i_m :

$1.n_{m-1}+1\ s_{n_{m-1}+1}$

$\vdots$

$1.n_m\ s_{n_m}$

default :

$1.n_{i,}+1\ s_{n_m+1}$

$\vdots$

$1.n_{m+1}\ s_{n_{m+1}}$

2. s

Meaning : For $j = 1,\ldots,m$ if $i_j =<$ *integer* $>$ then execute the statements $s_{n_{j-1}+1},\ldots,s_{n_j}$ only, where $n_0 = 0$; otherwise execute $s_{n_m+1},\ldots,s_{n_{m+1}}$. Then execute s.

for-to-do : A *for-to-do* statement is written as follows.

1. *for* $i = i_1$ *to* i_2 *do*

   1.1 s_1

   $\vdots$

   $1.n\ s_n$

2. s

where i is an integer variable and i_1 and i_2 are the integer values.

Meaning : If $i_2 \geq i_1$ then execute $s_1, \ldots, s_n$ with $i = i_1, i_1 + 1, \ldots, i_2$ and then execute s; otherwise execute s only.

for-by-do : A *for-by-do* statement is written as follows.

1. *for* $i = i_1$ *to* i_2 *by* i_3 *do*

   1.1 s_1

   ⋮

   1.n s_n

2. s

where i is an integer variable and i_1, i_2, and i_3 are integer values.

Meaning : If $i_1 \leq i_2$ and $i_3 > 0$ then execute $s_1, \ldots, s_n$ with $i = i_1, i_1 + i_3, \ldots, i_1 + mi_3$ where the integer m is such that $i_1 + mi_3 \leq i_2 < i_1 + (m+1)i_3$. If $i_1 \leq i_2$ and $i_3 \leq 0$ then do not execute $s_1, \ldots, s_n$. If $i_1 \geq i_2$ and $i_3 < 0$ then execute $s_1, \ldots, s_n$ with $i = i_1, i_1 + i_3, \ldots, i_1 + mi_3$ where the integer m is such that $i_1 + mi_3 \geq i_2 > i_1 + (m+1)i_3$. If $i_1 \geq i_2$ and $i_3 \geq 0$ then do not execute $s_1, \ldots, s_n$. Then execute s.

while-do : A *while-do* statement is written as follows.

1. *while* c *do*

   1.1 s_1

   .
   .
   .

   1.n s_n

2. s

Meaning : While c is true, execute $s_1, \ldots, s_n$ repeatedly; then execute s.

repeat-while-do : A *repeat-while-do* statement is written as follows.

1. *repeat*

   1.1 s_1

   .
   .
   .

   1.n s_n

   while c *do*

   1.$(n+1)$ s_{n+1}

   .
   .
   .

   1.$(n+m)$ s_{n+m}

2. s

Meaning : Execute $s_1, \ldots, s_n$; if c is true then execute $s_{n+1}, \ldots, s_{n+m}$ and go to step 1.1; else go to step 2.

{ } : A { } statement is written as follows

1. { }

Meaning : Do nothing.

Comment : A comment statement is written as follows.

! < *statement* >.

Meaning : The statement which is written after the symbol ! is treated as comment.

B.2 Procedure Declaration and Calling

In order to design a well-defined format for declaring and invoking procedures we need the following ideas.

(1) Input Parameters : These parameters are separated by commas. Each one supplies a value for computation and does not return a specific value at the end of the computation.

(2) Input-Output Parameters : These parameters are separated by commas. Each one supplies a value and returns a value at the end of the computation.

(3) Output Parameters : These parameters are separated by commas. They are declared and initialized during the computation and they return values at the end of the computation.

(4) *return* Statement : This statement is used to terminate the computation.

One type of procedure is used in this thesis only; its general form is as follows.

procedure < *name* > (< *input parameters* > ;
< *input – output parameters* > :
< *output parameters* >)

< *comment* >

< *block of statements* >

return □

where < *name* > is the name of the procedure, < *comment* > consists of several sentences which explain the procedure < *name* > and the parameters, < *block of statements* > consists of the *statements* which constitute the procedure < *name* > and the procedure is terminated by the *return* statement. An example follows.

procedure *gradient.test*($\underline{\hat{x}} \in I(R^n), n, l \in N, q \in N^{2n}, x.star.in.x \in B$;

$S_7 \in P, \underline{z} \in I(R^{3n}), \underline{\bar{f}} \in I(R), n_p \in N$:

$k \in N, delete.z \in B$)

The full procedure is given in §6.6.

Here, $\underline{\hat{x}}$, n, l, q, and $x.star.in.x$ are input parameters, S_7, $\underline{z}$, $\underline{\bar{f}}$, and n_p are input-output parameters, k, and $delete.z$ are output parameters and $I(R^n)$, N, N^{2n}, B, P, $I(R^{3n})$, and $I(R)$ are data types.

The data-types which are used in this thesis are given in the Table B.1.

Type	Notation	Vector Notation	Matrix Notations	
integer	N	N^n	–	–
real	R	R^n	$M(R^m, R^n)$	$M(R^n)$
string	S	–	–	–
boolean	B	B^n	–	–
interval	$I(R)$	$I(R^n)$	$I(M(R^m, R^n))$	$I(M(R^n))$
queue	Q	–	–	–
stack	P	–	–	–

Table B.1

In Chapter 6 it is convenient to use functions as well as procedures. The general form of a function is

function < *name* > (< *input parameters* >)

< *comment* >

< *block of statements* >

return □

in which < *block of statements* > contains at least one statement of the form

< *name* >:=< *expression* >

The function < *name* > is invoked by using a statement of the form

< *variable* >:=< *name* > (< *input parameters* >).

After the execution of this statement, < *variable* > contains the value of the single entity which is computed by the function < *name* >.

An example follows.

function $max(x, y \in R)$

! This function determines $max\{x, y\}$.

1. *if* $x > y$

   then

   1.1. $max := x$

   else

   1.2. $max := y$

2. *return* □

After the statement

$$z := max(x, y)$$

is executed, z has the value $max\{x, y\}$.

B.3 Algorithm Declaration

In this thesis, every method is expressed as a procedure (§B.2). All the procedures are combined with other blocks of statements (§B.2) to form an algorithm. The *stop* statement is used to terminate the algorithm. The parameters which are used in an algorithm consist of input, input-output, and output parameters described in (§B.2). The algorithm has the following form.

< *name*1 >

Data : < *initial values* >

< *comment* >

< *block of statements* >

procedure < *name* > (< *input parameters* > ;

< *input – output parameters* > :

< *output parameters* >)

stop □

where < *name1* > is the name of the algorithm, < *initial values* > are the values which are supplied by the user, < *comment* >, < *block of statements* >, and < *input parameters* >, < *input – output parameters* >, and < *output parameters* > are as in §*B*.2. The procedure < *name* > is a procedure which combines all the procedures which are needed by the algorithm < *name1* >.

APPENDIX C

Examples

This appendix contains the examples which are used to illustrate the effectiveness of the algorithms MW (Chapter 6) and H (Chapter 3).

Example 1 : [ViZS-75a]

$f(x) = x^2 - 100cos(x)$

$\underline{\hat{x}} = [-10, 10]$

$x^* = 0$

$f^* = -100.0$

Example 2 : [ViZS-75a]

$f(x) = (1/x)sin(x)$

$\underline{\hat{x}} = [-10. , -1]$

$x^* \approx -4.49341797$

$f^* \approx -0.217233628215$

Example 3 : [Han-79a]

$f(x) = -\sum_{k=1}^{5} ksin((k+1)x + k)$

$\underline{\hat{x}} = [-9, 9]$

$x^{*(1)} \approx -6.7745761445$

$x^{*(2)} \approx -0.49139218765$

$x^{*(3)} \approx 5.7917890155$

$f^* \approx -12.03125$

Example 4 :

$f(x) = (x_2 - x_1^2)^2 + (1 - x_1)^2$

$\underline{\hat{x}} = ([-1, 2])_{2\times 1}$

$x^* = (1)_{2\times 1}$

$f^* = 0.0$

Example 5 : [Ros–60a]

$f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$

$\underline{\hat{x}} = ([-1, 2])_{2\times 1}$

$x^* = (1)_{2\times 1}$

$f^* = 0.0$

Example 6 : [Han–80a]

$f(x) = 2x_1^2 - 1.05x_1^4 + x_1^6/6 - x_1x_2 + x_2^2$

$\underline{\hat{x}} = ([-4, 2])_{2\times 1}$

$x^* = (0)_{2\times 1}$

$f^* = 0.0$

Example 7 :

$f(x) = x_1^4 + x_1x_2 + (1 + x_2)^2$

$\underline{\hat{x}} = ([-100, 100])_{2\times 1}$

$x^* \approx (0.695884385 \ -1.34794219)^T$

$f^* \approx -0.582445174445$

Example 8 : [Bra–72a]

$f(x) = 16(x_1 + x_2)^2 + \{4(x_1 + x_2) + (x_1 - x_2)(x_1 - 2) + x_2^2 - 1\}^2$

$\underline{\hat{x}} = ([-2, 4])_{2\times 1}$

$x^{*(1)} \approx (-0.21525043\ 0.215255043)^T$

$x^{*(2)} \approx (1.5485837703548\ -1.5485837703549)^T$

$f^* = 0.0$

Example 9 : [Mad–73b]

$f(x) = (x_1^2 + x_2^2 + x_1 x_2)^2 + sin^2 x_1 + cos^2 x_2$

$\underline{\hat{x}} = ([-1, 2])_{2\times 1}$

$x^{*(1)} \approx (-0.15543723586\ 0.69456377530295)^T$

$x^{*(2)} \approx (0.15543723586\ -0.69456377530295)^T$

$f^* \approx 0.773199035$

Example 10 : [DiGH-75a]

$f(x) = x_1^6/3 - 2.1x_1^4 + 4x_1^2 + x_1 x_2 - 4x_2^2 + 4x_2^4$

$\underline{\hat{x}} = ([-3, 3]\ [-1.5, 1.5])^T$

$x^{*(1)} \approx (-0.08984201310025\ 0.712656403020735)^T$

$x^{*(2)} \approx (0.08984201310025\ -0.712656403020735)^T$

$f^* \approx -1.031628453489855$

Example 11 : Helical Valley Function [FleP-63a]

$f(x) = 100(x_3 - 10\theta)^2 + (\sqrt{x_1^2 + x_2^2} - 1)^2 + x_3^2$

$\theta(x_1, x_2) = (1/2\pi)arctan(x_2/x_1)$ if $x_1 > 0$

$\theta(x_1, x_2) = (1/2\pi)arctan(x_2/x_1) + 0.5$ if $x_1 < 0$

$\underline{\hat{x}} = ([0.991, 1.011]\ [-0.01, 0.01]\ [-0.01, 0.01])^T$

$x^* = (1\ 0\ 0)^T$

$f^* = 0.0$

Example 12 : [Eng–66a]

$f(x) = (x_1^2 + x_2^2 + x_3^2 - 1)^2 + (x_1^2 + x_2^2 + (x_3 - 2)^2 - 1)^2 + (x_1 + x_2 + x_3 - 1)^2 +$
$(x_1 + x_2 - x_3 + 1)^2 + (x_1^3 + 3x_2^3 + (5x_3 - x_1 + 1)^2 - 36)^2$

$\underline{\hat{x}} = ([-0.1, 0.2]\ [-0.1, 0.2]\ [0.9, 1.2])^T$

$x^* = (0,\ 0,\ 1)^T$

$f^* = 0\ 0$

Example 13 : Wood Function [Pea–69a]

$f(x) = 100(x_1^2 - x_2)^2 + (1 - x_1)^2 + 90(x_3^2 - x_4)^2 + (1 - x_3)^2 +$
$10.1((1 - x_2)^2 + (1 - x_4)^2) + 19.8(1 - x_2)(1 - x_4)$

$\underline{\hat{x}} = ([0.979, 1.001])_{4\times 1}$

$x^* = (1)_{4\times 1}$

$f^* = 0.0$

Example 14 : [Moh–78a]

$f(x) = x_1^4 - x_2 x_1^3 - x_2 x_3 x_1^2 + x_1 x_2 x_3 x_4$

$\underline{\hat{x}} = ([-1, 2])_{4\times 1}$

$x^* = (2,\ 2,\ 2,\ -1)^T$

$f^* = -24.0$

Example 15 : [Mir–79a]

$f(x) = (1 - x_1)^2 + (1 - x_6)^2 + \sum_{i=1}^{5}(x_i^2 - x_{i+1})^2$

$\underline{\hat{x}} = ([0.5, 1.045])_{6\times 1}$

$x^* = (1)_{6\times 1}$

$f^* = 0.0$

Example 16 : [Mir–79a]

$f(x) = (1 - x_1)^2 + (1 - x_8)^2 + \sum_{i=1}^{7}(x_i^2 - x_{i+1})^2$

$\underline{\hat{x}} = ([0.5, 1.1])_{8\times 1}$

$x^* = (1)_{8\times 1}$

$f^* = 0.0$

Example 17 : [Mir–79a]

$f(x) = (1 - x_1)^2 + (1 - x_n)^2 + \sum_{i=1}^{n-1}(x_i^2 - x_{i+1})^2$

$\underline{\hat{x}} = ([0.5, 1])_{n\times 1}$

$x^* = (1)_{n\times 1}$

$n = 2, \ldots, 8$

$f^* = 0.0$

APPENDIX D

This appendix contains the procedures *time*, *decode*, *function*, *gradient*, *jacobian*, *hessian.diagonal.element*, *constraint*, *F*, and *F.prime*.

procedure time

! This procedure returns a string in the form

! $n1, n2 : n3, n4 : n5, n6 : n7, n8$

! where the decimal digits $ni (i = 1, \ldots, 8)$ represent the time in hours, minutes,

! seconds, tenths of a second and hundredths of a second from midnight. □

procedure decode$(s \in S)$

! This procedure takes a string s as its argument and returns the integer value of the

! ASCII code for the first character of s as its result. □

procedure function$(\underline{x} \in I(R^n), n \in N \ : \ \underline{f} \in I(R))$

! This procedure computes an interval extension of the objective function

! $f : R^n \rightarrow R^1$ for problem P.

! On entry, $\underline{x}$ is the box over which $\underline{f}$ is evaluated, and n is the number of variables

! in $\underline{x}$.

! On return, $\underline{f} = \underline{f}(\underline{x})$.

1. $\underline{f} := \underline{f}(\underline{x})$! *Examples* $1 - 12$ *in Appendix C.*

2. *return* □

procedure gradient$(\underline{x} \in I(R^n), n \in N \ : \ \underline{g} \in I(R^n))$

! This procedure computes an interval extension of $f' : R^n \rightarrow R^n$ where

! $f' : R^n \rightarrow R^n$ is the derivative of $f : R^n \rightarrow R^1$.

! On entry, $\underline{x}$ is the box over which $\underline{f'}$ is evaluated, and n is the number of variables

! in $\underline{x}$.

! On return, $\underline{g} = \underline{f'}(\underline{x})^T = (\partial_1 f, \ldots, \partial_n f)^T$.

1. *for* $i = 1$ *to* n *do*

   1.1. $\underline{g_i} := \partial_i \underline{f}(\underline{x})$

2. *return* □

procedure $jacobian(\underline{x} \in I(R^n), n \in N \ : \ \underline{J} \in I(M(R^n))$

! This procedure computes an interval extension of $f'' \ : \ R^n \to M(R^n)$ for algorithm

! H (Chapter 3) only where $f'' \ : \ R^n \to M(R^n)$ is the jacobian of $f' \ : \ R^n \to R^n$.

! On entry, $\underline{x}$ is the box over which $\underline{f}''$ is evaluated, and n is the number of components

! in $\underline{x}$.

! On return, $\underline{J} = (\partial_j \partial_i f)_{n \times n}$.

1. *for* $i = 1$ *to* n *do*

   1.1. *for* $j = 1$ *to* n *do*

      1.1.1. $\underline{J}_{ij} := \partial_j \partial_i \underline{f}(\underline{x})$

2. *return* □

procedure $hessian.diagonal.element(\underline{x} \in I(R^n), n, i \in N \ : \ \underline{G}_{ii} \in I(R))$

! This procedure computes an interval extension of the ith diagonal element of the

! Hessian $G \ : \ R^n \to M(R^n)$ of $f \ : \ R^n \to R^1$ with the argument $(\underline{x}_1, \ldots, \underline{x}_n)$.

! On entry, $\underline{x}$ is the box over which $\underline{G}_{ii}$ $(i = 1, \ldots, n)$ is evaluated, n is the number

of

! variables in $\underline{x}$, and i is the ith diagonal element of the Hessian G.

! On return, $\underline{G}_{ii} = \partial_i \partial_i \underline{f}(\underline{x})$.

1. *case* i *of*

   $1, \ldots, n$:

   1.1. $\underline{G}_{ii} := \partial_i \partial_i \underline{f}(\underline{x})$

   default :

   1.2. *write* "*Error in hessian.diagonal.element.*"

   1.3. *stop*

2. *return* □

procedure constraint$(\hat{\underline{x}}, \underline{x} \in I(R^n), n \in N \ : \ \underline{c} \in I(R^{2n}))$

! This procedure computes the interval extension of the constraints $c_i : R^n \to R^1$

! for $i = 1, \ldots, 2n$.

! On entry, $\hat{\underline{x}}$ is the initial box, $\underline{x}$ is the box over which $\underline{c}_i$ $(i = 1, \ldots, 2n)$ is evaluated,

! and n is the number of components in $\underline{x}$.

! On return, $\underline{c} = \underline{c}(\underline{x})$.

1. *for* $i = 1$ *to* n *do*

1.1. $\underline{c}_i := \underline{x}_i - \hat{x}_{iI}$

1.2. $\underline{c}_{i+n} := \hat{x}_{iS} - \underline{x}_i$

2. *return* □

procedure $F(\underline{\hat{x}} \in I(R^n), \underline{z} \in I(R^{3n}), q \in N^{2n}, n, l \in N, x.star.in.x \in B$:

$\underline{F} \in I(R^{3n-l}))$

! This procedure computes an interval extension of $F : R^{3n} \rightarrow R^{3n}$ (Chapter 4)

! On entry, $\underline{\hat{x}}$ is the initial box in which $f : R^n \rightarrow R^1$ is defined, $\underline{z} = (\underline{x}^T, \underline{u}^T)^T$ is

! the current box in which $F : R^{3n} \rightarrow R^{3n}$ is to be computed, $q_i \in \{0, 1\}$

! $(i = 1, \ldots, 2n)$, $0 \leq l \leq 2n$, $x.star.in.x$ is such that $x.star.in.x = \underline{true}$ if $x^* \in int(\underline{\hat{x}})$

! and $x.star.in.x = \underline{false}$ if $x^* \in \underline{\hat{x}}$ where x^* is the global minimizer of $f : R^n \rightarrow R^1$.

! On return, if l Lagrange multiplier bounds in $\underline{z}$ are zero intervals then $\underline{F} = \underline{F}(\underline{z})$

! contains $3n - l$ components that is $\underline{F} = (\underline{F}_1, \ldots, \underline{F}_n, \underline{F}_{n+1}, \ldots, \underline{F}_{3n-l})^T$.

1. *if* $x.star.in.x$ *or* $l = 2n$

then

1.1. $m := n$

1.2. *for* $i = 1$ *to* $2n$ *do*

1.2.1. $\underline{z}_{n+i} := [0, 0]$

else

1.3. $m := 3n - l$

1.4. *for* $i = 1$ *to* $2n$ *do*

1.4.1. *if* $q_i = 0$ *do*

1.4.1.1. $\underline{z}_{n+i} := [0, 0]$

2. $k := n$

3. $\underline{F} := (\underline{0})_{m \times 1}$

4. *for* $i = 1$ *to* n *do*

4.1. $\underline{F}_i := \partial_i f(\underline{x}) \mid \underline{z} = (\underline{x}^T, \underline{u}^T)^T$.

5. *if* $\sim x.star.in.x$ *and* $l \neq 2n$ *do*

5.1. *for* $i = 1$ *to* n *do*

5.1.1. $\underline{F}_i := \underline{F}_i - \underline{z}_{n+i} + \underline{z}_{2n+i}$

5.2. *for* $i = 1$ *to* n *do*

5.2.1. *if* $q_i = 1$ *do*

5.2.1.1. $k := k + 1$

5.2.1.2. $\underline{F}_k := \underline{z}_{n+i}(\underline{z}_i - \hat{x}_{iI})$

5.3. *for* $i = n + 1$ *to* $2n$ *do*

5.3.1. *if* $q_i = 1$ *do*

5.3.1.1. $k := k + 1$

5.3.1.2. $\underline{F}_k := \underline{z}_{n+i}(\hat{x}_{i-nS} - \underline{z}_{i-n})$

6. *return* □

procedure $F.prime(\underline{x} \in I(R^n), \underline{z} \in I(R^{3n}), q \in N^{2n}, n, l \in N, x.star.in.x \in B$:

$$\underline{F'} \in I(M(R^{3n-l}))$$

! This procedure computes an interval extension of $F' : R^{3n} \to M(R^{3n})$.

! On entry, all the parameters are as in the procedure F.

! On return, if l Lagrange multiplier bounds in $\underline{z}$ are zero intervals then $\underline{F'} = \underline{F'}(\underline{z})$

! is a $(3n - l)$-dimensional matrix.

1. *if* $x.star.in.x$ *or* $l = 2n$

   then

   1.1. $m := n$

   1.2. *for* $i = 1$ *to* $2n$ *do*

      1.2.1. $\underline{z}_{n+i} := [0, 0]$

   else

   1.3. $m := 3n - l$

   1.4. *for* $i = 1$ *to* $2n$ *do*

      1.4.1. *if* $q_i = 0$ *do*

         1.4.1.1. $\underline{z}_{n+i} := [0, 0]$

2. $k := n$

3. $\underline{F'} := (\underline{0})_{m \times m}$

4. *if* $\sim x.star.in.x$ *and* $l \neq 2n$ *do*

4.1. *for* $j = 1$ *to* n *do*

4.1.1. *if* $q_j = 1$ *do*

4.1.1.1. $k := k + 1$

4.1.1.2. $\underline{F'_{kj}} := \underline{z_{n+j}}$

4.1.1.3. $\underline{F'_{jk}} := -[1, 1]$

4.1.1.4. $i := n + 1$

4.1.1.5. $p := j$

4.1.1.6. *while* $p > 1$ *do*

4.1.1.6.1. $p := p - 1$

4.1.1.6.2. *if* $q_p = 1$ *do*

4.1.1.6.2.1. $i := i + 1$

4.1.1.7. $\underline{F'_{ki}} := \underline{z_j} - \hat{x}_{jI}$

4.2. *for* $j = n + 1$ *to* $2n$ *do*

4.2.1. *if* $q_j = 1$ *do*

4.2.1.1. $k := k + 1$

4.2.1.2. $\underline{F'_{kj-n}} := -\underline{z_{n+j}}$

4.2.1.3. $\underline{F'_{j-nk}} := [1, 1]$

4.2.1.4. $i := n + 1$

4.2.1.5. $p := j$

4.2.1.6. *while* $p > 1$ *do*

4.2.1.6.1. $p := p - 1$

4.2.1.6.2. *if* $q_p = 1$ *do*

4.2.1.6.2.1. $i := i + 1$

4.2.1.7. $\underline{F}'_{ki} := \hat{x}_{jS} - z_{j-n}$

5. *for* $i = 1$ *to* n *do*

5.1. *for* $j = 1$ *to* n *do*

5.1.1. $\underline{F}'_{ij} := \partial_j \partial_i \underline{f}(\underline{x})$! $\underline{z} = (\underline{x}^T, \underline{u}^T)^T$.

6. *return* □

References

[Ale--84a] Alefeld, G., *On the Convergence of Some Interval-Arithmetic Modifications of Newton's Method*, SIAM Journal on Numerical Analysis, 21 (1984), pp. 363 - 372.

[AleH-83a] Alefeld, G., and Herzberger, J., *Introduction to Interval Computations*, Academic Press, New York, 1983.

[AleP-83a] Alefeld, G., and Platzöder, L., *A Quadratically Convergent Krawczyk - Like Algorithm*, SIAM Journal on Numerical Analysis, 20 (1983), pp. 210 - 219.

[And--72a] Anderssen, R. S., *Global Optimization*, in Optimization, Anderssen, R. S., Jennings, L. S., and Ryan, D. M. (Editors), University of Queensland Press, 1972.

[ArcB-78a] Archetti, F., and Betro, B., *A Priori Analysis of Deterministic Strategies for Global Optimisation Problems*, in Towards Global Optimisation 2, Dixon, L. C. W., and Szegö, G. P. (Editors), North Holland Publishing Company, 1978.

[AsSM-82a] Asaithambi, N. S., Sheu, Z., and Moore, R. E., *On Computing the Range of Values*, Computing, 28 (1982), pp. 225 - 237.

[BaCM-82a] Bailey, P. J., Cole, A. J., and Morrison, R., *Triplex User Manual CS/82/5*, University of St. Andrews, Department of Computational Science, North Haugh, St. Andrews, Fife, KY16 9SX, Scotland.

[BecL-70a] Becker, R. W., and Lago, G. V., *A Global Optimization Algorithm*, Proceedings of the Eighth Allerton Conference on Circuits and System Theory, 1970.

[BiaF-78a] Biase, L. D., and Frontini, F., *A Stochastic Method for Global Optimisation : Its Structure and Numerical Performance*, in Towards Global Optimisation 2, Dixon, L. C. W., and Szegö, G. P. (Editors), North Holland Publishing Company, 1978.

[Bra--71a] Branin Jr., F. H., *Solution of Nonlinear DC Network Problems via Differential Equations*, Memoirs Mexico 1971 IEEE Conference on Systems, Networks, and Computers, Oaxtepec, Mexico.

[Bra--72a] Branin Jr., F. H., *A Widely Convergent Method for Finding Multiple Solutions of Simultaneous Nonlinear Equations*, Tech. Report 21.466, IBM Systems Development Division Laboratory, Kingston, New York, 1972.

[BraH-72a] Branin Jr., F. H., and Hoo, S. K., *A Method for Finding Multiple Extrema of a Function of n Variables*, in Numerical Methods of Nonlinear Optimisation, Lootsma, F. (Editor), Academic Press, London, 1972.

[Bro--58a] Brooks, S. H., *A Discussion of Random Methods for Seeking Maxima*, Operations Research, 6 (1958), pp. 244 - 251.

[Brou-12a] Brouwer, L. E. J., *Uber die Abbildung von Mannigfaltigkeiten*, Math. Ann., 71 (1912), pp. 97 - 115.

[Broy-75a] Broyden, C. G., *Basic Matrices*, The Macmillan Press Ltd, 1975.

[Chi--67a] Chichinadze, V. K., *Eng. Cyb.*, No. 1 (1967), pp. 115 - 123.

[ChuM-72a] Chuba, W., and Miller, W., *Quadratic Convergence in Interval Arithmetic, Part I*, BIT, 12 (1972), pp. 284 - 290.

[ColM-82a] Cole, A. J., and Morrison, R., *An Introduction to Programming with S-Algol*, Cambridge University Press, Cambridge, 1982.

[ColM-82b] Cole, A. J., and Morrison, R., *Triplex : A System for Interval Arithmetic - Practice and Experience*, 12(1982), pp. 341 - 350.

[Cor--75a] Corles, C. R., *The Use of Regions of Attraction to Identify Global Minima*, in Towards Global Optimisation, Dixon, L. C. W., and Szegö, G. P. (Editors), North Holland Press, 1975.

[CorL-84a] Cornelius, H., and Lohner, R., *Computing the Range of Values of Real Functions with Accuracy Higher than Second Order*, Computing 33 (1984), pp. 331 - 347.

[DenS-83a] Dennis Jr, J. E., and Schnabel, R. B., *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice-Hall, New Jersey, 1983.

[DiGH-75a] Dixon, L. C. W., Gomulka, J., and Hersom, S. E., *Reflections on the Global Optimisation Problem*, Numerical Optimisation Centre, Technical Report No. 64, 1975.

[DiGS-74a] Dixon, L. C. W., Gomulka, J., and Szegö, G. P., *Towards Global Optimisation Techniques*, Numerical Optimisation Centre, Technical Report No. 61, 1974.

[Dix--74a] Dixon, L. C. W., *Nonlinear Optimization : A Survey of the State of Art*, in Software for Numerical Mathematics, Evans, D. J. (Editor), Academic Press, 1974.

[DixS-78a] Dixon, L. C. W., and Szegö, G. P., *The Global Optimisation Problem : An Introduction*, in Towards Global Optimisation 2, Dixon, L. C. W., and Szegö, G. P. (Editors), North Holland Publishing Company, 1978.

[Dus--72a] Dussel, R., *Einschliessung des Minimalpunktes einer Streng Konvexen Function auf einem n-dimensionalen Quader*, Internal Report no. 74, University Karlsruhe, Institute for Practical Mathematik, Karlsruhe, 1972.

[Eng--66a] Engwall, J. L., *Numerical Algorithms for Solving Overdetermined Systems of Nonlinear Equations*, NASA document N70-35600, 1966.

[Evt--71a] Evtushenko, Y. G., Zh. Vychisl. Mat. mat. Fiz., 11, 6 (1971), pp. 1390 - 1403.

[FaPZ-78a] Fagiuoli, E., Pianca, P., and Zecchin, M., *A Mixed Stochastic-Deterministic Technique for Global Optimisation*, in Towards Global Optimisation 2, Dixon, L. C. W., and Szegö, G. P. (Editors), North Holland Publishing Company, 1978.

[FiaM-68a] Fiacco, A. V., and McCormick, G. P., *Nonlinear Programming : Sequential Unconstrained Minimization Techniques*, John Wiley and Sons, Inc. 1968.

[Fle--80a] Fletcher, R., *Practical Methods of Optimization : Unconstrained Optimization*, Volume 1, John Wiley & Sons, New York, 1980.

[Fle--81a] Fletcher, R., *Practical Methods of Optimization : Constrained Optimization*, Volume 2, John Wiley & Sons, New York, 1981.

[FleP-63a] Fletcher, R., and Powell, M. J. D., *A Rapidly Convergent Descent Method for Minimization*, Computer Journal, 7 (1963), pp. 163 - 188.

[Gal--51a] Gale, D., *Convex Polyhedral Cones and Linear Inequalities*, in T.C. Koopmans (Editor), Activity Analysis of Production and Allocation, Wiley, New York, 1951.

[Gav--75a] Gaviano, M., *On the Convergence of Random Search Algorithms for Minimisation Problems*, in Towards Global Optimisation, Dixon, L. C. W., and Szegö, G. P. (Editors), North Holland Press, 1975.

[GolP-71a] Goldstein, A. A., and Price, J. F., *On Descent from Local Minima*, Mathematics of Computation, 25 (1971), pp. 569 - 574.

[Gom--75a] Gomulka, J., *Remarks on Branin's Method, A Counter Example*, in Towards Global Optimization, Dixon, L. C. W., and Szegö, G. P. (Editors), North Holland Press, 1975.

[Com--77a] Gomulka, J., *Deterministic vs Probabilistic Approaches to Global Optimisation*, Numerical Optimisation Centre, Technical Report No. 88, 1977.

[Gom--77b] Gomulka, J., *Two Implementations of Branin's Method : Numerical Experience*, Numerical Optimisation Centre, Technical Report No. 89, 1977.

[Gom--78a] Gomulka, J., *A Users Experience with Törn's Clustering Algorithm*, in Towards Global Optimization 2, Dixon, L. C. W., and Szegö, G. P. (Editors), North Holland Publishing Company, 1978.

[GooL-70a] Good, D. I., and London, R. L., *Computer Interval Arithmetic : Definition and Proof of Correct Implementation*, Journal of the Association for Computing Machinery, 17 (1970), pp. 603 - 612.

[Han--65a] Hansen, E. R., *Interval Arithmetic in Matrix Computations, Part I*, SIAM Journal on Numerical Analysis, 2 (1965), pp. 308 - 320.

[Han--68a] Hansen, E. R., *On Solving Systems of Equations Using Interval Arithmetic*, Mathematics of Computation, 22 (1968), pp. 374 - 384.

[Han--69a] Hansen, E. R. (Editor), *Topics in Interval Analysis*, Oxford University Press, 1969.

[Han--69b] Hansen, E. R., *On the Solution of Linear Algebraic Equations with Interval Coefficients*, Linear Algebra and Application, 2 (1969) pp. 153 - 165.

[Han--78a] Hansen, E. R., *A Globally Convergent Interval Method for Computing And Bounding Real Roots*, BIT, 13 (1978), pp. 415 - 424.

[Han--78b] Hansen, E. R., *Interval Forms of Newtons Method*, Computing, 20 (1978), pp. 153 - 163.

[Han--79a] Hansen, E. R., *Global Optimization Using Interval Analysis - The One-Dimensional Case*, J.O.T.A., 29 (1979), pp. 331 - 344.

[Han--80a] Hansen, E. R., *Global Optimization Using Interval Analysis - The Multi - Dimensional Case*, Numer. Math., 34 (1980), pp. 247 - 270.

[HanG-82a] Hansen, E. R., and Greenberg, R. I., *An Interval Newton Method*, Applied Mathematics and Computation, (1982), pp. 89 - 98.

[HanS-80a] Hansen, E. R., and Sengupta, S., *Global Constrained Optimization Using Interval Analysis*, in Interval Mathematics 1980, Nickel, K. L. E. (Editor), Academic Press, New York, 1980, pp. 25 - 47.

[HanS-81a] Hansen, E. R., and Sengupta, S., *Bounding Solutions of Systems of Equations Using Interval Analysis*, BIT, 21 (1981), pp. 203 - 211.

[HanS-67a] Hansen, E. R., and Smith, R. R., *Interval Arithmetic in Matrix Computations, Part II*, SIAM Journal on Numerical Analysis, 4 (1967), pp. 1 - 9.

[Har--75a] Hardy, J., *An Implemented Extension of Branin's Method*, in Towards Global Optimisation, Dixon, L. C. W., and Szegö, G. P. (Editors), North Holland Press, 1975.

[Hart-72a] Hartman, J. K., *Some Experiments in Global Optimization*, Naval Postgraduate School, Monterey, California, NPS 55HH72051A, 1972.

[Her--75a] Hersom, S. E., *The Practice of Optimisation, Part I*, in Towards Global Optimisation, Dixon, L. C. W., and Szegö, G. P. (Editors), North Holland Press, 1975.

[IchF-79a] Ichida, K., and Fujii, Y., *An Interval Arithmetic Method for Global Optimization*, Computing 23 (1979) pp. 85 - 97.

[IchF-85a] Ichida, K., and Fujii, Y., *Maximization of Multivariable Functions Using Interval Analysis*, in Interval Mathematics 1985, Nickel, K. L. E. (Editor). Freiburg, 1985, pp. 17 - 26.

[Jon--78a] Jones, S. T., *Searching for Solutions of Finite Nonlinear Systems - An Interval Approach*, Ph. D. Thesis, University of Wisconsin, Madison U. S. A., 1978.

[Jon--80a] Jones, S. T., *Locating Safe Starting Regions for Iterative Methods : A Heuristic Algorithm*, in Interval Mathematics 1980, Nickel, K. L. E. (Editor), Academic Press, New York, 1980, pp. 377 - 386.

[Kah--66a] Kahan, W. M., *A Computable Error-Bound for Systems of Ordinary Differential Equations*, Abstract, SIAM Review, 8 (1966), pp. 568 - 569.

[Kra--69a] Krawczyk, R., *Newton-Algorithmen Zur Bestimmung Von Nulstellen Mit Fehlerschranken*, Computing, 4 (1969), pp. 187 - 201.

[Kra--80a] Krawczyk, R., *Comparison of Diverse Iteration Methods for Sets*, in Interval Mathematics 1980, Nickel, K. L. E. (Editor), Academic Press, New York, 1980, pp. 387 - 396.

[Kra--80b] Krawczyk, R., *Interval Extensions and Interval Iterations*, Computing, 24 (1980), pp. 119 - 129.

[Kra--84a] Krawczyk, R., *Interval Iterations for Including a Set of Solutions*, Computing, 32 (1984), pp. 13 - 31.

[KraS-80a] Krawczyk, R., and Selsmark, F., *Order-Convergence and Iterative Interval Methods*, J.Math.Anal. and Appl., 73 (1980), pp. 1 - 23.

[Lid--76a] Liddell, H. M., *Use of Optimization Techniques in Optical Filter Design*, in Optimization in Action, Dixon, L. C. W. (Editor), Academic Press, London, 1976.

[Loo--72a] Lootsma, F. A., *A Survey of Methods for Solving Constrained Minimization Problems via Unconstrained Minimization*, in Numerical Methods of Nonlinear Optimisation, Lootsma, F. A. (Editor), Academic Press, London, 1972.

[Mad--73a] Madsen, K., *On The Solution of Nonlinear Equations in Interval Arithmetic*, BIT, 13 (1973), pp. 428 - 433.

[Mad--73b] Madsen, K., *An Algorithm for Minimax Solution of Overdetermined Systems of Nonlinear Equations*, AERE Report TP559, 1973.

[ManF-67a] Mangasarian, O. L., and Fromswitz, S., *The Fritz John Necessary Optimality Conditions in the Presence of Equality and Inequality Constraints*, J. Math. Anal. and Appl., 17 (1967), pp. 37 - 47.

[ManM-76a] Mancini, L. J., and McCormick, G. P., *Bounding Global Minima*, Mathematics of Operations Research, 1 (1976), pp. 50 - 53.

[Mc---72a] McCormick, G. P., *Attempts to Calculate Global Solutions of Problems that may have Local Minima*, in Numerical Methods of Nonlinear Optimization, Lootsma F. (Editor), Academic Press, London, 1972.

[Mc---83a] McCormick, G. P., *Nonlinear Programming: Theory, Algorithms and Applications*, John Willey & Sons, New York, 1983.

[MckN-76a] Mckeown, J. J., and Nag, A., *An Application of Optimization Techniques to the Design of an Optical Filter*, in Optimization in Action, Dixon, L. C. W. (Editor), Academic Press, London, 1976.

[Mil--72a] Miller, W., *Quadratic Convergence in Interval Arithmetic, Part II*, BIT, 12 (1972), pp. 291 - 298.

[Mir--79a] Mirnia-Harikandi, K., *Modifications of Some Algorithms for Unconstrained Optimization*, PH. D. Thesis, University of St. Andrews, Scotland, 1979.

[Moh--78a] Mohd, I. B., *Chebyshev Series Approximation in Optimization*, M. Sc. Dissertation, Loughborough University of Technology, U. K., 1978.

[Moh--84a] Mohd, I. B., *The Implementation of* [Han--79a], Unpublished Report, Department of Applied Mathematics, University of St. Andrews, Scotland, United Kingdom, 1984.

[Moo--62a] Moore, R. E., *Interval Arithmetic and Automatic Error Analysis in Digital Computing*, Applied Mathematics and Statistics Lab., Report 25, Stanford University, 1962.

[Moo--66a] Moore, R. E., *Interval Analysis*, Prentice-Hall, Englewood Cliffs, 1966.

[Moo--76a] Moore, R. E., *On Computing the Range of a Rational Function of n Variables over a Bounded Region*, Computing, 16 (1976), pp. 1 - 15.

[Moo--77a] Moore, R. E., *A Test for Existence of Solutions to Nonlinear Systems*, SIAM Journal on Numerical Analysis, 14 (1977), pp. 611 - 615.

[Moo--78a] Moore, R. E., *Bounding Sets in Function Spaces with Applications to Nonlinear Operator Equations*, SIAM Review, 20 (1978), pp. 492 - 512.

[Moo--78b] Moore, R. E., *A Computational Test for Convergence of Iterative Methods for Nonlinear Systems*, SIAM Journal on Numerical Analysis, 15 (1978), pp. 1194 - 1196.

[Moo--79a] Moore, R. E., *Methods and Applications of Interval Analysis*, SIAM, Philadelphia, 1979.

[Moo--80a] Moore, R. E., *Interval Methods for Nonlinear Systems*, Computing Supplementum, 2 (1980), pp. 113 - 120.

[Moo--80b] Moore, R. E., *New Results On Nonlinear Systems*, in Interval Mathematics 1980, Nickel, K. L. E. (Editor), Academic Press, New York, 1980, pp. 165 - 180.

[MooJ-77a] Moore, R. E., and Jones, S. T., *Safe Starting Regions for Iterative Methods*, SIAM Journal on Numerical Analysis, 14 (1977), pp. 1051 - 1065.

[MooK-80a] Moore, R. E., and Kioustelidis, *A Simple Test for Accuracy of Approximate Solutions to Nonlinear (or Linear) Systems*, SIAM Journal on Numerical Analysis, 17 (1980), pp. 521 - 529.

[MooQ-82a] Moore, R. E., and Qi, L., *A Successive Interval Test for Nonlinear Systems*, SIAM Journal on Numerical Analysis, 19 (1982), pp. 845 - 850.

[MorC-83a] Morrison, R., Cole, A. J., Bailey, P. J., Wolfe, M. A., and Shearer, M., *Experience in Using a High Level Language which Supports Interval Arithmetic*. In Proceedings of ARITH6, the Joint TCCA/IEEE sixth Symposium on Computer Arithmetic, Aarhus, Denmark, 1983.

[MoTZ-78a] Mockus, J., Tiesis, V., and Zilinskas, A., *The Application of Bayesian Methods for Seeking the Extremum*, in Towards Global Optimisation, Dixon, L. C. W., and Szegö, G. P. (Editors), North Holland Publishing Company, 1978.

[Nic--69a] Nickel, K., *Triplex-Algol and Its Applications*, in Topics In Interval Analysis, Hansen, E. R. (Editor), Oxford Press, 1969.

[Nic--71a] Nickel, K., *On the Newton Method in Interval Analysis*, MRC Technical Summary Report 1136 (1971), Madison, Wisconsin, U. S. A..

[NicR-72a] Nickel, K., and Ritter, K., *Termination Criterion and Numerical Convergence*, SIAM Journal on Numerical Analysis, 9 (1972), pp. 277 - 283.

[Pea--69a] Pearson, J. D., *Variable Metric Methods for Minimization*, Computer Journal, 12 (1969), pp. 171 - 178.

[Pri--78a] Price, W. L., *A Controlled Random Search Procedure for Global Optimization*, in Towards Global Optimisation 2, Dixon, L. C. W., Szegö, G. P. (Editors), North Holland Publishing Company, 1978.

[Qi---80a] Qi, L., *A Generalization of The Krawczyk-Moore Algorithm*, in Interval Mathematics 1980, Nickel, K. L. E. (Editor), Academic Press, New York, 1980, pp. 481 - 488.

[Qi---81a] Qi, L., *Interval Boxes of Solutions of Nonlinear Systems*, Computing, 27 (1981), pp. 137 - 144.

[Qi---82a] Qi, L., *A Note On The Moore Test for Nonlinear Systems*, SIAM Journal on Numerical Analysis, 19 (1982), pp. 851 - 857.

[Ral--74a] Rall, L. B., *A Note on the Convergence of Newton's Method*, SIAM Journal on Numerical Analysis, 11 (1974), pp. 34 - 36.

[Ral--80a] Rall, L. B., *A Comparison of the Existence Theorems of Kantorovich And Moore*, SIAM Journal on Numerical Analysis, 17 (1980), pp. 148 - 161.

[Rat--80a] Ratschek, H., *Centered Forms*, SIAM Journal on Numerical Analysis, 17 (1980), pp. 656 - 662.

[RatR-80a] Ratschek, H., and Rokne, J., *About the Centered Form*, SIAM Journal on Numerical Analysis, 17 (1980), pp. 333 - 337.

[RatR-80b] Ratschek, H., and Rokne, J., *Optimality of the Centered Form*, in Interval Mathematics, Nickel, K. L. E. (Editor), Academic Press, New York, 1980, pp. 490 - 508.

[RatR-84a] Ratschek, H., and Rokne, J., *Computer Methods for the Range of Functions*, Ellis Horwood Ltd., Chichester, 1984.

[Rob--73a] Robinson, S. M., *Computable Error Bounds for Nonlinear Programming*, Mathematical Programming, 5 (1973), pp. 235 - 242.

[Ros--60a] Rosenbrock, H. H., *An Automatic Method for Finding the Greatest or Least Value of a Function*, Computer Journal, 3 (1960), pp. 175 - 184.

[RubW-77a] Rubinstein, Y., and Weissman, I., *The Monte Carlo Method for Global Optimization*, Operation Research, Statistics and Economics - Mimeograph Series 187, 1977.

[ShaA-73a] Shampine, L. F., and Allen Jr, R. C., *Numerical Computing : An Introduction*, W.B. Saunders Company, Philadelphia, 1973.

[SheW-85a] Shearer, J. M., and Wolfe, M. A., *Some Computable Existence, Uniqueness, and Convergence Tests for Nolinear Systems*, SIAM Journal on Numerical Analysis, 22 (1985), pp. 1200 - 1207.

[SheW-85b] Shearer, J. M., and Wolfe, M. A., *A Note On the Algorithm of Alfeld and Platzöder*. To appear in SIAM Journal on Scientific and Statistical Computing, 1986.

[SheW-85c] Shearer, J. M., and Wolfe, M. A., *An Improved Form of the Krawczyk-Moore Algorithm*, Applied Mathematics and Computation, 17 (1985), pp. 229 - 239.

[Shu--72a] Shubert, B. O., *A Sequential Method Seeking the Global Maximum of a Function*, SIAM Journal on Numerical Analysis, 9 (1972), pp. 379 - 388.

[Ske--74a] Skelboe, S., *Computation of Rational Interval Functions*, BIT, 14 (1974), pp. 87 - 95.

[Ste--73a] Stewart, N., *Interval Arithmetic for Guaranteed Bounds in Linear Programming*, J.O.T.A., 12 (1973), pp. 1 - 5.

[Tor--78a] Törn, A. A., *A Search-Clustering Approach to Global Optimization*, in Towards Global Optimisation 2, Dixon, L. C. W., and Szegö, G. P. (Editors), North Holland Publishing Company, 1978.

[Tre--75a] Treccani, G., *A New Strategy for Global Minimization*, in Towards Global Optimisation, Dixon, L. C. W., and Szegö, G. P. (Editors), North Holland Press, 1975.

[TrTS-72a] Treccani, G., Trabattoni, L., and Szegö, G. P., *A Numerical Method for the Isolation of Minima*, in Minimization Algorithms, Szegö, G. P. (Editor), Academic Press, New York, 1972.

[Uei--72a] Ueing, U., *A Combinatorial Method to Compute a Global Solution of Certain Non-Convex Optimization Problems*, in Numerical Methods of Nonlinear Optimization, Lootsma, F. A. (Editor), Academic Press, London, 1972.

[Van--78a] Vandergraft, J. S., *Introduction to Numerical Computations*, Academic Press, New York, 1978.

[ViZS-75a] Vilkov, A. V., Zhidkov, N. P., and Shchedrin, B. M., *A Method of Finding the Global Minimum of a Function of one Variable*, U.S.S.R. Computational Mathematics and Mathematical Physics, 15 (1975), Number 4, pp. 221 - 224.

[Wol--78a] Wolfe, M. A., *Numerical Methods for Unconstrained Optimization : An Introduction*, Van Nostrand Reinhold Company, London, 1978.

[Wol--80a] Wolfe, M. A., *A Modification of Krawczyk's Algorithm*, SIAM Journal on Numerical Analysis, 17 (1980), pp. 376 - 379.

[Wol--80b] Wolfe, M. A., *Technical Note : An Existence Convergence Theorem for A Class of Iterative Methods*, J.O.T.A., 31 (1980), pp. 125 - 129.

[Wol--69a] Wolfe, P., *Convergence Conditions for Ascent Methods*, SIAM Review, 11 (1969), pp. 226 - 235.

[Yoh--73a] Yohe, J. M., *Interval Bounds for Square Roots and Cube Roots*, Computing, 11 (1973), pp. 51 - 57.

[Yoh--79a] Yohe, J. M., *Implementing Nonstandard Arithmetics*, SIAM Review, 21 (1979), pp. 34 - 56.