GLOBAL OPTIMIZATION USING INTERVAL ANALYSIS

Second Edition, Revised and Expanded

ELDON HANSEN
Consultant
Los Altos, California

G. WILLIAM WALSTER
Sun Microsystems Laboratories
Mountain View, California, U.S.A.

MARCEL DEKKER, INC. NEW YORK • BASEL

The first edition was *Global Optimization Using Interval Analysis*, Eldon Hansen, ed. (Marcel Dekker, 1992).

Library of Congress Cataloging-in-Publication Data
A catalog record for this book is available from the Library of Congress.

ISBN: 0-8247-4059-9

This book is printed on acid-free paper.

Headquarters
Marcel Dekker, Inc., 270 Madison Avenue, New York, NY 10016, U.S.A.
tel: 212-696-9000; fax: 212-685-4540

Distribution and Customer Service
Marcel Dekker, Inc., Cimarron Road, Monticello, New York 12701, U.S.A.
tel: 800-228-1160; fax: 845-796-1772

Eastern Hemisphere Distribution
Marcel Dekker AG, Hutgasse 4, Postfach 812, CH–4001 Basel, Switzerland
tel: 41-61-260-6300; fax: 41-61-260-6333

World Wide Web: http://www.dekker.com

The publisher offers discounts on this book when ordered in bulk quantities. For more information, write to Special Sales/Professional Marketing at the headquarters address above.

Current printing (last digit):

CR/SH 10 9 8 7 6 5 4 3 2 1

PRINTED IN THE UNITED STATES OF AMERICA

To Cecelia and Kaye

for their love and support.

Foreword

Take note, **mathematicians**. Here you will find a new extension of real arithmetic to interval arithmetic for containment sets (csets) in which there are no undefined operand-operator combinations such as previously "**indeterminate** forms" $0/0$, $\infty - \infty$, etc.

Take note, **hardware and software engineers**, **programmers and computer users**. Here you will find arithmetic with containment sets which is exception free, so exception event handling is unnecessary.

The main content of the volume consists of **interval algorithms for computing guaranteed enclosures** of the sets of points where constrained global optimization occurs. The use of interval methods provides computational proofs of existence and location of global optima. Computer software implementations use outwardly rounded interval (cset) arithmetic to guarantee that even rounding errors are bounded in the computations. The results are mathematically rigorous.

Computer-aided proofs of theorems and long-standing conjectures in analysis have been carried out using outwardly rounded interval arithmetic, including, for example, the Kepler conjecture — finally proved after 300 years. See "Perspectives on Enclosure Methods", U. Kulisch, R. Lohner and A. Fascius (eds.), Springer, 2001.

The earlier edition [Global Optimization Using Interval Analysis, Eldon Hansen, Marcel Dekker, Inc, 1992] has been expanded also by more recently developed methods and algorithms for global optimization problems with either (or both) inequality and equality constraints. In particular, constraint satisfaction and propagation techniques, using interval intersections for instance, discussed in the new chapter on "consistencies", are integrated with Newton-like interval methods, in a step towards bridging the gap between methods that work well “in the large” and those that work well “in the small”.

I wholeheartedly endorse this important new volume, and recommend its serious study by all who are concerned with global optimization.

Ramon Moore

Contents

Preface

The primary purpose of this book is to describe and discuss methods using interval analysis for solving nonlinear equations and the global optimization problem. The overall approach is the same as in the first edition. However, various new procedures are included. Many of them have previously not been published. The methods discussed find the global optimum and provide bounds on its value and location(s). All solution bounds are guaranteed to be correct despite errors from uncertain input data, approximations, and machine rounding.

The global optimization methods considered here are those developed by the authors and their collaborators. Other methods using interval analysis can be found in the literature. Most of the published methods use only subsets of the procedures described herein.

In the first edition of this book, the interval Newton methods for solving systems of nonlinear equations were the most important part of our global optimization algorithms. In the second edition, this place is shared with consistency methods that are used to speed up the initial convergence of algorithms. As in the first edition, these central methods are discussed in detail.

We show that interval Newton and consistency methods can prove the existence and uniqueness of a solution of a system of nonlinear equations in a given region. This has important practical implications for the discussed global optimization algorithms. Proof of existence and/or uniqueness by an interval Newton or consistency method follows as a by-product of either algorithm and requires no extra computing. As before, these proofs hold true in the presence of errors from rounding, approximation and data uncertainty bounded with intervals.

In addition to many new algorithm improvements that result from integrating consistency and classical interval approaches, there is an additional new feature in the second edition. Using a set-theoretic foundation for computing with intervals, it is possible to close interval computing systems. This means that there are no undefined interval operations or functions. The new system works over the set of extended real numbers including infinities. This new system increases the generality of algorithms and simplifies their development and construction.

The first edition contained an extensive set of numerical test results. They are now obsolete. The current edition contains many illustrative numerical examples, but no results from a list of standard tests.

Finally, our work together and its presentation in this book could not have been accomplished without the support and encouragement of far too many individuals and organizations to list. However, for the reasons cited, we want to especially mention the following:

- Ramon Moore, for starting the field of *interval analysis*; for his many and continuing contributions to the field; for his tireless encouragement and support; and for his personal friendship.

- Sun Microsystems Inc., for financial support during the preparation of the manuscript.

- Jeff Tupper, for creating GrafEq™ and for his generous help with final preparation of manuscript Figures.

- Melissa Harrison, for expert consultation and support developing interval-specific LaTeX styles for Scientific Workplace™ and with final preparation of the manuscript.

Eldon Hansen and Bill Walster

Chapter 1

INTRODUCTION

1.1 AN OVERVIEW

In mathematics, there are real numbers, a real arithmetic for combining them, and a real analysis for studying the properties of the numbers and the arithmetic. Interval mathematics is a generalization in which interval numbers replace real numbers, interval arithmetic replaces real arithmetic, and interval analysis replaces real analysis.

Numerical analysis is the study of computing with real (and other kinds of) numbers. Theoretical numerical analysis considers exact numbers and exact arithmetic, while practical numerical analysis considers finite precision numbers in which rounding errors occur. This book is concerned with both theoretical and practical interval analysis for computing with interval numbers.

In this book we limit our attention almost exclusively to real interval analysis. However, an analysis of complex intervals has been defined and used, beginning with Boche (1966). A complex "interval" can be a rectangle, a circle; or a more complicated set. Intervals of magnitude and phase can also be used. Some early publications discussing complex intervals are Alefeld (1968), Alefeld and Herzberger (1974), Gargantini (1975, 1976, 1978), Gargantini and Henrici (1972), Glatz (1975), Henrici (1975), Krier and Spellucci (1975), and Nickel (1969).

1.2 THE ORIGIN OF INTERVAL ANALYSIS

There are several types of mathematical computing errors. Data often contain measurement errors, or are otherwise uncertain because rounding errors generally occur, and approximations are made, etc. The purpose of interval analysis is to provide upper and lower bounds on the effect all such errors and uncertainties have on a computed quantity.

It is desirable to make interval bounds as narrow as possible. A major focus of interval analysis is to develop practical interval algorithms that produce sharp[1] (or nearly sharp) bounds on the solution of numerical computing problems. However, in practical problems with interval inputs, it is often sufficient to simply compute reasonably narrow interval bounds.

Several people independently had the idea of bounding rounding errors by computing with intervals; e.g., see Dwyer (1951), Sunaga (1958), Warmus (1956), (1960) and Wilkinson (1980). However, interval mathematics and analysis can be said to have begun with the appearance of R. E. Moore's book *Interval Analysis* in 1966. Moore's work transformed this simple idea into a viable tool for error analysis. In addition to treating rounding errors, Moore extended the use of interval analysis to bound the effect of errors from all sources, including approximation errors and errors in data.

1.3 THE SCOPE OF THIS BOOK

In this book we focus on a rather narrow part of interval mathematics. One of our goals is to describe algorithms that use interval analysis to solve the global (unconstrained or constrained) nonlinear optimization problem. We show that such problems can be solved with a guarantee that the computed bounds on the location and value of a solution are numerically correct. If there are multiple solutions, all will be found and correctly bounded. It is also guaranteed that the solution(s) are global and not just local.

Our optimization algorithms use interval linear algebra and interval Newton algorithms that solve systems of nonlinear equations. Consequently, we discuss these topics in some detail. Our discussion includes

[1]An interval bound is said to be *sharp* if it is as narrow as possible.

some historical information but is not intended to be exhaustive in this regard.

We also describe and use an extended interval arithmetic. In the past, it has been customary to exclude certain arithmetic operations in both real and interval arithmetic. Hanson (1968) and Kahan (1968) each described incomplete extensions of interval arithmetic in which endpoints of intervals are allowed to be infinite. The foundation for complete interval arithmetic extensions is described in Chapter 4. Alefeld (1968) (See also Hansen (1978b)) described a practical interval Newton algorithm in which division by an interval containing zero is allowed.

The extension of interval arithmetic that we describe is a closed[2] system with no exclusions of any arithmetic operations or values of operands. It includes division by zero and indeterminate forms such as $\frac{0}{0}$, $\infty - \infty$, $0 \times \infty$, and $\frac{\infty}{\infty}$, etc., that are normally excluded from real and extended (i.e., including infinities) real arithmetic systems. It is remarkable that interval analysis allows closure of systems containing such indeterminate forms and infinite values of variables. All the algorithms in this book can be implemented using these closed interval systems. The resulting benefits are increased generality and simpler code.

1.4 VIRTUES AND DRAWBACKS OF INTERVAL MATHEMATICS

The history of floating-point computing and resulting rounding errors are described in Section 4.11 of Hennessy and Patterson (1994). Interval analysis began as a tool for bounding rounding errors. Nevertheless, the belief persists that rounding errors can be easily detected in another way. The contention is that one need only compute a given result using, say single and double precision. If the two results agree to some number of digits, then these digits are correct.

[2]A closed system is one in which there are no undefined arithmetic operand-operator combinations.

1.4.1 Rump's Example

An example of Rump (1988) shows that this argument is not necessarily valid. Using IEEE-754 computers, the following form (from Loh and Walster (2002)) of Rump's expression with $x_0 = 77617$ and $y_0 = 33096$ replicates his original IBM S/370 results.

$$f(x, y) = (333.75 - x^2)y^6 + x^2(11x^2y^2 - 121y^4 - 2) + 5.5y^8 + \frac{x}{2y} \tag{1.4.1}$$

With round-to-nearest (the usual default) IEEE-754 arithmetic, the expression in (1.4.1) produces:

32-bit:	$f(x_0, y_0) = 1.172604$
64-bit:	$f(x_0, y_0) = 1.1726039400531786$
128-bit:	$f(x_0, y_0) = 1.1726039400531786318588349045201838$

All three results agree in the first seven decimal digits and thirteen digits agree in the last two results. Nevertheless, they are all completely incorrect. Even their sign is wrong.

Loh and Walster (2001) show that both Rump's original and the expression for $f(x, y)$ in (1.4.1) reduce to:

$$f(x_0, y_0) = \frac{x_0}{2y_0} - 2, \tag{1.4.2}$$

from which

$$f(x_0, y_0) = -0.827396059946821368141165095479816... \tag{1.4.3}$$

with the above values for x_0 and y_0.

Evaluating $f(x_0, y_0)$ in its unstable forms using interval arithmetic of moderate accuracy produces a wide interval (containing the correct value of

$f(x_0, y_0)$). The fact that the interval is wide even though the argument values are machine-representable is a warning that roundoff and catastrophic cancellation have probably occurred; and therefore higher-accuracy arithmetic is needed to get an accurate answer. In some cases, as is seen in the above example, rearranging the expression can reduce or eliminate the catastrophic cancellation.

1.4.2 Real Examples

Rump's example is contrived. However, rounding errors and the effects of cancellation impact computed results from important real world problems, as documented in:

www.math.psu.edu/dna/disasters/

and by Daumas (2002). For example, the failure of the Patriot Missile battery at Daharan was directly attributable to accumulation of roundoff errors; and the explosion of the Ariane 5 was caused by overflow. The Endeavour US Space Shuttle maiden flight suffered a software failure in its Intelsat satellite rendezvous maneuver and the Columbia US Space Shuttle maiden flight had to be postponed because of a clock synchronization algorithm failure.

Use of standard interval analysis could presumably have detected the roundoff difficulty in the first example. The extended interval arithmetic discussed in Chapter 4 and used in this book would have produced a correct interval result in the second example, even in the presence of overflow. See Walster (2003b) for an extended interval arithmetic implementation standard in which underflow and overflow are respectively distinguished from zero and infinity. The third failure was traced to an input-dependent software error that was not detected in spite of extensive testing. Intervals can be used to perform exhaustive testing that is otherwise impractical. Finally, the fourth failure occurred after the algorithm in question had been subjected to a three year review process and formally *proved* to be correct. Unfortunately, the proof was flawed. Although it is impossible to know, we believe that all of these and similar errors would have been detected if interval rather than floating-point algorithms had been used.

1.4.3 Ease of Use

Despite the value of interval analysis for bounding rounding errors in problems such as these, interval mathematics is less used in practice than one might expect. There are several reasons for this. Undoubtedly, the main reasons are the (avoidable) lack of convenience, the (avoidable) slowness of many interval arithmetic packages, the (occasional) slowness of some interval algorithms, and the (unavoidable) difficulty of some interval problems.

For programming convenience, an interval data type is needed to represent interval variables and interval constants as single entities rather than as two real interval endpoints. This was made possible early in the history of interval computations by the use of precompilers. See, for example, Yohe (1979). However, the programs they produced were quite slow because each arithmetic step was invoked with a subroutine call. Moreover, subroutines to evaluate transcendental functions were inefficient or lacking and interval programs were available on only a few computers.

Eventually, some languages (e.g., Pascal-SC, Ada, and C++) made programming with intervals convenient and reasonably fast by supporting user defined types and operator overloading.

Microprogramming can be fruitful in improving the speed of interval arithmetic. See Moore (1980). However, this has rarely been considered.

Convenient programming of interval computations was made available as part of ACRITH. See Kulisch and Miranker (1983) and IBM (1986a, 1986b). However, the system was designed for accuracy with exact (degenerate interval) inputs rather than speed with interval inputs that are not exact. Because binary-coded decimal arithmetic was used, it was quite slow.

The M77 compiler was developed at the University of Minnesota. See Walster, *et al* (1980). It was available only on certain computers manufactured by Control Data Corp. With this compiler, interval arithmetic was roughly five times slower than ordinary arithmetic. All the numerical results contained in the first edition of this book were computed using the M77 compiler.

More recently compilers have been developed by Sun Microsystems Inc. that represent the current state of the art. See Walster (2000c) and

Walster and Chiriaev (2000). These compilers implement a limited version of the closed numerical system described briefly in Chapter 4. This "Simple" system is designed to be fast when implemented in software. Nevertheless, it permits calculation of interval bounds (although not as narrow as possible) on functions having singularities and indeterminate forms.

Support for computing with intervals has been introduced into popular symbolic computing tools, including:

- Mathematica (see: `www.wolfram.com/`),
- Maple (see: `www.scg.uwaterloo.ca/`),
- MuPad (see: `www.mupad.de/`), and
- Matlab (see: `www.mathworks.com/`).

Using intervals to graph relations that otherwise would be impossible to rigorously visualize has been accomplished in:

- GrafEq (see: `www.peda.com/grafeq`) and
- Graphical Calculator (see: `www.nucalc.com/`).

Good interval arithmetic software for various applied problems is now often available. Nevertheless, except when written in pure Java™, portable codes are rare.

Unfortunately, at least one commercial product uses interval algorithms with a quasi-interval arithmetic that does not produce rigorous interval bounds. This was done for speed, but at the sacrifice of being able to legitimately claim that computed results are interval bounds in the commonly accepted use of the term. All the algorithms in this book produce rigorous interval bounds.

Ideally, interval hardware will simultaneously compute both endpoints of the four basic interval arithmetic operations. When such a computer is built the speed of interval computations will be comparable to that of floating-point arithmetic and there will be no benefit from cutting corners in rigor for speed. See:

`www.sun.com/processors/whitepapers`

There is another reason why interval analysis was slow to become popular. In its early history, computed bounds on the solution of certain problems were very far from sharp. Subsequent analysis by many researchers has made it possible to compute excellent bounds for solutions to a wide variety of applied problems. As yet, this early stigma has been erased only slowly.

1.4.4 Performance Benchmarks

The relative speed of interval and point algorithms is often the cause of confusion and misunderstanding. People unfamiliar with the virtues of interval algorithms often ask what is the relative speed of interval and point operations and intrinsic function evaluations. Aside from the fact that relatively little time and effort have been spent on interval system software implementation and almost no time and effort implementing interval-specific hardware, there is another reason why a different question is more appropriate to ask. Interval and point algorithms solve different problems. Comparing how long it takes to compute guaranteed bounds on the set of solutions to a given problem, as compared to providing an approximate solution of unknown accuracy, is not a reasonable way to compare the speed of interval and point algorithms.

Gustafson (1994a, 1994b, and 1995) has proposed a computer system benchmarking strategy that focuses on the time require to do real work (including to compute results with a known accuracy) rather than solely on the time required to perform a fixed set of arbitrary numerical computations (without regard to their accuracy). By requiring different systems to compute comparable results, his strategy eliminates the kind of confusion that occurs when fundamentally uncomparable point and interval computations are nevertheless compared.

Independently, Walster (2001) has proposed a way of formulating interval performance benchmark problems, designed to clear up this confusion and to provide standards with which to compare different interval implementation systems. The following is a summary of this proposal.

> Floating-point performance benchmark problems are used routinely to measure the performance of floating-point hardware and software systems. As intervals become more widely used,

interval-specific performance tests will be developed. With interval performance benchmarks, there is a need to measure both run-time given the width of computed interval bounds, and the width of computed bounds within a given runtime. Because the quality of interval bounds is self-evident, there need be no requirement that interval benchmark codes are the same, although, they can be. Rather, standard problem statements are needed against which *any* algorithm and computing system, interval or not, can be compared. The following proposals seem reasonable:

- Interval benchmarks must be written as a mathematical problem statement with no specification of how bounds are to be computed. *Bounds*, however, must be produced. In other words, it is an error if computed bounds fail to contain the set of all possible results.
- At least some input data items must be non-degenerate (non-zero width) intervals, to unambiguously reflect the benchmark's interval nature. The width of input data items might be fixed or relative to the magnitude of interval data.
- When possible, benchmarks need to scale as a function of the number of independent variables, so that efficiency can be estimated as a function of problem size and number of processors.
- Single and double precision versions of problems will be included in benchmark tests. Benchmarks can be any problem, including:
    - integration of ordinary or partial differential equations,
    - solution of linear and nonlinear systems of equations,
    - linear or dynamic programming problems, or
    - nonlinear constrained or unconstrained global optimization (nonlinear programming) problems.

For uncomparable fixed sequences of operations, current interval implementations are slower than real (i.e., noninterval) counterparts. As mentioned, above, this is not a necessary limitation.

For uncomparable problems, current interval algorithms can require more interval operations than real counterparts. For example to get narrow bounds on the solution to linear algebraic equations, interval methods sometimes require about six times as many arithmetic operations as real methods require to compute an approximate solution. See Chapter 5. We hope that future research produces more efficient interval algorithms for this important problem. We also hope that comparisons between point and interval algorithms will be confined to comparable problems, such as those described above.

For many even not comparable problems, the operation counts in interval algorithms are similar to those in noninterval algorithms. For example, the number of iterations to bound a polynomial root to a given (guaranteed) accuracy using an interval Newton method (see Section 9.2) is about the same as the number of iterations of a real Newton method to obtain the same (not guaranteed) accuracy.

For some problems, an interval method is faster than a noninterval one. For example, to find *all* the roots of a polynomial requires fewer steps for an interval Newton method than for a noninterval one. This is because the latter generally must do some kind of explicit or implicit deflation. The interval method does not. Another area in which interval algorithms have been reported to be faster than point algorithms is in robust control. See Nataraj and Sadar (2000), and Nataraj (2002a) and (2002b).

1.4.5 Interval Virtues

The transcendent virtue of interval analysis is that it enables the solution of certain problems that cannot be solved by noninterval methods. The primary example is the global optimization problem, which is the major topic of this book. Even if interval procedures for this problem were slow, this fact cannot be considered a flaw. Fortunately, the procedures are quite efficient for most problems. This is in spite of the fact that even computing sharp bounds on the values of a function of n-variables is known to be an NP-hard problem. See Kreinovich (1998). As with many well known point algorithms that efficiently solve NP-hard problems, interval algorithms seek to capitalize on the structure of real world problems. Walster and Kreinovich (2003) characterize the nature of this structure. Many of the new algorithm innovations described in this book, particularly box and hull consistency in Chapter 10, use this strategy to achieve significant performance increases.

The obvious comment regarding the apparent slowness of interval methods for some problems (especially if they lack the structure often found in real world problems) is that a price must be paid to have a reliable algorithm with guaranteed error bounds that non-interval methods do not provide. For some problems, the price is somewhat high; for others it is negligible or nonexistent. For still others, interval methods are more efficient.

Consider a problem in which the input is a degenerate (zero width) interval (or intervals) and we simply wish to bound the effect of rounding errors. For such a problem, we need to do more than just compare the time for the interval and noninterval programs to execute. We also need to compare the time it takes to solve the problem in the interval case with both: the time it takes in the noninterval case, and the time (and effort) it takes in the noninterval case to somehow perform a rigorous error analysis. The proposed interval benchmark standard seeks to expose the time and effort required to produce rigorous bounds using noninterval algorithms.

Next, consider a problem in which the input is a nondegenerate interval (or intervals). For this problem, the interval approach produces a set of answers to a set of problems. In so doing, it provides a rigorous sensitivity analysis (see Chapter 17). For such a problem, it might be difficult or impossible to do the sensitivity analysis by noninterval methods. When it is possible to compare (as above) the speeds of the interval and noninterval approaches to a given problem, the interval approach is often faster.

There are several other virtues of interval methods that make them well worth paying even a real performance price. In general, interval methods are more reliable. As we shall see, some interval iterative methods *always* converge, while their noninterval counterparts do not. An example is the Newton method for solving for the zeros of a nonlinear equation. See Theorem 9.6.2 on page 183.

Also, natural stopping criteria exist for interval iterations. One can simply iterate until either the bounds are sufficiently narrow or no further reduction of the interval bounds occurs. The latter case happens when rounding errors prevent further accuracy. A comparable heuristic stopping criteria used in noninterval algorithms can be difficult to devise and be quite complicated to implement.

Interval methods can yield a valuable by-product. As we shall see, algorithms for solving systems of nonlinear equations can provide *proof of existence* and *uniqueness* of a solution without the need for any computations not already performed in solving the problem. This occurs only for simple (i.e., nonmultiple) zeros.

Interval methods find *all* solutions to a set of nonlinear equations in a given interval vector or box (see Section 5.1 for a formal definition of a box). They do so without the extra analysis, programming, and computation that are necessary for a deflation process that is required by most noninterval methods.

Probably the transcendent virtue of interval mathematics is that it provides solutions to otherwise unsolvable problems. Prior to the use of interval methods, it was impossible to solve the nonlinear global optimization problem except in special cases. In fact, various authors have written that in general it is impossible *in principle* to numerically solve such problems. Their argument is that by sampling values of a function and some of its derivatives at isolated points, it is impossible to determine whether a function dips to a global minimum (say) between the sampled points. Such a dip can occur between adjacent machine-representable floating point values.

Interval methods avoid this difficulty by computing information about a function over continua of points even if interval endpoints are constrained to be machine-representable. As we show in this book, it is not only possible but relatively straightforward to solve the global optimization problem using interval methods.

For an example illustrating how an interval method detects a sharp dip in an objective function, see Moore (1991).

1.5 THE FUTURE OF INTERVALS

Three forces are converging to offer unprecedented computing opportunities and challenges:

- Computer performance continues to double every 18 months (Moore's law),
- Parallel architectures with tens of thousands or even millions of processors will soon be routinely available, and
- Interval algorithms to solve nonlinear systems and global optimization problems are naturally parallel.

With the inherent ability of intervals to represent errors from all sources and to rigorously propagate their interactions, the validity of answers from the most extensive computations can now be guaranteed. With the natural parallel character of nonlinear interval algorithms, it will be possible to efficiently use even the largest parallel computing architectures to safely solve large practical problems.

Computers are attaining the speed required to replace physical experiments with computer simulations. Gustafson (1998) has written that using computers in this way might turn out to be as scientifically important as the introduction of the experimental method in the Renaissance. One difficulty is how to validate computed results from huge simulations. A second difficulty is how to then synthesize simulation results into optimal designs. With interval algorithms, simulation validity can be verified. Moreover, interval global optimization can use the mathematical models derived from validated simulations to solve for optimal designs.

Chapter 2

INTERVAL NUMBERS AND ARITHMETIC

2.1 INTERVAL NUMBERS

Consider a closed[1], real interval $X = [a, b]$. An *interval number* X is such a closed interval. That is, it is the set $\{x \mid a \leq x \leq b\}$ of all real numbers between and including the endpoints a and b. We use the terms "interval number" and "interval" interchangeably. An interval number can be an interval constant or a value of an interval variable.

A real number x is equivalent to an interval $[x, x]$, which has zero width. Such an interval is said to be *degenerate*. When we express a real number as an interval, we usually retain the simpler noninterval notation. For example, we often write 2 in place of [2, 2] or x in place of $[x, x]$

The endpoints a and b of a given interval might not be representable on a given computer. Such an interval might be a datum or the result of a computation on the computer. In such a case, we round a down to the largest machine-representable number that is less than a and round b up to the smallest machine-representable number that is greater than b. Thus, the retained interval contains $[a, b]$. This process is called *outward rounding*.

[1]The word "closed" in this context is short hand for "topologically closed". A closed interval includes the interval's endpoints. An open interval does not.

Directed rounding is rounding that is specified to be either up or down. That is, it is rounding to either a (specified) larger of smaller number than the number being rounded. Directed rounding is used to achieve the outward rounding used in practical interval arithmetic. The IEEE-754 (1985) standard for floating-point arithmetic specifies that directed rounding be an option in computer arithmetic. Directed rounding has been available in hardware since the Intel 8087 chip was introduced in 1981. See Palmer and Morse (1984).

2.2 NOTATION AND RELATIONS

When a real (i.e., noninterval) quantity is expressed in lower case, we generally use the corresponding capital letter to denote the corresponding interval quantity. For example, if x denotes a real variable then X denotes an interval variable. If the real quantity is denoted by a capital letter, we denote the corresponding interval quantity by attaching a superscript "I". For example, if a real matrix is denoted by $\mathbf{A}$, the corresponding interval matrix is denoted by $\mathbf{A}^{\mathbf{I}}$. See Chapter 5.

A superscript "I" on the symbol for a function indicates that it is an interval function. Thus, $f^{\mathbf{I}}$ is an interval function. However, if $f(x)$ is a real function of a real variable x, then $f(X)$ also denotes the corresponding interval function. This fact is indicated by the presence of the interval argument X. For a definition and discussion of an interval function, see Chapter 3. A thorough treatment of the notation used in this book is presented at the end of Chapter 4.

An underbar indicates the lower endpoint of an interval; and an overbar indicates the upper endpoint. For example, if $X = [a, b]$, then $\underline{X} = a$ and $\overline{X} = b$. Similarly, we write $f(X) = [\underline{f}(X), \overline{f}(X)]$.

An interval $X = [a, b]$ is said to be *positive* if $a > 0$ and *nonnegative* if $a \geq 0$. It is said to be *negative* if $b < 0$ and *nonpositive* if $b \leq 0$.

Two intervals $[a, b]$ and $[c, d]$ are *equal* if and only if $a = c$ and $b = d$.

Interval numbers are partially ordered. We have $[a, b] < [c, d]$ if and only if $b < c$.

2.3 FINITE INTERVAL ARITHMETIC

Let $+$, $-$, $\times$, and $\div$ denote the operations of addition, subtraction, multiplication, and division, respectively. If $\bullet$ denotes any one of these operations for arithmetic on real numbers x and y, then the corresponding operation for arithmetic on interval numbers X and Y is

$$X \bullet Y = \{x \bullet y \mid x \in X,\ \ y \in Y\} \tag{2.3.1}$$

Thus the interval $X \bullet Y$ resulting from the operation contains every possible number that can be formed as $x \bullet y$ for each $x \in X$, and each $y \in Y$.

This definition produces the following rules for generating the endpoints of $X \bullet Y$ from the two intervals $X = [a, b]$ and $Y = [c, d]$.

$$X + Y = [a + c, b + d] \tag{2.3.2}$$

$$X - Y = [a - d, b - c] \tag{2.3.3}$$

$$X \times Y = \begin{cases} [ac, bd] & \text{if } a \geq 0 \text{ and } c \geq 0 \\ [bc, bd] & \text{if } a \geq 0 \text{ and } c < 0 < d \\ [bc, ad] & \text{if } a \geq 0 \text{ and } d \leq 0 \\ [ad, bd] & \text{if } a < 0 < b \text{ and } c \geq 0 \\ [bc, ac] & \text{if } a < 0 < b \text{ and } d \leq 0 \\ [ad, bc] & \text{if } b \leq 0 \text{ and } c \geq 0 \\ [ad, ac] & \text{if } b \leq 0 \text{ and } c < 0 < d \\ [bd, ac] & \text{if } b \leq 0 \text{ and } d \leq 0 \\ [\min(bc, ad), & \\ \ \max(ac, bd)] & \text{if } a < 0 < b \text{ and } c < 0 < d \end{cases} \tag{2.3.4}$$

If we exclude division by an interval containing 0 (that is, $c < 0$ or $d > 0$), we have

$$\frac{1}{Y} = \left[\frac{1}{d}, \frac{1}{c}\right] \tag{2.3.5}$$

and

$$\frac{X}{Y} = X \times \left(\frac{1}{Y}\right) \tag{2.3.6}$$

The case of division by an interval containing zero is covered in Chapter 4. For n a nonnegative integer, we also define

$$X^n = \begin{cases} [1, 1] & \text{if } n = 0 \\ [a^n, b^n] & \text{if } a \geq 0 \text{ or if } n \text{ is odd} \\ [b^n, a^n] & \text{if } b \leq 0 \text{ and } n \text{ is even} \\ [0, \max(a^n, b^n)] & \text{if } a \leq 0 \leq b \text{ and } n > 0 \text{ is even.} \end{cases} \tag{2.3.7}$$

2.4 DEPENDENCE

Suppose we subtract the interval $X = [a, b]$ from itself. As a result of using the rule (2.3.3) for subtraction of intervals, we obtain the interval $[a-b, b-a]$. We might expect to obtain $[0, 0]$. However, we do not (unless $b = a$). The result is $\{x - y \mid x \in X, y \in X\}$ instead of $\{x - x \mid x \in X\}$.

In general, each occurrence of a given variable in an interval computation is treated as a *different* variable. Thus $X - X$ is computed as if it were $X - Y$ with Y numerically equal to, but independent of X. This causes widening of computed intervals and makes it difficult to compute sharp numerical results of complicated expressions.

This unwanted extra interval width is called the *dependence problem* or simply *dependence.* One should always be aware of this difficulty and, when possible, take steps to reduce its effect. We discuss some ways to do this in Section 3.3 and elsewhere in this book.

Equation (2.3.7) defines the n-th power of an interval. It is included to overcome the dependence problem in multiplication. For example, when $n = 2$, the definition is equivalent to $X^2 = \{x^2 \mid x \in X\}$ rather than $X \times X = \{x \times y \mid x \in X, y \in X\}$. Using (2.3.7), we compute $[-1, 2]^2 = [0, 4]$ rather than $[-1, 2] \times [-1, 2] = [-2, 4]$ using (2.3.4)

Moore (1966) notes that if a particular interval variable occurs only once in a given form of a function, then it cannot give rise to excess width because of dependence. Suppose every variable occurs only once in a function. Then an (exact) interval evaluation yields the exact range of the function as variables range over their interval bounds.

Thus, dependence can occur in evaluating a function $f(X, Y)$ in the form $\frac{X-Y}{X+Y}$, but not if it is written in the form $1 - \frac{2}{1+\frac{X}{Y}}$. If we evaluate $f(X, Y)$ in the latter form and if no division by an interval containing zero occurs, then the resulting interval is the exact range of $f(x, y)$ for $x \in X$ and $y \in Y$. We discuss the case of division by zero in Chapter 4.

Widening of intervals from dependence can occur even when evaluating a real (i.e., degenerate interval) function with a real argument. An example of this is Rump's expression in (1.4.1). Assume we use interval arithmetic to bound rounding errors. As soon as a rounding occurs, a non-degenerate interval is introduced. If this interval is again used in the computation, dependence can cause widening of the final interval bound on the function value. As we shall see in Chapter 5 when doing Gaussian elimination to solve systems of linear equations, dependence can cause catastrophic numerical instability, which is exposed by the widening of intervals. Numerical instability can remain hidden in the result of evaluating a floating-point expression, but not in an interval expression result.

2.4.1 Dependent Interval Arithmetic Operations

We now describe a useful arithmetic procedure called *dependent subtraction.* In other publications we have called this procedure *cancellation*. To motivate it, assume we have n intervals X_i and, for each $i = 1, \cdots, n$, we want the sum of all but the i-th interval. Suppose we first compute the sum $S_1 = X_2 + \cdots + X_n$. Next, we want the sum $S_2 = X_1 + X_3 + \cdots + X_n$.

Instead of computing the entire sum S_2, we want to use the previous result. We note that

$$S_2 = S_1 + X_1 - X_2. \tag{2.4.1}$$

Therefore, we can compute S_2 by adding X_1 to S_1 and then cancelling X_2 from the result by subtraction. But $X_2 - X_2 = [\underline{X}_2 - \overline{X}_2, \overline{X}_2 - \underline{X}_2]$ which is not the (degenerate) zero interval. Therefore, we do not get a sharp result if we compute S_2 using 2.4.1 (unless X_2 is degenerate).

Instead of subtracting using (2.3.3), we use the special dependent subtraction rule which we write as

$$X \ominus Y = [a - c, b - d] \tag{2.4.2}$$

As usual, we must round outward when computing this interval. This can be implemented by defining the "interval" $[d, c]$ and using the subtraction rule (2.3.3) to compute $[a, b] - [d, c]$. Note that $[d, c]$ is not an interval when $c < d$. Alternatively, if dependent interval operations are allowed in an interval supporting compiler, an expression such as `X.DSUB.Y` can be used to represent the operation $X \ominus Y$. The Sun Microsystems Inc. Fortran and C++ compilers support dependent subtraction using the `.DSUB.` syntax, (see Walster (2000c)).

Two points to make regarding dependent subtraction are:

1. For $X \ominus A$ to be legal, X must be additively dependent on A. This is true if $X = A + B$ for some interval B.

2. Suppose $|B| << |A|$, so $X = A + B$ is dominated by A. Then rounding prevents dependent subtraction from recovering a sharp bound on B. In this case, B must be saved or directly recomputed to avoid excess width. The width (see Chapter 3) of $X \ominus A$ can be checked for this event.

See Sections 6.2 and 10.5 for example uses of dependent subtraction.

In addition to the dependent subtraction operation, each interval basic arithmetic operation (BAO) has a corresponding dependent form. For example, dependent division, denoted $\oslash$, is used to recover either A or B from $X = A \times B$. The key requirement to use a dependent operation is: The dependent operation must be the inverse of an operation already performed on the same variable or subfunction being "removed". Dependent operations cannot be performed on interval *constants*, as they cannot be dependent. In this respect the distinction between constants and variables is much more important for intervals than for points.

2.5 EXTENDED INTERVAL ARITHMETIC

In the above rules of interval arithmetic, we excluded division by an interval containing zero. Nevertheless, it is often useful to remove this restriction. The resulting arithmetic is called *extended interval arithmetic*. This arithmetic was first discussed (independently) by Hanson (1968) and Kahan (1968). An example of its utility is that it allows the derivation of an interval Newton method guaranteed to find all real zeros of a function of one variable in a given interval. See Alefeld (1968), Hansen (1978b), and Section 9.6.

In Chapter 4, we give an even more general interval arithmetic. It not only allows use of intervals with unbounded endpoints but allows for computation of expressions containing indeterminate forms such as $0 \div 0$, $0 \times \infty$, $\infty \div \infty$, $\infty - \infty$, etc. This arithmetic system is closed under all arithmetic operations and the evaluation of all arithmetic expressions, whether they are single-valued functions or multi-valued relations.

Chapter 3

FUNCTIONS OF INTERVALS

3.1 REAL FUNCTIONS OF INTERVALS

There are a number of useful real-valued functions of intervals. In this section, we list those that we use and our notation for them.

The *midpoint* or *center* of an interval $X = [a, b]$ is

$$\mathrm{m}(X) = \frac{a+b}{2}.$$

The *width* of X is

$$\mathrm{w}(X) = b - a.$$

The *magnitude* is defined to be the maximum value of $|x|$ for all $x \in X$. Thus,

$$\mathrm{mag}\,(X) = \max\,(|a|, |b|) \tag{3.1.1}$$

The magnitude is also called the absolute value by some authors. We use the notation $|X|$ to denote $\mathrm{mag}\,(X)$ in the development and analysis of our algorithms. The *mignitude* is defined to be the minimum value of $|x|$ for all $x \in X$. Thus,

$$\operatorname{mig}(X) = \begin{cases} \ \ a \text{ if } a > 0 \\ -b \text{ if } b < 0 \\ 0 \text{ otherwise} \end{cases} \tag{3.1.2}$$

The interval version of the absolute value function abs(X) can be defined in terms of the magnitude and mignitude:

$$\operatorname{abs}(X) = \operatorname{mag}(X) - \operatorname{mig}(X). \tag{3.1.3}$$

We also use the notation $|X|$ to denote abs (X) in two contexts: discussing slope expansions of nonsmooth functions in Section 7.11; and applications involving nondifferentiable functions in Chapter 18.

Various other real-valued functions of intervals have been defined and used. For a discussion of many such functions, see Ris (1975).

3.2 INTERVAL FUNCTIONS

An *interval function* is an interval-valued function of one or more interval arguments. Thus, an interval function maps the value of one or more interval arguments onto an interval. Consider a real-valued function f of real variables $x_1, \cdots, x_n$ and a corresponding interval function $f^{\mathbf{I}}$ of interval variables $X_1, \cdots, X_n$. The interval function $f^{\mathbf{I}}$ is said to be an *interval extension* of f if $f^{\mathbf{I}}(x_1, \cdots, x_n) = f(x_1, \cdots, x_n)$ for any values of the argument variables. That is, if the arguments of $f^{\mathbf{I}}$ are degenerate intervals, then $f^{\mathbf{I}}(x_1, \cdots, x_n)$ is a degenerate interval equal to $f(x_1, \cdots, x_n)$.

This definition presupposes the use of exact interval arithmetic when evaluating $f^{\mathbf{I}}$. In practice, with rounded interval arithmetic, we are only able to compute F, an interval enclosure of $f^{\mathbf{I}}$. Therefore, we have

$$f(x_1, \cdots, x_n) \in F(x_1, \cdots, x_n)$$

even when $f^{\mathbf{I}}$ is an interval extension of f.

An interval function $f^{\mathbf{I}}$ if said to be *inclusion isotonic* if $X_i \subset Y_i$ $(i = 1, \cdots, n)$ implies $f^{\mathbf{I}}(X_1, \cdots, X_n) \subset f^{\mathbf{I}}(Y_1, \cdots, Y_n)$. It follows from the defining relation (2.3.1) that finite interval arithmetic is inclusion isotonic. That is, if $\bullet$ denotes $+, -, \times$, or $\div$, then $X_i \subset Y_i$ $(i = 1, 2)$ implies $(X_1 \bullet X_2) \subset (Y_1 \bullet Y_2)$. If outward rounding is used, then interval arithmetic is inclusion isotonic even when rounding occurs. See Alefeld and Herzberger (1974, p.49) or Alefeld and Herzberger (1983, p.41).

Chapter 2 contains a brief description of the finite interval system as originally conceived by Moore (1966). Chapter 4 contains the main features of closed interval systems that eliminate many of the limitations in the finite system. In particular, for the closed system version (Theorem 4.8.14) of the fundamental theorem of interval analysis (Theorem 3.2.2), the requirement that interval functions be inclusion isotonic interval extensions is removed. To provide a baseline from which to compare the finite and closed systems, the remainder of this chapter is developed in the finite system. The algorithms for solving nonlinear systems and global optimization problems can be implemented in either system, although there are significant advantages to the closed system.

Except where stated otherwise (for example in Chapter 18), all interval enclosures are assumed to be inclusion isotonic interval extensions of real valued continuous functions. In closed systems (Chapter 4), these assumptions are unnecessary. When the closed system is used and an assumption of continuity is required, there are at least three alternatives, none of which require dealing with undefined outcomes: Constraints can be introduced to exclude points of discontinuity, expressions can be transformed to be continuous mappings, or Theorem 4.8.15 can be used to enforce continuity.

For example, whenever division by an expression E occurs, the constraints $E < 0$ or $0 < E$ can be explicitly introduced to preclude division by zero. Whenever a ratio is assumed to be continuous this has the effect of precluding division by an interval containing zero. The even better alternative is to transform the ratio into a continuous function of the same independent variables. See Walster (2003a) for a detailed analysis of how this can be done. The third alternative is to use Theorem 4.8.15 to introduce a continuity constraint.

To simplify notation, we remove the superscript "I" on f when it is unambiguous to do so and simply let $f(X_1, \cdots, X_n)$ denote an interval extension of a given real-valued function $f(x_1, \cdots, x_n)$. Any function written with interval argument(s) is implicitly an interval function. This notation is ambiguous because there is no unique interval function that is an enclosure of a given function f. (See Section 3.3). Also, we shall say that we evaluate a real function with interval arguments. What we really mean is: we evaluate some interval function that is an enclosure of f. We ask the reader's indulgence in these conventions. They simplify exposition.

There is another ambiguity in notation. It is standard practice in mathematics to use the same notation for more than one purpose. The notation $f(x)$ can denote a function in a theoretical sense without a specific expression, or it can denote one specific expression. It can also denote the numerical value of a function at a point x. Usually, a different notation is used to denote an approximate value computed, for example, using rounded arithmetic.

In the interval case, we compound this ambiguity. The notation $f(X)$ can refer to a theoretical function or one of many expressions for it. Although a different notation is usually employed in this case, $f(X)$ can also denote the interval that is the range of values of $f(x)$ for all $x \in X$. $f(X)$ can even denote a bound on the range that is understood to be unsharp because of rounding errors and dependence, however we prefer the notation $F(X)$ in this case.

It is common practice to let context imply the interpretation for a given notation. We shall usually follow this practice. However, we sometimes use special notation to distinguish cases. For example, $f(X)$ often denotes the range of the function f over the interval X, whereas or $F(X)$ denotes an interval bound on $f(X)$ that is computed by some (unspecified) finite precision numerical procedure. The width of $F(X)$ includes both the range of f over X and any numerical errors arising from rounding and dependence. Similarly, $F(x)$ usually denotes the numerically computed interval bound (including any numerical errors) on the single number $f(x)$. Occasionally we want to denote the fact that a real function f of a real value x is computed using rounded interval arithmetic to bound rounding errors. To emphasize that the result is an interval, we often append a superscript I and denote it by $f^{\mathbf{I}}(x)$.

From the fact that the interval arithmetic operators are inclusion isotonic, it follows that rational interval functions are inclusion isotonic. However, for this to be true, we must restrict a given rational function to a single form using only interval arithmetic operations. The following example from Caprani and Madsen (1980) shows that if a given rational function is evaluated in different ways for different intervals, the results might not exhibit inclusion isotonicity.

Suppose we rewrite the function $f(x) = x(1 - x)$ in the form

$$f(x) = c(1 - c) + (1 - 2c)(x - c) - (x - c)^2. \qquad (3.2.1)$$

These two forms of $f(x)$ are equivalent for an arbitrary value of c. Let $X = [0, 1]$ and $c = \mathrm{m}(X) = 0.5$. Evaluating $f(X)$ in the form in (3.2.1), we compute $f([0, 1]) = [0, 0.25]$. Now replace $X = [0, 1]$ by $X' = [0, 0.9]$. Also replace $c = 0.5$ by $c' = \mathrm{m}(X') = 0.45$. We compute $f(X') = [0, 0.2925]$. Thus, $f(X')$ is not contained in $f(X)$ even though $X' \subset X$. Inclusion isotonicity failed because we changed the functional form of f when we replaced c by c'.

In this example, we could say that the functional form is the same for each evaluation since $c = \mathrm{m}(X)$ and $c' = \mathrm{m}(X')$. However, the midpoint $\mathrm{m}(X)$ of an interval cannot be evaluated using only the interval arithmetic operations of addition, subtraction, multiplication, and division. A separate computation involving the endpoints of X is required for $\mathrm{m}(X)$.

The following Theorem shows that, for rational functions, inclusion isotonicity is easily assured.

Theorem 3.2.1 Let $F(X_1, \cdots, X_n)$ be a rational function evaluated using finite precision interval arithmetic. Assume that F is evaluated using a fixed form with a fixed sequence of operations involving only interval addition, subtraction, multiplication, and division. Then F is inclusion isotonic.

Proof of Theorem 3.2.1 is omitted. It follows easily from inclusion isotonicity of the four basic interval arithmetic operations.

Common practice makes use of monotonicity over an interval to sharply bound the range of a real function. We discuss this topic in Section 3.6. When we use monotonicity to compute our result, we do not limit expressions to those made up of the four basic interval arithmetic operations. However, it is easy to assure that computed results are inclusion isotonic.

Irrational functions can be treated as follows or using the more general closed interval system discussed in Chapter 4. Let f be a real irrational function of a real vector $\mathbf{x} = (x_1, \cdots, x_n)$. Assume that a rational approximation $r(\mathbf{x})$ is known such that $|f(\mathbf{x}) - r(\mathbf{x})| < \varepsilon$ for all $\mathbf{x}$ given that $a_i \leq x_i \leq b_i$ $(i = 1, \cdots, n)$ for some a_i and b_i. Then

$$f(x_1, \cdots, x_n) \subseteq r(x_1, \cdots, x_n) + \varepsilon[-1, +1]$$

for any points $x_i \in [a_i, b_i]$ $(i = 1, \cdots, n)$. Thus the range of f over the region with $x_i \in X_i$ $(i = 1, \cdots, n)$ can be bounded by evaluating $r(X_1, \cdots, X_n)$ using interval arithmetic and adding the error bound $[-\varepsilon, \varepsilon]$, provided:

- $X_i \subseteq [a_i, b_i]$. This is assured through the choice of X_i.
- $r(x_1, \cdots, x_n) \in r(X_1, \cdots, X_n)$ for all $x_i \in X_i$ $(i = 1, \cdots, n)$. This follows from the fundamental Theorem of interval arithmetic (Theorem 3.2.2), because $r(X_1, \cdots, X_n)$ is an inclusion isotonic interval extension of the rational function $r(x_1, \cdots, x_n)$, for all $x_i \in [a_i, b_i]$ $(i = 1, \cdots, n)$.

This "interval evaluation" of the irrational function f is inclusion isotonic if the interval evaluation of r is inclusion isotonic. The result is *not* an interval extension of f because $F(x_1, \cdots, x_n) = r(x_1, \cdots, x_n) + [-\varepsilon, \varepsilon]$ instead of $f(x_1, \cdots, x_n)$. Nevertheless

$$f(X_1, \cdots, X_n) \subseteq F(X_1, \cdots, X_n),$$

which is the critically important result of the fundamental theorem.

Interval rational function approximations of irrational functions are inclusion isotonic interval extensions provided the rational operations are those described in Theorem 3.2.1. More importantly, they bound the range of the approximated irrational function.

Unless otherwise stated, we shall assume that any interval function used in this book is an enclosure of the corresponding real function. This is true either because the considered function is itself an inclusion isotonic interval extension, or it is an inclusion isotonic interval bound on the considered function.

The range of a function can be expressed in interval form as

$$\begin{aligned} \text{range}\,(f) &= f(X_1, \cdots, X_n) \\ &= [\inf f(x_1, \cdots, x_n), \sup f(x_1, \cdots, x_n)] \end{aligned} \tag{3.2.2}$$

where the inf and sup are taken for all $x_i \in X_i$ $(i = 1, \cdots, n)$. The following theorem due to Moore (1966) shows how easy it is to bound the range of a function. It is undoubtedly the most important theorem in interval analysis. Rall (1969) aptly calls it the fundamental theorem of interval analysis. One of its far reaching consequences is that it makes possible the solution to the global optimization problem.

Theorem 3.2.2 Let $f(X_1, \cdots, X_n)$ be an infinite precision inclusion isotonic interval extension of a real function $f(x_1, \cdots, x_n)$. Then $f(X_1, \cdots, X_n)$ contains the range of values of $f(x_1, \cdots, x_n)$ for all $x_i \in X_i$ $(i = 1, \cdots, n)$.

Proof. Assume that $x_i \in X_i$ for all $i = 1, \cdots, n$. By inclusion isotonicity, $f(X_1, \cdots, X_n)$ contains $f([(x_1, x_1], \cdots, [x_n, x_n]) = f(x_1, \cdots, x_n)$ because $f(X_1, \cdots, X_n)$ is an interval extension of f. Since this is true for all $x_i \in X_i$ $(i = 1, \cdots, n)$, $f(X_1, \cdots, X_n)$ contains the range of f over these points. ■

If f is a rational function, then direct evaluation using interval arithmetic produces bounds on the set of all function values over the argument intervals. While $f(X_1, \cdots, X_n)$ contains the values of $f(x_1, \cdots, x_n)$, given $x_i \in X_i$ for $i = 1, \ldots, n$; the bounds on the set of f-values is not sharp in general. This is because of dependence (See Section 2.4). If a given endpoint of $f(X_1, \cdots, X_n)$ is exactly the correct bound for the range, we say that the endpoint is *sharp*. If both endpoints are sharp, we shall say that $f(X_1, \cdots, X_n)$ is sharp.

Using finite precision interval arithmetic and directed rounding means that in practice we are only able to compute an interval enclosure F of f. When the width of F is as small as possible for a given word length, we also call F sharp.

3.3 THE FORMS OF INTERVAL FUNCTIONS

When evaluating a function with interval arguments, the computed interval depends on the form in which the function is written. One example of this follows from the fact that interval arithmetic fails to satisfy the distributive law of algebra. Instead, as shown by Moore (1966), it satisfies the *subdistributivity law* which states that if X, Y, and Z are intervals, then

$$X(Y + Z) \subset XY + XZ. \tag{3.3.1}$$

Therefore, interval expressions are written in factored form when possible.

If we compute $X(Y + Z)$, we always obtain the exact range of the function $f(x, y, z) = x(y + z)$ (if exact interval arithmetic is used). This is because each variable occurs only once in the expression of the function, so dependence (see Section 2.4) can cause no widening of the computed intervals.

This fact holds in general. Moore (1966) notes the following. Suppose the expression for a rational function f is such that each interval variable occurs only once. Then evaluation of f using exact interval arithmetic produces the exact range of the function over the region defined by the interval variables. If $X(Y+Z)$ is computed using exact interval arithmetic, the result is sharp.

A common way to rewrite a quadratic function to remove multiple occurrences of a variable is to complete the square. For example, we can rewrite $x(x - 2)$ as $(x - 1)^2 - 1$.

Considerable effort has been expended by interval analysts in attempting to produce systematic methods with which to create an interval function that more sharply bounds the range of a given function. For example, see Ratschek and Rokne (1984), Neumaier (1989), and Rokne (1986). Such methods are important to improve the efficiency of optimization algorithms. However, we shall discuss them only briefly. The range of a function can also be bounded by expanding the function in Taylor series and bounding the remainder by interval methods. See Chapter 7.

Let $f(X_1, \cdots, X_n)$ denote the true range (expressed as an interval) of a function $f(x_1, \cdots, x_n)$ for all x_i in an interval X_i $(i = 1, \cdots, n)$. See (3.2.2). Denote the difference between the width of a bound $F(X_1, \cdots, X_n)$

on the range of f and the true width of the range by

$$E\,[f\,(X_1, \cdots, X_n)] = \mathrm{w}\,[F\,(X_1, \cdots, X_n)] - \mathrm{w}\,[f\,(X_1, \cdots, X_n)]\,.$$

Also denote

$$d = \max_{1 \le i \le n} \mathrm{w}(X_i).$$

Moore (1966) proved that

$$E[f(X_1, \cdots, X_n)] = O(d) \tag{3.3.2}$$

for a rational function f in any form. He noted that $f(x_1, \cdots, x_n)$ can be written as

$$f_c(x_1, \cdots, x_n) = f(c_1, \cdots, c_n) + g(x_1 - c_1, ...x_n - c_n)$$

where g is rational and $c_i = \mathrm{m}(X_i)$ $(i = 1, \cdots, n)$. He conjectured that

$$E[f_c(X_1, \cdots, X_n)] = O(d^2). \tag{3.3.3}$$

That is, the form f_c (called the centered form by Moore) has an excess width that is of second order in the "width" d of the box $(X_1, \cdots, X_n)$. This conjecture was proved to be true by Hansen (1969c).

Various centered forms have been derived. By using expansions of appropriate orders, it is possible to derive centered forms f_c for which

$$E[f_c(X_1, \cdots, X_n)] = O(d^k) \tag{3.3.4}$$

for arbitrary $k = 2, 3, \cdots$. For a thorough discussion, see Ratschek and Rokne (1984).

Note that equations (3.3.2), (3.3.3), and (3.3.4) are asymptotic statements. They are useful expressions only when d is small. Consider two different formulations f_1 and f_2 for the same function. Suppose

$E[f_1(X)] = O(d)$ and $E[f_2(X)] = O(d^2)$. If d is not small, it is quite possible that $f_1(X)$ is narrower than $f_2(X)$.

An unsolved problem in interval analysis is how to know how small the width of an interval must be for the centered form to always yield a result at least as sharp as can be computed using the "original" form of the function.

Centered forms are very useful and give good results when $\mathrm{w}(X)$ is reasonably small. However, the extra effort to use them is generally not warranted when $\mathrm{w}(X)$ is not small. An apparent drawback is that they are not inclusion isotonic unless c is fixed. See the example in Section 3.2. However, the fixed c requirement no longer exists. As mentioned earlier, in the closed interval systems described in Chapter 4, an explicit inclusion isotonicity assumption is not required.

In this book, we are particularly interested in the evaluation of interval functions when finding zeros of systems of nonlinear functions and when finding minima of functions. For this purpose, we often use Taylor expansions which we discuss in Chapter 7. Centered forms and Taylor expansions yield increasingly sharp bounds as the interval bounding a solution converges toward a point.

3.4 SPLITTING INTERVALS

In whatever form we express a function, when we evaluate it with interval arguments, we tend to get narrower bounds on its range as argument widths decrease. One way to compute narrower range-bounds is to subdivide an interval argument and compute the union of results for each subinterval. If we subdivide a given interval X into subintervals X_i $(i = 1, ...m)$ so that

$$X = \bigcup_{i=1}^{m} X_i,$$

we have

$$\begin{aligned} f(X) &\subseteq \bigcup_{i=1}^{m} f^{\mathbf{I}}(X_i) \\ &\subseteq f^{\mathbf{I}}(X). \end{aligned}$$

We have used a superscript "I" on f to emphasize that because of dependence, the computed value of $f^{\mathrm{I}}(X_i)$ is generally not sharp, even if infinite precision interval arithmetic is used. Each computed interval $f^{\mathrm{I}}(X_i)$ tends to suffer less from dependence than the computed bound $f^{\mathrm{I}}(X)$ on $f(X)$. Therefore, the union is generally sharper. The same inclusions hold if rounded finite precision interval arithmetic is used, in which case

$$\begin{aligned} f(X) &\subseteq \bigcup_{i=1}^{m} F(X_i) \\ &\subseteq F(X). \end{aligned}$$

However, interval splitting can be costly in the multidimensional case. If we divide in half each of n interval variables on which a function depends, we must evaluate the function with 2^n sets of different arguments. The amount of extra computing can be prohibitive. However, if we subdivide only a few interval variables, we do not generate as many cases. Therefore, we shall only selectively use this approach later when solving systems of nonlinear equations and optimization problems; that is, when all other less costly alternatives fail to make sufficiently good progress. We discuss other aspects of splitting in Sections 11.8 and 12.13.

Even when n is relatively large, only a few dimensions of a solution set can be graphically displayed. This is typically done by fixing values of variables in the remaining dimensions. This fact makes it practical to subdivide displayed boxes so "slices" of the solution region can be accurately displayed.

3.5 ENDPOINT ANALYSIS

Generally, the computed interval value of a function is not sharp, even when exact interval arithmetic is used. This is because of dependence (see Section 2.4). An exception occurs when each variable appears only once in the expression used to compute the function. In such cases the discussion in Section 3.3 is not relevant because it deals only with how much the computed interval value of a function asymptotically exceeds its range.

Unfortunately, we can not know, in general, whether a computed result is sharp or not. If we wish to know, we can do so by using an *endpoint analysis* which was described by Hansen (1997b). It requires extra (non-numerical) computing which increases as a linear function of the amount of computing necessary to evaluate the function. We briefly describe endpoint analysis in this section.

The rules for interval arithmetic (see Section 2.3) are expressed in terms of the endpoints of the intervals involved. For example, suppose we add an interval $X = [\underline{X}, \overline{X}]$ to itself and subtract X from itself. We obtain

$$X + X = [2\underline{X}, 2\overline{X}] \text{ and}$$
$$X - X = [\underline{X} - \overline{X}, \overline{X} - \underline{X}].$$

Note that the lower endpoint of $X + X$ is a function of $\underline{X}$ only and the upper endpoint of $X + X$ is a function of $\overline{X}$ only. However, each endpoint of $X - X$ is a function of both $\underline{X}$ and $\overline{X}$.

As we shall see, these facts tell us that not only is $X + X$ sharp and $X - X$ is not sharp, but also that the function $f(x) = x + x$ is a monotonically increasing function of x. (Remember, however, that $X \ominus X = [0, 0]$ is sharp.)

Suppose we compute an interval extension $F(X_1, \cdots, X_n)$ of a rational function $f(x_1, \cdots, x_n)$. Following the rules of interval arithmetic, the lower endpoint $\underline{F}$ and the upper endpoint $\overline{F}$ of $F(X_1, \cdots, X_n)$ are computed as functions of the endpoints of X_i $(i = 1, \cdots, n)$. However, we do not denote this fact in our notation because the form of dependence can change when the values of the X_i change. For example, let $f(x) = x(x - 3)$. If $X = [1, 2]$ then $\underline{F} = \underline{X}\left(\underline{X} - 3\right)$. However, if $X = [2, 5]$ then $\underline{F} = \overline{X}\left(\underline{X} - 3\right)$.

An endpoint of the range of a function is a value of the function at a point. Therefore, if $\underline{F}$ or $\overline{F}$ is computed using more than one endpoint of a given variable, then it cannot be sharp. A kind of converse is expressed in the following Theorem from Hansen (1997b).

Theorem 3.5.1 If $\underline{F}$ is computed in terms of a single endpoint of each of the variables on which F depends, and if the same is true of $\overline{F}$, then $\underline{F}$ and $\overline{F}$ are sharp.

Note that this theorem permits both $\underline{F}$ and $\overline{F}$ to be a function of the same endpoint of one or more of the variables. For example, suppose $F(X_1, X_2) = X_1X_2$. If $X_1 > 0$ and $0 \in X_2$, then $F(X_1, X_2) = [\overline{X}_1\underline{X}_2, \overline{X}_1\overline{X}_2]$. In this case, both endpoints of $F(X_1, X_2)$ are functions of the same endpoint of X_1. Yet, $F(X)$ is sharp.

Of course, both endpoints of F cannot be functions of the same endpoint of *all* the components of X. If they were, the two endpoints of F would be the same and F could not contain the range of $f(\mathbf{x})$ for all $\mathbf{x} \in \mathbf{x}^\mathbf{I}$.

Hansen (1997) describes four applications of endpoint analysis. In addition, it can also be used to verify monotonicity. Theorem 3.5.2 below indicates how this can be done. To state the theorem, we need the following definition:

Definition 3.5.2 A product of two intervals is a "$P_0(X_i)$ product" if both intervals contain zero as an interior point and one is a function of X_i, but the other is not.

Theorem 3.5.3 Assume that when $F(X)$ is computed, no $P_0(X_i)$ product occurs (for some $i = 1, \cdots, n$). If $\underline{F}$ is a function of $\underline{X}_i$ but not $\overline{X}_i$ and $\overline{F}$ is a function of $\overline{X}_i$ but not $\underline{X}_i$, then $f(\mathbf{x})$ is a monotonically increasing function of x_i for $\mathbf{x} \in \mathbf{x}^\mathbf{I}$. If $\underline{F}$ is a function of $\overline{X}_i$ but not $\underline{X}_i$ and $\overline{F}$ is a function of $\underline{X}_i$ but not $\overline{X}_i$, then $f(\mathbf{x})$ is a monotonically decreasing function of x_i for $\mathbf{x} \in \mathbf{x}^\mathbf{I}$.

Monotonicity of a function over an interval can be proved by verifying that the value of a derivative does not contain zero as an interior point. However, lack of sharpness (due to dependence) of a computed derivative can sometimes prevent verification while endpoint analysis succeeds in doing so. The reverse can also occur since dependence can cause loss of sharpness when the function is evaluated (while doing endpoint analysis).

In practice, both endpoint analysis and evaluation of derivatives can be used to check for monotonicity.

In the next section, we show how to constructively use monotonicity when computing an interval function. However, if endpoint analysis reveals that a function is monotonic, we already have computed the function; and no extra sharpness can be obtained by using verified monotonicity.

Suppose we try to prove monotonicity by evaluating a derivative over an interval. Dependence might cause the computed interval value of the derivative to contain zero so that monotonicity is not verified. No such failure can occur using endpoint analysis.

3.6 MONOTONIC FUNCTIONS

Sharp bounds on the range of a function can be computed if the function is monotonic. Suppose $f(x)$ is a monotonically nondecreasing function in an interval $X = [a, b]$. Then $f(X) = [f(a), f(b)]$.

In practice, when rounding is present, we evaluate $f(a)$ and $f(b)$ using outward rounding and obtain $[\underline{F}(a), \overline{F}(a)]$ and $[\underline{F}(b), \overline{F}(b)]$, respectively. Then

$$f(X) \subset [\underline{F}(a), \overline{F}(b)].$$

Thus, we compute bounds on $f(X)$ even in the presence of dependence. With exact arithmetic, we obtain the exact range of a monotonic function over an interval.

Using rounded interval arithmetic of sufficiently high precision yields bounds on $f(X) = \{f(x) \mid x \in X\}$ that are as accurate as desired when f is monotonic. Without additional analysis, sharp bounds are rarely computed for non-monotonic functions. Dependence (see Section 2.4) generally precludes sharp bounds even if exact interval arithmetic is used.

However, arbitrarily sharp bounds are generally obtained for solutions to problems such as nonlinear equations and optimization because the intervals involved are successively narrowed during the progression of the algorithms. Therefore, as guaranteed by (3.3.2), bounds of decreasing width are obtained on the range of a function as the widths of function arguments decrease. See the algorithms described in later chapters.

Monotonicity can also be used to more narrowly bound the range of functions of more than one variable. For simplicity, let us assume that $f(x_1, \cdots, x_n)$ is a monotonically nondecreasing function of $x_1, \cdots, x_m$ for some $m < n$, but is not monotonic in $x_{m+1}, \cdots, x_n$ for $x_i \in X_i$ $(i = 1, \cdots, n)$. Denote $X_i = [a_i, b_i]$ for $i = 1, \cdots, n$. Evaluate

$$f(a_1, \cdots, a_m, X_{m+1}, \cdots, X_n)$$

obtaining $[f_1, f_2]$ and evaluate $f(b_1, \cdots, b_m, X_{m+1}, \cdots, X_n)$ obtaining $[f_3, f_4]$. Then $f(x_1, \cdots, x_n) \in [f_1, f_4]$ for all $x_i \in X_i$ $(i = 1, \cdots, n)$.

Note that we can check for monotonicity by evaluating partial derivatives. For example, if

$$\frac{\partial}{\partial x_i} f(X_1, \cdots, X_n) \geq 0,$$

then f is a monotonically nondecreasing function of x_i for all $x_i \in X_i$ $(i = 1, \cdots, n)$. When we evaluate a derivative, we generally don't obtain its range exactly. Again, this is because of dependence. However, the computed interval contains the range. Therefore, if the computed interval does not contain zero, neither does the range of the partial derivative. Therefore, monotonicity can be proved even in the presence of both rounding and dependence.

Monotonicity can often be used after subdividing an interval into parts so that a given function is monotonic in one or more subinterval. Monotonicity can also be used even when the function is not monotonic in a given interval provided the behavior of the function is sufficiently well known.

For example, $\sin(X)$ can be evaluated over any given interval X by evaluating the function at the endpoints only. We need only check whether the interval contains a point or points where $\sin(x)$ is known to have an extremum.

To illustrate, suppose $X = [1, 2]$. We find $\sin(1) < \sin(2)$. We observe that $\frac{\pi}{2} \in X$, and we know that $\sin(x)$ is a maximum at $\frac{\pi}{2}$ and that $\sin(\frac{\pi}{2}) = 1$. Therefore, we obtain $\sin([1, 2]) = [\sin(1), 1]$. We can evaluate $\sin(1)$ as accurately as we like and thus compute $\sin([1, 2])$ as accurately as we like.

3.7 PRACTICAL EVALUATION OF INTERVAL FUNCTIONS

In this section, we consider how an interval function can be evaluated in practice.

When doing scientific computing with real numbers, we approximate irrational functions by rational functions. We generally have a list of subroutines available to generate (approximate) values of trigonometric functions, the exponential function, the logarithm function, etc.

We need a similar list of subroutines in the interval case. Consider an interval $X = [a, b]$ and suppose we want a subroutine to compute a sharp single-precision bound on $\exp(X)$. Since the exponential function is monotonic, we know that $\exp(X) = [\exp(a), \exp(b)]$. One way to compute the desired sharp interval is as follows.

Evaluate $\exp(a)$ using a standard double-precision noninterval arithmetic subroutine which is guaranteed to produce a value of $\exp(a)$ accurate to more than single-precision. Round the result down to a double-precision number guaranteed to be a lower bound for $\exp(a)$. Then round this intermediate double-precision result to the largest single-precision computer number not exceeding the double precision value of $\exp(a)$. This is the desired left endpoint of $\exp(X)$.

Next, evaluate $\exp(b)$ using the double precision subroutine. Use the same two step procedure applied to $\exp(a)$ to compute a single-precision result. Now, however, rounding is upward. This produces the desired right endpoint of $\exp(X)$.

We now illustrate another alternative with a simple example. Suppose we want to compute $\arctan(X)$ and will be satisfied with about three decimal digit accuracy. We can take advantage of the monotonicity of $\arctan(x)$. However, let us use an approximation found in Fike (1968).

The polynomial

$$p(x) = x(0.079331x^4 - 0.288679x^2 + 0.995354) \tag{3.7.1}$$

approximates $\arctan(X)$ for $x \in [-1, 1]$. The error is bounded by

$$\max |p(x) - \arctan(x)| < .00061 \tag{3.7.2}$$

From this approximation and bound, we have

$$\arctan(X) \in p(X) + 0.00061 \times [-1, 1] \tag{3.7.3}$$

for $X \in [-1, 1]$ where $p(X)$ is obtained by replacing x with X in (3.7.1).

Our subroutine for computing $\arctan(X)$ computes the right member of (3.7.3) and returns the resulting interval as the "value" of $\arctan(X)$.

When evaluating $p(X)$ in practice, we do not use the form given by (3.7.1). Instead, we complete the square and write

$$p(X) = X[0.079331(X^2 - 1.81946)^2 + 0.732734] \tag{3.7.4}$$

to reduce the number of occurrences of the variable X and thus produce sharper results. See Section 3.3.

We make rounding errors when deriving (3.7.4) from (3.7.1). We must assure that (3.7.2) remains true when p is written in the form (3.7.4). An easy way to do so is to compute the coefficients in (3.7.4) from (3.7.1) as intervals. This illustrates the extra care that must be taken in writing interval subroutines to bound the range of irrational functions.

As an illustrative example, let us evaluate the function

$$f(x) = x^2 + \arctan\left[\frac{\sin(x)}{x}\right]$$

with x replaced by the interval $X = [1, 2]$. To do so, let us define and evaluate the following functions:

$$f_1(x) = \sin(x),$$

$$f_2(x) = \frac{f_1(x)}{x},$$

$$f_3(x) = \arctan[f_2(x)].$$

Then

$$f(x) = x^2 + f_3(x).$$

We first evaluate $f_1(X)$ by calling a subroutine that computes $\sin(X)$. This subroutine can use the monotonicity properties of $\sin(x)$ as is done above for $\exp(x)$. Alternatively, it can use an appropriate rational approximation (with error bound) as is done above for $\arctan(x)$. As shown in Section 3.6, the desired interval $\sin([1, 2]) = [\sin(1), 1]$. An interval containing $[\sin(1), 1]$ is returned. Thus, suppose the interval $[0.84147, 1]$ is returned and becomes the "value" of $f_1(X)$.

To compute $f_2(X)$, we simply divide our value of $f_1(X)$ by X using interval arithmetic. We obtain (using six decimals)

$$f_2(X) = [0.420735, 1].$$

Next, we call the subroutine to evaluate $\arctan[f_2(X)]$. If the subroutine uses (3.7.3), we obtain

$$f_3(X) = [0.303230, 0.958296].$$

For our final step, we add X^2 to $f_3(X)$ using interval arithmetic. We obtain

$$f(X) = [1.30323, 4.95830].$$

This result is not sharp. We lost sharpness because of dependence. Also, (3.7.3) is only accurate to about three decimals. See (3.7.2). The range of f over X rounded outward to six decimals is $[1.69952, 4.42672]$. However, the final result contains the range as promised by the fundamental Theorem 3.2.2 of interval analysis.

Note that we used outwardly rounded interval arithmetic throughout the computation. This assures that the computed result contains the range of $f(x)$ despite the presence of rounding errors.

Note, also, that computing the interval $f(x)$ involves essentially the same operations as evaluating $f(x)$ using real arithmetic. In each case, we use available subroutines to compute irrational functions. These routines are written in much the same way in either the interval or noninterval case.

3.8 THICK AND THIN FUNCTIONS

We can distinguish two types of interval functions. Suppose we evaluate a function using infinite precision interval arithmetic. Assume the function involves only degenerate interval parameters[1]. If the argument of the interval function is a degenerate interval, the "value" of the function is also degenerate if the function is an interval extension of its arguments and parameters. We call such a function "thin".

Suppose, however, that the function involves an interval parameter of nonzero width. If we evaluate such a function using exact interval arithmetic over a degenerate interval, the value of the function is a nonzero-width interval. For example, if we evaluate $f(X) = X + [1, 2]$ over the degenerate interval $X = [0, 0]$, we obtain the nondegenerate result $[1, 2]$. We call such a function "thick".

In most of this book, we discuss functions as if they are thin. We usually consider functions that arise as noninterval functions and this, of course, implies that they do not contain interval parameters. For example, our primary concern is to find zeros of functions and to find global minima of functions that arise in a noninterval context. We merely use intervals to solve such problems.

The algorithms that we discuss also serve to solve problems involving thick functions. In this case, a solution is generally an extended set of points instead of a single point. Unless otherwise stated, we assume for simplicity that functions are thin or "nearly thin" so that extended solutions are of reasonably narrow width. Special methods involving "very thick" functions are discussed separately in various sections. For example, see Chapter 17.

In many practical computations, however, input data are inexact and are input as intervals to bound their true value. See for example:

`http://physics.nist.gov/cuu/Constants`

for internationally recommended values of fundamental physical constants. Even if data are exact, they might not be machine-representable. If round-

[1]A function's parameter is a fixed constant that therefore cannot depend on any of the function's arguments.

ing is necessary because real or interval data are input, they must be rounded outward. In addition, (outward) rounding as a result of interval computing causes a thin function to become thick. Therefore, we sometimes distinguish between exact and rounded interval arithmetic.

In Chapter 17 and in various other places in this book, we explicitly consider perturbed or thick functions. Results from the analysis of problems with narrow width parameters and subdivision can be used to compute and display narrow interval bounds on solution sets to problems with large numbers of arbitrarily thick parameters. Thus concentrating attention on problems with no parameters or thin parameters is not limiting in practice.

Chapter 4

CLOSED INTERVAL SYSTEMS

4.1 INTRODUCTION

Interval algorithms for solving nonlinear systems of equations and global optimization problems can be more general, simple and efficient if the interval arithmetic system used to implement them is closed. A closed system (or a system that is closed[1] under a set of arithmetic operations) is one that contains no undefined operator-operand combinations, such as division by zero in the real number system. How closed systems provide these benefits is described in Section 4.2.

Closed interval systems are a step in the evolution of mathematical systems starting with the positive integers. Each step in this evolution was motivated by a requirement to perform operations that are undefined in the earlier system. For example, in the system of positive integers the difference $3 - 3$ is undefined. This fact motivated adding zero to the system of positive integers. Similarly, fractions motivated rational numbers and square (and other) roots of positive rational numbers motivated irrational numbers. Roots of negative real numbers motivated complex numbers.

[1]The word "closed" is used in analysis and topology. In analysis, a closed system produces only members of the system. An interval $[a, b]$ that includes its endpoints is topologically closed. An open interval (a, b) does not include the endpoints a or b.

Throughout this evolution, division by zero, operations on infinite values such as $+\infty - (+\infty)$ or $\frac{+\infty}{+\infty}$, and other indeterminate forms have remained undefined. The real interval system was built up from sets of real numbers and operations on them. As a consequence, existing limitations in the real point system were inherited by the real interval system.

There is an alternative development path. Because intervals are *sets* of real numbers, it is possible to first consider building systems of sets of *extended* real numbers including infinite values. These systems can be closed. Intervals can then be constructed from the convex hulls of sets of extended reals. If this is done, the resulting interval systems are closed. This chapter describes one such closed interval system. It is consistent with both the existing real interval system and with basic arithmetic operations (BAOs) on all possible operands including those that lead to indeterminate forms.

Different closed cset-based systems all have these properties, but they differ in the width of computed intervals and the complexity of their definitions. The system described herein is a compromise between the simplest system currently implemented in Sun Microsystems' Fortran and C++ compilers, and more sophisticated and narrow systems described in Walster, Pryce, and Hansen (2002).

Because closed arithmetic systems contain no undefined operator--operand combinations, their implementation on a computer can never produce "exceptional events". This means that all arithmetic operations on intervals containing infinities and zero can be defined. Consequently, computer hardware and software can be simpler because exceptional event handling is unnecessary, even for arguments outside a function's domain of definition.

New analysis is required for arithmetic on infinite intervals to be consistent with arithmetic on finite intervals and with the axioms of real analysis. Let $\mathbb{R} = \{z \mid -\infty < z < +\infty\}$ denote the set of real numbers. The closed interval system described herein includes all extended intervals with endpoints in the set of real numbers, but augmented with plus and minus infinity. That is, the set of extended real numbers is $\mathbb{R}^* = \mathbb{R} \cup \{-\infty, +\infty\}$, which is also the topologically closed interval $[-\infty, +\infty]$. See Section 4.8.1. This chapter contains the main results of this new analysis along with the principles that motivate its form. Mathematical details and other possible systems are presented in referenced papers.

The idea of using sets as the basis for interval analysis is discussed in Section 4.3. The critical concepts of the *containment constraint* and the *containment set* are introduced in Sections 4.4 and 4.5. Arithmetic over the extended real numbers is presented in Section 4.6. Closed interval systems are defined in Section 4.7. The fundamental theorem of interval analysis is extended to the closed interval system in Section 4.8.

4.2 CLOSED SYSTEM BENEFITS

A number of characteristics can be used to evaluate different interval systems and the algorithms they produce. These include: generality, speed, and interval width. Using these characteristics, this section describes positive features of closed interval systems.

4.2.1 Generality

If algorithms are more general, they are usually simpler and require fewer special-case branches. The same is true for interval systems: Generality is good.

Algorithms that use the closed interval system accept more inputs and are therefore more *general* because:

- Defensive code is not needed to avoid exceptional events, such as division by intervals containing zero.
- Arguments that are partially or totally outside a function's natural domain of definition are accepted.
- Intervals with infinite endpoints are included in the arithmetic system and therefore can be used in algorithms.

- Interval expressions[2] need not be inclusion isotonic interval extensions for them to be covered by the extended fundamental theorem of interval analysis (Theorem 4.8.14 on page 77).

- Interval expressions can be used to bound the range of both single-valued functions (Chapter 3) and multi-valued relations.

A possible objection to closed interval systems is that their benefits come at the cost of being unable to detect when an expression either is not defined, or is discontinuous. This objection is groundless. When there is a requirement for a function to be defined over a given interval, this can be enforced by introducing explicit constraints that delete points outside the function's natural domain. For example the appearance of $\ln(g(x))$ in a composition imposes the implicit constraint that $g(x) \geq 0$.

When the continuity assumption is required, three approaches are possible:

1. Introduce explicit constraints that enforce the assumption by deleting points of discontinuity;

2. Work instead with a continuous version of the discontinuous function; or

3. Use Theorem 4.8.15 to test for continuity over any given interval, or impose a continuity constraint.

The first option can be easily implemented if the given function is available for analysis. The second alternative is described in Walster (2003a) and can be used when discontinuities arise at branch cuts. The third alternative can be automated and does not require knowing the given function. However, an enclosure of the function's derivative is required. These methods can be used to eliminate the requirement for defensive code to guarantee algorithm assumptions are satisfied when using an exception-free closed interval system. Also see Section 4.8.5.

[2]Throughout this book, the term "expression" refers to any sequence of arithmetic operations, and/or compositions of single-valued functions and multivalued relations. In this chapter, the term "function" is reserved for single-valued mappings of points onto points. The concept of an interval function is discussed in Chapter 3.

4.2.2 Speed and Width

A good interval system facilitates writing algorithms that quickly produce narrow-width results. Algorithm speed generally *increases* whenever the width of computed intermediate intervals is reduced. This can even be true when local speed is significantly *decreased*. For example, Corliss (1995) developed an interval ordinary differential equation integration algorithm in which speed *decreased* by a factor of 200 to compute narrow intervals using methods described in Corliss and Rall (1991) pages 195–212. Ultimately, overall algorithm speed was *increased* by a factor of 2 because of the above decrease in interval width. Thus, it is generally good interval algorithm development practice to optimize the performance of relatively large algorithms rather than to focus on the runtime performance of small code fragments. The reason is that the relationship between overall interval algorithm speed and intermediate interval result-width can be complicated.

All other things being equal, "more is better" when it comes to speed. However, there can be "too much of a good thing" with narrow width. The quest for narrow width or speed must *never* come at the cost of the failure to contain the set of required results. This set is called the *containment set* of a given expression. Failure to enclose an expression's containment set in a computed interval is a *containment failure*. Interval systems must not produce a containment failure by violating the *containment constraint*. Interval algorithms can be slow and produce wide intervals, but they must always satisfy the containment constraint.

Interestingly, the containment set of some expressions is not always clearly defined in finite interval systems. Closed interval systems precisely define containment sets, thereby making clear what is required to produce a sharp (narrow as possible) interval result. The containment-set concept is so fundamental to computing with intervals that the collection of containment-set results is known as containment-set theory. The term "containment set" is abbreviated "cset", so the study of csets becomes "cset theory".

4.3 THE SET FOUNDATION FOR CLOSED INTERVAL SYSTEMS

Finite interval arithmetic, as introduced by Moore (1966) and briefly described in Chapters 2 and 3, is based on intervals of real numbers. As such, finite interval arithmetic inherits the assumptions, axioms, and limitations of real analysis. Intervals have a dual identity both as numeric entities and as sets. Recognizing this duality is not new. See Moore (1966). What *is* new is the recognition that because intervals are *sets* of numbers, the fundamental analysis of intervals can be based on sets as opposed to individual real numbers. See Walster (1996). It is the set-theoretic interval foundation that enables limitations of real and finite interval analysis to be removed.

The set-based development of interval arithmetic, together with additional motivation and justification, are described in the following sections. The motivation for the set-based development is to produce a closed interval system in which there are no undefined operand-operator combinations. As a consequence, expressions can be algebraically transformed even by a compiler without regard to the consequences of division by zero or other induced indeterminate forms. The objective of such transformations is narrow width and speed, but the transformed expression's cset must enclose the original expression's cset. Two expressions with the same cset are said to be *cset-equivalent* expressions. See page 53 in Section 4.5.2.

4.4 THE CONTAINMENT CONSTRAINT

For all finite intervals, X and Y, interval arithmetic operations must satisfy:

$$X \bullet Y \supseteq \{x \bullet y \mid x \in X,\ y \in Y\}, \tag{4.4.1}$$

where $\bullet \in \{+,\ -,\ \times,\ \div\}$. Because division by zero is undefined for real numbers, division by intervals containing zero is undefined in the finite interval system (See Chapter 2).

Bold letters $\mathbf{x}$ and $\mathbf{x^I}$ are used to denote respectively, an n-dimensional point $(x_1, \cdots, x_n)^T$ and an interval vector or box $(X_1, \cdots, X_n)^T$.

To be valid, any interval numerical evaluation F of a real function f of n-variables must satisfy:

$$F\left(\mathbf{x}^{\mathrm{I}}\right) \supseteq \left\{f(\mathbf{x}) \mid \mathbf{x} \in \mathbf{x}^{\mathrm{I}}\right\}. \tag{4.4.2}$$

Clearly, (4.4.1) is a special case of (4.4.2).

The fundamental requirement in (4.4.2) of any interval arithmetic system is referred to as the *containment constraint* of interval arithmetic. Satisfying this constraint is necessary to produce rigorous bounds, the key to numerical proofs, such as those produced by the algorithms in this book. Because this constraint is so obvious and simple, it was never even named until the possibility of extending it to include infinite intervals became evident.

4.4.1 The Finite Interval Containment Constraint

In finite interval arithmetic, the containment constraint requires computed intervals to contain the set of all possible real results, as defined by the right hand side of (4.4.2). To close interval systems, the containment constraint concept must be extended to include point operations and functions that are normally undefined in real arithmetic.

4.4.2 The (Extended) Containment Constraint

For points within the domain of real BAOs and functions, the existing containment constraint must not be changed. Rather, the existing definition must be extended to include otherwise undefined interval operation-operand and function-argument combinations.

The approach taken is to *almost*, but not quite, beg the question of the extended containment constraint's definition. Rather than focusing only on the containment constraint of any single expression, consideration is broadened to include the set of *all possible* containment constraints that result from *any* algebraic transformation of the given expression. By making this one simple change, the door is opened to the extended containment constraint definition. When algebraically transformed, the result of any expression evaluation must not cause the transformed expression's containment constraint to be violated. This is the key that unlocks the door to the needed generalization.

The *containment set* of an expression is now introduced and extended.

4.5 THE (EXTENDED) CONTAINMENT SET

When evaluated over a box $\mathbf{x}^{\mathbf{I}} = (X_1, \cdots, X_n)^T$ in the finite interval system, the containment set that a function f must contain is given by the right-hand side of (4.4.2) and is conveniently denoted by $f\left(\mathbf{x}^{\mathbf{I}}\right)$. That is

$$f\left(\mathbf{x}^{\mathbf{I}}\right) = \left\{f(\mathbf{x}) \mid \mathbf{x} \in \mathbf{x}^{\mathbf{I}}\right\}. \tag{4.5.1}$$

The containment set (or cset) of *any* expression (whether a single-valued function or a multi-valued relation) is the set of possible values the given expression can take on. Thus, the cset of a given expression is the union of the csets of all possible algebraic transformations of the given expression. For example, suppose $g(x)$ is an algebraic transformation of the given expression $f(x)$. Then evaluating g must not violate f's containment constraint. The motivation is to permit f to be replaced by g. See, for example, f and g in Section 4.5.2. An equivalent way of stating this is that algebraically equivalent expressions must have the same csets. A trivial way to guarantee this is to let the cset of every expression be the interval $[-\infty, +\infty]$, but this is not useful. Therefore an additional restriction on the cset of an expression is required: An expression's cset must be the *smallest possible* set that satisfies the given expression's containment constraint and the containment constraint of all the expression's possible algebraic transformations.

4.5.1 Historical Context

Various authors (including Hanson (1968) and Kahan (1968), and more recently Hickey, Ju and Van Emden (1999)) have proposed generalizations of finite interval arithmetic in unsuccessful attempts to extend interval arithmetic. Others (including Ratschek and Rokne (1988), while developing conventions to support infinite starting box-widths in global optimization) have come close to discovering how to close interval systems. The paper by Walster, Pryce, and Hansen (2002), is the first to contain a mathematically consistent[3] system that holds for all extended real expression arguments. Extending the containment constraint and cset concepts is required.

4.5.2 A Simple Example: $\frac{1}{0}$

A simple example illustrates how the cset and containment constraint concepts permit otherwise undefined operations to be given consistent csets. Consider the problem of defining the cset of $\frac{1}{0}$. In the real analysis of points, division by zero is undefined. A reason is that the resulting value turns out to be a set, not a point.

Temporarily, ignore the fact that the cset is not defined until Section 4.5.5. Consider instead:

$$f(x) = \frac{x}{x+1}. \tag{4.5.2}$$

If $x = 0$, $f(0) = \frac{0}{1} = 0$. However, suppose x is replaced by an interval. To eliminate excess interval width caused by dependence as described in Section 2.4 (see also Section 3.3), an astute interval analyst will choose to compute an interval enclosure of $f(x)$ using the algebraically equivalent expression:

$$g(x) = \frac{1}{1+\frac{1}{x}}. \tag{4.5.3}$$

An interval extension of (4.5.3) often produces narrower results than does an interval extension of (4.5.2). The reason is because the dependence from multiple occurrences of the variable x is avoided in (4.5.3). See Section

[3]An arithmetic system is *consistent* if there are no contradictory operand-operator combinations. For example, if $\frac{1}{0}$ were defined to be 1, then $g(0) = \dfrac{1}{2}$ in (4.5.3). However, this contradicts the fact that $f(0) = 0$ in (4.5.2).

2.4. However, when $x = 0$, (4.5.3) is undefined because $\frac{1}{0}$ is undefined. Therefore, whenever the interval X contains zero, the interval expression

$$g(X) = \frac{1}{1 + \frac{1}{X}} \tag{4.5.4}$$

is also undefined in the finite interval system.

To make matters worse, both $f(-1)$ and $g(-1)$ are also undefined. Therefore, both their interval extensions are undefined for any interval argument X that contains -1. One way to motivate the closed interval system is to determine how $h(x) = \frac{1}{x}$ can be defined so that $g(x) = f(x)$ whenever $f(x)$ is defined, and also to consistently define $g(x)$ and $f(x)$ when $x = -1$.

Let $\mathcal{D}_f$ and $\mathcal{D}_g$ denote the natural domains (or simply the domains) of the expressions f and g — that is, $\mathcal{D}_f$ and $\mathcal{D}_g$ are, respectively, the sets of arguments for which f and g are defined and finite. The following facts are known:

1. Because their domains are different, the functions f and g are different. Using $\mathbb{R}$ to denote the set of real numbers $\{z \mid -\infty < z < +\infty\}$, and $M \backslash N$ to denote the complement of the set N in M (or all the points of M not in N):

$$\begin{aligned}\mathcal{D}_f &= \{-\infty < x < -1\} \cup \{-1 < x < +\infty\} \\ &= \mathbb{R} \backslash \{-1\},\end{aligned}$$

   and

$$\begin{aligned}\mathcal{D}_g &= \{-\infty < x < -1\} \cup \{-1 < x < 0\} \cup \{0 < x < +\infty\} \\ &= \mathbb{R} \backslash \{-1, 0\}.\end{aligned}$$

2. Let x_0 denote a specific value of the variable, x. As long as $x_0 \in \mathcal{D}_f \cap \mathcal{D}_g$ (so both $f(x_0)$ and $g(x_0)$ are defined) then $f(x_0) = g(x_0)$.

3. "Algebra" includes the set of transformations through which g and f can be derived from each other with $f(x) = g(x)$ for all $x \in \mathcal{D}_f \cap \mathcal{D}_g$.

In the present context, extending the definition of an expression's cset provides a consistent definition for the cset of both f and g that applies for all values of x, whether finite or infinite. With this definition, f and g are seen to have identical csets for all extended real values of x. When any two expressions have identical csets for all possible arguments, the expressions (in this case f and g) are said to be *cset-equivalent*. This is yet another important new construct. Cset-equivalent expressions can be interchanged without fear of violating their common containment constraint. The choice of which cset-equivalent expression to compute can be freely made on the basis of width and speed. This is the principle practical consequence of, and motivation for, cset theory.

4.5.3 Cset Notation

Denote the cset of the expression f evaluated at all the points x_0 in the set S :

$$\text{cset}\,(f, S)\,.$$

The zero subscript on x is used to connote the fact that an expression's cset depends on the specific value(s) of the expression's argument(s).

For convenience and without loss of generality, the following development uses scalar points and sets, rather than n-dimensional vectors. When the set S is the singleton set $\{x_0\}$ and $x_0 \in \mathcal{D}_f$, then

$$\text{cset}\,(f, \{x_0\}) = \{f\,(x_0)\}\,.$$

The notation $\{f\,(x_0)\}$ denotes the value of the function f, but viewed as a singleton set. Whether x_0 is inside or outside the domain of f, it is notationally convenient to permit "$f\,(x_0)$" to be understood to mean $\text{cset}\,(f, \{x_0\})$. Otherwise, a plethora of braces "$\{\cdots\}$" is needed to distinguish points from singleton sets. Therefore, for example, when $x_0 = 0$ in (4.5.3), it is understood that

$$g\,(0) = \text{cset}\,(g, \{0\})\,.$$

There is an additional bit of notation that is necessary to explicitly convey how csets of expressions with non-singleton set arguments are defined: When the set S is not a singleton set, then $f(S) = \text{cset}(f, \{S\})$ is understood to be the set:

$$\bigcup_{z_0 \in \{S\}} f(z_0) = \bigcup_{z_0 \in S} \text{cset}(f, \{z_0\}). \tag{4.5.5}$$

The reason for using the union to define $f(S)$ in (4.5.5) is that $f(z_0)$ is now a set. Therefore, $\{f(z_0) \mid z_0 \in S\}$ is properly interpreted as a set of sets, rather than $\text{cset}(f, S)$. The expression in (4.5.5) for $f(S)$ is exactly analogous to the definition of $f\left(\mathbf{x^I}\right)$ in (4.5.1) when $\mathbf{x^I}$ is a nondegenerate interval vector. Note the difference in notation between the set S of scalars and the interval vector $\mathbf{x^I}$. See Section 2.2.

Finally, when it is important to distinguish between a variable x and a value x_0 of it, the zero subscript notation is used. Otherwise, let it be understood that the point arguments of expressions in csets are specific given values and not simply the names of variables.

4.5.4 The Containment Set of $\frac{1}{0}$

With the above notation conventions, the value of the containment set (cset) of $\frac{1}{0}$ is now addressed. Continue to use the expression definitions: $f(x) = \frac{x}{x+1}$ and $g(x) = \frac{1}{1+\frac{1}{x}}$ in (4.5.2) and (4.5.3). The question to be answered is: What is the smallest set of values (if any) that can be assigned to $h(x) = \frac{1}{x}$ when $x_0 = 0$ so that $g(x_0) = 0$. In fact the only way for $g(x_0)$ to equal zero is if $h(x_0) = -\infty$ or $+\infty$. Therefore $\{-\infty, +\infty\}$ is the set of all possible values that the cset of $h(x_0)$ must include when $x_0 = 0$. Moreover, when $x_0 = 0$, if the cset of $h(x_0)$ includes any value other than $-\infty$ or $+\infty$, then $g(x_0) \neq 0$. Therefore, a mathematically consistent way for $g(0)$ to equal zero is if

$$\frac{1}{0} = \{-\infty, +\infty\}. \tag{4.5.6}$$

In this case, if

$$h(x) = \frac{1}{x}, \tag{4.5.7}$$

then because $h(0)$ is understood to mean $\operatorname{cset}(h, \{0\})$, when $h(x)$ is otherwise undefined,

$$h(0) = \{-\infty, +\infty\}.$$

Then and only then (in the current cset system) is

$$\begin{aligned} g(0) &= \frac{1}{1+\{-\infty\}} \cup \frac{1}{1+\{+\infty\}} \\ &= 0. \end{aligned}$$

Having informally established (4.5.6), both $f(-1)$ and $g(-1)$ are seen to be $\{-\infty, +\infty\}$ as well. That $g(-1) = \{-\infty, +\infty\}$ can also be seen by writing g in terms of h as defined in (4.5.7):

$$g(x) = h(1 + h(x)).$$

Similar arguments to that given above can be developed to find the cset of any indeterminate form in any closed cset-based system.

4.6 ARITHMETIC OVER THE EXTENDED REAL NUMBERS

For a rigorous development of csets, see Walster, Pryce, and Hansen (2002). With their development and/or analyses similar to that of $\frac{1}{0}$ above, csets for the basic arithmetic operations (BAOs) displayed in Tables 4.1 through 4.4 have been derived and proved to be consistent in $\mathbb{R}^*$; where $\mathbb{R}$ denotes the real numbers $\{z \mid -\infty < z < +\infty\}$, and the set of extended real numbers $\mathbb{R}^*$ is the set of real numbers to which the elements $\{-\infty\}$ and $\{+\infty\}$ are adjoined. This compact[4] set is the same as the interval $[-\infty, +\infty]$ and is also denoted:

$$\mathbb{R}^* = \mathbb{R} \cup \{-\infty, +\infty\}.$$

In each of tables 4.1 through 4.4, the upper left corner cell contains the given operation in terms of specific values x_0 and y_0 of the variables x and y. The first column and first row in each table contains the values of x_0 and y_0 for which different cset values are produced.

The value of $\frac{1}{0}$ discussed above is found in Table 4.4.

4.6.1 Empty Sets and Intervals

To close the set-based arithmetic system, it is necessary to define arithmetic operations on empty sets. The logically consistent definition is for an arithmetic operation to produce the empty set if one or both operands is empty. This result is consistent with (4.5.5) and is also consistent with the implicit constraint that the arguments of any expression E must be within or at least on the boundary of E's domain of definition, $\mathcal{D}_E$. This implicit constraint can be made explicit to limit values of independent variables in different expressions. For example, if $X = [-\infty, +\infty]$ and $Y = \sqrt{X-1}$, then $0 \leq X - 1$ or $X \geq 1$. If in addition $Z = \sqrt{1-X}$, then $0 \leq 1 - X$ or $X \leq 1$. Therefore, the only way both Y and Z can be defined is to impose the constraint on X that $X = [1, 1]$, in which case: $Y = Z = [0, 0]$.

More generally, for any expression E that appears as the argument of a function whose domain is a proper subset of $\mathbb{R}^*$, an implicit constraint on the values of E is imposed. For example, with $E = \frac{x}{x+1,}$, the expression $\sqrt{E}$ imposes the constraint on x that $x \in [-\infty, -1] \cup [0, +\infty]$. In algorithms such as those in this book that delete impossible solutions, such constraints can be useful when made explicit. See Chapter 10. The following is a

[4]Compactness is necessary for all needed sequences of $\mathbb{R}^*$ elements to have limits. Compactness of $\mathbb{R}^*$ is established in Walster et al (2002) by noting that the interval $[-\infty, +\infty]$ can be mapped onto the compact interval $[-1, +1]$ using, for example, the hyperbolic tangent function.

$x_0 + y_0$	$-\infty$	$y_0 \in \mathbb{R}$	$+\infty$
$-\infty$	$-\infty$	$-\infty$	$\mathbb{R}^*$
$x_0 \in \mathbb{R}$	$-\infty$	$x_0 + y_0$	$+\infty$
$+\infty$	$\mathbb{R}^*$	$+\infty$	$+\infty$

Table 4.1: Addition over the extended real numbers.

$x_0 - y_0$	$-\infty$	$y_0 \in \mathbb{R}$	$+\infty$
$-\infty$	$\mathbb{R}^*$	$-\infty$	$-\infty$
$x_0 \in \mathbb{R}$	$+\infty$	$x_0 - y_0$	$-\infty$
$+\infty$	$+\infty$	$+\infty$	$\mathbb{R}^*$

Table 4.2: Subtraction over the extended real numbers.

$x_0 \times y_0$	$-\infty$	$y_0 \in -(\mathbb{R}^*)^2$	0	$y_0 \in (\mathbb{R}^*)^2$	$+\infty$
$-\infty$	$+\infty$	$+\infty$	$\mathbb{R}^*$	$-\infty$	$-\infty$
$x_0 \in -(\mathbb{R}^*)^2$	$+\infty$	$x_0 \times y_0$	0	$x_0 \times y_0$	$-\infty$
0	$\mathbb{R}^*$	0	0	0	$\mathbb{R}^*$
$x_0 \in (\mathbb{R}^*)^2$	$-\infty$	$x_0 \times y_0$	0	$x_0 \times y_0$	$+\infty$
$+\infty$	$-\infty$	$-\infty$	$\mathbb{R}^*$	$+\infty$	$+\infty$

Table 4.3: Multiplication over the extended real numbers.

$x_0 \div y_0$	$-\infty$	$y_0 \in -(\mathbb{R}^*)^2$	0	$y_0 \in (\mathbb{R}^*)^2$	$+\infty$
$-\infty$	$[0, +\infty]$	$+\infty$	$\{-\infty, +\infty\}$	$-\infty$	$[-\infty, 0]$
$x_0 \in \mathbb{R} - \{0\}$	0	$x_0 \div y_0$	$\{-\infty, +\infty\}$	$x_0 \div y_0$	0
0	0	0	$\mathbb{R}^*$	0	0
$+\infty$	$[-\infty, 0]$	$-\infty$	$\{-\infty, +\infty\}$	$+\infty$	$[0, +\infty]$

Table 4.4: Division over the extended real numbers.

Note that $-(\mathbb{R}^*)^2 = [-\infty, 0]^2$ and $(\mathbb{R}^*)^2 = [0, \infty]^2$.

source of implicit constraints that *must* be made explicit. If the interval version of Newton's method is used to find roots of a function f over an interval X, then f must be defined and continuous over X. See Chapter 9.

Another possibility (to explicitly impose a continuity constraint) is mentioned in Section 4.2.1. For example, because zero is a point of discontinuity for the expression $\frac{1}{x}$, if this expression is used in a context where continuity is assumed, then the implicit constraints ($x < 0$ and $x > 0$) on x can be made explicit. See also Section 4.8.5.

4.6.2 Cset-Equivalent Expressions

Two expressions that have the same csets for all possible expression arguments are said to be *cset-equivalent*. Two cset-equivalent expressions can be interchanged without fear of violating their containment constraints. For example,

$$f(x) = \frac{x}{x+1}$$

and

$$g(x) = \frac{1}{1+\frac{1}{x}},$$

can be interchanged without loss of containment because f and g are cset-equivalent expressions. In fact, any cset enclosure of g can be used to bound values of f. This example illustrates an important general result: A necessary condition for an analyst or a compiler to substitute one expression g for another f, is that g is a cset-enclosure of f. See Definition 4.8.12.

4.7 CLOSED INTERVAL SYSTEMS

A closed system is one in which there are no undefined operator-operand or function-argument combinations. All that is necessary to construct a closed interval system is to guarantee that *any* interval expression evaluation produces an enclosure of the expression's cset. If the resulting interval is the hull of the expression's cset, then the computed interval is *sharp*. For example, hull $(\{-\infty, +\infty\}) = [-\infty, +\infty]$ is the narrowest *interior* interval containing $\frac{1}{0}$. In practice (for example when implementing the interval Newton algorithm in Section 9.2), the fact that $\frac{1}{0} = [-\infty, -\infty] \cup [+\infty, +\infty]$ is used, whether or not the interval system supports exterior intervals[5].

More generally, when evaluated over the interval box

$$\mathbf{x}^{\mathbf{I}} = (X_1, \cdots, X_n),$$

the usual notation can be used for the set that an enclosure F of the expression f must contain:

$$F\left(\mathbf{x}^{\mathbf{I}}\right) \supseteq \text{hull}\left(f\left(\mathbf{x}^{\mathbf{I}}\right)\right).$$

Upper case F denotes an interval enclosure, for example, as can be obtained by evaluating the expression f using finite precision interval arithmetic. Thus, F is not uniquely defined, except to the extent that it must satisfy the containment constraint of interval arithmetic. The notation distinguishes between: hull $\left(f\left(\mathbf{x}^{\mathbf{I}}\right)\right)$, which is the sharp interval enclosure of f's cset over the box, $\mathbf{x}^{\mathbf{I}}$; and $F\left(\mathbf{x}^{\mathbf{I}}\right)$, which is only *an* enclosure of hull $\left(f\left(\mathbf{x}^{\mathbf{I}}\right)\right)$, and which therefore might be less than sharp.

Adopting the mathematically standard zero-subscript notation (e.g.: x_0) to denote a specific, but unspecified value of the variable x, let it be understood that $F(\mathbf{x}_0)$, rather than $F([\mathbf{x}_0, \mathbf{x}_0])$ or $F(\{\mathbf{x}_0\})$, can be used to denote an interval expression evaluated at the degenerate interval $[\mathbf{x}_0, \mathbf{x}_0]$, or equivalently at the singleton set $\{\mathbf{x}_0\}$.

4.7.1 Closed Interval Algorithm Operations

Different closed interval implementations of the same cset system are possible to construct. They can produce different width enclosures of interval expression csets, but they all must produce enclosures of:

[5]An exterior interval is the union of two semi-infinite intervals, such as $[-\infty, a] \cup [b, +\infty]$ with $a < b$. The notion of exterior intervals was first conceived by Kahan (1968).
Alefeld (1968) was the first to use division by intervals containing zero to extend the interval version of Newton's method to find *all* the roots of nonlinear functions.

1. BAO csets, that (in the present system) are given in Tables 4.1 through 4.4; and

2. the csets of expressions evaluated over sets as given in equation (4.5.5).

All the definitions of finite interval arithmetic in Section 2.3 carry over to closed systems because the cset of a defined function is simply the defined function's value, but viewed as a singleton set. The cases that require additional analysis are those that implicitly or explicitly use undefined point operations, such as $(\pm\infty) - (\pm\infty)$, $\frac{1}{0}$, $0 \times (\pm\infty)$, and $\frac{\pm\infty}{\pm\infty}$.

It is a tempting mistake to conclude that each of the following three examples is non-negative, because there is no obvious way to produce a negative result:

$$[0, 1] \times [2, +\infty] = [0, +\infty], \tag{4.7.1a}$$

$$\frac{[1, 2]}{[0, 1]} = [1, +\infty] \tag{4.7.1b}$$

and

$$\frac{[0, 1]}{[0, 1]} = [0, +\infty]. \tag{4.7.1c}$$

Combining the rules of interval arithmetic in Section 2.3 with csets from Tables 4.1 through 4.4 produces the correct results, which are:

$$[0, 1] \times [2, +\infty] = [-\infty, +\infty], \tag{4.7.2a}$$

$$\frac{[1, 2]}{[0, 1]} = \{-\infty\} \cup [1, +\infty] \tag{4.7.2b}$$

and

$$\frac{[0, 1]}{[0, 1]} = [-\infty, +\infty]. \tag{4.7.2c}$$

In two out of the above three cases, it works to compute interval endpoints using the formulas in Section 2.3 and csets from Tables 4.1 through 4.4. For example, $[0, 1] \times [2, +\infty] = \mathbb{R}^*$, because $0 \times (\pm\infty) = \mathbb{R}^*$ from Table 4.3. The exception is division by an interval with zero as an interior point, such as happens with

$$\frac{[1, 1]}{[-1, 1]}. \tag{4.7.3}$$

However, by splitting the denominator interval into the union of two interior intervals, both of which have zero as an endpoint, the entries in Table 4.4 and the formulas in Section 2.3 interact to produce the sharp result. From (4.5.5) and the entries in Table 4.4, Section 4.6 page 81

$$\begin{aligned}
&\text{cset}\,(\div, (1, [-1, 1])) \\
&\qquad = \text{cset}\,(\div, (1, [-1, 0])) \cup \text{cset}\,(\div, (1, [0, 1])) && \text{(4.7.4a)} \\
&\qquad = ([-\infty, -1] \cup \{+\infty\}) \cup (\{-\infty\} \cup [1, +\infty]) && \text{(4.7.4b)} \\
&\qquad = [-\infty, -1] \cup [1, +\infty]\,. && \text{(4.7.4c)}
\end{aligned}$$

From BAO csets with extended interval operands, different closed interval arithmetic systems are possible to implement. They differ in how narrowly BAO csets are enclosed. For example, if exterior intervals are not explicitly supported, then the result of computing (4.7.3) is the hull of (4.7.4c), which is $\mathbb{R}^*$. Walster (2000) proposed a "Simple" closed interval system that has been fully implemented in Sun Microsystem's Fortran 95 and C++ compilers. See Walster (2002).

Interval algorithms, including those in the remainder of this book, can be realized in any closed interval system using the extended BAOs displayed in Tables 4.1 through 4.4, or their equivalent in other different cset-based systems. To accomplish this, the fundamental theorem of interval analysis (Theorem 3.2.2) is generalized in Theorem 4.8.13 to include csets and rational expressions. Finally, using the closed interval system, this more general fundamental theorem is further extended in Theorem 4.8.14 to include irrational composite expressions.

With compiler support for interval data types in a closed interval system, the mechanics of writing code to implement interval algorithms is *easier* in some important respects than writing noninterval programs. See, for example Walster (2000b, 2000c) and Walster and Chiriaev (2000). Generalizing and extending the fundamental theorem completes the theoretical foundation for the compiler support of interval data types in closed interval systems.

4.8 EXTENDED FUNDAMENTAL THEOREM

So far in the above development, closed interval systems are limited to the BAOs. As described in Chapter 3, repeated application of the fundamental theorem to compositions of finite BAOs proves that over the argument intervals, computed rational functions using interval arithmetic produce bounds on the value of the underlying rational function. Closing interval systems requires the following extensions to the fundamental theorem:

- extending function values to expression csets,
- extending csets to their enclosures, and
- removing the requirement that computed expressions be inclusion isotonic interval extensions.

At the same time these extensions are made, the fundamental theorem is further extended to any expressions, whether a single-valued continuous function with finite domain that is a subset of the real numbers, or a multi-valued relation that is always defined. Examples of single-valued continuous functions include the log or square root functions. A simple example of a multi-valued relation is a single-valued function to which is added a nondegenerate interval constant.

4.8.1 Containment Sets and Topological Closures

In this and the following two subsections, the explicit notation, $\text{cset}(f, x_0)$ is used. This is necessary to clearly distinguish between a function and its cset. The implicit notation in which $f(x)$ represents its cset is resumed in Section 4.8.4 for the statement and proof of Theorem 4.8.14 on page 77. The notation used throughout remaining chapters is described in Section 4.9.

When an expression is a function with a non-empty domain (the converse case is treated in Section 4.8.2), the function's *topological closure* is the function's cset if the function is undefined at the given point, say x_0. Note that a function's closure is not the same concept as a closed system. The closure of a set S, denoted $\overline{S}$ in this chapter[6], is the union of the set S and all its accumulation points.

Definition 4.8.1 The closure of the expression f, evaluated at the point x_0 is denoted $\overline{f}(x_0)$, and is defined to be the set of all possible limits of sequences $\{y_j\}$, where $y_j = f(x_j)$ and $\{x_j\}$ is any sequence converging to x_0 from within f's domain $\mathcal{D}_f$. Symbolically,

$$\overline{f}(x_0) = \left\{ z \,\middle|\, \begin{array}{l} z = \lim_{j\to\infty} y_j, \\ y_j \in f(x_j)\ ^{7}, \text{ and} \\ \lim_{j\to\infty} x_j = x_0 \end{array} \right\}. \tag{4.8.1}$$

The closure of f is always defined, but might be the empty set. Namely, for y_j to exist, $f(x_j)$ must be non-empty, which, by definition of f's domain, means that $x_j \in \mathcal{D}_f$. Therefore, if $x_0 \notin \overline{\mathcal{D}_f}$, or if $\mathcal{D}_f = \emptyset$, there are no sequences $\{x_j\}$, hence no y_j, and $\overline{f}(x_0)$ is empty. The domain of $\overline{f}$ is the set of argument values x_0 for which $\overline{f}(x_0) \neq \emptyset$, which is $\overline{\mathcal{D}_f}$, the closure of f's domain. Using the compactness of $\mathbb{R}^*$, Walster, Pryce, and Hansen (2002) proved that $\overline{f}(x_0)$ is never empty if $\mathcal{D}_f \neq \emptyset$ and $\mathbf{x}_0 \in \overline{\mathcal{D}_f}$.

[6]In this chapter the notation $\overline{S}$ denotes the closure of the set S. Readers should not confuse this commonly used mathematical notation with the interval notation $X = [\underline{X}, \overline{X}]$. The former is *only* used in this chapter. The latter is used in the remaining chapters of this book to denote the *infimum* and *supremum* of the interval X.

[7]Note that the $f(x_j)$ are single valued, so for $x_j \in \mathcal{D}_f$, this sequence is well defined.

Therefore,

$$\mathcal{D}_{\overline{f}} = \overline{\mathcal{D}}_f . \tag{4.8.2}$$

Definition 4.8.1 imposes no restriction on the point x_0 other than $x_0 \in \mathbb{R}^*$.

When x_0 is on the boundary of f's domain (or $x_0 \in \overline{D}_f \backslash D_f$), the closure of a function satisfies all the requirements imposed on the function's cset. This result is also proved in Walster, Pryce, and Hansen (2002).

The examples below illustrate features of cset theory at points in Tables 4.1 through 4.4 where the normal value is *not* defined. All sequences are indexed by $n \to +\infty$.

Example 4.8.2

The expression $(+\infty) + (-\infty)$ is the set of all $\lim (x_n + y_n)$ where both $x_n \to +\infty$, and $y_n \to -\infty$. Any finite limit a can be achieved (e.g. $x_n = a + n$, $y_n = -n$) as well as $\pm\infty$ (e.g. $x_n = \pm 2n$, $y_n = \mp n$), so

$$(+\infty) + (-\infty) = \mathbb{R}^*. \tag{4.8.3}$$

Example 4.8.3

The expression $0 \div 0$ is the set of all $\lim (x_n \div y_n)$ where $y_n < 0$, or $y_n > 0$, and both $x_n \to 0$ and $y_n \to 0$. Any finite limit a can be achieved (e.g. $x_n = \frac{a}{n}$, $y_n = \frac{1}{n}$) as well as $\pm\infty$ (e.g. $x_n = \pm\frac{1}{n}$, $y_n = \pm\frac{1}{n^2}$), so

$$0 \div 0 = [-\infty, +\infty] . \tag{4.8.4}$$

Example 4.8.4

Let $a > 0$ be finite. Then $a \div 0$ is the set of all $\lim \left(\frac{a}{\pm x_n}\right)$, where $x_n > 0$ and $x_n \to 0$. If $x_n = \frac{1}{n}$, then $\lim \left(\frac{a}{\pm x_n}\right) = \lim (\pm an)$, so for any finite $a > 0$

$$a \div 0 = \{-\infty, +\infty\}. \tag{4.8.5}$$

This result implies, for instance, that $\frac{1}{[0,1]} = \{-\infty\} \cup [1, +\infty]$.

4.8.2 Multi-Valued Expressions

The natural domain of an expression that is never single valued is the empty set. As a simple example, the expression

$$f(x) = \frac{1}{x - x} \tag{4.8.6}$$

is undefined for all values of x. Another example is:

$$f(x) = \frac{x}{[-1, 1]}, \tag{4.8.7}$$

the cset of which is $[-\infty, -|x|] \cup [|x|, +\infty]$. In these cases there is no single function, the closure of which is the cset of such an expression. For all expressions to have csets, whether the expressions are single-valued functions or multi-valued relations, additional analysis is required. In Walster, Pryce, and Hansen (2002), the concept of *constant variable expressions* (CVEs) is introduced. Briefly, $\frac{1}{x-x}$ is a CVE that can be replaced by the expression $\frac{1}{y_0}$ given that $y_0 = 0$. Thus, the cset of $\frac{1}{x-x}$ is unconditionally equal to $\{-\infty, +\infty\}$. Note that CVEs are independent. This means that each occurrence of the same CVE must be replaced by a different variable, or by a zero-subscripted symbolic constant.

Defining the cset of $\frac{x}{[-1,1]}$ can be accomplished using composite expressions and the union of all possible function closures in (4.8.8). This same device can be used for CVEs and is described now.

Let a composite expression be given, such as

$$f(\mathbf{x} \mid \mathbf{c}) = g(h(\mathbf{x} \mid \mathbf{c}), \mathbf{x} \mid \mathbf{c}),$$

in which the elements of the vector $\mathbf{c}$ are fixed constants. Further, assume that $\mathcal{D}_{g(y,\mathbf{x}|\mathbf{c})} \neq \emptyset$, but assume there are no values of $\mathbf{x}$ for which the expression $h(\mathbf{x} \mid \mathbf{c})$ is a single-valued function. For example, with $f(x) = \frac{x}{[-1,1]}$, let $g(y, x) = \frac{x}{y}$, and $h(x \mid [-1, 1]) = [-1, 1]$. Therefore the natural domain of h given $\mathbf{c}$ is the empty set. Additionally, $\mathcal{D}_{f(\mathbf{x}|\mathbf{c})} = \emptyset$ because $\mathcal{D}_{h(\mathbf{x}|\mathbf{c})} = \emptyset$. Let $H(\mathbf{x}_0, \mathbf{c})$ denote a set of values that depends on the value $\mathbf{x}_0$ of $\mathbf{x}$ and the constant vector $\mathbf{c}$. $H(\mathbf{x}_0, \mathbf{c})$ might be simply

$\overline{h}(\mathbf{x}_0,\mathbf{c})$, or it might be a set that is determined in some other way. The important point is that $H(\mathbf{x}_0 \mid \mathbf{c})$ need not be empty because it is the closure of a function h with an empty natural domain. In the present case $H(x_0 \mid [-1, 1]) = [-1, 1]$. When g's domain is not empty, the only way for f's domain to be empty is if h's domain is empty. In this case, the cset of f is simply:

$$\text{cset}\,(f, (\mathbf{x}_0 \mid \mathbf{c})) = \bigcup_{h_0 \in H(\mathbf{x}_0,\mathbf{c})} \overline{g}\,(h_0, \mathbf{x} \mid \mathbf{c}) \tag{4.8.8}$$

where $H(\mathbf{x}_0, \mathbf{c}) = \text{cset}\,(h, (\mathbf{x}_0 \mid \mathbf{c}))$. Combining this case with the usual case in which $\mathcal{D}_{f(\mathbf{x}|\mathbf{c})} \neq \emptyset$ yields the four cases in (4.8.10) to be distinguished if

$$f(\mathbf{x} \mid \mathbf{c}) = g(h(\mathbf{x} \mid \mathbf{c}), \mathbf{x} \mid \mathbf{c}): \tag{4.8.9}$$

$$\begin{array}{ll}
\text{if } \mathcal{D}_f \neq \emptyset \text{ and } \left\{ \begin{array}{ll} \mathbf{x}_0 \in \mathcal{D}_f & \textbf{Case 1} \\ \mathbf{x}_0 \notin \mathcal{D}_f & \textbf{Case 2} \end{array} \right. \\
\text{if } \mathcal{D}_f = \emptyset \text{ and } \left\{ \begin{array}{ll} \mathbf{x}_0 \in \overline{\mathcal{D}}_h \text{ and } \overline{\mathcal{D}}_g \cap \text{cset}\,(h, (\mathbf{x}_0 \mid \mathbf{c})) \neq \emptyset & \textbf{Case 3} \\ \mathbf{x}_0 \notin \overline{\mathcal{D}}_h \text{ or } \overline{\mathcal{D}}_g \cap \text{cset}\,(h, (\mathbf{x}_0 \mid \mathbf{c})) = \emptyset & \textbf{Case 4} \end{array} \right.
\end{array} \tag{4.8.10}$$

Then, the cset of (4.8.9) can be written as follows:

$$\text{cset}\,(f, (\mathbf{x}_0 \mid \mathbf{c})) = \begin{cases} f(\mathbf{x}_0 \mid \mathbf{c}), & \text{if } \textbf{Case 1} \\ \overline{f}(\mathbf{x}_0 \mid \mathbf{c}), & \text{if } \textbf{Case 2} \\ \displaystyle\bigcup_{y_0 \in \text{cset}(h,(\mathbf{x}_0|\mathbf{c}))} \text{cset}\,(g, (y_0, \mathbf{x}_0 \mid \mathbf{c})), & \text{if } \textbf{Case 3} \\ \emptyset, & \text{if } \textbf{Case 4} \end{cases} \tag{4.8.11}$$

The cset of the example in (4.8.7) can be represented in a number of equivalent ways:

$$\begin{aligned}
\text{cset}\,(f\,(x_0)) &= \bigcup_{h_0 \in [-1,1]} \frac{x_0}{h_0} && \text{(4.8.12a)}\\
&= \frac{x_0}{[-1,\,1]} && \text{(4.8.12b)}\\
&= x_0 \times ([-\infty,\,-1] \cup [+1,\,+\infty]) && \text{(4.8.12c)}\\
&= [-\infty,\,-|x_0|] \cup [|x_0|\,,\,+\infty]\,. && \text{(4.8.12d)}
\end{aligned}$$

To avoid a continuing plethora of zero subscripts, let it be understood that given argument values of expression csets are always given points, even though they are written without zero subscripts. Only where it is particularly important to emphasize the distinction will cset (f, x) be written as cset (f, x_0) .

The following example uses simple special cases of (4.8.11) to illustrate the important distinction between constants and variables in defining expression's cset. The examples all use only scalar (not vector) functions h, so $f\,(x, y, z) = g\,(h\,(x, y)\,, z)$.

Example 4.8.5

The following cases (a through d) use variations on a common theme: The expression

$$\left(\frac{x}{y}\right) \times (xy) \qquad \text{(4.8.13a)}$$

can be simplified to

$$x^2 \qquad \text{(4.8.13b)}$$

because the two occurrences of the variables x and y in equation (4.8.13a) are dependent. However, the expression

$$\left(\frac{x}{[1,\,2]}\right) \times (x \times [1,\,2]) \qquad \text{(4.8.14a)}$$

can be simplified no more than to

$$x^2 \times \left[\tfrac{1}{2}, 2\right] \tag{4.8.14b}$$

because the two occurrences of the interval [1, 2] in (4.8.14a) are independent.

Let $f(x, y, z) = g(h(x, y), z)$, with $h(x, y) = \frac{x}{y}$, and $g(x, y) = xy$, or equivalently, with $h(x, y) = xy$ and $g(x, y) = \frac{x}{y}$. Therefore, $f(x, y, z) = \left(\frac{x}{y}\right) z$, or equivalently $\frac{(xz)}{y}$. The cset of f, depends on whether arguments are *independent constants* or *dependent variables* (see Section 2.4.1), together with whether any of the arguments of f repeat the same variable. In the following examples, the zero subscript, as in x_0 is used to denote a constant.

(a) If $g_1(x, y) = f(x, y, x) = \frac{x^2}{y}$, not $\frac{x \times x}{y}$ because multiple occurrences of x are dependent, then

$$\operatorname{cset}(g_1, (x, y)) = \operatorname{cset}(f, (x, y, x)) \tag{4.8.15a}$$

$$= \frac{x^2}{y}. \tag{4.8.15b}$$

(b) If $g_2(x, z) = f(x, x, z) = z$, then

$$\operatorname{cset}(g_2, (x, z)) = \operatorname{cset}(f, (x, x, z)) \tag{4.8.16a}$$

$$= z. \tag{4.8.16b}$$

(c) If $h_2(z \mid x_0) = g_2(x_0, z) = f(x_0, x_0, z)$, then the cset of $h_2(z \mid x_0)$ depends on the value of the constant x_0. In particular,

$$\begin{aligned}
&\text{cset}\,(h_2, (z \mid x_0)) \\
&\quad = \text{cset}\,(g_2, (x_0, z)) && (4.8.17\text{a}) \\
&\quad = \text{cset}\,(f, (x_0, x_0, z)) && (4.8.17\text{b}) \\
&\quad = \begin{cases} z, & \text{if } x_0 \in \mathbb{R}\setminus\{0\} \\ [0, +\infty] \text{ for all } z, & \text{if } x_0 \in \{-\infty, +\infty\} \\ [-\infty, +\infty] \text{ for all } z, & \text{if } x_0 = 0 \end{cases} && (4.8.17\text{c})
\end{aligned}$$

Note the difference between the above results and

$$\begin{aligned}
\text{cset}\,(g_2, (x, z)) &= \text{cset}\,(f, (x, x, z)) && (4.8.18\text{a}) \\
&= z, \text{ for all } z. && (4.8.18\text{b})
\end{aligned}$$

(d) If $h_3\,(x \mid y_0) = g_3\,(x, y_0) = f\,(x, y_0, y_0)$, then the cset of $h_3\,(x \mid y_0)$ depends on the value of the constant $y_{0.}$ in precisely the same way that $h_2\,(z \mid x_0)$ depends on the value of the constant x_0.

Because variables can be dependent and constants cannot, variables and constants must be carefully distinguished. For example, without additional information it is impossible to tell: whether $f\,(x, 0, 0) = x$ because $y = 0$ in $f\,(x, y, y)$; or whether $f\,(x, 0, 0) = [-\infty, +\infty]$ because both $y = 0$ and $z = 0$ in $f\,(x, y, z)$.

The above distinctions are important to make both in mathematical notation and in computer languages that support intervals, such as Fortran C, C++, or Java™. Otherwise, to guarantee containment, when there is *any* expression ambiguity, the widest interval result must be returned.

The required distinctions can be made by introducing a computer language variable dependence attribute: A computer variable or symbolic constant can be explicitly declared to have either the mathematical dependence or independence property. A computer variable or symbolic constant with the dependence attribute has the properties of a mathematical variable —

namely dependence. A computer variable or symbolic constant with the independence attribute has the mathematical properties of a constant — namely, independence. Literal constants in computer languages represent mathematical constants and are therefore unconditionally independent. In mathematical notation, a zero subscript, as in x_0, is used to identify a symbolic constant as contrasted with a mathematical variable.

Henceforth, by letting the function h be a vector $\mathbf{h}$, the cset of a composite expression f is defined:

Definition 4.8.6 (Containment-set) When $\mathbf{x}$, and $\mathbf{y}$ appear as expression arguments in a cset expression, let it be understood that they denote specific values $\mathbf{x}_0$, and $\mathbf{y}_0$ of the corresponding vector variables. Given the composition

$$f(\mathbf{x} \mid \mathbf{c}) = g(\mathbf{h}(\mathbf{x} \mid \mathbf{c}), \mathbf{x} \mid \mathbf{c}), \tag{4.8.19}$$

and the following case definitions

$$\begin{array}{ll} \text{if } \mathcal{D}_f \neq \emptyset \text{ and} \begin{cases} \mathbf{x}_0 \in \mathcal{D}_f & \textbf{Case 1} \\ \mathbf{x}_0 \notin \mathcal{D}_f & \textbf{Case 2} \end{cases} \\ \text{if } \mathcal{D}_f = \emptyset \text{ and} \begin{cases} \mathbf{x}_0 \in \overline{\mathcal{D}}_{\mathbf{h}} \text{ and } \overline{\mathcal{D}}_g \cap \text{cset}(\mathbf{h}, (\mathbf{x}_0 \mid \mathbf{c})) \neq \emptyset & \textbf{Case 3} \\ \mathbf{x}_0 \notin \overline{\mathcal{D}}_{\mathbf{h}} \text{ or } \overline{\mathcal{D}}_g \cap \text{cset}(\mathbf{h}, (\mathbf{x}_0 \mid \mathbf{c})) = \emptyset & \textbf{Case 4} \end{cases} \end{array} \tag{4.8.20}$$

then the containment set of the composition in (4.8.19) is defined to be:

$$\text{cset}(f, (\mathbf{x} \mid \mathbf{c})) = \begin{cases} f(\mathbf{x} \mid \mathbf{c}), & \text{if } \textbf{Case 1} \\ \overline{f}(\mathbf{x} \mid \mathbf{c}), & \text{if } \textbf{Case 2} \\ \bigcup\limits_{\mathbf{y} \in \text{cset}(\mathbf{h}, (\mathbf{x}\mid\mathbf{c}))} \text{cset}(g, (\mathbf{y}, \mathbf{x} \mid \mathbf{c})), & \text{if } \textbf{Case 3} \\ \emptyset, & \text{if } \textbf{Case 4} \end{cases} \tag{4.8.21}$$

The following example demonstrates the use of vector functions $\mathbf{h}$ in compositions and how different compositions can have different csets.

Example 4.8.7

Let $f(x) = g(\mathbf{h}(x))$ with $g(u, v) = u + v$ and

$$\mathbf{h}(x) = \left(h_1(x), h_2(x)\right) \tag{4.8.22a}$$

$$= \left(\log(x-2), \log(-x-2)\right). \tag{4.8.22b}$$

The cset of f is the empty set because the closure $\overline{\mathcal{D}}_h$ of $\mathbf{h}$'s domain is empty. In fact, if the domains of h_1 and h_2 are used to impose constraints on x, then x must be empty. Such *domain constraints* can be used like any constraints to reduce the set of possible solutions to a given problem. See Section 4.8.5; and Chapters 6, 11, and 16.

If $f(x)$ is defined using

$$f(x) = \log((x-2)(-x-2)) \tag{4.8.23}$$

instead of the composition in (4.8.22), f's cset is not empty if $x \in [-2, 2]$. This is the domain constraint on x imposed using this alternative definition. With the domain constraint $x \in [-2, 2]$, the composition

$$f(x) = \log((2-x)(x+2)) \tag{4.8.24}$$

$$= \log(2-x) + \log(x+2) \tag{4.8.25}$$

can also be used.

If the expression f in Definition 4.8.6 is evaluated over some set $\mathbf{X}$ or box $\mathbf{x}^{\mathbf{I}}$ of values, then as usual over the set $\mathbf{X}$

$$\text{cset}(f, \mathbf{X} \mid \mathbf{c}) = \bigcup_{\mathbf{x} \in \mathbf{X}} \text{cset}(f, \mathbf{x} \mid \mathbf{c}), \tag{4.8.26a}$$

and over the box $\mathbf{x}^{\mathbf{I}}$,

$$\text{cset}\left(f, \mathbf{x}^{\mathbf{I}} \mid \mathbf{c}\right) = \bigcup_{\mathbf{x}\in\mathbf{x}^{\mathbf{I}}} \text{cset}\left(f, \mathbf{x} \mid \mathbf{c}\right). \tag{4.8.26b}$$

It does not matter whether the vector of constants **c** is a singleton set, a more general set, or an interval vector. Note that as a consequence of (4.8.26a),

$$\bigcup_{\mathbf{y}\in\text{cset}(\mathbf{h},(\mathbf{x}|\mathbf{c}))} \text{cset}\left(g, (\mathbf{y}, \mathbf{x} \mid \mathbf{c})\right) \tag{4.8.27}$$

in (4.8.21) can be written as

$$\text{cset}\left(g, \left(\text{cset}\left(\mathbf{h}, (\mathbf{x} \mid \mathbf{c})\right), \mathbf{x} \mid \mathbf{c}\right)\right). \tag{4.8.28}$$

If it is understood that $f\,(\mathbf{x} \mid \mathbf{c})$ represents $\text{cset}\left(f, (\mathbf{x} \mid \mathbf{c})\right)$, then (4.8.28) can be simply represented as the right hand side of (4.8.19). In this way the value of any expression can be replaced by its cset.

The simple example in (4.8.6) illustrates how the cset definition in (4.8.21) provides a non-empty value for an expression whose cset is otherwise empty. Let $f\,(x)$ be the composite function $g\,(h\,(x))$ with $g\,(y) = \frac{1}{y}$ and $h\,(x) = x - x = 0$. Therefore

$$f\,(x) = \frac{1}{x - x}. \tag{4.8.29}$$

The domain of f is empty. Combining cset Definition 4.8.6 and the fact that $\text{cset}\,(h, x) = 0$ for all x, it follows that $\text{cset}\,(f, x) = \text{cset}\,(g, 0) = \{-\infty, +\infty\}$, the same result obtained above using CVEs.

4.8.3 Containment-Set Inclusion Isotonicity

Let **X** represent a vector of sets. The following lemma is used in the next section to eliminate the inclusion isotonicity requirement in the original fundamental theorem.

Lemma 4.8.8 Expression csets are inclusion isotonic. That is, given the expression f and its cset evaluated over the sets $\mathbf{X} \subseteq \mathbf{X}'$, then

$$\text{cset}\,(f, \mathbf{X}) \subseteq \text{cset}\left(f, \mathbf{X}'\right).$$

Proof. From the hypothesis that $\mathbf{X} \subseteq \mathbf{X}'$, the set $\mathbf{X}'$ can be partitioned into two mutually exclusive and exhaustive sets:

$$\mathbf{X}' = \mathbf{X} \cup \left(\mathbf{X}' \backslash \mathbf{X}\right). \tag{4.8.30}$$

From the definition of the cset of f at a singleton set in (4.8.21) and over a non-singleton set in (4.8.26),

$$\operatorname{cset}\left(f, \mathbf{X}'\right) = \operatorname{cset}\left(f, \mathbf{X}\right) \cup \operatorname{cset}\left(f, \left(\mathbf{X}' \backslash \mathbf{X}\right)\right) \tag{4.8.31a}$$

$$\supseteq \operatorname{cset}\left(f, \mathbf{X}\right), \tag{4.8.31b}$$

the required result. ■

Note that $\mathbf{x}^{\mathbf{I}}$ represents a vector of intervals and that $f\left(\mathbf{x}\right)$ and $f\left(\mathbf{x}^{\mathbf{I}}\right)$ can continue to be placeholders for there respective csets. Note also that Lemma 4.8.8 holds if set vectors $\mathbf{X}$ and $\mathbf{X}'$ are replaced by interval vectors $\mathbf{x}^{\mathbf{I}}$ and $\mathbf{x}^{\mathbf{I}'}$, respectively.

4.8.4 Fundamental Theorem of Interval Analysis

The original fundamental theorem of interval analysis (Theorem 3.2.2 due to Moore) is remarkable. It guarantees that the values of a real function over an interval argument can be rigorously bounded with a single interval expression evaluation. No assumptions of monotonicity or continuity are required.

The practical consequence of the original fundamental theorem is to provide a simple method of constructing enclosures of real functions. While the theorem is important, it can be made more general, even in the finite interval system. With a closed interval system, such as that in Section 4.7, the fundamental theorem can be extended to include composite expressions that are undefined in the finite interval system. However, even the general finite-system requires all sub-expressions to be inclusion isotonic. Theorems 4.8.13 and 4.8.14, below, first proved in Walster (2000a) and Walster and Hansen (1997), respectively, are the needed extensions of the original fundamental theorem. In Walster, Pryce, and Hansen (2002), the equivalent fundamental theorem for csets is also proved.

First the original fundamental theorem is restated. Next the original theorem is extended to the closed interval system and the restriction that interval expressions be interval extensions is removed. At this point, the theorem applies to inclusion isotonic cset enclosures of real functions in the closed interval system. Finally, these results are further extended to any composite function or multi-valued relation using the implicit notation for csets and the fact that csets, themselves, are inclusion isotonic (Lemma 4.8.8).

Definition 4.8.9 Interval extension: An interval expression, F, evaluated at the point $\mathbf{x}_0$ is an *interval extension* of the real function, f, evaluated at the point, $\mathbf{x}_0$ if $F(\mathbf{x}_0) = f(\mathbf{x}_0)$ for all $\mathbf{x}_0 \in \mathcal{D}_f$. See Section 3.2 or Moore (1979), page 21.

Definition 4.8.10 Inclusion isotonicity: An interval expression, F, is *inclusion isotonic* if for every pair of interval boxes, $\mathbf{x^I}_0 \subseteq \mathbf{x^{I'}}_0 \subseteq \mathcal{D}_f$, then $F\left(\mathbf{x^I}_0\right) \subseteq F\left(\mathbf{x^{I'}}_0\right)$. See Chapter 3 or inclusion monotonicity in Moore (1966).

Theorem 4.8.11 (The original Fundamental Theorem) Let $F\left(\mathbf{x^I}\right)$ be an inclusion isotonic interval extension of the real function $f(\mathbf{x})$. Then $F\left(\mathbf{x^I}_0\right)$ contains $f(\mathbf{x}_0)$ for all $\mathbf{x}_0 \in \mathbf{x^I}_0 \subseteq \mathcal{D}_f$; where $\mathcal{D}_f$ is the domain of f.

Proof. See Theorem 3.2.2. ■

Because the four interval arithmetic operators are inclusion isotonic interval extensions, interval arithmetic operations are enclosures of point arithmetic operations. Repeated application of Theorem 4.8.11 yields enclosures of rational functions. However, for Theorem 4.8.11 to hold, an interval expression must be an interval *extension* (Definition 4.8.9) and inclusion isotonic (Definition 4.8.10). As a consequence, four important cases are not covered by the original fundamental theorem.

1. Theorem 4.8.11 cannot be used to prove an interval expression is an interval enclosure of a function if the expression is not an interval extension. For example, suppose an interval expression is the interval evaluation of a real approximation $g(x)$ of some function

$f(x)$, to which is added an interval bound on the approximation error, $\varepsilon[-1,1]$, for some $\varepsilon > 0$. Let the approximating expression evaluated at the point x_0 be $F(x_0) = g^{\mathbf{I}}(x_0) + \varepsilon[-1,1]$. Because $F(x_0) \neq g(x_0)$, and $F(x_0) \neq f(x_0)$, $F(x_0)$ is an interval *extension* neither of f nor of g. Therefore, in this case, Theorem 4.8.11 does not apply to the expression $F(x_0)$. (Also see the discussion of bounding irrational functions in Section 3.7.)

2. Theorem 4.8.11 only applies to continuous functions. It cannot be invoked to construct an enclosure of a function at either a singular point or an indeterminate form, such as $f(x, y) = x/y$; either when $y = 0$, or when both $x = 0$ and $y = 0$.

3. Theorem 4.8.11 does not define how to construct an enclosure when an interval argument is partially or completely outside the domain of a function. For example, suppose $f(x) = \ln(x)$. What is the set of values that must be contained by an enclosure of $f(x)$ over the interval $X = [-1, 1]$? This question arises because $\ln(x)$ is not defined for $x < 0$. This situation can arise not because of an analysis mistake or coding error, but simply as the consequence of computed interval widening because of dependence.

4. Theorem 4.8.11 does not define how to construct an enclosure of a composition from enclosures of component expressions if the component expressions are not inclusion isotonic. For example, given enclosures of the subexpressions $g(y, \mathbf{x})$ and $h(\mathbf{x})$, what are sufficient conditions under which $G(H(\mathbf{x}_0), \mathbf{x}_0)$ is an enclosure of $f(\mathbf{x}_0) = g(h(\mathbf{x}_0), \mathbf{x}_0)$?

Cases 1 through 3 are covered by simply replacing any of the given expression's possible interval extensions (Definition 4.8.9) in Theorem 4.8.11 by any of the given expression's *cset enclosures*.

Definition 4.8.12 Containment-set enclosure: An interval expression, F, evaluated over an interval, $\mathbf{x}^{\mathbf{I}}{}_0$, is a *containment-set enclosure* of the expression, f, if $F\left(\mathbf{x^I}_0\right) \supseteq \text{cset}\left(f, \mathbf{x^I}_0\right)$ for all $\mathbf{x^I}_0 \in \left(\mathbb{IR}^*\right)^n$, where

$$\mathbb{IR}^* = \left\{[a, b] \mid a \in \mathbb{R}^*,\ b \in \mathbb{R}^*,\ a \leq b\right\},$$

the set of extended real intervals.

In particular, an interval expression F, evaluated at the point $\mathbf{x}_0$, is a *cset enclosure* of the real function $f(\mathbf{x}_0)$ if $F(\mathbf{x}_0) \supseteq \text{cset}(f, \mathbf{x}_0)$ for all $\mathbf{x}_0 \in (\mathbb{R}^*)^n$. In the following theorem, this result replaces the unnecessarily stringent requirement that an interval expression be an *extension* (Definition 4.8.9) of the given function.

Theorem 4.8.13 Let the function f have an inclusion isotonic cset enclosure, $F(\mathbf{x}_0)$, of the expression $f(\mathbf{x}_0)$. Then $F\left(\mathbf{x^I}_0\right)$ is an enclosure of f's cset for all $\mathbf{x^I}_0 \in \left(\mathbb{IR}^*\right)^n$. That is,

$$F\left(\mathbf{x^I}_0\right) \supseteq \text{hull}\left(\text{cset}\left(f, \mathbf{x^I}_0\right)\right). \tag{4.8.32}$$

The proof parallels the proof of the original fundamental theorem.

Proof. Assume $\mathbf{x}_0 \in \mathbf{x^I}_0$. From the inclusion isotonicity hypothesis, $F\left(\mathbf{x^I}_0\right)$ contains $F(\mathbf{x}_0)$, which in turn contains $\text{cset}(f, \mathbf{x}_0)$ because $F(\mathbf{x}_0)$ is a cset enclosure of f. Since this is true for all $\mathbf{x}_0 \in \mathbf{x^I}_0 \in \left(\mathbb{IR}^*\right)^n$, $F\left(\mathbf{x^I}_0\right)$ contains the cset of f for all $\mathbf{x^I}_0$. Since $\mathbf{x^I}_0$ is an arbitrary member of $\left(\mathbb{IR}^*\right)^n$, this completes the proof. ■

Theorem 4.8.13 guarantees that extended real interval arithmetic operations contain the values produced by the corresponding point operation over all elements of the extended interval operands. This is true even for combinations of operations and operands for which the corresponding point operation is undefined. The evaluation of any expressions that are compositions of inclusion isotonic cset enclosures is an enclosure of the corresponding composition. For example, the value of any rational function must be contained in the corresponding enclosure defined by a the same set of extended interval operations.

While the consequences of the simple change from Theorem 4.8.11 to 4.8.13 cover cases 1 through 3 on page 74, neither theorem covers case

4 on page 75. Without an additional extension, it remains unclear how to construct enclosures of composite expressions from sub-expression enclosures that are not inclusion isotonic. Walster and Hansen (1998) cover this case by extending the fundamental theorem to include compositions of expressions that are not inclusion isotonic.

The implicit notation (using $f\left(\mathbf{x^I}_0\right)$ in place of cset $\left(f, \mathbf{x^I}_0\right)$, and $f^{\mathbf{I}}(\mathbf{x}_0)$ in place of cset $(f, \mathbf{x}_0)$) for expression csets is now used for the remainder of this book. Because it is understood that $F\left(\mathbf{x^I}_0\right)$ is the result of evaluating a cset enclosure of f at $\mathbf{x^I}_0$, it follows that hull $\left(f\left(\mathbf{x^I}_0\right)\right) \subseteq F\left(\mathbf{x^I}_0\right)$.

Theorem 4.8.14 (Extended Fundamental Theorem) Given real expressions, $g(y, \mathbf{x})$ and $h(\mathbf{x})$, the composite expression $f(\mathbf{x}) = g(h(\mathbf{x}), \mathbf{x})$, and cset enclosures, $Y_0 = H\left(\mathbf{x^I}_0\right)$ and $G\left(Y_0, \mathbf{x^I}_0\right)$. Then $G\left(Y_0, \mathbf{x^I}_0\right)$ is a cset enclosure of $f\left(\mathbf{x^I}_0\right)$ for all $\mathbf{x^I}_0 \in \left(\mathbb{IR}^*\right)^n$.

Proof. From the definition of csets and their inclusion isotonicity (Lemma 4.8.8)

$$f(X_0) \subseteq G\left(H\left(\mathbf{x^I}_0\right), \mathbf{x^I}_0\right). \tag{4.8.33}$$

Because $G\left(Y_0, \mathbf{x^I}_0\right)$ and $H\left(\mathbf{x^I}_0\right)$ are cset enclosures, $h\left(\mathbf{x^I}_0\right) \subseteq H\left(\mathbf{x^I}_0\right) = Y_0$ and over $\mathbf{x^I}_0$:

$$g\left(h\left(\mathbf{x^I}_0\right), \mathbf{x^I}_0\right) \subseteq G\left(\text{hull}\left(h\left(\mathbf{x^I}_0\right)\right), \mathbf{x^I}_0\right) \tag{4.8.34a}$$

$$\subseteq G\left(Y_0, \mathbf{x^I}_0\right). \tag{4.8.34b}$$

Since $\mathbf{x^I}_0$ is an arbitrary member of $\left(\mathbb{IR}^*\right)^n$, this completes the proof. ■

The cset enclosures on which Theorem 4.8.14 depends can be created from the definition of closures or by prior application of Theorem 4.8.14. Inclusion isotonicity of g and h is not required.

4.8.5 Continuity

When using the closed interval system and continuity is required (as for example is the interval version of Newton's method), then there are three options:

1. Introduce explicit constraints to eliminate points of discontinuity from consideration;

2. Transform discontinuous expressions into continuous expressions, see Walster (2003a); or

3. Use Theorem 4.8.15 below to select intervals that can be proved to be continuous, or impose a containment constraint.

The alternative of using exceptions in the finite system to flag when interval arguments are outside an intrinsic function's natural domain is problematic for at least two reasons:

1. Explicit code is required to prevent any function argument or operation operand from being outside the function or operator's natural domain.

2. Not all intrinsic functions will raise an exception at points of discontinuity, so this approach is not fail safe. The sign [8] and Heaviside[9]functions are good examples.

The following theorem provides a way to automate the test for continuity over a given interval. This, combined with explicit domain constraints can be used to create fail-safe (at least with respect to assumptions of existence and continuity) algorithms for finding roots and fixed points using the interval version of Newton's method and the Brouwer fixed-point theorem. See Chapters 6, 11, and 16 for the algorithms needed to apply domain and continuity constraints.

Theorem 4.8.15 Let f be a function of n variables $x_1, \cdots, x_n$. Let $\mathbf{x}^{\mathbf{I}}$ be an interval vector in the closure of f's domain. That is, $\mathbf{x}^{\mathbf{I}} \subseteq \overline{\mathcal{D}}_f$. Let

[8]The sign function is defined:

$$\operatorname{sign}(x) = \begin{cases} -1 & \text{if } x < 0 \\ 0 & \text{if } x = 0 \\ +1 & \text{if } x > 0 \end{cases} \tag{4.8.35}$$

[9] The Heaviside function is defined:

$$\operatorname{hv}(x) = \begin{cases} 0 & \text{if } x < 0 \\ +1 & \text{if } x \geq 0 \end{cases} \tag{4.8.36}$$

$\mathbf{g}(\mathbf{x}) = \left(\frac{\partial f(\mathbf{x})}{\partial x_1}, \cdots, \frac{\partial f(\mathbf{x})}{\partial x_n}\right)^T$ denote the gradient of f evaluated at $\mathbf{x}$. Then, if all the elements of $\mathbf{g}\left(\mathbf{x^I}\right)$ are bounded (that is, $\left|g_i\left(\mathbf{x^I}\right)\right| < +\infty$, for $i = (1, \cdots, n)$), then f is Lipschitz continuous (or L-continuous) over the box, $\mathbf{x^I}$.

Proof. Begin with the open interval (a, b) and assume $\left|f'(x_0)\right| < +\infty$ for all $x_0 \in (a, b)$. This implies there is a positive finite constant C, say, such that

$$\lim_{\varepsilon \to 0^+} \frac{|f(x_0) \pm f(x_0 \pm \varepsilon)|}{\varepsilon} \leq C,$$

or from the product of limits theorem,

$$\begin{aligned} \lim_{\varepsilon \to 0^+} |f(x_0) \pm f(x_0 \pm \varepsilon)| &\leq C \lim_{\varepsilon \to 0^+} \varepsilon \\ &= 0, \end{aligned}$$

which proves f is continuous in the open interval (a, b).

At the endpoints of the closed interval $[a, b]$, even if they are on the boundary of f's domain, $\left|f'(x_0)\right| \leq C$ guarantees continuity from within the closed interval.

Finally, if f is a function of n variables, the above argument is applied to each variable to obtain the required result. ■

4.9 VECTOR AND MATRIX NOTATION

Using cset enclosures, interval algorithms, including those in the remaining chapters, can be implemented using any expression, whether a function or a relation. With compiler support for interval data types and cset enclosures of interval expressions, any interval algorithms can be implemented without regard to the form of the expressions contained therein. Consequently, and without loss of containment, any enclosure of a cset-equivalent expression can be chosen by a compiler to produce narrow bounds on expression values.

The remaining chapters can be implemented using either the finite interval system discussed in Chapters 2 and 3, or using any more general cset-based interval system such as the one discussed in this chapter.

Unless explicitly stated otherwise, in the remainder of this and subsequent chapters upper case letters such as X without a superscript "I" denote intervals, not sets.

In closed (cset-based) interval systems, $f\,(x)$ represents the cset of an extended real expression that might be multi-valued. On the other hand, $f^{\mathbf{I}}\,(x)$, $f\,(X)$, and $F\,(X)$ denote interval enclosures of the hull of f's cset. The notation $f^{\mathbf{I}}\,(x)$ is used to denote an interval bound on f's cset $f\,(x)$ at the point x. This notation serves to denote a (non-sharp) interval bound on the cset of $f\,(x)$ when $f\,(x)$ is a single point. It is convenient to use $f\,(X)$, $f^{\mathbf{I}}\,(X)$, and $F\,(X)$ to represent interval bounds on the union of f's csets over the interval X. When convenient to do so, $f\,(X)$ can be used to denote *the* hull of f's csets over the interval X. In other situations, $f\,(X)$ can be used to denote the infinite precision interval (not necessarily sharp because of dependence) interval evaluation of the expression f over the interval X. Finally, $F\,(X)$ is normally used to represent the finite precision interval evaluation of f over the interval X.

Table 4.5 contains these representations. The first and second rows contain point and interval arguments, respectively. The first column is labeled "Point Function Notation" rather than "Point Functions", because $f\,(X)$ is used to denote an interval extension using the point function symbol f. The second column contains interval functions.

	Point Function Notation	Interval Functions
Point Argument	$f\,(x)$	$f^{\mathbf{I}}\,(x)$
Interval Argument	$f\,(X)$	$f^{\mathbf{I}}\,(X)\quad F\,(X)$

Table 4.5: Scalar Point and Interval Functions of One Variable

Note that even the above notation is not completely consistent. For example, in some cases point functions of interval arguments are required. Two such examples are the width $\mathrm{w}\,(X)$ and the midpoint $\mathrm{m}\,(X)$ of an interval. Explicitly identified notation overloading can improve exposition clarity, if done carefully and judiciously. Any residual ambiguity is resolved with explicit qualifying remarks.

The notation shown in Table 4.6 is used to generalize point and interval functions and expressions to include vector and matrix arguments and to produce vector and matrix results. A balance has been struck between completely consistent, but verbose notation on the one hand, and inconsistent notation that requires continuous restatement of context on the other hand.

A Bold upright font is used to denote vectors and matrices, with lower and upper case used, respectively, for vectors and matrices. Point functions of point arguments are shown in the upper-left cell of Table 4.6. The lower-left and upper-right cells are used to denote vector and matrix analogs of the corresponding cells in Table 4.5. In fact (with one exception) the (1, 1)-elements in each cell of Table 4.6 contain the elements in Table 4.5. The exception is in the lower-right cell in which there is no analogue in Table 4.6 of $f^{\mathbf{I}}(X)$ in Table 4.5. The reason is that once fine distinctions have been made for scalars, there is no need to notationally carry them over to vector and matrix generalizations.

	Point Function Notation			Interval Functions		
Point Arguments	$f(x)$	$f(\mathbf{x})$	$f(\mathbf{X})$	$f^{\mathbf{I}}(x)$	$f^{\mathbf{I}}(\mathbf{x})$	$f^{\mathbf{I}}(\mathbf{X})$
	$\mathbf{f}(x)$	$\mathbf{f}(\mathbf{x})$	$\mathbf{f}(\mathbf{X})$	$\mathbf{f}^{\mathbf{I}}(x)$	$\mathbf{f}^{\mathbf{I}}(\mathbf{x})$	$\mathbf{f}^{\mathbf{I}}(\mathbf{X})$
	$\mathbf{F}(x)$	$\mathbf{F}(\mathbf{x})$	$\mathbf{F}(\mathbf{X})$	$\mathbf{F}^{\mathbf{I}}(x)$	$\mathbf{F}^{\mathbf{I}}(\mathbf{x})$	$\mathbf{F}^{\mathbf{I}}(\mathbf{X})$
Interval Arguments	$f(X)$	$f\left(\mathbf{x}^{\mathbf{I}}\right)$	$f\left(\mathbf{X}^{\mathbf{I}}\right)$	$F(X)$	$F\left(\mathbf{x}^{\mathbf{I}}\right)$	$F\left(\mathbf{X}^{\mathbf{I}}\right)$
	$\mathbf{f}(X)$	$\mathbf{f}\left(\mathbf{x}^{\mathbf{I}}\right)$	$\mathbf{f}\left(\mathbf{X}^{\mathbf{I}}\right)$	$\mathbf{f}^{\mathbf{I}}(X)$	$\mathbf{f}^{\mathbf{I}}\left(\mathbf{x}^{\mathbf{I}}\right)$	$\mathbf{f}^{\mathbf{I}}\left(\mathbf{X}^{\mathbf{I}}\right)$
	$\mathbf{F}(X)$	$\mathbf{F}\left(\mathbf{x}^{\mathbf{I}}\right)$	$\mathbf{F}\left(\mathbf{X}^{\mathbf{I}}\right)$	$\mathbf{F}^{\mathbf{I}}(X)$	$\mathbf{F}^{\mathbf{I}}\left(\mathbf{x}^{\mathbf{I}}\right)$	$\mathbf{F}^{\mathbf{I}}\left(\mathbf{X}^{\mathbf{I}}\right)$

Table 4.6: Point and Interval Functions/Expression Notation

The above notation is not always followed. An example is $\mathrm{N}\left(\mathbf{x},\mathbf{x}^{\mathbf{I}}\right)$ to denote the result of the n-dimensional Newton operation. It is both traditional and appropriate to use upper case in tribute to Sir Isaac, rather than to use the more consistent notation $\mathbf{n}\left(\mathbf{x},\mathbf{x}^{\mathbf{I}}\right)$.

It is unfortunate that interval notation cannot be as simple as in real analysis. The need for extra notation illustrates the greater variety and (greater) generality of information provided when computing with intervals.

4.10 CONCLUSION

With this chapter, the algorithms the remaining chapters can be implemented using interval cset enclosures of any expression, whether a function or a relation. With compiler support for interval data types and cset enclosures of interval expressions, any interval algorithms can be implemented without regard to the form of the expressions contained therein. Consequently, and without loss of containment, any enclosure of a cset-equivalent expression can be chosen to produce narrow bounds on expression values.

Chapter 5

LINEAR EQUATIONS

5.1 DEFINITIONS

An *interval vector* is a vector whose components are intervals. An *interval matrix* $\mathbf{A^I}$ is a matrix whose elements are intervals. Let $\mathbf{x}$ be a real vector with components x_i $(i = 1, \cdots, n)$, and let $\mathbf{x^I}$ be an interval vector with components X_i $(i = 1, \cdots, n)$. We say $\mathbf{x}$ is contained in $\mathbf{x^I}$ (and write $\mathbf{x} \in \mathbf{x^I}$) if and only if $x_i \in X_i$ for all $i = 1, \cdots, n$. Let $\mathbf{A}$ be a real matrix with elements a_{ij} and let $\mathbf{A^I}$ be an interval matrix with elements A_{ij} for $i = 1, \cdots, m$ and $j = 1, \cdots, n$. We say $\mathbf{A}$ is contained in $\mathbf{A^I}$ (and write $\mathbf{A} \in \mathbf{A^I}$) if and only if $a_{ij} \in A_{ij}$ for all $i = 1, \cdots, m$ and all $j = 1, \cdots, n$.

Similarly, for interval vectors $\mathbf{x^I}$ and $\mathbf{y^I}$ we write $\mathbf{x^I} \subset \mathbf{y^I}$ if and only if $X_i \subset Y_i$ for all $i = 1, \cdots, n$, where $Y_1, \cdots, Y_n$ are the components of $\mathbf{y^I}$. Also, we write $\mathbf{A^I} \subset \mathbf{B^I}$ if and only if $A_{ij} \subset B_{ij}$ for all $i = 1, \cdots, m$ and all $j = 1, \cdots, n$ where the B_{ij} $(i = 1, \cdots, m;\ j = 1, \cdots, n)$ are elements of $\mathbf{B^I}$.

The set of real points (i.e., vectors) $\mathbf{x}$ in an interval vector $\mathbf{x^I}$ form an n-dimensional parallelepiped with sides parallel to the coordinate axes. We often refer to an interval vector as a *box*.

We define the *center* of an interval vector $\mathbf{x^I}$ to be the real vector $\mathrm{m}(\mathbf{x^I}) = (\mathrm{m}(X_1), \cdots, \mathrm{m}(X_n))^T$. Similarly, the center of an interval matrix $\mathbf{A^I}$ is the real matrix $\mathrm{m}(\mathbf{A^I})$ whose elements are the midpoints of the corresponding elements of $\mathbf{A^I}$. We define the *width* of an interval vector (matrix) to be the width of the widest component (element).

An interval matrix $\mathbf{A^I}$ is said to be *regular* if every real matrix $\mathbf{A} \in \mathbf{A^I}$ is nonsingular. Otherwise, it is *irregular*. (Some authors use the term singular in place of irregular.)

We use only one norm for interval vectors. It is the extension of the max norm for real vectors and is defined by

$$||\mathbf{x^I}|| = \max |X_i|$$

where the max is over $i = 1, \cdots, n$. The magnitude of an interval is defined in (3.1.1).

We say that an interval matrix $\mathbf{A^I}$ is *diagonally dominant* if

$$\text{mig}(A_{ii}) \geq \sum_{\substack{j=1 \\ j \neq i}}^{n} |A_{ij}| \text{ for } (i = 1, \cdots, n)$$

where the mignitude mig$(\cdot)$ is defined in (3.1.2).

We sometimes refer to an M-matrix. Let a square matrix $\mathbf{A}$ have non-positive off-diagonal elements. If there exists a positive vector u such that $\mathbf{A}u > \mathbf{0}$, then $\mathbf{A}$ is an M-matrix.

Let a square matrix of order n have eigenvalues λ_i $(i = 1, \cdots, n)$. The *spectral radius* $\rho(\mathbf{A})$ is defined to be

$$\rho(\mathbf{A}) = \max |\lambda_i|$$

for $i = 1, \cdots, n$.

5.2 INTRODUCTION

In this section, we introduce systems of interval linear equations and define the solution set. In Section 5.3, we discuss the solution set and give an illustrative example. We then consider interval methods for bounding the (to be defined) hull of the solution set. We introduce an interval form of Gaussian elimination in Section 5.4 and point out in Section 5.5 that it cannot generally be used in a straightforward manner without considerable overestimation of the solution set. In Section 5.6, we show how to reduce the overestimation by use of preconditioning. We describe an interval version of the Gauss-Seidel method in Section 5.7. In Section 5.8, we describe a procedure for determining the exact hull of the solution set of a preconditioned system. Section 5.10 describes a way to compute the hull of a solution set without preconditioning. In Section 5.11, we consider use of an alternative to the inverse of a matrix when the matrix is singular. Section 5.12 discusses overdetermined systems.

Consider the real system of equations

$$\mathbf{Ax} = \mathbf{b}. \tag{5.2.1}$$

There are many applications in which the elements of the matrix $\mathbf{A}$ and/or the components of the vector $\mathbf{b}$ are not precisely known. If we know an interval matrix $\mathbf{A^I}$ bounding $\mathbf{A}$ and an interval vector $\mathbf{b^I}$ bounding $\mathbf{b}$, we can replace (5.2.1) by

$$\mathbf{A^I x} = \mathbf{b^I}. \tag{5.2.2}$$

We define the solution to (5.2.2) to be the set

$$\mathbf{s} = \{\mathbf{x} : \mathbf{Ax} = \mathbf{b}, \mathbf{A} \in \mathbf{A^I}, \mathbf{b} \in \mathbf{b^I}\}. \tag{5.2.3}$$

That is, $\mathbf{s}$ is the set of all solutions of (5.2.1) for all $\mathbf{A} \in \mathbf{A^I}$ and all $\mathbf{b} \in \mathbf{b^I}$. This set is generally not an interval vector. In fact, it is usually difficult to describe $\mathbf{s}$, as we show by example in Section 5.3. In Section 17.1, we describe a method for approximating $\mathbf{s}$ as closely as desired by covering it with boxes of arbitrarily small size. Other sections in Chapter 17 provide a means for bounding the hull (defined below) of the solution set. See especially Section 17.10.

Because $\mathbf{s}$ is generally so complicated in shape, it is usually impractical to try to use it. Instead, it is common practice to seek the interval vector $\mathbf{x^I}$ containing $\mathbf{s}$ that has the narrowest possible interval components. This interval vector is called the *hull of the solution set* or simply the *hull.* We say we "solve" the system when we find the hull $\mathbf{x^I}$.

The problem of finding the hull is known to be NP-hard. See, for example, Heindl *et al* (1998). Therefore, we generally compute only outer bounds for the hull. Various iterative and direct methods for computing such bounds have been published. We discuss some of these methods later in this chapter. Many publications have discussed this topic. For example, see Alefeld and Herzberger (1983), Hansen (2000a, b), Kearfott (1996), Neumaier (1986, 1990), Shary (1995, 1999). There is a case in which the exact hull is easy to compute. This is the case in which the system has been preconditioned. See Section 5.8.

It is difficult to write (5.2.2) in an unambiguous way. Consider the scalar case $\mathbf{A^I} = [1, 2]$ and $\mathbf{b^I} = [4, 5]$. The solution set is the interval

$$\mathbf{x^I} = \frac{[4, 5]}{[1, 2]} = [2, 5].$$

But, the product of $\mathbf{A^I}$ times $\mathbf{x^I}$ is [2, 10], which does not equal $\mathbf{b^I}$. That is, we cannot substitute the solution into the given equation and get equality. All we can say is that $\mathbf{A^I x^I} \supset \mathbf{b^I}$.

To understand what happens in this example, note that $\mathbf{x^I} = \frac{\mathbf{b^I}}{\mathbf{A^I}}$ and hence $\mathbf{A^I x^I} = \mathbf{A^I}\left(\frac{\mathbf{b^I}}{\mathbf{A^I}}\right)$. This formulation shows that $\mathbf{A^I}$ occurs twice in the computation of $\mathbf{A^I x^I}$. Therefore, dependence (as discussed in Section 2.4) causes loss of sharpness in the computed result.

In this scalar example, it is possible to compute $\mathbf{A^I}\left(\frac{\mathbf{b^I}}{\mathbf{A^I}}\right)$ correctly to be $\mathbf{b^I}$ using dependent multiplication described in Section 2.4.1. However, when $\mathbf{A^I}$ is a matrix, this does not seem to be possible.

We continue to write an equation in the form (5.2.2) wherein $\mathbf{x}$ occurs as if it were a real vector. The obvious incongruity emphasizes the fact that the "equation" requires interpretation.

5.3 THE SOLUTION SET

To illustrate that the solution set $\mathbf{s}$ given by (5.2.2) is not simple, we now give an example from Hansen (1969b). See also Deif (1986). Consider the equations

$$[2,3]x_1 + [0,1]x_2 = [0,120], \tag{5.3.1a}$$
$$[1,2]x_1 + [2,3]x_2 = [60,240]. \tag{5.3.1b}$$

When $\mathbf{x}$ is in the first quadrant, we have $x_1 \geq 0$ and $x_2 \geq 0$ and hence (5.3.1) can be written

$$\begin{aligned} [2x_1, \quad 3x_1 + x_2] &= [0,120], \\ [x_1 + 2x_2, \; 2x_1 + 3x_2] &= [60,240]. \end{aligned}$$

If $\mathbf{x}$ is to be a point of the solution set $\mathbf{s}$, it must be such that the left member intersects the right member in each equation. Therefore,

$$2x_1 \leq 120, \quad 3x_1 + x_2 \geq 0 \tag{5.3.2a}$$
$$x_1 + 2x_2 \leq 240, \; 2x_1 + 3x_2 \geq 60. \tag{5.3.2b}$$

The relation $3x_1+x_2 \geq 0$ is automatically satisfied because we are considering points in the first quadrant only. The remaining three inequalities, when rewritten as equalities, provide boundary lines for $\mathbf{s}$ in the first quadrant.

The boundary of $\mathbf{s}$ in the other quadrants can be found in a similar way. The result is shown in Figure 5.3.1. The boundary is composed of eight line segments. In higher dimensions, $\mathbf{s}$ can be quite complicated. For an explicit description, see Hartfiel (1980).

The set $\mathbf{s}$ has been discussed in contexts wherein interval analysis is not used. For example, see Oettli (1965), Oettli Prager and Wilkinson (1965), and Rigal and Gaches (1967).

We consider interval methods for bounding $\mathbf{s}$ in Sections 5.4, 5.7, 5.8, and 5.10. The bounds are in the form of a box (i.e., an interval vector) containing $\mathbf{s}$. The smallest such box (the hull) for the solution set of (5.3.1) is

$$\mathbf{x}^{\mathbf{I}} = \begin{bmatrix} [-120, 90] \\ [-60, 240] \end{bmatrix}.$$

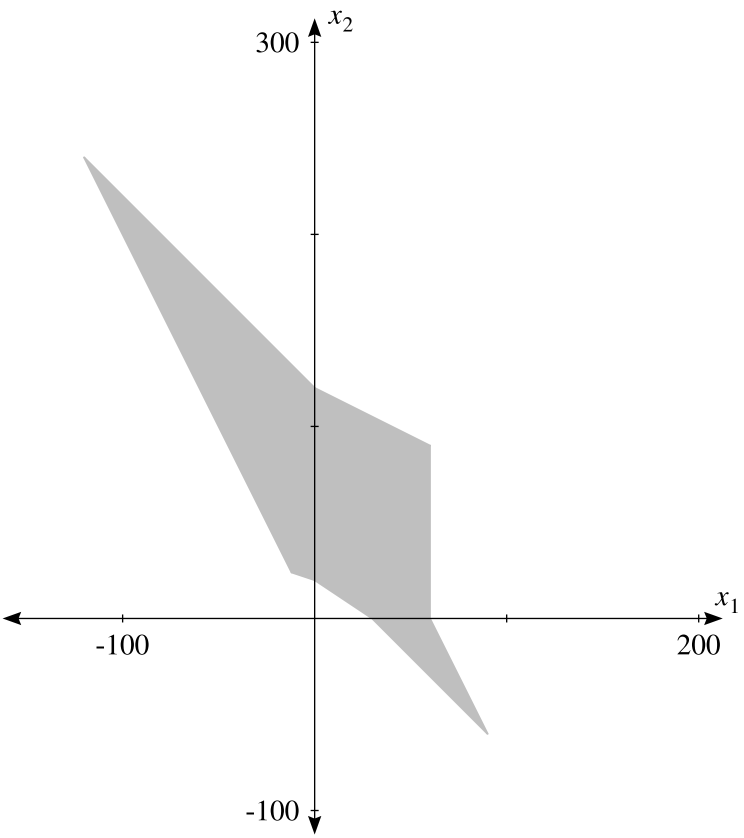


Figure 5.3.1: Solution Set S for Equations in (5.3.1)

Note that $\mathbf{x}^\mathbf{I}$ contains points that are not in $\mathbf{s}$. For example, the point $(0, 200)^T$ is in $\mathbf{x}^\mathbf{I}$ but not in $\mathbf{s}$.

The method we have described can determine the solution set in a given orthant. To completely determine the solution set in this way for an n-dimensional problem requires finding the solution set in 2^n orthants. This suggests that it is difficult to determine the solution set. It is shown in Heindl *et al* (1998) that even the simpler problem of computing the hull of the solution set is NP-hard.

5.4 GAUSSIAN ELIMINATION

There are several variants of methods for solving linear equations that can be labeled as Gaussian elimination. See the outstanding book by Wilkinson (1965). An interval version of any of them can be obtained from one using ordinary real arithmetic by simply replacing each real arithmetic step by the corresponding interval arithmetic step.

One standard method involves factoring the coefficient matrix into the product of a lower and an upper triangular matrix. An interval version of this method with iterative improvement of the triangular factors is discussed by Alefeld and Rokne (1984). Most papers on interval linear equations have not used factorization and we do not do so.

If the coefficient matrix $\mathbf{A}^\mathbf{I}$ and the vector $\mathbf{b}^\mathbf{I}$ are real (i.e., noninterval), then the interval version of Gaussian elimination applied to $\mathbf{Ax} = \mathbf{b}$ simply bounds rounding errors. If the coefficient matrix $\mathbf{A}^\mathbf{I}$ and/or the vector $\mathbf{b}^\mathbf{I}$ have any interval elements, the interval solution vector computed using Gaussian elimination contains the set $\mathbf{s}$.

Suppose the elimination procedure does not fail because of division by an interval containing zero. Then it produces an upper triangular matrix. If no diagonal element of the upper triangular matrix contains zero, then $\mathbf{A}^\mathbf{I}$ is regular (i.e., each real matrix contained in $\mathbf{A}^\mathbf{I}$ is nonsingular). If $\mathbf{A}^\mathbf{I}$ is degenerate, this result proves that $\mathbf{A}^\mathbf{I}$ is nonsingular. That is, the interval method can prove that a real matrix is nonsingular.

Note that regularity can be proved even when (outwardly) rounded interval arithmetic is used because the rounding merely widens intervals. If the widened diagonal elements of the transformed (by elimination) matrix do not contain zero, then they do not contain zero if exact interval arithmetic is used. Therefore, even when rounding occurs, we can numerically prove that a given real matrix is nonsingular or that every real matrix in an interval matrix is nonsingular.

If $\mathbf{A}^I$ and/or $\mathbf{b}^I$ has at least one nondegenerate interval element, then the solution set $\mathbf{s}$ generally consists of more than one point. The box $\mathbf{x}^I$ computed by interval Gaussian elimination always contains the solution set $\mathbf{s}$. However, it is generally not the smallest possible box (the hull) containing $\mathbf{s}$. This is partly because of rounding errors and partly because of dependence.

The following argument shows that the solution $\mathbf{x}^I$ computed using interval Gaussian elimination contains $\mathbf{s}$. When we do Gaussian elimination, using real arithmetic, we compute each component of the solution by, in effect, evaluating a real rational function of the elements of $\mathbf{A}$ and $\mathbf{b}$.

By replacing these quantities by intervals and doing the operations using interval arithmetic, we replace this real rational function by an inclusion isotonic interval extension of the rational function. By Theorem 3.2.2, each such component of the interval solution contains the corresponding real component of the real solution for any $\mathbf{A} \in \mathbf{A}^I$ and any $\mathbf{b} \in \mathbf{b}^I$. Since these real components constitute a point in $\mathbf{s}$, the interval solution must contain $\mathbf{s}$.

Unfortunately, simply replacing real Gaussian elimination by an interval version generally does not yield a suitable algorithm. Bounds of intermediate quantities tend to grow rapidly because of accumulated rounding errors and especially because of dependence among generated intervals. Thus, the computed solution is far from sharp, in general.

An alternative algorithm is described in Section 5.6. It is more costly in numerical operations than simple Gaussian elimination, but produces excellent results when the given system's elements are degenerate or nearly so.

Special consideration must be given to pivot selection in interval Gaussian elimination. In the real algorithm, if two elements are candidates for a pivot, the better choice is the one of larger magnitude. However, if two intervals overlap, then all the real numbers in one are not larger than all the real numbers in the other. A good choice is the one of largest mignitude. See 3.1.2.

Wongwises (1975a, 1975b) simply selects the candidate of largest magnitude (see the definition in Section 3.1) as the pivot. Hebgen (1974) discusses a scaling invariant pivot search procedure.

5.5 FAILURE OF GAUSSIAN ELIMINATION

Since (outward) rounding occurs in interval computations, a solution computed using interval Gaussian elimination is generally not as narrow as possible. In fact, one might expect that the algorithm occasionally fails because of division by an interval containing zero even when the solution set **s** is bounded.

What is worse, however, is that dependence can cause such failure even when exact interval arithmetic is used. In such cases a long accumulator is obviously no help.

In practice, failure can occur even when each real matrix in the interval coefficient matrix is positive definite. This is proved using a three dimensional example by Reichmann (1979).

Consider a system $\mathbf{Ax} = \mathbf{b}$ in which **A** and **b** are real (i.e., noninterval). Suppose we solve this system by interval Gaussian elimination simply to bound rounding errors. Hansen and Smith (1967) observed that the interval bounds grow as the order of the matrix grows.

They experimented with the method on a computer with 27 bits in the mantissa (which is the equivalent of about 8.5 decimal digits of accuracy). they found that for well conditioned real matrices, the endpoints of the interval components of the solution vector generally differed in the first digit for matrices of order 10 or more. Thus when using single precision interval arithmetic, simple interval Gaussian elimination cannot generally be used to solve systems of order 10 or more.

Wongwises (1975a, 1975b) conducted much more extensive experiments. Her results are a definitive determination of the properties of the interval Gaussian elimination algorithm.

5.6 PRECONDITIONING

Simply replacing a real Gaussian elimination algorithm by an interval version cannot generally be recommended in practice (although there are exceptions). In this section, we give an alternative algorithm. It gives excellent results when applied to degenerate or near degenerate systems.

The method was derived by Hansen (1965). A thorough study of the method was made by Wongwises (1975a, 1975b). She showed that, for a noninterval system the guaranteed accuracy of the interval result (as specified by the interval solution) is comparable with the (unknown in practice) accuracy of the result computed using a real Gaussian algorithm.

The procedure we now describe is the same whether the system is real or interval. However, we describe and discuss the interval version.

Let $\mathbf{A}^c$ denote the center $\mathrm{m}(\mathbf{A}^\mathbf{I})$ of $\mathbf{A}^\mathbf{I}$. Thus, if $A_{ij} = [\underline{A}_{ij}, \overline{A}_{ij}]$, then $A_{ij}^c = \frac{1}{2}\left(\underline{A}_{ij} + \overline{A}_{ij}\right)$. In practice, $\mathbf{A}^c$ need not be exact. By some means (for example, real Gaussian elimination), we compute an approximate inverse $\mathbf{B}$ of $\mathbf{A}^c$. We use $\mathbf{B}$ as a preconditioning matrix. Using interval arithmetic to bound rounding errors, we compute $\mathbf{M}^\mathbf{I} = \mathbf{B}\mathbf{A}^\mathbf{I}$ and $\mathbf{r}^\mathbf{I} = \mathbf{B}\mathbf{b}^\mathbf{I}$. We then solve the preconditioned system

$$\mathbf{M}^\mathbf{I}\mathbf{x} = \mathbf{r}^\mathbf{I}. \tag{5.6.1}$$

It might happen that $\mathbf{B}^\mathbf{I}$ cannot be computed because $\mathbf{A}^c$ is singular or near singular. In such a case, the solution set of $\mathbf{A}^\mathbf{I}\mathbf{x} = \mathbf{b}^\mathbf{I}$ might be unbounded. Thus, it is reasonable simply to abandon trying to compute a solution. A program for this case might return a message such as "Solution might be unbounded". We describe an alternative procedure in Section 5.11.

When exact arithmetic is used, the center of $\mathbf{M}^\mathbf{I}$ is the identity matrix. If the interval elements of $\mathbf{A}^\mathbf{I}$ are not wide, then $\mathbf{M}^\mathbf{I}$ approximates the identity matrix in some sense. In this case, we can solve (5.6.1) using rounded interval arithmetic with little growth of interval widths. This is the motivation for the procedure just described.

Computing $\mathbf{M}^\mathbf{I}$ is done without any loss of sharpness due to dependence. This is because each element of $\mathbf{A}^\mathbf{I}$ enters only once in computing a given element of $\mathbf{M}^\mathbf{I}$. Thus, only rounding errors add width to the computed elements of $\mathbf{M}^\mathbf{I}$.

We can now solve (5.6.1) by the interval Gaussian elimination method described above. If the interval elements of $\mathbf{A}^\mathbf{I}$ are narrow, then so are those of $\mathbf{M}^\mathbf{I}$. Therefore, a component of X_i $(i = 1, \cdots, n)$ differs little from $\frac{R_i}{M_{ii}}$. Dependence arising from the small off-diagonal elements of $\mathbf{M}^\mathbf{I}$ has little effect.

When the elements of $\mathbf{A}^\mathbf{I}$ are real (i.e., degenerate intervals), computed results using this procedure are excellent. There is little growth of intervals from either rounding errors or dependence. For experimental results verifying this fact, see Wongwises (1975a, 1975b).

To use the method we have described for solving systems of linear equations, we do a considerable amount of work that is not needed in noninterval Gaussian elimination when we merely compute an approximate solution. The extra work consists of computing an approximate inverse $\mathbf{B}$ of the center $\mathbf{A}^c$ of $\mathbf{A}^\mathbf{I}$ and multiplying $\mathbf{A}^\mathbf{I}$ and $\mathbf{b}^\mathbf{I}$ by $\mathbf{B}$. Consequently, our procedure uses about six times as many operations as ordinary Gaussian elimination.

Extra work of this sort seems to be unnecessary for other kinds of interval computations. It is unfortunate that the one place where it seems to be necessary is in such a fundamental problem.

The extra work can pay a dividend other than producing good results in interval Gaussian elimination. The approximate inverse $\mathbf{B}$ of $\mathbf{A}^c$ is a valuable commodity in certain applications. In Section 11.4, we describe how we use it when solving systems of nonlinear equations. In Section 12.6, we discuss its use in our global optimization algorithm.

Suppose the coefficient matrix $\mathbf{A}^\mathbf{I}$ is not degenerate. Then the solution set is not a single point (in a multidimensional space) but is a set of points. See Section 5.3. Another unfortunate aspect of preconditioning is that it generally enlarges this solution set. See Hansen (1992) or Neumaier (1990). However, this enlargement is generally much less than that caused by the growth of error bounds due to dependence. The solution set of the preconditioned system contains the solution set of the original system $\mathbf{A}^\mathbf{I}\mathbf{x}^\mathbf{I} = \mathbf{b}^\mathbf{I}$.

There are methods that avoid preconditioning when bounding the solution set. See, for example, Shary (1995).

It is not always necessary to precondition a system of linear equations. If the coefficient matrix $\mathbf{A^I}$ is an M-matrix, then interval Gaussian elimination produces the smallest box containing the solution set (i.e., the hull). See Neumaier (1990). In this case, preconditioning not only is unnecessary, but should be avoided to not enlarge the solution set.

Preconditioning can be done in ways other than that we have discussed. See Kearfott (1996).

Suppose that at least one of the elements of $\mathbf{A^I}$ is a nondegenerate interval. Then the solution of $\mathbf{A^I x} = \mathbf{b^I}$ is not a single point but is an extended set. See Section 5.3. The solution of the preconditioned system (5.6.1) contains the solution set of $\mathbf{A^I x} = \mathbf{b^I}$; but is generally larger. The inner region of Figure 5.6.1 is the solution set of the original system (5.3.1). The outer region is the solution set of this system after it has been preconditioned. Despite this enlargement of the solution set, it is generally advisable to precondition the system before solving.

The method we have described in this section is suitable when the elements of $\mathbf{A^I}$ are degenerate or narrow intervals. However, when the elements of $\mathbf{A^I}$ are wide intervals, dependence generally causes interval widths to grow rapidly when applying Gaussian elimination to the preconditioned system (5.6.1). In this wide interval case, it is better to use the hull method described in Section 5.8. The hull method uses preconditioning, but can fail. It is generally advisable to precondition.

When the interval elements of $\mathbf{A^I}$ are wide, Gaussian elimination is prone to failure whether preconditioning is used or not. Failure occurs when it is necessary to divide by an interval containing zero in the elimination procedure. A virtue of the interval Gauss-Seidel method in the next section is that it is always applicable.

In Section 5.8, we discuss yet another method which we call the hull method. The hull method computes the exact hull of the preconditioned system. However, it requires some extra computing. It is reasonable to always use the hull method to solve the preconditioned system; and we recommend its use. However, some computing effort can be saved by

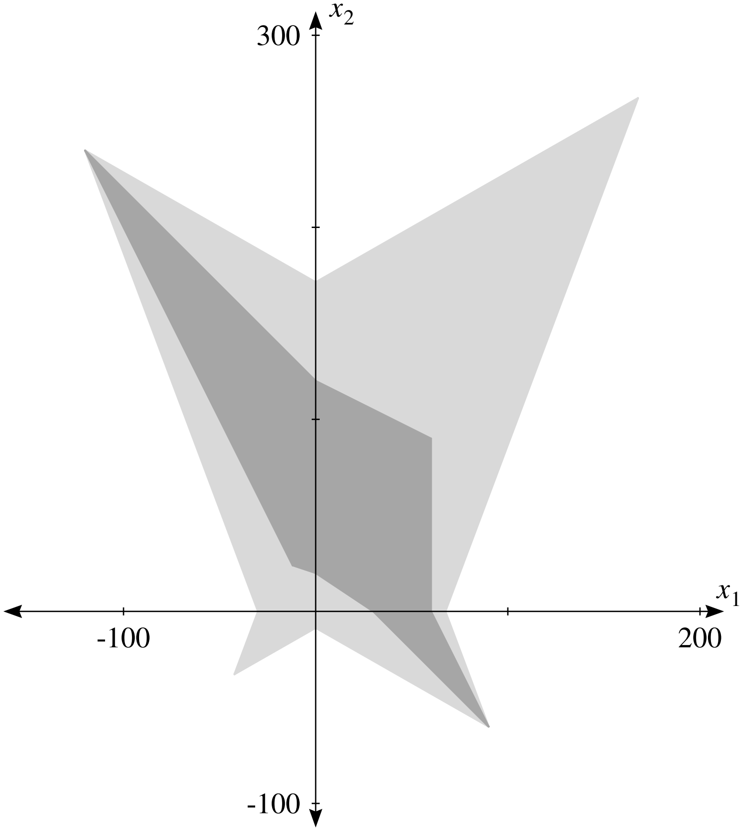


Figure 5.6.1: Solution Set S and the enlarged solution set due to preconditioning for Equations in (5.3.1)

solving the preconditioned system using Gaussian elimination as described above. The hull method generally provides sharper results.

5.7 THE GAUSS-SEIDEL METHOD

In some applications, we have crude bounds on the solution $\mathbf{x}^{\mathbf{I}}$ of $\mathbf{A}^{\mathbf{I}}\mathbf{x} = \mathbf{b}^{\mathbf{I}}$. For example, this is the case when solving the linearized equations in the interval Newton method to be described in Chapter 11. When such bounds are known, it is possible to solve the preconditioned system $\mathbf{M}^{\mathbf{I}}\mathbf{x} = \mathbf{r}^{\mathbf{I}}$ by alternative procedures.

An interval version of the Gauss-Seidel method was first discussed by Alefeld and Herzberger (1970). See also Ris (1972) and Alefeld and Herzberger (1974, 1983). In this section, we describe an extension of the interval Gauss-Seidel method due to Hansen and Sengupta (1981). We assume that the method is applied to the preconditioned system $\mathbf{M}^{\mathbf{I}}\mathbf{x} = \mathbf{r}^{\mathbf{I}}$. Preconditioning enhances performance by reducing the spectral radius of the coefficient matrix. See below.

The i-th equation of the system $\mathbf{M}^{\mathbf{I}}\mathbf{x} = \mathbf{r}^{\mathbf{I}}$ is

$$M_{i1}x_1 + \cdots + M_{in}x_n = R_i.$$

Assume we have interval bounds X_j on the variables x_j for all $j = 1, \cdots, n$. Solving for x_i and replacing the other components of $\mathbf{x}$ by their bounds, we obtain the new bound

$$Y_i = \frac{1}{M_{ii}}(R_i - M_{i1}X_1' - \cdots - M_{i,i-1}X_{i-1}' - M_{i,i+1}X_{i+1} - \cdots - M_{in}X_n). \tag{5.7.1}$$

The intersection

$$X_i' = X_i \cap Y_i \tag{5.7.2}$$

now replaces X_i.

We successively solve using (5.7.1) and (5.7.2) with $i = 1, \cdots, n$. The intersecting process given by (5.7.2) is done at each step so that the newest bound is used in (5.7.1) for each variable x_j for $j < i$.

Note that M_{ii} $(i = 1, \cdots, n)$ might contain zero. If so, the division in (5.7.1) yields a result for Y_i which is not finite. Suppose the numerator also contains zero. Then, if we compute Y_i, the result is $[-\infty, +\infty]$, and $X_i \cap Y_i$ is just X_i; and we have made no progress. Hence, if zero is in both the numerator and denominator in (5.7.1), we do not try to compute Y_i from (5.7.1). We simply skip to the next value of i.

Now suppose $0 \in M_{ii}$ but zero is not in the numerator interval of (5.7.1). Then Y_i is composed of two semi-infinite intervals. Thus, $X_i \cap Y_i$ can be one (finite) interval, the union of two (finite) intervals, or empty. The arithmetic for the case when $0 \in M_{ii}$ is given in Chapter 4.

If $X_i \cap Y_i$ is empty, we have proof that equation (5.6.1) does not have a solution in $\mathbf{x}^\mathbf{I}$. Therefore the original equation $\mathbf{A}^\mathbf{I}\mathbf{x} = \mathbf{b}^\mathbf{I}$ does not have a solution in $\mathbf{x}^\mathbf{I}$. For the remainder of this section, we assume that $X_i \cap Y_i$ is not empty.

Suppose X_i' (computed from (5.7.2) is composed of two intervals. Then X_i' is just the original interval with a gap missing. Since it is tedious to work with an interval with a gap missing, we simply (temporarily) ignore the gap and use the original interval X_i instead of X_i' when using (5.7.1) for the next value of i. The gap might be used later if the box is split in its i-th component.

Suppose we have solved for X_i' for all $i = 1, \cdots, n$. In the interval Newton method discussed in Chapter 11, we might decide to split the new box in one or more dimensions. We can delete gaps in the process. Therefore, we save information of where gaps occur.

Although (5.7.1) and (5.7.2) are written with i in the natural order $1, \cdots, n$, we do not use their elements in this order. If $0 \notin M_{ii}$, then X_i' (from (5.7.2)) might be strictly contained in X_i. If so, sharper results are computed from (5.7.1) for subsequent values of i.

Hence, we first solve for Y_i (and X_i') for those values of i for which $0 \notin M_{ii}$. Then we solve using the remaining values of i. It might be that, for a subsequent value of i, the numerator does not contain zero, whereas it does when using the natural ordering $i = 1, \cdots, n$.

The Hansen-Sengupta method differs from the basic Gauss-Seidel method in that:

(a) preconditioning is used,

(b) division by an interval containing zero is allowed,

(c) gaps in intervals (i.e., exterior intervals) are used,

(d) the order in which equations are solved is variable, and

(e) the intersection (5.7.1) is determined before Y_{i+1} is computed.

Hereafter, when we refer to the interval Gauss-Seidel method, we shall mean the Hansen-Sengupta version. However, we shall retain the more familiar Gauss-Seidel name.

If the off-diagonal elements of $\mathbf{M^I}$ are "wide" intervals, a single iteration of the Gauss-Seidel method improves the bounds on the solution by little or not at all. If the off-diagonal elements are "narrow" intervals, one step of the Gauss-Seidel can give nearly sharp results. In either case, the Gauss-Seidel step, with fewer operations than elimination, can be the preferred procedure. This cheaper procedure was introduced into interval Newton methods by Hansen and Sengupta (1981).

More than one Gauss-Seidel step can be used, of course. In fact, one can iterate until no further improvement occurs. Termination occurs in a finite number of steps. However, this is not an efficient use of the method.

It is fruitful to solve the equations in different orders for different steps of the Gauss-Seidel method. See, for example, Alefeld (1977).

Let λ_i $(i = 1, \cdots, n)$ denote the eigenvalues of a real matrix $\mathbf{M}$ of order n. The spectral radius of $\mathbf{M}$ is defined by $\rho(\mathbf{M}) = \max |\lambda_i|$ where the maximum is over $i = 1, \cdots, n$. The spectral radius of an interval matrix $\mathbf{M^I}$ is $\rho(\mathbf{M^I}) = \max \rho(\mathbf{M})$ where the max is over all $\mathbf{M} \in \mathbf{M^I}$.

The iterated Gauss-Seidel method in interval form converges if $\rho(|\mathbf{M^I}|) < 1$, where $|\mathbf{M^I}|$ denotes the matrix whose elements are the magnitudes of the corresponding elements of $\mathbf{M^I}$. See Alefeld and Herzberger (1974, 1983).

If the original vector $\mathbf{x^I}$ used in (5.7.1) contains a solution of $\mathbf{M^I x} = \mathbf{r^I}$, then the new vector computed using (5.7.1) also contains the solution. If the intersection $X_i' \cap X_i$ is empty for any $i = 1, \cdots, n$, then this fact provides proof that the original vector $\mathbf{x^I}$ did not contain a solution. Note that this is true even when (outward) rounding occurs.

We discuss a modified version of the Gauss-Seidel method in Section 5.10.

5.8 THE HULL METHOD

The *hull* of the solution set of a system of interval linear equations is defined to be the smallest box containing the solution set of the system. For brevity, we speak of the "hull of a system". In general, finding the hull of a system is NP-hard. See Heindl *et al* (1998).

Suppose a given interval system $\mathbf{A^I x} = \mathbf{b^I}$ has been preconditioned using the inverse of the center of $\mathbf{A^I}$ as described in Section 5.6. If the preconditioned system $\mathbf{M^I x} = \mathbf{r^I}$ is regular, its hull can be determined exactly by a fairly simple procedure. In this section, we describe how this is done. We refer to the procedure as the *hull method*.

A procedure for computing this hull was given by Hansen (1992) and independently by Bliek (1992). An improved version was given by Rohn (1993). The procedure we describe in this section is a further improved version from Hansen (2000a). See also Neumaier (1999, 2000). Ning and Kearfott (1997) gave a method for bounding the hull when the coefficient matrix is an H-matrix. Their bounds are sharp when the center of the H-matrix is diagonal. We do not discuss H-matrices in this book. For a definition and discussion of H-matrices, see Neumaier (1990).

We give a procedure for computing the hull; but do not derive it. For a derivation, see Hansen (2000). We first give a theoretical algorithm and then a practical one.

5.8.1 Theoretical Algorithm

In Section 5.6, we denoted the center of $\mathbf{A^I}$ by $\mathbf{A}^c$ and used a matrix $\mathbf{B}$, which approximates $(\mathbf{A}^c)^{-1}$, to precondition the system $\mathbf{A^I x} = \mathbf{b^I}$. We can write $\mathbf{A^I} = \mathbf{A}^c + \mathbf{Q}[-1, 1]$ where $\mathbf{Q}$ is a real matrix. Therefore, the preconditioned matrix (see (5.6.1)) is

$$\mathbf{M^I} = \mathbf{BA^I} = \mathbf{I} + \mathbf{BQ}[-1, 1].$$

That is, the center of $\mathbf{M^I}$ is the identity matrix if $\mathbf{B}$ is the exact inverse of $\mathbf{A}^c$.

Denote $\mathbf{M^I} = [\underline{\mathbf{M^I}}, \overline{\mathbf{M^I}}]$ and $\mathbf{r^I} = [\underline{\mathbf{r^I}}, \overline{\mathbf{r^I}}]$. Then for $i, j = 1, \cdots, n$,

$$\underline{\mathbf{M^I}}_{ij} = -\overline{\mathbf{M^I}}_{ij} \ (i \neq j), \tag{5.8.1a}$$

$$\underline{\mathbf{M^I}}_{ii} + \overline{\mathbf{M^I}}_{ii} = 2. \tag{5.8.1b}$$

Below we give a method for testing whether $\mathbf{M^I}$ is regular. See Theorem 5.8.1. For now assume it is. In this case, $\underline{\mathbf{M^I}}$ is nonsingular and we define $\mathbf{P} = \underline{\mathbf{M^I}}^{-1}$.

We modify the system $\mathbf{M^I x} = \mathbf{r^I}$ so that $\mathbf{r^I}$ takes a particular form. Suppose we multiply the i-th equation of the system by -1 and simultaneously change the sign of x_i. From (5.8.1a), the off-diagonal elements are unchanged. The diagonal elements change sign twice so they have no net change. Thus, the coefficient matrix is unchanged while x_i and R_i change sign.

We can assure that $\overline{R}_i \geq 0$ by changing the sign of R_i (and x_i) if necessary. Assume this is the case. If $0 \in R_i$, we can change the sign of R_i (and x_i) if necessary and obtain $-\underline{R}_i \leq \overline{R}_i$. Therefore, we can always assure that

$$0 \leq |\underline{R}_i| \leq \overline{R}_i. \tag{5.8.2}$$

Hereafter, we assume that (5.8.2) is satisfied for all $i = 1, \cdots, n$. This simplifies the procedure for finding the hull of $\mathbf{M^I x} = \mathbf{r^I}$.

Define $C_i = \frac{1}{2P_{ii}-1}$ and $Z_i = (\underline{R}_i + \overline{R}_i)P_{ii} - \mathbf{e}_i^T \mathbf{P}\overline{\mathbf{r^I}}$ where $\mathbf{e}_i$ is the i-th column of the identity matrix. Denote the hull by $\mathbf{h^I}$. Then

$$\underline{H}_i = \begin{cases} C_i Z_i \text{ if } Z_i > 0, \\ Z_i \text{ if } Z_i \leq 0 \end{cases} \quad (i = 1, \cdots, n) \tag{5.8.3a}$$

$$\overline{\mathbf{h^I}} = \mathbf{P}\overline{\mathbf{r^I}} \tag{5.8.3b}$$

To obtain this result, we assume that the center of $\mathbf{M^I}$ is exactly the identity matrix. In practice, it is not because of rounding errors made in computing $\mathbf{M^I}$ as the product $\mathbf{BA^I}$ and because $\mathbf{B}$ is only an approximate inverse of the approximate center of $\mathbf{A^I}$. We now describe a practical procedure that takes these inaccuracies into account.

5.8.2 Practical Procedure

Denote the computed approximation for $\mathbf{M^I}$ by $(\mathbf{M^I})' = [\underline{\mathbf{M^I}}', \overline{\mathbf{M^I}}']$. We widen the elements of $(\mathbf{M^I})'$ just enough so that the resulting matrix has center $\mathbf{I}$. We change no more than one endpoint of each element.

If $\underline{\mathbf{M^I}}'_{ij} \leq -|\overline{\mathbf{M^I}}'_{ij}|$ for $j \neq i$, we leave $\underline{\mathbf{M^I}}'_{ij}$ unchanged. Otherwise, we replace $\underline{\mathbf{M^I}}'_{ij}$ by $-|\overline{\mathbf{M^I}}'_{ij}|$. This requires no arithmetic. To change an endpoint of a diagonal element requires arithmetic; so rounding might occur. We change $\underline{\mathbf{M^I}}'_{ii}$ (if necessary) so that for the modified matrix $\mathbf{M^I}$, we have $\underline{\mathbf{M^I}}_{ii} \leq 2 - \overline{\mathbf{M^I}}_{ii}$. To do so, we compute $2 - \overline{\mathbf{M^I}}'_{ii}$ using interval arithmetic. We then set $\underline{\mathbf{M^I}}_{ii}$ equal to the smaller of $\underline{\mathbf{M^I}}'_{ii}$ and the lower endpoint of the computed interval bound for $2 - \overline{\mathbf{M^I}}'_{ii}$.

Note that it is not necessary to explicitly modify an upper endpoint of $(\mathbf{M^I})'$. This is because only the lower endpoints of elements of $\mathbf{M^I}$ are used in computing the hull. The modified matrix (implicitly) has the identity as its center.

Our procedure for finding the hull is valid only if $\mathbf{M^I}$ is regular. The following theorem from Hansen (2000) enables us to verify regularity as a by-product of the computation of the hull.

Theorem 5.8.1 Assume $\underline{\mathbf{M^I}}$ is nonsingular so that $\mathbf{P} = \underline{\mathbf{M^I}}^{-1}$ exists. Also assume that $\underline{\mathbf{M^I}}_{ii} > 0$ for all $i = 1, \cdots, n$. Then $\mathbf{M^I}$ is regular if and only if $\mathbf{P} \geq \mathbf{I}$.

Let $\mathbf{B}_0$ denote the exact inverse of the exact center of $\mathbf{A}$. Recall that instead of $\mathbf{B}_0$, we compute an approximation $\mathbf{B}$ for $\mathbf{B}_0$ and therefore we must widen the approximate matrix $\mathbf{M^I}'$ to obtain a matrix $\mathbf{M^I}$ whose center is the identity matrix. Suppose that, using this theorem, we verify that the computed matrix $\mathbf{M^I}$ is regular. Then we have proved that both $\mathbf{A}$ and the result $\mathbf{B}_0\mathbf{A}$ of exact preconditioning are regular. However, the converse is not true. If $\mathbf{M^I}$ is irregular, both $\mathbf{A}$ and $\mathbf{B}_0\mathbf{A}$ can be regular. Also, since preconditioning enlarges the solution set of a system of equations, $\mathbf{B}_0\mathbf{A}$ can be irregular when $\mathbf{A}$ is regular.

If, using this theorem, we find that $\mathbf{M^I}$ is regular, we can compute the hull using (5.8.3). We have noted, however, that the hull of the preconditioned system $\mathbf{M^I x} = \mathbf{r^I}$ is generally larger than that of the original system $\mathbf{A^I x} = \mathbf{b^I}$. We have also noted that if $\mathbf{A^I}$ is an M-matrix, then Gaussian elimination applied to $\mathbf{A^I x} = \mathbf{b^I}$ yields its hull. Therefore, if $\mathbf{A^I}$ is an M-matrix, Gaussian elimination without preconditioning is preferred to the hull method. This not only avoids the work of preconditioning, but Gaussian elimination requires less computing than the hull method.

5.9 COMBINING GAUSS-SEIDEL AND HULL METHODS

Both the Gauss-Seidel method of Section 5.7 and the hull method of Section 5.8 begin by preconditioning a given system. In the resulting system

$$\mathbf{M^I x} = \mathbf{r^I}, \tag{5.9.1}$$

the center of $\mathbf{M^I}$ is the identity matrix. If $\underline{\mathbf{M^I}}_{ii} \leq 0$ for some $i = 1, \cdots, n$, then $\mathbf{M^I}$ is irregular. In this case, the hull method is not applicable. However, it can be combined with the Gauss-Seidel method as we describe in this section.

Assume that $\underline{\mathbf{M^I}}_{ii} > 0$ for one or more values of $i = 1, \cdots, n$. For simplicity, assume that $\underline{\mathbf{M^I}}_{ii} > 0$ for $i = 1, \cdots, s$ and $\underline{\mathbf{M^I}}_{ii} \leq 0$ for $i = s+1, \cdots, n$ for some s where $1 \leq s < n$. We temporarily ignore the last $n - s$ equations. We write the first ($i = 1, \cdots, s$) equations as

$$\mathbf{M^I}_{i1}x_1 + \cdots + \mathbf{M^I}_{is}x_s = \mathbf{r^I}_i - \mathbf{M^I}_{i,s+1}x_{s+1} - \cdots - \mathbf{M^I}_{in}x_n \tag{5.9.2}$$

When applying the Gauss-Seidel method, we assume that we have bounds on the components of x. In the right member of (5.9.2), we replace x_{s+1} through x_n by their interval bounds. We then have s equations in s unknowns in which the center of the coefficient matrix is the identity matrix of order s. We can use the hull method of Section 5.8 to solve this new system. This can fail if the new coefficient matrix is irregular. In this case, we can use the Gauss-Seidel method in the form described in Section 5.8.

If the new coefficient matrix is regular, the hull method obtains a sharp solution of equations (5.9.2) for the first s components of the solution vector of (5.9.1). Solving (5.9.2) by the Gauss-Seidel method does not produce a sharp solution.

On the other hand, the Gauss-Seidel method has the following virtue. Suppose we apply it to (5.9.1) rather than (5.9.2). It is possible that a sharpened bound on one component will sharpen another component that is obtained when solving a different equation of the system. This can occur because of the use of the intersection step expressed by equation (5.7.2) (which is a feature of the Hansen-Sengupta version of Gauss-Seidel). Such a sharpening does not occur when solving (5.9.2) by the hull method.

Note that such sharpening occurs only for those components for which the solution (before intersection) is the entire real line with a gap missing. If the intersection with the input interval is the union of two finite intervals, then no sharpening of the component occurs during the Gauss-Seidel procedure. Therefore, we can expect that the method of this section will produce sharper bounds on the solution of (5.9.1) than will Gauss-Seidel. A reasonable procedure is to use both methods and take the intersection of the two results.

Note that once new bounds on $x_1, \cdots, x_s$ have been obtained, we can obtain new bounds on the remaining components of $\mathbf{x}$ using the Gauss-Seidel method as described in Section 5.7.

5.10 THE HULL OF THE SOLUTION SET OF $\mathbf{A}^I\mathbf{x} = \mathbf{b}^I$

We have noted that the problem of determining the hull of the solution set of an arbitrary (non-degenerate) linear system is NP-hard. Nevertheless, the hull of the solution set of general systems of small order can be computed. We now describe how this can be done.

Given an interval system $\mathbf{A}^I\mathbf{x} = \mathbf{b}^I$, suppose we define a real system by choosing one endpoint of each element of $\mathbf{A}^I$ and one endpoint of each component of $\mathbf{b}^I$. Suppose we do this for every possible combination of endpoints and solve each resulting real systems.

Nickel (1977) showed that the smallest value of a component x_i of $\mathbf{x}$ among all the solutions is the lower endpoint of the i-th component of the hull and the largest such value is the upper endpoint. Therefore, we can determine the hull by solving $2^{n(n+1)}$ real systems. The number of real systems to be solved can sometimes be greatly reduced. See Hansen (2002b).

5.11 A SPECIAL PRECONDITIONING MATRIX

In various sections, we have discussed the inverse $\mathbf{B}$ of the center $\mathbf{A}^c$ of an interval matrix $\mathbf{A}^I$. We have usually ignored the fact that $\mathbf{B}$ does not exist if $\mathbf{A}^c$ is singular. Suppose, we try to invert $\mathbf{A}^c$ using Gaussian elimination. If $\mathbf{A}^c$ is singular, this fails because a pivot element is zero.

If our final goal is simply to compute a solution to the linear system, we can stop. However, in later sections of this book, our goal is to find the intersection of the (unbounded) solution of the system with a given box. Therefore we wish to continue. We can use a Gauss-Seidel step to compute a bound on this intersection. To do so, we want an alternative preconditioning procedure.

When we are going to use Gauss-Seidel, we can regard preconditioning as an effort to compute a matrix that is diagonally dominant. Even when $\mathbf{A}^c$ is singular, we can generally compute a matrix $\mathbf{B}$ such that some diagonal elements dominate the off-diagonal elements in their row. This improves the performance of Gauss-Seidel.

To compute $\mathbf{B}$ when $\mathbf{A}^c$ is singular, we can begin the Gaussian elimination procedure. Assume we use a Gauss-Jordan form of elimination in which a pivot element is used to zero elements both above and below the diagonal. Suppose we arrive at a stage where a nonzero pivot element cannot be obtained using row and column pivoting. The (incomplete) elimination steps at this stage serve to determine a preconditioning matrix that can improve the performance of Gauss-Seidel.

5.12 OVERDETERMINED SYSTEMS

In Hansen and Walster (2003) an algorithm is developed to compute interval bounds on solutions to overdetermined system of interval linear equations. This procedure and extensions of it can be used when more interval equations than unknowns are given. Overdetermined systems are better conditioned than the corresponding least squares problem defined by “normal equations”. Therefore, when overdetermined systems of linear equations arise as part of solving nonlinear systems or optimization problems, this procedure can, and should, be used.

Chapter 6

INEQUALITIES

6.1 INTRODUCTION

In this chapter, we consider inequalities that might or might not involve interval parameters that are independent of the variable arguments of the inequalities. Most of our discussion concerns linear inequalities with interval coefficients. In this case, they are generally obtained as linear expansions of nonlinear functions. We discuss such expansions in Chapter 7.

In optimization problems, inequality constraints are often imposed on the solution. See Chapters 13 and 14. In Section 12.5, 14.3, and 15.3, we discuss another inequality that we introduce when solving optimization problems.

Consider such a set of linear or nonlinear inequalities

$$p_i(\mathbf{x}) \leq 0 \text{ for } (i = 1, \cdots, m) \tag{6.1.1}$$

where $\mathbf{x}$ is a real vector of order n.

Using standard definitions from optimization theory, we say a point $\mathbf{x}$ is *feasible* if $p_i(\mathbf{x}) \leq 0$ for all $i = 1, \cdots, m$. Otherwise $\mathbf{x}$ is *infeasible*.

In practice, we make rounding errors when evaluating $p_i(\mathbf{x})$. Hence, there might be uncertainty whether $p_i(\mathbf{x}) \leq 0$ or not. Furthermore, p_i might be an interval function, in which case the meaning of the inequality requires clarification.

Therefore, we consider the set of inequalities

$$P_i(\mathbf{x}) \leq 0 \text{ for } (i = 1, \cdots, m) \tag{6.1.2}$$

where P_i is an interval function. Suppose we evaluate $P_i(\mathbf{x})$ using outwardly rounded interval arithmetic and compute the interval $[\underline{P}_i(\mathbf{x}), \overline{P}_i(\mathbf{x})]$. We say the point $\mathbf{x}$ is *certainly feasible* if $\overline{P}_i(\mathbf{x}) \leq 0$ for all $i = 1, \cdots, m$. We say that $\mathbf{x}$ is *certainly strictly feasible* if $\overline{P}_i(\mathbf{x}) < 0$ for all $i = 1, \cdots, m$. We say that $\mathbf{x}$ is *certainly infeasible* if $\underline{P}_i(\mathbf{x}) > 0$ for at least one value of $i = 1, \cdots, m$.

A point $\mathbf{x}$ is feasible if it is certainly feasible. Also, $\mathbf{x}$ is infeasible if it is certainly infeasible. Thus, it is possible to know without question whether a point is feasible or infeasible even when rounding is present. A point that is neither certainly feasible not certainly infeasible might be feasible or infeasible. In this case, more accurate evaluation of the constraint functions at a point might change the status of the point. If P_i contains an interval parameter, there are points that are neither certainly feasible nor certainly infeasible even when exact interval arithmetic is used.

Suppose we evaluate P_i over a box $\mathbf{x}^\mathbf{I}$ and obtain the interval $[\underline{P}_i(\mathbf{x}^\mathbf{I}), \overline{P}_i(\mathbf{x}^\mathbf{I})]$. We say the box $\mathbf{x}^\mathbf{I}$ is certainly feasible if $\overline{P}_i(\mathbf{x}^\mathbf{I}) \leq 0$ for all $i = 1, \cdots, m$. We say $\mathbf{x}^\mathbf{I}$ is certainly infeasible if $\underline{P}_i(\mathbf{x}^\mathbf{I}) > 0$ for any $i = 1, \cdots, m$. The strict cases for a box are defined similarly.

In noninterval problems, a goal might be to find a single feasible point. When solving inequality constrained optimization problems, the corresponding process is to eliminate certainly infeasible points from a given box. Thus, all feasible points are retained, even when there is uncertainty because of roundoff. When there is uncertainty in data, an inequality might be expressed using interval parameters. Uncertainty does not alter the fact that feasible points are always retained.

The process of interest for our purposes is the following: Given a box $\mathbf{x}^\mathbf{I}$, find the smallest subbox $\mathbf{x}^{\mathbf{I}\prime}$of $\mathbf{x}^\mathbf{I}$ such that every point in $\mathbf{x}^\mathbf{I}$ that is outside $\mathbf{x}^{\mathbf{I}\prime}$ is certainly infeasible. If we replace $\mathbf{x}^\mathbf{I}$ by $\mathbf{x}^{\mathbf{I}\prime}$, we know we have not discarded any feasible point. In this way, we narrow the region of search for (say) a minimum feasible point. In practice, we do not generally obtain the smallest such subbox, $\mathbf{x}^{\mathbf{I}\prime}$. However, that is our ideal goal.

6.2 A SINGLE INEQUALITY

We now consider how to use a single nonlinear inequality $p(\mathbf{x}) \leq 0$. Assume we wish to use it to eliminate at least some of the certainly infeasible points from a box $\mathbf{x}^{\mathbf{I}}$. One way to do this is to use hull consistency or box consistency which we discuss in Chapter 10. In this section, we consider linearizing and solving an interval linear inequality.

A given inequality might already be linear in its variables. When it is not, we can linearize it by using a Taylor expansion. See Chapter 7. An expansion using slopes (as defined in Chapter 7) can also be used. We now consider a single linear inequality. Thus, consider

$$C_0 + C_1 x_1 + \cdots + C_n x_n \leq 0 \tag{6.2.1}$$

where each C_i $(i = 0, \cdots, n)$ is an interval.

Assume we wish to eliminate points from a box $\mathbf{x}^{\mathbf{I}}$ that do not satisfy this inequality. We first solve for x_1. We replace the other variables by their interval bounds and obtain

$$C_0 + C_1 x_1 + C_2 X_2 + \cdots + C_n X_n \leq 0.$$

We solve this inequality for x_1 and obtain a new interval bound X_1' for x_1. We replace X_1 by $X_1 \cap X_1'$ and repeat the procedure to get a new bound on $x_2, \cdots, x_n$.

Note that when we solve for x_2, we can simplify the computation by using dependent subtraction. This dependent operation is defined by (2.4.2). When solving for x_1, we compute the sum $C_2 X_2 + \cdots + C_n X_n$. If we cancel $C_2 X_2$ from this sum and add $C_1 X_1'$, we have the needed sum to solve for x_2. We do not have to recompute the sum of the other $n - 2$ terms.

In each step, we solve an inequality of the form

$$U + Vt \leq 0 \tag{6.2.2}$$

for a variable t where U and V are fixed intervals. That is, we solve for a set T of values of t for which there exists at least one value of $u \in U$ and at least one value of $v \in V$ such that $u + vt \leq 0$. Thus,

$$T = \{t \mid \exists u \in U,\ \exists v \in V,\ u + vt \leq 0\}. \tag{6.2.3}$$

A simple way to solve $U + Vt \leq 0$ is to rewrite it as the equation

$$U + Vt = [-\infty, 0].$$

Then $T = \frac{1}{V}([-\infty, 0] - U)$, which we can evaluate using extended interval arithmetic. Explicit results can be useful. To list them, denote $U = [a, b]$ and $V = [c, d]$. Then

$$T = \frac{[-\infty, -a]}{[c, d]}.$$

From the rules for cset-based interval arithmetic in Chapter 4, we obtain

$$T = \begin{cases} \left[-\frac{a}{d}, +\infty\right] & \text{if } a \leq 0 \text{ and } d < 0 \\ \left[-\frac{a}{c}, +\infty\right] & \text{if } a > 0,\ c < 0, \text{ and } d \leq 0 \\ [-\infty, +\infty] & \text{if } a \leq 0 \text{ and } c \leq 0 \leq d \\ \left[-\infty, -\frac{a}{d}\right] \cup \left[-\frac{a}{c}, +\infty\right] & \text{if } a > 0 \text{ and } c < 0 < d \\ \left[-\infty, -\frac{a}{c}\right] & \text{if } a \leq 0 \text{ and } c > 0 \\ \left[-\infty, -\frac{a}{d}\right] & \text{if } a > 0,\ c \geq 0, \text{ and } d > 0 \\ \{-\infty\} \cup \{+\infty\} & \text{if } a > 0 \text{ and } c = d = 0 \end{cases} \tag{6.2.4}$$

The last entry in this list is the set of values of t such that $[a, b] + [0, 0]t \leq 0$ when $a > 0$. In practice, we generally seek finite solutions to an inequality. When $a > 0$ and $c = d = 0$ the set of finite solution points is empty.

Note that if $a > 0$ and $c < 0 < d$, the solution consists of two semi-infinite intervals. This occurs because we divide by an interval whose interior contains zero. If this solution is intersected with a finite interval (see above), the result can be empty, a single interval or two disjoint intervals. In the latter case, we speak of an interval with an (open) gap removed. The gap consists of certainly infeasible points.

6.3 SYSTEMS OF INEQUALITIES

Inequality constrained optimization problems generally have more than one inequality constraint. Each can be separately used to reduce the box $\mathbf{x}^{\mathbf{I}}$ as described in Section 6.2. However, combinations of the inequalities are generally more successful in reducing $\mathbf{x}^I$. In this section, we describe how inequalities can be combined.

To solve a system of linear equations, we precondition the system and solve the preconditioned system using Gaussian elimination, the Gauss-Seidel method, or the hull method. See Chapter 5. We use a similar approach to solve linear systems of inequalities. In this case, the hull method is not applicable. In this chapter we describe a procedure for inequalities that is similar to the Gauss-Seidel method for solving linear equalities.

It is convenient to say we "solve" a system of inequalities. However, this merely means that we apply a procedure that eliminates some certainly infeasible subboxes from a given box. As when applying a Gauss-Seidel step to solve a system of linear equations, a step of the method does not produce the smallest possible solution subbox.

The procedure has a certain similarity to the Fourier-Motzkin method which is described, for example, by Dantzig and Eaves (1975). In the Fourier-Motzkin method, the number of generated inequalities increases and can become quite large. In our procedure, we also generate more inequalities than occur in the original system. However, the number of generated inequalities to be solved by the Gauss-Seidel-like procedure is always less than twice the number of original inequalities.

Because we restrict the number of inequalities that can be generated, we delete fewer infeasible points from a box than is possible with greater effort. However, there is good reason for not expending too much effort. In practice, one or more of the inequalities in a given optimization problem is generally nonlinear. We linearize them to compute a solution. The coefficients in the linear expansion are functions of the box in which we are solving the system of inequalities. Therefore, the inequalities change as the box is reduced. There is little point in getting the very best solution to a linear system that is not the final one to be solved.

Once we have linearized the inequalities, we have a system of the form

$$\mathbf{A}^I\mathbf{x} \leq \mathbf{b}^I \tag{6.3.1}$$

where $\mathbf{A}^I$ is an interval matrix. It has as many rows (say m) as there are inequalities. It has n columns where n is the number of variables. The interval vector $\mathbf{b}^{\mathbf{I}}$ has m components.

If we multiply an inequality by a positive constant, we do not change the inequality's sense. That is, we do not change $\leq$ to $\geq$. Also, we can add two inequalities together if they have the same sense. Hence, a positive linear combination of inequalities (having the same sense) yields another valid inequality. Therefore, we can perform Gaussian elimination on the set of inequalities provided we use positive multipliers. To eliminate a given element in the coefficient matrix, the given element and the pivot element must have opposite sign.

As pointed out in Section 5.6, to solve a set of linear equations by interval Gaussian elimination, we first multiply by an approximate inverse of the center of the coefficient matrix. This step reduces the growth due to dependence of interval widths in the elimination process.

We use a similar procedure for systems of inequalities. It is somewhat more complicated, however. The purpose in both procedures is to reduce the effect of dependence.

Let $\mathbf{A}^c$ denote the center, $\mathrm{m}(\mathbf{A}^{\mathbf{I}})$, of $\mathbf{A}^{\mathbf{I}}$. Using $\mathbf{A}^c$, we generate a real matrix $\mathbf{B}$ of nonnegative elements such that the modified system $\mathbf{B}\mathbf{A}^{\mathbf{I}} \leq \mathbf{B}\mathbf{b}^{\mathbf{I}}$ can be solved with reduced interval width. Thus, $\mathbf{B}$ is a preconditioning matrix similar to that used when solving linear interval equations. Now, however, the number of rows of $\mathbf{B}$ might be any number from m to $2m-1$ depending on the problem.

The matrix $\mathbf{B}$ can be computed in the same way a matrix inverse is generated. To aid in understanding, we now describe this more familiar procedure.

Let $\mathbf{Q}$ be a square, nonsingular, real matrix. Initially, set $\mathbf{B}$ equal to the identity matrix $\mathbf{I}$. Use (for example) the Gauss-Jordan method of elimination (e.g., see Stewart (1973)) to transform $\mathbf{Q}$ into the identity matrix. Simultaneously, do every arithmetic operation on $\mathbf{B}$ that is done in the elimination process on $\mathbf{Q}$. When $\mathbf{Q}$ is finally transformed to $\mathbf{I}$, the same operations on $\mathbf{B}$ produce the inverse of $\mathbf{Q}$.

Suppose $\mathbf{Q}$ is an m by n matrix and $m \neq n$. If $m \geq n$, the elimination procedure can produce zero elements in position (i, j) for all i and j with $i \neq j$. If $m < n$, then zeros are produced only in the first m columns.

Now consider the case $m = n$ so that the system of interval equations

$$\mathbf{Q}^{\mathbf{I}}\mathbf{x} = \mathbf{c}^{\mathbf{I}} \tag{6.3.2}$$

is square. Let $\mathbf{Q}^c$ be the center $m(\mathbf{Q}^{\mathbf{I}})$ of $\mathbf{Q}^{\mathbf{I}}$. Let $\mathbf{B}$ be an approximate inverse of $\mathbf{Q}^c$. We can compute $\mathbf{B}$ as just described. Multiplying (6.3.2) by $\mathbf{B}$, we obtain

$$\mathbf{B}\mathbf{Q}^{\mathbf{I}}\mathbf{x} = \mathbf{B}\mathbf{c}^{\mathbf{I}}.$$

In this new equation, the coefficient matrix tends to be diagonally dominant and can be solved without undue growth (from dependence) of interval widths. See Section 5.6 for a more thorough discussion of this procedure.

When solving inequalities, we use a similar procedure. We generate a preconditioning matrix $\mathbf{B}$ in essentially the same way. However, in the case of inequalities, the elements of $\mathbf{B}$ must now be nonnegative. This restriction might prevent completion of the elimination process to get the desired $\mathbf{B}$. However, this does not mean the process fails. It merely means we delete fewer points from a box.

Having computed $\mathbf{B}$, we multiply (6.3.1) by $\mathbf{B}$, getting

$$\mathbf{B}\mathbf{A}^{\mathbf{I}}\mathbf{x} \leq \mathbf{B}\mathbf{b}^{\mathbf{I}} \tag{6.3.3}$$

We can solve this relation with less growth of interval widths than for the original relation $\mathbf{A}^{\mathbf{I}}\mathbf{x} \leq \mathbf{b}^{\mathbf{I}}$.

In general, we do column interchanges to compute $\mathbf{B}$. Therefore, instead of (6.3.3), our new system is

$$(\mathbf{B}\mathbf{A}^{\mathbf{I}}\mathbf{P})(\mathbf{P}^{-1}\mathbf{x}) \leq \mathbf{B}\mathbf{b}^{\mathbf{I}} \tag{6.3.4}$$

where $\mathbf{P}$ is a permutation matrix and $\mathbf{B}\mathbf{A}^{\mathbf{I}}\mathbf{P}$ is the new coefficient matrix. Note that the inverse $\mathbf{P}^{-1}$ of $\mathbf{P}$ is the transpose $\mathbf{P}^T$.

The order in which the inequalities are combined by the elimination process is important. We discuss this aspect in the next section and return to the solution process in Section 6.5.

6.4 ORDERING INEQUALITIES

Consider a set of inequalities $p_i(\mathbf{x}) \leq 0$ $(i = 1, \cdots, m)$. If we evaluate p_i for some $i = 1, \cdots, m$ over a box $\mathbf{x^I}$, we compute an interval

$$P_i(\mathbf{x^I}) = [\underline{P}_i(\mathbf{x^I}), \overline{P}_i(\mathbf{x^I})].$$

If $\overline{P}_i(\mathbf{x^I}) \leq 0$, then $p_i(\mathbf{x}) \leq 0$ for all $\mathbf{x} \in \mathbf{x^I}$, and this particular constraint cannot cause a point in $\mathbf{x^I}$ to be infeasible. Therefore, this constraint can be ignored when considering the box $\mathbf{x^I}$.

If $\underline{P}_i(\mathbf{x^I}) > 0$, then $p_i(\mathbf{x}) > 0$ for all $\mathbf{x} \in \mathbf{x^I}$. That is, every point $\mathbf{x} \in \mathbf{x^I}$ is certainly infeasible. For these extreme cases, the effect of the particular inequality is known. The case of interest is when

$$\underline{P}_i(\mathbf{x^I}) \leq 0 < \overline{P}_i(\mathbf{x^I}). \tag{6.4.1}$$

Hereafter in this section, we assume that this condition holds for all $i = 1, \cdots, m$.

We want to know which inequalities are more helpful in deleting certainly infeasible points of a given box $\mathbf{x^I}$. The corresponding question in the noninterval case is: "Which constraints are most strongly violated at some point?" This question is complicated by the fact that the different inequalities might be scaled differently. In the interval case, we address this complication by (implicitly) normalizing.

Consider the quantity

$$s_i = \frac{\overline{P}_i(\mathbf{x^I})}{\overline{P}_i(\mathbf{x^I}) - \underline{P}_i(\mathbf{x^I})} \tag{6.4.2}$$

for $i = 1, \cdots, m$. If $\overline{P}_i(\mathbf{x^I}) = 0$, then $s_i = 0$ and (as pointed out above) the constraint $p_i(\mathbf{x}) \leq 0$ is of no help in eliminating certainly infeasible point from $\mathbf{x^I}$.

If $\underline{P}_i(\mathbf{x^I}) = 0$ and $\overline{P}_i(\mathbf{x^I}) > 0$, then $s_i = 1$ and $\mathbf{x^I}$ cannot contain interior points of the feasible region. That is, the constraint is about as useful as possible in eliminating certainly infeasible points of $\mathbf{x^I}$. For constraints of interest, $0 < s_i \leq 1$; and the larger s_i, the greater help the constraint tends to be in eliminating points of $\mathbf{x}^{\mathbf{I}}$.

Assume (6.4.1) holds for a set of m constraints. We order them so that $s_i \geq s_{i+1}$ for all $i = 1, \cdots, m-1$. Then the smaller the index i, the more useful the constraint tends to be.

We proceed as follows. We first evaluate $P_i(\mathbf{x}^{\mathbf{I}})$ for all $i = 1, \cdots, m$. (We discuss an alternative in Section 10.10.) If $\underline{P}_i(\mathbf{x}^{\mathbf{I}}) > 0$ for some i, our process for solving the constraint inequalities is finished. There is no feasible point in $\mathbf{x}^{\mathbf{I}}$. If $\overline{P}_i(\mathbf{x}^{\mathbf{I}}) \leq 0$, then (while $\mathbf{x}^{\mathbf{I}}$ is the current box) we drop the i-th constraint from the list of constraints to be solved. We linearize those that remain.

Assume $\mathbf{x}^{\mathbf{I}}$ is not certainly infeasible and that all constraints for which $\overline{P}_i(\mathbf{x}^{\mathbf{I}}) \leq 0$ (i.e., that are known to be inactive in $\mathbf{x}^{\mathbf{I}}$) have been removed from the list. We order the remaining constraints according to the value of s_i as described above.

6.5 SECONDARY PIVOTS

We now return to the elimination process. Consider the step in which we generate the preconditioning matrix $\mathbf{B}$ occurring in (6.3.3). We use Gaussian elimination operations to zero appropriate elements of $\mathbf{A}^c = \mathrm{m}(\mathbf{A}^{\mathbf{I}})$. The same operations applied to the identity matrix generates $\mathbf{B}$.

We shall denote the matrix that begins as $\mathbf{A}^c$ by $\mathbf{A}$ at any stage of the elimination process even though it changes from step to step. Assume we have generated the desired zero elements in the first $r-1$ columns of $\mathbf{A}$. We describe the pattern of zero elements below. Next, we generate zero elements in column r.

To do so, we use the element in position (r, r) as pivot element and apply an elimination step to produce the desired zero elements in column r. The pivot element is chosen to be the one of largest magnitude in row r and it is placed in position (r, r) by interchanging columns if necessary. This element a_{rr} is designated as the "primary pivot".

When doing elimination among inequalities, the multiplier must be positive so that sense of the inequality is not reversed. Therefore a given pivot element can only be used to zero an element which is opposite in sign to the pivot element. Therefore, we determine a "secondary pivot" whose sign is opposite to that of the primary pivot. The secondary pivot in column r is used to zero appropriate elements that have the same sign as the primary pivot a_{rr}.

The secondary pivot is chosen (as described below) from the column which is interchanged into column r when choosing the primary pivot. It is chosen from rows $r + 1, \cdots, m$. A copy of the secondary pivot row is placed as row $r + m$

The secondary pivot is used to eliminate elements in positions (i, r) for $i = r + 1, \cdots, r + m - 1$ which are opposite in sign to it. Then the primary pivot is used to zero elements in these positions which are opposite in sign to the primary pivot. This latter step includes zeroing of the secondary pivot. Note that the secondary pivot is zeroed only in its original position and not in the copy placed in row $r + m$.

In earlier steps in which zeros are generated in columns $1, \cdots, r - 1$, copies of the secondary pivot rows are placed in rows $m + 1, \cdots, m + r - 1$. When generating zeros in column r, elements in these rows are zeroed.

Suppose the primary pivot is positive. Then the primary pivot row can be used to obtain a lower bound on x_r; and the secondary pivot row (copied in row $r + m$) can obtain an upper bound on x_r. The roles are reversed if the primary pivot is negative.

It might happen that when we want to zero elements in column r that all the elements in positions (i, r) for $i = r, ..., m$ have the same sign. If this is also true for all columns $j > r$ (that might be interchanged with row r), then we are unable to continue the elimination process that we have been describing.

Let r' denote the last column index for which zeros can be generated as described above. We now do a second phase of elimination. We zero elements above the diagonal. That is, we zero elements in positions (i, j) for which $i < j$ $(j = 2, \cdots, r')$. A primary pivot element is used to zero opposite sign elements above it in its column. Elements that are copies of secondary pivots and now occur in rows $m + 1, \cdots, m + r'$ are used to zero opposite sign elements in their respective columns.

6.6 COLUMN INTERCHANGES

Assume the inequalities have been placed in the order described in Section 6.4. Because this is a desirable order, we wish to not interchange rows of **A** even though interchanges for pivot selection might enhance numerical stability. However, we are free to interchange columns. In this section, we discuss how to do so to get a "well positioned" secondary pivot and improve numerical stability.

A procedure was described by Hansen and Sengupta (1983) for choosing a pivotal column. We describe a simpler procedure in this section.

We do not perform row interchanges unless the current row is zero. In decreasing order of importance, we want

1. the secondary pivot to occur as high in its column as possible,
2. the secondary pivot to be as large in magnitude as possible, and
3. the primary pivot to be as large in magnitude as possible.

Consider what happens when we use a (primary or secondary) pivot element to eliminate an element in another row. We first "scale" the pivot row by multiplying it by the multiplier. Then we add the scaled pivot row to another row. If the multiplier is large in magnitude, the scaled pivot row tends to dominate and information in the other row is lost by being shifted off the accumulator in the addition step.

That is, information in the pivot row dominates and is saved while information in the other row is lost. Conversely, if the multiplier is small in magnitude, the information in the other row is retained. Preserving information in the other row is why we want a multiplier to be small in magnitude when doing ordinary Gaussian elimination. For the same reason, we want the pivot elements to be large in magnitude.

In our case, if we have to use a pivot element small in magnitude, we want the pivot row to contain highly useful information because it will be retained intact. Thus, we want the dominating pivot row to correspond to an inequality that is "strongly violated" in $\mathbf{x}^{\mathbf{I}}$.

Because of the way rows are initially ordered (see Section 6.4), higher rows are more useful in this sense than lower rows. It is for this reason that we want the secondary pivot (which occurs lower than the primary pivot) to occur as high in the column as possible. The smaller the pivot, the more the pivot row dominates.

We wish to allow more useful information rather than less useful information to dominate in this way. Thus, it is more important for the secondary pivot to be large in magnitude than it is for the primary pivot to be large in magnitude.

The reader might find it odd that we are willing to sacrifice numerical stability (i.e., accuracy) in favor of maintaining the order of the inequalities. The reason is that much of the time we solve the inequalities over a relatively large box. Therefore, the linearization of the original nonlinear inequalities does not produce very accurate approximation.

Moreover, we are satisfied merely to delete large infeasible portions of such boxes. Accuracy is of little importance until the boxes become small near the end of the optimization process. Therefore, we opt for efficiency over accuracy in this part of the process.

In the next section, we describe our algorithm for computing the preconditioning matrix **B** discussed in Sections 6.3 and 6.5. The above arguments are used in choosing the columns to interchange for the pivot selection and elimination procedure.

6.7 THE PRECONDITIONING MATRIX

In this section, we give an algorithm for doing Gaussian elimination to transform $\mathbf{A}^c$ into a matrix with the desired zero elements. The algorithm saves both the primary and the secondary pivot rows. The preconditioning matrix **B** is computed by performing the same operations on a matrix that begins as the identity matrix of order m.

We let **A** denote the matrix being transformed. Initially, $\mathbf{A} = \mathbf{A}^c$. We retain the same name for the matrix throughout the elimination process even though it changes with each step.

1. Set $r = 0$.

2. Replace r by $r + 1$.

3. If $r = m$, set $r' = m$ and go to step 13.

4. If $r > n$, set $r' = n$ and go to step 13.

5. If $a_{ij} = 0$ for all $i = r, \cdots, m$ and all $j = 1, \cdots, n$, set $r' = r - 1$ and go to step 13.

6. If row r of $\mathbf{A}$ is zero, move rows $r + 1, \cdots, m$ up by one and move the old r-th row to become the m-th row.

7. Determine the smallest index s such that, for some $j = r, \cdots, n$, the elements a_{rj} and a_{sj} are nonzero and have opposite signs. If no such index exists, set $r' = r - 1$ and go to step 13. Suppose a_{rj} and a_{sj} have opposite signs for all j in some set J of indices $r, \cdots, n$. Let j' denote the index $j \in J$ for which a_{sj} is largest in magnitude. If there is more than one such index, choose j' to be the index for which a_{rj} (with $j \in J$) is largest in magnitude.

8. Interchange columns r and j'.

9. Use the secondary pivot element (in position (s, r)) to zero the elements opposite in sign to it in positions (i, r) for $i = r + 1, \cdots, m$.

10. Put a copy of row s into row $m + r$.

11. Use the primary pivot element (in position (r, r)) to zero the elements opposite in sign to it in positions (i, r) for $i = r + 1, \cdots, m$.

12. Go to step 2.

13. The first m rows of $\mathbf{A}$ are now in upper trapezoidal form. A submatrix of $r' - 1$ rows and n columns has been appended to $\mathbf{A}$. This submatrix is composed of secondary pivot rows and is also in upper trapezoidal form. We now begin zeroing elements above the diagonal of each of the two submatrices. Set $r = 0$.

14. Set $r = r + 1$.

15. If $r = r'$, go to step 17. If $r = r' + 1$, go to step 19.

16. Use $a_{m+r,r}$ as a pivot to zero any element (except the one in position (r,r)) of opposite sign in column r.

17. Use a_{rr} as a pivot to zero any element (except the one in position $(m+r,r)$) of opposite sign in column r.

18. Go to step 14.

19. Terminate.

6.8 SOLVING INEQUALITIES

In this section, we discuss how to "solve" inequalities with interval coefficients after they have been preconditioned.

Assume we have computed the preconditioning matrix **B** as described in Section 6.7. Recall that, while performing the matrix operations given by the steps of the algorithm, we compute **B** by doing the same operations on a matrix that is initially the identity matrix of order m.

Recall that we wish to "solve" a set of linear inequalities

$$\mathbf{A}^{\mathbf{I}}\mathbf{x} \leq \mathbf{b}^{\mathbf{I}}. \tag{6.8.1}$$

We precondition these inequalities by transforming them into

$$(\mathbf{B}\mathbf{A}^{\mathbf{I}}\mathbf{P})(\mathbf{P}^{-1}\mathbf{x}) \leq \mathbf{B}\mathbf{b}^{\mathbf{I}}. \tag{6.8.2}$$

where **P** is the permutation matrix effecting the column interchanges described in Section 6.7.

To simplify the following discussion, we assume no column interchanges were necessary. In this case, (6.8.2) takes the simpler form

$$\mathbf{M}^{\mathbf{I}}\mathbf{x} \leq \mathbf{c}^{\mathbf{I}} \tag{6.8.3}$$

where $\mathbf{M}^{\mathbf{I}} = \mathbf{B}\mathbf{A}^{\mathbf{I}}$ and $\mathbf{c}^{\mathbf{I}} = \mathbf{B}\mathbf{b}^{\mathbf{I}}$.

When we precondition a system of linear equations, we obtain a similar system except that they are equations. See (5.6.1). We then choose one of three procedures. The preferred choice is to use the hull method of Section 5.8. However, this method has no counterpart when solving inequalities; and therefore we ignore it in the current context. Another choice is to perform interval Gaussian elimination on the preconditioned system. A third choice is to use the Gauss-Seidel method and simply solve the i-th equation ($i = 1, \cdots, n$) for the i-th variable (after replacing the other variables by their interval bounds).

We have similar choices when solving (6.8.3). A procedure that uses interval Gaussian elimination on the preconditioned system is difficult to describe and complicated to program. Partly, this is because a positive or negative element before elimination is applied to the real matrix $\mathbf{A}^c$ might contain zero after elimination. We shall not describe such a procedure.

We note the following for a reader who might use such a method. Suppose an interval element that we wish to eliminate contains zero as an interior point. We can use a positive (primary or secondary) pivot to eliminate the negative part of such an element and a negative (secondary or primary) pivot to eliminate the positive part of such an element. In this way, the elimination step is done without changing the sense of an inequality.

We now describe a procedure similar to a Gauss-Seidel that can be used to "solve" (6.8.3). To use this procedure, we make use of information obtained when computing the preconditioning matrix $\mathbf{B}$.

Suppose that when transforming $\mathbf{A}^c$ to obtain $\mathbf{B}$, the element that ends up in position (i, i) is used as a primary pivot. Then, in the i-th inequality of (6.8.3), we replace all variables except the i-th by their interval bounds. We then solve this inequality for the i-th variable. Before this primary pivot was used, a row containing the corresponding secondary pivot is added to row $m + i$ of the list of inequalities. We solve this inequality for the same variable x_i in the same way as the i-th inequality.

Note that if row i provides an upper bound on x_i, then row $m+i$ provides a lower bound and vice versa.

This might not exhaust all of the inequalities in the system (6.8.3). We might not be able to complete the elimination process in $\mathbf{A}^c$ in getting $\mathbf{B}$ because of the lack of a secondary pivot at some stage. Suppose we are able to produce the desired zero elements in only some number k of the columns where $k < m$. Then rows $k + 1$ through m of $\mathbf{BA}^c$ might all be nonzero in all of columns $k+1$ through n. We do not use these rows of $\mathbf{M^I}$ in our procedure resembling Gauss-Seidel. They simply can be ignored or treated separately using hull or box consistency as described in Chapter 10.

Chapter 7

TAYLOR SERIES AND SLOPE EXPANSIONS

7.1 INTRODUCTION

In our optimization algorithms, we frequently expand the objective function and constraint functions to first or second order terms. These expansions can be in terms of slopes (see Section 7.7) or can be Taylor expansions using derivatives. Our algorithms are more efficient if slope expansions are used. However, we expect readers to be more familiar with derivatives than slopes; and therefore we discuss the algorithms (in other chapters) as if derivatives are used.

In this chapter, we discuss expansions of both kinds. We show how interval methods can bound the remainder in Taylor series. There is no remainder in a slope expansion.

Although slope expansions can replace those using derivatives, slopes cannot replace derivatives in all situations. Note that monotonicity is expressed in terms of derivatives; and there is no counterpart in terms of slopes. We make extensive use of monotonicity in this book.

We consider Taylor expansions for the one-dimensional case in Section 7.2 and for the multidimensional case in Section 7.3. Jacobians and Hessians are discussed in Section 7.4. In Section 7.5, we discuss automatic procedures to compute numerical values of derivatives evaluated over intervals. In Section 7.6, we describe special procedures for sharpening the bounds on the range of a function by more effective use of Taylor expansions.

We introduce slope expansions of rational functions in Section 7.7 and of irrational functions in Section 7.8. We discuss multidimensional slope expansions in Section 7.9 and higher order slope expansions in Section 7.10. We describe slope expansions of nonsmooth functions in Section 7.11 and the automatic computation of slopes in Section 7.12.

7.2 BOUNDING THE REMAINDER IN TAYLOR EXPANSIONS

Interval methods can be used very conveniently to bound the remainder when truncating Taylor series. Consider a function f of a single variable. For simplicity, assume f has continuous derivatives of any necessary order. Expanding $f(y)$ about a point x,

$$f(y) = f(x) + (y-x)f'(x) + \ldots + \frac{(y-x)^m}{m!} f^{(m)}(x) + R_m(x, y, \xi) \tag{7.2.1}$$

where the remainder term (in the Lagrange form) is

$$R_m(x, y, \xi) = \frac{(y-x)^{m+1}}{(m+1)!} f^{(m+1)}(\xi).$$

The point ξ lies between x and y. Hence, if x and y are in an interval X, then ξ must be in X. Therefore, $f^{(m+1)}(\xi) \in f^{(m+1)}(X)$ (see Theorems 3.2.2 and 4.8.14) and

$$R_m(x, y, X) = \frac{(y-x)^{m+1}}{(m+1)!} f^{(m+1)}(X) \tag{7.2.2}$$

bounds the remainder for any $x \in X$ and any $y \in X$.

The cases $m = 0$ and $m = 1$ are of special interest later. For $m = 0$, we have

$$f(y) \in f(x) + (y-x)f'(X) \tag{7.2.3}$$

Since this relation holds for all $y \in X$, we have

$$f(X) \subset f(x) + (X - x)f'(X'). \tag{7.2.4}$$

Note that we have replaced X in the argument of f' by X'. We explain below why this is done. For $m = 1$,

$$f(y) \in f(x) + (y - x)f'(x) + \frac{(y - x)^2}{2} f''(X) \tag{7.2.5}$$

and

$$f(X) \subset f(x) + (X - x)f'(x) + \frac{(X - x)^2}{2} f''(X'). \tag{7.2.6}$$

We now explain why we have replaced X by X' in (7.2.4) and (7.2.6). We noted above that the quantity ξ in (7.2.1) must be in X. Therefore, we replaced ξ by X in (7.2.3). However, this is a bound with the same numeric value as X but is not analytically identical to X. Thus, in (7.2.4), while X and X' are numerically equal, they are not the same variable and are therefore independent.

To illustrate this fact, consider the example in which $f(x) = \frac{1}{x}$. Since $f'(x) = \frac{-1}{x^2}$, (7.2.4) can be written as

$$f(X) \subset \frac{1}{x} - \frac{X - x}{X'^2}. \tag{7.2.7}$$

If we assume that X' is identically equal to X, we can replace $\frac{X}{X'^2}$ by $\frac{1}{X}$ and write (7.2.4) as

$$f(X) \subset \frac{1}{x} - \frac{1}{X} + \frac{x}{X^2}.$$

Completing the square to sharpen the bound on $f(X)$, we rewrite this as

$$f(X) \subset x\left(\frac{1}{X} - \frac{1}{2x}\right)^2 + \frac{3}{4x}. \tag{7.2.8}$$

Let $X = [1, 4]$ and $x = 2.5$. Then the interval value of the right member of (7.2.8) is $[0.30625, 1.9]$. This interval does not contain the entire range $[0.25, 1]$ of f over X.

If we evaluate the right member of (7.2.7) without analytic changes, we obtain $[-1.1, 1.9]$ which contains $f(X)$.

Elsewhere in this book, we do not distinguish between X and X' in relations such as (7.2.4) or (7.2.6). Instead, we replace X' by X and rely upon the reader to remember that, while $X' = X$, the two intervals are independent.

7.3 THE MULTIDIMENSIONAL CASE

Now let f be a function of n variables. Assume f has continuous partial derivatives of any necessary order with respect to each variable. We first expand f by a standard method and then by ways that produce better results in interval applications. We begin with the case $m = 0$. Thus, the remainder is in terms of first derivatives.

Let $\mathbf{x}$ and $\mathbf{y}$ be vectors of n components and let α be a scalar. We can view $f[\mathbf{x} + \alpha(\mathbf{y} - \mathbf{x})]$ as a function of the single variable α and expand f using (7.2.1). Expanding about $\alpha = 0$ and then setting $\alpha = 1$, we obtain

$$f(\mathbf{y}) = f(\mathbf{x}) + (\mathbf{y} - \mathbf{x})^T \mathbf{g}[\mathbf{x} + \xi(\mathbf{y} - \mathbf{x})]$$

where $0 \le \xi \le 1$ and where $\mathbf{g}$ is the gradient of f. Thus, the components of $\mathbf{g}$ are $g_i = \frac{\partial f}{\partial x_i}$ $(i = 1, \cdots, n)$.

If X_i is an interval containing both x_i and y_i $(i = 1, \cdots, n)$, then $x_i + \xi(y_i - x_i) \in X_i$. Therefore,

$$f(\mathbf{y}) \in f(\mathbf{x}) + (\mathbf{y} - \mathbf{x})^T \mathbf{g}(X_1, \cdots, X_n). \tag{7.3.1}$$

Note that all the arguments of $\mathbf{g}$ are intervals.

We now describe a method due to Hansen (1968) that permits some of these arguments to be replaced by real (i.e., noninterval) quantities. This means that the components of $\mathbf{g}$ are narrower intervals resulting in sharper bounds on $f(\mathbf{y})$, in general. See also Rokne and Bao (1987) and Bao and Rokne (1988).

We illustrate the method of derivation by considering the case $n = 3$. We first regard $f(y_1, y_2, y_3)$ as a function of y_3 only. Using (7.2.1) to expand about x_3, we obtain

$$f(y_1, y_2, y_3) = f(y_1, y_2, x_3) + (y_3 - x_3)g_3(y_1, y_2, \xi_3). \tag{7.3.2}$$

We now expand $f(y_1, y_2, x_3)$ about x_2 as a function of y_2 and obtain

$$f(y_1, y_2, x_3) = f(y_1, x_2, x_3) + (y_2 - x_2)g_2(y_1, \xi_2, x_3). \tag{7.3.3}$$

Finally, we expand $f(y_1, x_2, x_3)$ as a function of y_1 and obtain

$$f(y_1, x_2, x_3) = f(x_1, x_2, x_3) + (y_1 - x_1)g_1(\xi_1, x_2, x_3). \tag{7.3.4}$$

Combining equations (7.3.2), (7.3.3), and (7.3.4), we obtain

$$\begin{aligned} f(\mathbf{y}) =& f(\mathbf{x}) + (y_1 - x_1)g_1(\xi_1, x_2, x_3) + (y_2 - x_2)g_2(y_1, \xi_2, x_3) \\ & + (y_3 - x_3)g_3(y_1, y_2, \xi_3). \end{aligned} \tag{7.3.5}$$

If $\mathbf{x} \in \mathbf{x}^{\mathbf{I}}$ and $\mathbf{y} \in \mathbf{x}^{\mathbf{I}}$, then $\xi_i \in X_i$ $(i = 1, 2, 3)$.

In applications, we sometimes want a linear bound on $f(\mathbf{y})$ for all $\mathbf{y} \in \mathbf{x}^{\mathbf{I}}$. Therefore, we replace components of $\mathbf{y}$ in the arguments of the components of $\mathbf{g}$ by $\mathbf{x}^{\mathbf{I}}$ and we replace ξ_i $(i = 1, 2, 3)$ by the bounding interval X_i. We obtain

$$\begin{aligned} f(\mathbf{y}) \in& f(\mathbf{x}) + (y_1 - x_1)g_1(X_1, x_2, x_3) + (y_2 - x_2)g_2(X_1, X_2, x_3) \\ & + (y_3 - x_3)g_3(X_1, X_2, X_3). \end{aligned}$$

For n variables, the corresponding expression is

$$f(\mathbf{y}) \in f(\mathbf{x}) + \sum_{i=1}^{n} (y_i - x_i)g_i(X_1, \cdots, X_i, x_{i+1}, \cdots, x_n). \tag{7.3.6}$$

Compare this expression with (7.3.1). In (7.3.1), all the arguments of all components of $\mathbf{g}$ are intervals. In (7.3.6), a fraction $\frac{1}{2}\left(1 - \frac{1}{n}\right)$ are real. Therefore, an interval bound on $f(\mathbf{y})$ using (7.3.1) is generally not as narrow as the corresponding one computed using (7.3.6). The amount of work to compute the bound is the same in either case. This is because, to bound rounding errors, we treat real arguments as degenerate intervals.

To illustrate the superiority of (7.3.6) over (7.3.1), consider the function

$$f(x_1, x_2, x_3) = \frac{x_1}{x_2 + 2} + \frac{x_2}{x_3 + 2} + \frac{x_3}{x_1 + 2}. \tag{7.3.7}$$

Let $X_i = [-1, 1]$ $(i = 1, 2, 3)$. Using (7.3.1), we find $f(X_1, X_2, X_3) \subset [-6, 6]$. Using (7.3.6), we compute $[-4, 4]$, a much better result. The actual range of f over the given box is $[-3, 3]$. Even using (7.3.6), the result is generally not sharp because of dependence.

Note that if we evaluate f in its original form (7.3.7) (i.e.,without expanding) over $\mathbf{x}^{\mathbf{I}}$, we compute the sharp result $[-3, 3]$. For this example, the Taylor expansion yields a *wider* interval result than direct evaluation of f.

This exemplifies an unsolved problem in interval analysis. It is not known when a centered form or Taylor expansion yields a better or worse result than direct evaluation. Generally, they yield a sharper result when the width of the interval (or box) is small.

The width of the bound given by the right member of (7.3.6) depends on the order in which the variables are indexed. It is difficult to determine the best order in this regard.

We can use the same process of sequential expansion for higher order Taylor expansions. For example, a first order expansion (with error in term of second derivatives) yields

$$f(\mathbf{y}) \in f(\mathbf{x}) + (\mathbf{y} - \mathbf{x})^T \mathbf{g}(\mathbf{x}) + \frac{1}{2}(\mathbf{y} - \mathbf{x})^T \mathbf{H}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})(\mathbf{y} - \mathbf{x}) \tag{7.3.8}$$

where $\mathbf{H}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$ is an interval enclosure of the Hessian. For $n = 3$,

$$\mathbf{H}(\mathbf{x}, \mathbf{x}^{\mathbf{I}}) = \begin{bmatrix} h_{11}(X_1, x_2, x_3) & 0 & 0 \\ h_{21}(X_1, x_2, x_3) & h_{22}(X_1, X_2, x_3) & 0 \\ h_{31}(X_1, x_2, x_3) & h_{32}(X_1, X_2, x_3) & h_{33}(X_1, X_2, X_3) \end{bmatrix}$$

where

$$h_{ij}(\mathbf{x}) = \begin{cases} \frac{\partial^2 f}{\partial x_i^2} & \text{for } j = i \ \ (i = 1, \cdots, n), \\ \frac{2\partial^2 f}{\partial x_i \partial x_j} & \text{for } j < i \ \ (i = 2, \cdots, n; \ \ j = 1, \cdots, i-1), \\ 0 & \text{otherwise.} \end{cases}$$

We have chosen to write **H** as a lower triangular matrix, rather than a symmetric one, so that fewer terms occur in evaluating the bound on $f(\mathbf{y})$. The fewer terms in an expression, the less likely dependence is to cause loss of sharpness when evaluating the expression over a box.

In n dimensions, the arguments for the i-th column of **H** (on and below the diagonal) are $(X_1, ...X_i, x_{i+1}, \cdots, x_n)$ for all $i = 1, \cdots, n$.

So far in this section, we have been concerned with the number of real versus interval arguments in an expansion. We can also focus attention on the amount of computation required. Still another consideration is dependence. The form in which an expansion is written can be just as important as the proportion of real arguments. We now consider some examples.

Let f_1 and f_2 be functions of n variables. Suppose we wish to expand their product $f_{1\cdot 2} = f_1 f_2$. To discuss ways to obtain this expansion, we introduce some shorthand notation.

We let (i) denote $(X_1, \cdots, X_i, x_{i+1}, \cdots, x_n)$ for $i = 1, \cdots, n$. Thus, for example, $f_j(i)$ denotes $f_j(X_1, \cdots, X_i, x_{i+1}, \cdots, x_n)$. We could let (0) denote $(x_1, \cdots, x_n)$. However, we use the more common notation (x). We use $(\mathbf{x}^\mathbf{I})$ to denote $(X_1, \cdots, X_n)$ when convenient. However, when a summation index j in the symbol (j) takes the value n, then (n) also denotes $(X_1, \cdots, X_n)$.

Denote

$$g_{ij} = \frac{\partial f_i}{\partial x_j} \ \ (i = 1, 2; \ \ j = 1, \cdots, n).$$

If we expand f_i $(i = 1, 2)$ using (7.3.6), we obtain

$$f_i(\mathbf{y}) \in f_i(\mathbf{x}) + \sum_{j=1}^{n} (y_j - x_j) g_{ij}(j). \tag{7.3.9}$$

This expansion is valid for $x_i \in X_i$ and $y_i \in X_i$ $(i = 1, \cdots, n)$. Using this same method to expand $f_{1\cdot 2}$ we obtain

$$f_{1\cdot 2}(\mathbf{y}) \in f_{1\cdot 2}(\mathbf{x}) + \sum_{j=1}^{n}(y_j - x_j)[f_1(j)g_{2j}(j) + g_{1j}(j)f_2(j)]. \tag{7.3.10}$$

Now suppose that we simply take the product of the expanded forms of f_1 and f_2 as given by (7.3.9). By combining terms appropriately, we obtain

$$\begin{aligned} f_{1\cdot 2}(\mathbf{y}) \in & f_{1\cdot 2}(\mathbf{x}) + f_1(\mathbf{x})\sum_{j=1}^{n}(y_j - x_j)g_{2j}(j) \\ & + f_2(\mathbf{x}^{\mathbf{I}})\sum_{j=1}^{n}(y_j - x_j)g_{1j}(j). \end{aligned} \tag{7.3.11}$$

The factor $f_2(\mathbf{x}^{\mathbf{I}})$ in the right member occurs because we have replaced the argument $\mathbf{y}$ by $\mathbf{x}^{\mathbf{I}}$ in the (unexpanded) function $f_2(\mathbf{y})$. Other similar forms are possible by combining terms in other ways

Let us now compare the two forms (7.3.10) and (7.3.11). The latter has advantages that we list and then discuss.

(a) Fewer multiplications are required. Once the required function evaluations are done, (7.3.10) requires $3n + 1$ multiplications while (7.3.11) requires only $2n + 3$.

(b) A given function is evaluated using fewer sets of arguments. In (7.3.10), f_1 and f_2 must each be evaluated for n different sets of arguments. In (7.3.11), f_1 and f_2 are each evaluated for only one set of arguments.

(c) The form given in (7.3.11) is factored. That is, dependence is reduced.

The functions $g_{ij}(j)$ $(i = 1, 2;\ j = 1, \cdots, n)$ are evaluated with the same sets of arguments for either form of expansion. The difference in real versus interval arguments occurs only in the arguments of f. For (7.3.10), the ratio of the number of real to the number of interval arguments is $1 - \frac{2}{n+1}$. For (7.3.11), it is 1. For moderately large n, these ratios are much the same.

Despite the fact that (7.3.11) has more interval arguments than (7.3.10), it has a more favorable factored form. This permits it to provide sharper results by taking advantage of subdistributivity. (See Section 3.3.) In general, (7.3.11) can be the preferable form despite the fact that it contains more interval arguments than (7.3.10). This is especially true if computation speed is important.

As an example, let $f_1(\mathbf{x}) = x_1^2 + x_2^2$ and $f_2(\mathbf{x}) = \frac{x_1}{x_2}$. Let us use the two forms to evaluate expansions of $f_{1 \cdot 2}(\mathbf{x}) = f_1(\mathbf{x}) f_2(\mathbf{x})$ for $X_1 = [-1, 3]$ and $X_2 = [1, 3]$. From (7.3.10), we compute $f_{1 \cdot 2}(\mathbf{y}) \in [-88.5, 93.5]$. From (7.3.11), we compute $f_{1 \cdot 2}(\mathbf{y}) \in [-71.5, 76.5]$. Thus (7.3.11) produces sharper bounds than (7.3.10) even though a larger proportion of its arguments are intervals.

Less dependence is what enables (7.3.11) to give sharper results in this example. For narrow interval arguments, dependence is a less important concern. Therefore, the relative performance of the two expansion methods can depend on interval widths. The number of variables also affects the choice of form.

Instead of expanding the quotient of two functions, we can use the quotient of their expansions in a similar way. Thus, we can write the expansion of $f_{1/2}(\mathbf{x}) = \frac{f_1(\mathbf{x})}{f_2(\mathbf{x})}$ as

$$\begin{aligned} f_{1/2}(\mathbf{y}) = f_{1/2}(\mathbf{x}) &+ \frac{1}{f_2(\mathbf{x}^\mathbf{I})} \sum_{j=1}^{n} (y_j - x_j) g_{1j}(j) \\ &- \frac{f_1(\mathbf{x})}{f_2(\mathbf{x}) f_2(\mathbf{x}^\mathbf{I})} \sum_{j=1}^{n} (y_j - x_j) g_{2j}(j). \end{aligned} \tag{7.3.12}$$

We have argued that when expanding a product (quotient) of two functions in terms of first order derivatives, it is better to use a product (quotient) of expansions rather than to directly expand the product (quotient). This is also true when the expansions are in terms of slopes.

It also seems likely that functions more complicated than simple products or quotients should be expanded piecemeal rather than as a single function. This is likely to be true for higher order expansions as well.

7.4 THE JACOBIAN AND HESSIAN

Consider a vector **g** that is the gradient of a function f of n variables. The Jacobian **J** of **g** has elements

$$J_{ij} = \frac{\partial g_i}{\partial x_j} = \frac{\partial^2 f}{\partial x_i \partial x_j} \ (i, j = 1, \cdots, n).$$

As a noninterval matrix, **J** is symmetric. To compute **J**, we need only compute the lower (or upper) triangle and use symmetry to get the elements above the diagonal.

But the situation differs in the interval case if we want to have some noninterval arguments as discussed in Section 7.3. Suppose we expand each component of g as in (7.3.6) using the same pattern of real and interval arguments given therein. The resulting Jacobian is not symmetric. A real argument of J_{ij} might be an interval argument in J_{ji}. To compute an interval enclosure $\mathbf{J}^{\mathbf{I}}$ in this case, we must compute all n^2 elements. We can have symmetry by using intervals for all the arguments. But then the interval elements of $\mathbf{J}^{\mathbf{I}}$ are wider than necessary.

In this section, we consider how to have symmetry without using all interval arguments. We also consider how to compute both the Hessian of f and the Jacobian of the gradient of f. In the interval case, they can differ (if we want some arguments to be real) because the pattern of real versus interval arguments can differ.

Consider the case $n = 2$. Let us expand g_1 with respect to x_2 and then x_1. Let us expand g_2 in the opposite order. Then

$$\mathbf{g}(\mathbf{y}) \in \mathbf{g}(\mathbf{x}) + \mathbf{J}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})(\mathbf{y} - \mathbf{x}) \qquad (7.4.1)$$

where

$$\mathbf{J}(\mathbf{x}, \mathbf{x}^{\mathbf{I}}) = \begin{bmatrix} J_{11}(X_1, x_2) & J_{12}(X_1, X_2) \\ J_{21}(X_1, X_2) & J_{22}(x_1, X_2) \end{bmatrix}.$$

If $\mathbf{g}$ is the gradient of f, then $\frac{\partial g_1}{\partial x_2} = \frac{\partial g_2}{\partial x_1}$ and, hence, $J_{21}(X_1, X_2) = J_{12}(X_1, X_2)$ and $\mathbf{J}(\mathbf{x}, \mathbf{x}^\mathbf{I})$ is symmetric. It can be shown that, in general, $\mathbf{J}(\mathbf{x}, \mathbf{x}^\mathbf{I})$ cannot be made symmetric for any $n > 2$ and still have the maximum possible number of real arguments.

However, we can have symmetry without replacing all arguments by intervals; but, we must use fewer than the maximum possible number of real arguments. As an example, consider the case $n = 3$. Expand g_1 and g_2 in the order (of indices of variables) 3, 2, 1. Expand g_3 in the order 1, 2, 3. Then

$$\mathbf{J}(\mathbf{x}, \mathbf{x}^\mathbf{I}) = \begin{bmatrix} J_{11}(X_1, x_2, x_3) & J_{12}(X_1, X_2, x_3) & J_{13}(X_1, X_2, X_3) \\ J_{21}(X_1, x_2, x_3) & J_{22}(X_1, X_2, x_3) & J_{23}(X_1, X_2, X_3) \\ J_{31}(X_1, X_2, X_3) & J_{32}(x_1, X_2, X_3) & J_{33}(x_1, x_2, X_3) \end{bmatrix}.$$

In this form, $\mathbf{J}(\mathbf{x}, \mathbf{x}^\mathbf{I})$ is not symmetric. However, if we replace $J_{21}(X_1, x_2, x_3)$ by $J_{21}(X_1, X_2, x_3)$ and $J_{32}(x_1, X_2, X_3)$ by $J_{32}(X_1, X_2, X_3)$ we gain symmetry and still have real arguments for some element of $\mathbf{J}(\mathbf{x}, \mathbf{x}^\mathbf{I})$.

We have no general rule for retaining the maximum number of real arguments while gaining symmetry in this way.

In later chapters, we sometimes want to use both of the expansions

$$f(\mathbf{y}) \in f(\mathbf{x}) + (\mathbf{y} - \mathbf{x})^T \mathbf{g}(\mathbf{x}) + \frac{1}{2}(\mathbf{y} - \mathbf{x})^T \mathbf{H}(\mathbf{x}, \mathbf{x}^\mathbf{I})(\mathbf{y} - \mathbf{x})$$

and

$$\mathbf{g}(\mathbf{y}) \in \mathbf{g}(\mathbf{x}) + \mathbf{J}(\mathbf{x}, \mathbf{x}^\mathbf{I})(\mathbf{y} - \mathbf{x}).$$

The former is a repeat of (7.3.8) and the latter is a repeat of (7.4.1). Functionally, $\mathbf{H}(\mathbf{x}, \mathbf{x}^\mathbf{I})$ and $\mathbf{J}(\mathbf{x}, \mathbf{x}^\mathbf{I})$ in these equations are the same matrix. However, if we wish to have real (instead of interval) arguments everywhere possible in their matrix elements, then they become different matrices when evaluated. This is because their patterns of real and interval arguments differ.

Suppose, we use an expansion of the form (7.3.6) for each component of $\mathbf{g}$. Then the arguments of J_{ij} are $(X_1, ...X_j, x_{j+1}, \cdots, x_n)$ for all $i = 1, \cdots, n$. These arguments are the same as those for the elements of the lower triangle of the Hessian **H** as derived in Section 7.3. That is, if we compute the Jacobian **J** as just indicated, we have the necessary data to form the Hessian. To get the elements of **H** below the diagonal, we need only multiply the corresponding elements of **J** by 2.

If we replace certain real arguments of $\mathbf{J}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$ by intervals to get symmetry, the new lower triangle of $\mathbf{J}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$ still yields a lower triangle that can be used to determine $\mathbf{H}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$. That is, $\mathbf{H}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$ need not be computed separately despite such changes in $\mathbf{J}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$.

7.5 AUTOMATIC DIFFERENTIATION

Nonlinear equations and optimization problems can be solved without resorting to expansions using derivatives or slopes. For example, see the discussion of hull consistency in Chapter 10. However, use of expansions can speed the solution process. This is especially true when the region of search becomes asymptotically small. In this section, we briefly discuss how derivatives can be automatically computed. This is an extremely valuable asset because it not only saves the human effort of differentiating and coding of derivatives, but avoids possible errors in doing so.

In Section 7.12, we give a similar discussion for generation of slopes. Divided differences with error bounds could be used but this approach should be avoided.

One way to obtain derivatives is to first use a program such as MACSYMA, Maple, Mathematica, or REDUCE to perform the algebra of the differentiation process. Such a process is sometimes called *symbolic differentiation*. Analytic expressions for derivatives of a given function are produced automatically. Coding these expressions to evaluate the derivatives can also be automated.

If properly implemented, this can be the more desirable way to compute derivatives in interval applications. This is because the symbolic expressions can be written to reduce the effect of dependence. However, this approach can often lead to lengthy formulas for derivatives that require considerable computation to evaluate. The complexity of such formulas can often be greatly reduced with an algorithm that identifies and eliminates multiple common subexpressions. Proper implementation of symbolic differentiation in a form suitable for use in interval programs is not generally available.

A more commonly used procedure is *automatic differentiation.* This is a procedure (in two forms with variations) for numerically generating values of derivatives without obtaining their analytic expressions. It is an idea that has occurred independently to various people. One of them was Moore (1965) (see also Moore (1966)), who was the first to apply it to interval analysis. However, earlier references to noninterval applications can be found in Griewank (1989, 2000).

Rall (1969, 1981, 1983, 1984, 1987) and his colleagues have made considerable use of automatic differentiation in interval applications. Some other publications on the subject are Iri (1984) and Speelpenning (1980). For a thorough discussion of automatic differentiation in noninterval applications, see Griewank (2000).

Automatic differentiation is very efficient (see below). Some authors have pointed out that it is much more efficient than symbolic differentiation. For example, see Griewank (1991, 2000). However, a proper comparison for interval applications must take into account the fact that, a symbolic expression for a derivative can be written to reduce the effect of dependence. The sharper bounds available using symbolic differentiation might provide better overall efficiency in solving a problem by interval methods.

Assume that a computer program exists for evaluating a given function f. Automatic differentiation can be performed using a precompiler that generates code for computing the derivative of the function as defined by the code to evaluate it. A better method is to implement a user-defined type together with operator overloading. This is called a "class" in the language C++ and a "module" in Fortran.

We now briefly discuss the so-called "forward form" of automatic differentiation to illustrate the ideas involved. A "backward form" is discussed in various publications. For a brief discussion, see Kearfott (1996). For a complete discussion of both forms and variations, see Griewank (2000).

The steps to evaluate a rational function involve only the arithmetic operations of addition, subtraction, multiplication and division. Raising a quantity to a power can also be included in the set of operations. Each of the four basic operations involves two quantities that either have been computed in a previous step or are primitives such as a constant or a single variable.

Consider the one-variable case. The derivative of the result of a computational step involving functions f_1 and f_2 can be obtained using Table 7.1. The derivative of each primitive and each irrational function used must

Computational step	**Step result derivative**
$f_1 \pm f_2$	$f_1' \pm f_2'$
$f_1 f_2$	$f_1 f_2' + f_1' f_2$
$\frac{f_1}{f_2}$	$\frac{f_1' f_2 - f_1 f_2'}{f_2^2}$

Table 7.1: Computational Step Derivatives

be known.

We illustrate the method with a simple example. Consider the function $f(x) = \frac{1}{x} + x \sin x$. Define the primitives $f_1 = 1$ and $f_2 = x$ and their derivatives $f_1' = 0$ and $f_2' = 1$. A code to evaluate f might involve generation of values on the functions

$$\begin{aligned} f_3 &= \frac{f_1}{f_2}, \\ f_4 &= \sin x, \\ f_5 &= f_2 f_4, \\ f &= f_3 + f_5. \end{aligned}$$

In some manner or other, we must know that the derivative of $\sin x$ is $\cos x$. Using the definitions of the functions f_3, f_4, and f_5 and the rules in Table 7.1, we obtain

$$\begin{aligned} f_3' &= \frac{f_1' f_2 - f_1 f_2'}{f_2^2}, \\ f_5' &= f_2 f_4' + f_2' f_4, \end{aligned}$$

and

$$f' = f_3' + f_5'.$$

Since we know the primitives f_1 and f_2 and their derivatives and the special function $f_4 = \sin x$ and its derivative, we can compute the derivative of f using the above equations. Code to compute the right-hand side of each equation can be generated automatically. Note that the evaluation is numerical, not analytic.

Hopefully, the readers understand what is involved in automatic differentiation from this abbreviated introduction. Details can be found in the references cited above. We merely wish to make readers aware that automatic procedures exist that are alternatives to symbolic differentiation.

The procedure we have described is suitable for computing the derivative of a function of a single variable. It can also serve for the multidimensional case. However, as we showed in Section 7.3, it is better to compute the expansion of a product or quotient by using the product or quotient of the individual expansions. This alternative method also can be incorporated into an automatic procedure.

Automatic differentiation can be used, for example, to evaluate the gradient of a multivariable function. Let r denote the ratio of the amount of work to evaluate the gradient of a function and the work to evaluate the function itself. Wolfe (1982) observed that the value of r is usually around 1.5 in practice.

Bauer and Strassen (1983) used complexity theory to show that, for a certain way of counting arithmetic operations, the theoretical value of r is at most 3. For another proof, see Iri (1984). Griewank (1989) gives an explicit algorithm and shows that, for it, $r \leq 5$.

We now consider an example to illustrate the advantage of symbolic over automatic differentiation from consideration of dependence. Let $f(x) = \frac{u(x)}{v(x)}$ where $u(x) = x^2 + x$ and $v(x) = x^2 - x$. The derivative is

$$f'(x) = \frac{v(x)u'(x) - u(x)v'(x)}{v(x)^2}.$$

Using automatic differentiation, we get, in effect,

$$f'(X) = \frac{(X^2 - X)(2X + 1) - (X^2 + X)(2X - 1)}{(X^2 - X)^2}.$$

For $X = [4, 6]$, automatic differentiation produces $f'(X) = [-3.72, 2.76]$.

But, the numerator in this expression for $f'(X)$ can be simplified to $-2X^2$. If we do so, and evaluate the result, we compute the better result $[-0.72, -0.0312]$.

7.6 SHARPER BOUNDS USING TAYLOR EXPANSIONS

Expansions that generally give sharper interval bounds than (7.2.3) or (7.3.6) can be computed at the expense of additional computing. We simply use each occurrence of a given variable as a separate variable and use the sequential expansion process described in Section 7.3.

For example, we rewrite $F(x) = x^2 \sin x$ in the form

$$f(x_1, x_2, x_3) = x_1 x_2 \sin x_3.$$

We use the expansion (7.3.6) and then set $x_1 = x_2 = x_3 = x$ and $y_1 = y_2 = y_3 = y$. We obtain

$$\begin{aligned} f(y, y, y) = {} & f(x, x, x) \\ & + (y - x)[g_1(\xi_1, x, x) + g_2(y, \xi_2, x) + g_3(y, y, \xi_3)]. \end{aligned}$$

If $x \in X$ and $y \in X$, then $\xi_i \in X$ $(i = 1, 2, 3)$. Hence, since $f(x, x, x) = F(x)$, we have

$$F(y) \in F(x) + (X - x)(x \sin x + X_1 \sin x + X_1 X_2 \cos X_3). \tag{7.6.1}$$

The intervals X_1, X_2, and X_3 each equal X. However, they must be treated as independent because they represent bounds on the separate quantities ξ_i $(i = 1, 2, 3)$. For example, we cannot write $X_1 X_2$ as X^2.

If we simply use (7.2.4) to expand F, we obtain

$$F(X) \subset F(x) + (X - x)(2X \sin X + X^2 \cos X) \tag{7.6.2}$$

which produces a wider interval bounding $F(X)$, in general. For example, if $X = [0, 1]$, and if we choose $x = \mathrm{m}(X) = 0.5$, then (7.6.1) yields $F([0, 1]) \subset [-0.7398, 0.9795]$ while (7.6.2) yields $F([0, 1]) \subset [-1.2217, 1.4614]$. The correct range of F over $X = [0, 1]$ is $[0, \sin(1)]$ which to four significant digits is $[0, 0.8415]$.

The extra accuracy is obtained at the cost of some extra work. Note that (7.6.1) is somewhat more complicated than (7.6.2). For more complicated functions, the extra work is greater.

This procedure was introduced and discussed in more detail by Hansen (1978a). Alefeld (1981) showed that when F is a polynomial, the same improved expansion can be obtained by more direct means.

Note that procedures such as this can never produce sharper results than the use of monotonicity as described in Section 3.4. For example, the function $F(x) = x^2 \sin x$ discussed above is easily shown to be monotonic over [0, 1] by evaluating its derivative over this interval. Therefore, the method of Section 3.6 is applicable and yields exact bounds on the range of F over [0, 1].

7.7 EXPANSIONS USING SLOPES

Krawczyk and Neumaier (1985) described a systematic way to obtain expansions similar to those discussed in Section 7.6. This is done by use of slopes that we define and discuss below. Their method is applicable for a rational function of a single variable. Neumaier (1989) extended the procedure to the multidimensional case which we discuss in Section 7.9. The procedure has also been extended to the case of irrational functions which we discuss in Section 7.8. See Rump (1996). Ratz (1998) extended the use of slopes to nonsmooth functions. See Section 7.11. Automatic evaluation of slope is discussed in Section 7.12. Other similar expansions are mentioned in Section 7.13. Automatic evaluation of slopes is discussed in Section 7.12. Other similar expansions are mentioned in 7.13.

We first discuss a function f of a single variable. We want an expansion of the form

$$f(y) = f(x) + g(x, y)(y - x). \tag{7.7.1}$$

This equation is an identity if

$$g(x, y) = \frac{f(y) - f(x)}{y - x}. \tag{7.7.2}$$

Note that equation (7.7.1) is the same as Moore's centered form described in Section 3.3.The function $g(x, y)$ is called the *slope function.* It is the slope of the chord joining the ordinates of f at x and y. Its limit as y approaches x is the slope of the tangent to f at x. That is, in the limit, the slope is the derivative $f'(x)$.

We can regard (7.7.1) as a factorization of $f(y) - f(x)$. For example, if $f(x) = x^2$, then $g(x, y) = y + x$ and (7.7.1) can be written as

$$y^2 - x^2 = (y + x)(y - x)$$

Given an interval X, inclusion isotonicity of containment sets (See Lemma 4.8.8) assures that

$$f(y) \in f(x) + g(x, X)(y - x) \tag{7.7.3}$$

for arbitrary x and all $y \in X$. Therefore,

$$f(X) \subset f(x) + g(x, X)(X - x).$$

The right member does not generally provide sharp bounds on $f(X)$ despite the fact that it arises from the identity (7.7.1). To illustrate this fact, let $f(x) = x^2$. We obtain

$$X^2 \subset x^2 + (X + x)(X - x).$$

For $X = [-1, 3]$ and $x = 1$, the left member is $[0, 9]$, but the right member is the wider interval $[-7, 9]$. Dependence has caused widening of the latter result.

If f is rational, we can analytically divide $f(y) - f(x)$ by $y - x$ and obtain $g(x, y)$ explicitly as we did above for $f(x) = x^2$. When f is not rational, we can sometimes obtain $g(x, X)$ numerically. See Section 7.8.

The relation (7.7.3) is a (first order) slope expansion. Compare it with the corresponding Taylor form

$$f(y) \in f(x) + f'(X)(y - x), \tag{7.7.4}$$

which is valid provided both x and y are in X. However, in the slope expansion (7.7.3), x need not be in X. When we do have $x \in X$ and $y \in X$, if follows that $g(x, X) \in f'(X)$. Therefore, the slope expansion provides at least as narrow a bound on $f(y)$ for $y \in X$ as does the Taylor expansion.

In effect, some of the occurrences of the interval X in the expression of the derivative $f'(X)$ have been replaced in $g(x, X)$ by the degenerate interval x. Therefore, (7.7.3) generally produces a narrower interval bound on $f(y)$ than (7.7.4).

In Section 7.2, we noted that from (7.2.3) (which is the same as (7.7.4)), the Taylor expansion gives the relation

$$f(X) \subset f(x) + f'(X')(X - x)$$

where $X' = X$ but that X' and X are independent intervals. In the corresponding relation

$$f(X) \subset f(x) + g(x, X)(X - x)$$

from (7.7.3) for slopes, the interval X in $g(x, X)$ and in the factor $X - x$ are identically the same. This is another advantage of a slope expansion over a Taylor expansion because there is an opportunity to analytically reduce multiple occurrences of the interval X.

When dependence does not cause widening, the interval value of the slope expression $f(x)+g(x, X)(X-x)$ provides sharp bounds on the range of y for $y \in X$. The corresponding Taylor expression $f(x)+f'(X')(X-x)$ does not. However, dependence generally prevents us from computing sharp results when evaluating the former expression.

For a small box, the (exact) slope expression generally yields sharper bounds on $f(X)$ than direct evaluation of $f(X)$ because $f(x)$ is a good approximation to values of f for all x in the (small) box. Therefore, loss of sharpness due to dependence occurs only in $g(x, X)(X - x)$ which is relatively smaller in magnitude than $f(x)$. Note that for rational functions, the slope form is the same as rewriting f in centered form.

For the above example in which $f(x) = x^2$, we have $g(x, X) = x + X$. For $X = [1, 2]$ and $x = 2$, we find $g(x, X) = [3, 5]$. For the corresponding Taylor form, (7.7.4), we have $f'(X) = 2X = [2, 6]$, which is a wider interval. Note that if we write $f'(x)$ as $X + X$, then the function $g(x, X) = x + X$ can be viewed as being obtained from $f'(X)$ by replacing one occurrence of X by x. As a result, $g(x, X)$ is a narrower interval than $f'(X)$.

For any function, the wider the interval, the more the width of $f'(X)$ exceeds that of $g(x, X)$. If we let the width of the interval approach zero asymptotically, the difference between derivatives and slopes disappears.

The slope of a composite function is easily obtained. Suppose $f(x) = v(u(x))$.Then

$$g(x, y) = \frac{v(u(y)) - v(u(x))}{y - x} \tag{7.7.5a}$$

$$= \left(\frac{v(u(y)) - v(u(x))}{u(y) - u(x)}\right)\left(\frac{u(y) - u(x)}{y - x}\right). \tag{7.7.5b}$$

Therefore, the slope of $v(u(x))$ is the product of two slopes. The first factor is the slope of v with $u(y)$ regarded as the variable and $u(x)$ regarded as the fixed point. The other factor is the slope of u.

We give an example illustrating the computation of the slope of a composite function at the end of Section 7.8.

An extension of (7.7.5b) gives a chain rule for slopes similar to that for derivatives. For example, if

$$f(x) = w(u(x), v(x)),$$

then

$$\begin{aligned}g(x,y) &= \frac{w(u(y),v(y)) - w(u(x),v(x))}{y-x} \\ &= \left(\frac{w(u(y),v(y)) - w(u(x),v(y))}{u(y)-u(x)}\right)\left(\frac{u(y)-u(x)}{y-x}\right) \\ &\quad + \left(\frac{w(u(x),v(y)) - w(u(x),v(x))}{v(y)-v(x)}\right)\left(\frac{v(y)-v(x)}{y-x}\right).\end{aligned}$$

In the remaining chapters of this book, we discuss the use of derivatives in solving optimization problems. We do so since derivatives are more familiar than slopes. In practice, use slopes rather than derivatives whenever possible.

Slope expansions have a useful aspect. The mean value theorem does not hold for functions of a complex variable. Let z and w be complex points in a box $\mathbf{z}^{\mathbf{I}}$ in the complex plane. There need not exist a point ξ in $\mathbf{z}^{\mathbf{I}}$ such that

$$f(w) = f(z) + (w-z)f'(\xi). \tag{7.7.6}$$

Therefore, we cannot obtain the relation

$$f(w) \in f(z) + (w-z)f'(Z)$$

corresponding to (7.2.4) which holds for real variables.

However, the identity

$$f(w) = f(z) + (w-z)g(z,w)$$

where $g(z,w)$ is a complex version of the slope yields

$$f(w) \in f(z) + (w-z)g(z,\mathbf{z}^{\mathbf{I}})$$

for arbitrary z and any $w \in \mathbf{z}^{\mathbf{I}}$.

This enables us to derive an interval Newton method for finding complex zeros of a complex function. We do not do so; but the derivation follows that for the real case in Section 11.2.

7.8 SLOPES FOR IRRATIONAL FUNCTIONS

We have noted that for any rational function, $f(x)$, we can analytically divide $f(y) - f(x)$ by $y - x$. This can also be done for certain algebraic functions. Note that, for $f(x) = x^{\frac{1}{n}}$ $(n = 2, 3, 4, \cdots)$ we have

$$f(y) - f(x) = \frac{y - x}{\sum_{k=0}^{n-1} x^{\frac{k}{n}} y^{\frac{n-1-k}{n}}}.$$

In particular, for $n = 2$, we obtain the slope of the square root function. That is, if $f(x) = x^{\frac{1}{2}}$, then

$$g(x, y) = \frac{1}{x^{\frac{1}{2}} + y^{\frac{1}{2}}}.$$

However, such analytic division is not possible when f is a transcendental function. Nevertheless, we can compute numerical values of slopes for certain transcendental functions; and we can compute bounds on slopes for others. What we need for a given function f is to be able to compute a value, or at least, a bound, for the ratio (i.e., the slope) $\frac{f(y)-f(x)}{y-x}$ over an interval.

Let x be fixed and consider $g(x, y)$ as a function of y. Rump (1996) (see also Ratz (1998)) proves that $g(x, y)$ is a monotonic function of y if $f(y)$ is convex or concave. His proof is as follows. Assume that $f(y)$ is continuously differentiable. Then

$$\frac{\partial}{\partial y} g(x, y) = \frac{\partial}{\partial y}\left(\frac{f(y) - f(x)}{y - x}\right) = \frac{(y - x)f'(y) - f(y) + f(x)}{(y - x)^2}.$$

If f is convex, the numerator of this quotient is ≥ 0. If f is concave, it is ≤ 0. That is, $g(x, y)$ is a monotonically nondecreasing function of y if f is convex, and is a monotonically nonincreasing function of y if f is concave.

If f is convex over an interval X, it follows that

$$g(x, X) = \left[\frac{f(\underline{X}) - f(x)}{\underline{X} - x}, \frac{f(\overline{X}) - f(x)}{\overline{X} - x}\right] \tag{7.8.1}$$

and if f is concave,

$$g(x, X) = \left[\frac{f(\overline{X}) - f(x)}{\overline{X} - x}, \frac{f(\underline{X}) - f(x)}{\underline{X} - x} \right]. \tag{7.8.2}$$

If x is an endpoint of X, then an endpoint of $g(x, X)$ in these expressions becomes an indeterminate form whose value is the derivative $f'(x)$.

Even though x is real, the value of $f(x)$ is an interval because rounded interval arithmetic must be used to bound its value. This must be taken into account when determining the slope. It is only a slight complication.

If $|\underline{X} - x|$ or $|\overline{X} - x|$ is small, rounding errors can be large when computing the slope using (7.8.1) or (7.8.2). When one or both of these quantities become too small, it is best to use a derivative rather than a slope when the slope is to be computed as a difference quotient.

Let ε denote the smallest positive machine number such that $1 + \varepsilon$ is represented as a machine number greater than 1 in the number system used on the relevant computer. A standard rule when computing difference quotients as approximations for derivatives has been to choose the difference in variable values to be greater than $\varepsilon^{\frac{1}{2}}$. Following this rule, we use a derivative rather than a slope (expressed in terms of a difference quotient) if $|\underline{X} - x| < \varepsilon^{\frac{1}{2}}$ of if $|\overline{X} - x| < \varepsilon^{\frac{1}{2}}$.

Note that e^x is a convex function of x. So is x^n for n even or for n odd if $x \geq 0$. Also, $\ln(x)$ is concave and so is x^n when n is odd and $x \leq 0$.

The slopes of some functions can be obtained because they are compositions of other functions whose slopes are known. For example, the inverse hyperbolic functions can be expressed in terms of square roots and the logarithm. An example is $\operatorname{arcsinh}(x) = \ln[x + (x^2 + 1)^{\frac{1}{2}}]$.

Use of monotonicity can also be used to reduce the effort to obtain the slope for the function $f(x) = x^n$. Its slope can be expressed as

$$\frac{y^n - x^n}{y - x} = \sum_{k=0}^{n-1} x^k y^{n-1-k}. \tag{7.8.3}$$

It requires quite a bit of computing to evaluate this sum if n is large. However, if n is even and x is an interior point of an interval X, then the slope is

$$g(x, X) = \left[\frac{\underline{X}^n - x^n}{\underline{X} - x}, \frac{\overline{X}^n - x^n}{\overline{X} - x} \right]. \tag{7.8.4}$$

This form requires less computing for $n > 4$.

It can be advantageous to use (7.8.3) rather than (7.8.4) because the former can be manipulated analytically before numerical values are inserted for evaluation. For example, consider the function

$$f(x) = x^4 + 3x^3 - 96x^2 - 388x + 480$$

which we consider again in Section 9.9. If we determine its slope analytically using (7.8.4), we can collect terms and write the slope as

$$g(x, X) = X^3 + (x + 3)X^2 + (x^2 + 3x - 96)X$$
$$+ x^3 + 3x^2 - 96x - 388.$$

Let $X = [0, 4]$ and $x = 2$. Evaluating $g(x, X)$ using Horner's rule, we obtain $g(x, X) = [-904, -560]$. If we use (7.8.4) to compute the slope, we are not able to collect terms; and we obtain $g(x, X) = [-1043, -271]$. this result is wider than the previous one by a factor of approximately 2.2.

Suppose that a given function $f(x)$ is not convex or concave over an interval X. But, suppose we can subdivide X into two subintervals X_1 and X_2 such that $X = X_1 \cup X_2$ and $f(x)$ is convex for $x \in X_1$ and concave for $x \in X_2$. The slope of $f(x)$ over X is

$$g(x, X) = g(x, X_1) \cup g(x, X_2).$$

The intervals $g(x, X_1)$ and $g(x, X_2)$ can be determined using monotonicity as described above. Therefore, we can determine $g(x, X)$. Note that x is arbitrary; but the same value must be used in computing both $g(x, X_1)$ and $g(x, X_2)$.

This approach can be generalized by subdividing X into more than two subintervals. This allows us to determine the slope of functions such as $\sin(x)$.

Suppose we want the slope of a function over an interval X and it contains a transcendental subfunction $u(x)$ and we do not know its slope $g_u(x, X)$. We can replace $g_u(x, X)$ by a derivative that bounds the slope. However, the argument of the derivative must be chosen correctly. Recall that for a slope, the fixed point x need not be in X. The argument of the derivative that bounds the slope must contain both x and X. Let X' be the smallest interval containing x and X. Then $g_u(x, X) \subset \frac{d}{dx}u(X')$. Therefore, when computing the slope of the original function, we can replace the slope of $u(x)$ by $\frac{d}{dx}u(X')$.

To illustrate the computation of the slope of an irrational composite function, let us find the slope of $f(x) = \exp(x^2 + x)$ over an interval X. Define $u(x) = x^2 + x$ and $v(u) = e^u$. To obtain the slope of $u(x)$, we divide $u(y) - u(x)$ by $y - x$ and get $x + y + 1$. Thus

$$g_u(x, X) = x + X + 1. \tag{7.8.5}$$

The exponential function is convex. Therefore, from (7.8.1),

$$g_v[u(x), u(X)] = \frac{e^{u(X)} - e^{u(x)}}{u(X) - u(x)} = \left[\frac{e^{\underline{u}} - e^{u(x)}}{\underline{u} - u(x)}, \frac{e^{\overline{u}} - e^{u(x)}}{\overline{u} - u(x)}\right] \tag{7.8.6}$$

where $\underline{u}$ and $\overline{u}$ are the endpoints of $u(X)$. That is, $u(X) = [\underline{u}, \overline{u}]$.

For a general function u, we might not be able to determine $\underline{u}$ and $\overline{u}$ sharply. However, we can compute bounds on $u(X)$ by simply evaluating u over X. The result might not be sharp because of dependence. Therefore, the bounds on the slope of f might not be sharp. If u is monotonic over X, this fact can be used to compute $u(X)$ sharply. See Section 3.6. Since we use the slope of u to obtain an expression for the slope of f, we can also bound $u(X)$ using the slope expansion

$$u(X) \subset u(x) + g_u(x, X)(X - x).$$

In our case, we can write $u(X) = (X + \frac{1}{2})^2 - \frac{1}{4}$ and compute $u(X)$ sharply because X occurs only once in this expression for $u(X)$. (See Section 2.4.)

From (7.7.5b), the slope of f is

$$g_f(x, X) = g_v[u(x), u(X)]g_u(x, X).$$

Therefore, (7.8.3) or (7.8.4) yield the desired slope of f.

7.9 MULTIDIMENSIONAL SLOPES

Multidimensional slopes can be obtained using the sequential expansion procedure for derivatives discussed in Section 7.3. See Equations (7.3.2) through (7.3.5). Consider the three dimensional case. If we regard $f(y_1, y_2, y_3)$ as a function of y_3, then

$$f(y_1, y_2, y_3) = f(y_1, y_2, x_3) + (y_3 - x_3)g_3(y_1; y_2; x_{3,} y_3)$$

where

$$g_3(y_1; y_2; x_{3,} y_3) = \frac{f(y_1, y_2, y_3) - f(y_1, y_2, x_3)}{y_3 - x_3}.$$

We now expand $f(y_1, y_2, x_3)$ as a function of y_2 and obtain

$$f(y_1, y_2, x_3) = f(y_1, x_2, x_3) + (y_2 - x_2)g_2(y_1; x_2, y_2; x_3)$$

where

$$g_2(y_1; x_2, y_2; x_3) = \frac{f(y_1, y_2, x_3) - f(y_1, x_2, x_3)}{y_2 - x_2}.$$

Finally, we expand $f(y_1, x_2, x_3)$ as a function of x_1 and obtain

$$f(y_1, x_2, x_3) = f(x_1, x_2, x_3) + (y_1 - x_1)g_1(x_1, y_1; x_2; x_3)$$

where

$$g_1(x_1, y_1; x_2; x_3) = \frac{f(y_1, x_2, x_3) - f(x_1, x_2, x_3)}{y_1 - x_1}.$$

Combining these results, we obtain

$$\begin{aligned} f(y_1, y_2, y_3) = & f(x_1, x_2, x_3) \\ & + (y_1 - x_1) g_1(x_1, y_1; x_2; x_3) \\ & + (y_2 - x_2) g_2(y_1; x_2, y_2; x_3) \\ & + (y_3 - x_3) g_3(y_1; y_2; x_{3,} y_3). \end{aligned} \tag{7.9.1}$$

Each of the functions g_1, g_2, and g_3 are obtained using one-dimensional expansions and hence can be computed or bounded as discussed above.

The form of the expansion depends on the order in which the sequential expansion is done. Since (7.9.1) is an identity, any sequence produces an analytically equivalent (and therefore cset-equivalent) form. However, different forms can produce different numerical results because of differing effects from dependence.

In Section 7.3, we discuss two procedures for obtaining the Taylor expansion of the product of two or more multidimensional function. In one procedure, the expansion is done directly (see (7.3.10)). In the other, it is obtained as the product of the expansions of the individual functions (see (7.3.11)). This discussion applies equally well for slope expansions. For the same reasons given in Section 7.3, we prefer to use a product of slope expansions rather than a slope expansion of products.

Similarly, the slope expansion of a quotient of multidimensional functions is obtained as the quotient of slope expansions. Compare the above with (7.3.12).

Consider a vector function $\mathbf{f}(\mathbf{x})$. If we expand each component of $\mathbf{f}$ in a form such as (7.9.1), we can combine the results in matrix form as

$$\mathbf{f}(\mathbf{y}) - \mathbf{f}(\mathbf{x}) = \mathbf{G}(\mathbf{x}, \mathbf{y})(\mathbf{y} - \mathbf{x})$$

where $\mathbf{G}(\mathbf{x}, \mathbf{y})$ is the appropriate matrix. That is, we can generate a slope expansion of a vector function.

Note that $\mathbf{G}(\mathbf{x}, \mathbf{y})$ takes the place of the Jacobian in a Taylor expansion. In Section 7.4, we discussed how to make the Jacobian symmetric when $\mathbf{f}$ is the gradient of some function. Unfortunately, the slope expansion does not seem to provide a means for making $\mathbf{G}(\mathbf{x}, \mathbf{y})$ symmetric.

Anticipating our discussion of solving systems of nonlinear equations in Chapter 11, we note the following. If $\mathbf{y}$ is a zero of $\mathbf{f}$ in a box $\mathbf{x}^\mathbf{I}$, then $\mathbf{f}(\mathbf{y}) = \mathbf{0}$ and we can seek a solution $\mathbf{y}$ of $\mathbf{f}(\mathbf{y}) = \mathbf{0}$ by solving

$$\mathbf{G}(\mathbf{x}, \mathbf{x}^\mathbf{I})(\mathbf{y} - \mathbf{x}) = -\mathbf{f}(\mathbf{x}).$$

See Chapter 11 for further discussion.

7.10 HIGHER ORDER SLOPES

We now consider how to obtain higher order slopes comparable to higher order derivatives. We consider the one-dimensional case. As we have seen, the first order expansion of f is

$$f(y) = f(x) + (y - x)g(x, y).$$

To obtain a second order expansion, we can expand $g(x, y)$ to first order. Thus, we want an expansion of the form

$$g(x, y) = g(x, x) + (y - x)h(x, y).$$

The slope $g(x, y)$ is defined to be $\frac{f(y)-f(x)}{y-x}$. Therefore,

$$g(x, x) = \lim_{y \to x} \frac{f(y) - f(x)}{y - x} = f'(x).$$

Therefore, the second order expansion is of the form

$$f(y) = f(x) + (y - x)f'(x) + (y - x)^2 h(x, y).$$

That is, the slope $\frac{g(x,y)-g(x,x)}{y-x}$ of g becomes

$$h(x, y) = \frac{f(y) - f(x) - (y - x)f'(x)}{(y - x)^2}.$$

For a rational function f, the numerator can be explicitly divided by the denominator. For example, if $f(x) = x^4$, then $h(x, y) = 3x^2 + 2xy + y^2$.

If $g(x, X)$ is composed of functions for which we can compute the slopes, then we can compute the second order slope $h(x, X)$. If $g(x, X)$ is convex or concave, we can compute $h(x, X)$ as described in Section 7.8.

Note that slopes of arbitrarily high order can be obtained iteratively. If we have an expansion of order n, we need only expand the highest order term to first order to obtain an expansion of order $n + 1$. This is how we obtained the second order expansion from the first order expansion.

7.11 SLOPE EXPANSIONS OF NONSMOOTH FUNCTIONS

In theory, slope bounds can be determined for any function f. What is needed is a bound on the difference quotient $\frac{f(y)-f(x)}{y-x}$ for arbitrary but fixed x and all y in an arbitrary interval X. Such bounds can be obtained for certain nonsmooth functions.

In Section 18.1, we show that the absolute value of a function and the max of two functions can be replaced by smooth functions plus constraints. An alternative exists in which a slope expansion is used for these functions. Kearfott (1996) provides the slopes of these functions and also for the so-called if-then-else function defined below. See also Ratz (1998), (1999).

Consider the absolute value function

$f(x) = |x|$.

We can determine the slope of this function for an interval X by separately considering positive and negative values in X. We obtain

$$g(x, X) = \begin{cases} [1, 1] & \text{if } x \geq 0 \text{ and } X \geq 0, \\ [-1, -1] & \text{if } x \leq 0 \text{ and } X \leq 0, \\ \left[\frac{|\underline{X}|-x}{\underline{X}-x}, 1\right] & \text{if } x \geq 0 \text{ and } 0 \in X, \\ \left[-1, \frac{|\overline{X}|-|x|}{\overline{X}-x}\right] & \text{if } x \leq 0 \text{ and } 0 \in X. \end{cases}$$

Next consider the function

$$f(x) = \max\{u(x), v(x)\}.$$

For this function, we merely bound the slope. If $u(x) \geq v(x)$ for all x in an interval X, then $f(x) = u(x)$ for $x \in X$; and the slope of f over X is the slope $g_u(x, X)$ of u. Similarly, if $u(x) \leq v(x)$ for all $x \in X$, the slope of f over X is the slope $g_v(x, X)$ of v. Otherwise, the slope of f for a given point in X must be either the slope of u or of v for the point. Therefore, the slope for the point is in the union of the slope of u and of v.

That is, the slope of f is contained in

$$g(x, X) = \begin{cases} g_u(x, X) \text{ if } u(x) \geq v(x) \text{ for all } x \in X, \\ g_v(x, X) \text{ if } u(x) \leq v(x) \text{ for all } x \in X, \\ g_u(x, X) \cup g_v(x, X), \text{ otherwise.} \end{cases}$$

The if-then-else function is defined to be

$$\text{ite}(u, v, z) = \begin{cases} u \text{ if } z < 0, \\ v \text{ otherwise.} \end{cases}$$

Its slope can be bounded in a way similar to that for the maximum function. We obtain

$$g(x, X) = \begin{cases} g_u(x, X) \text{ if } z < 0, \\ g_v(x, X) \text{ if } z \geq 0, \\ g_u(x, X) \cup g_v(x, X), \text{ otherwise.} \end{cases}$$

7.12 AUTOMATIC EVALUATION OF SLOPES

To evaluate a rational function on a computer, we program a sequence of steps involving the four arithmetic operations of addition, subtraction, multiplication, and division. Krawczyk and Neumaier (1985) point out that we can automatically obtain the slope of the function in the same way automatic differentiation obtains the derivative of a functions. See Section 7.5.

If

$$f_i(y) = f_i(x) + g_i(x, y)(y - x) \tag{7.12.1}$$

for $i = 1$ and 2, then

$$f_1(y) \pm f_2(y) = f_1(x) \pm f_2(x) + [g_1(x, y) \pm g_2(x, y)](y - x).$$

Hence, if $f = f_1 \pm f_2$, then

$$f(y) = f(x) + g(x, y)(y - x)$$

provided

$$g(x, y) = g_1(x, y) \pm g_2(x, y). \tag{7.12.2}$$

For multiplication, we have

$$\begin{aligned} f(y) &= f_1(y) f_2(y) \\ &= [f_1(x) + g_1(x, y)(y - x)] f_2(y) \\ &= f_1(x)[f_2(x) + g_2(x, y)(y - x)] + g_1(x, y) f_2(y)(y - x). \end{aligned}$$

Hence,

$$g(x, y) = f_1(x) g_2(x, y) + g_1(x, y) f_2(y). \tag{7.12.3}$$

Note that one of the functions f_1 and f_2 has argument x and the other has argument y. If we interchange the roles of f_1 and f_2, we obtain $g(x, y)$ in a different form. Analytically, the two forms are interchangeable, because they are containment-set equivalent (see Chapter 4) algebraic rearrangements of one another. Nevertheless, the effect of dependence on the computed values might be different for the two cases; so different results might be produced.

The slope of the quotient of two functions is unique and is given in Table 7.2. Note that the table includes the slopes of the primitives $f =$ constant and $f = x$ which are necessary for starting the procedure for automatic evaluation of slopes.

Ideally, we want merely to program the evaluation of a function and have code generated automatically to evaluate its slope. When a slope cannot be determined, such a program can produce a bound for the slope. Such a bound can be in the form of a derivative. The approach is the same as for automatic differentiation as described in Section 7.5. It automatically yields numerical values of the slope.

$f(x)$	$g(x, y)$
constant	0
x	1
$f_1(x) \pm f_2(x)$	$g_1(x, y) \pm g_2(x, y)$
$f_1(x)f_2(x)$	$f_1(y)g_2(x, y) + f_2(x)g_1(x, y)$
$\frac{f_1(x)}{f_2(x)}$	$\frac{f_2(x)g_1(x,y)-f_1(y)g_2(x,y)}{f_2(x)f_2(y)}$

Table 7.2: Slopes for the Basic Arithmetic Operations on Two Functions.

In the multivariable case, special procedures are used to generate the expansion of the product and quotient of functions. See Sections 7.3 and 7.5 for discussion of expansions using derivatives.

Alternatively, the slope can be generated in a way similar to symbolic differentiation (see Section 7.5) using algebraic manipulation by a program such as MACSYMA, Maple, Mathematica, or REDUCE. It is then possible to express the slope function analytically to produce sharper numerical results by reducing dependence (see Section 2.4). Identification and use of common subexpressions should be included.

A simple example shows how such an analytic approach can yield sharper results than the automatic procedure. Consider the function $f(x) = x^2 - x^3$ with $X = [0, 1]$ and $x = 0.5$. The automatic procedure using Table 7.2 produces numerical values as if the slope is computed using

$$g(x, y) = (x + y) - (x^2 + xy + y^2) \tag{7.12.4}$$

with y replaced by X. This results in $g(x, X) = [-1.25, 1.25]$.

Suppose we obtain $g(x, y)$ explicitly in the algebraic form given by (7.12.4). We can rewrite $g(x, y)$ as

$$g(x, y) = \frac{1 - 3x^2}{4} - \left(y - \frac{x-1}{2}\right)^2.$$

Evaluating g using this form, we produce the much better result $g(x, X) = [-0.5, 0.0625]$. It can be shown that this is the best possible result. That is, it is the range of $g(x, y)$ for $y \in X$ (when $x = 0.5$).

Krawczyk and Neumaier point out that the computational effort to compute a slope of a function f using the rules in Table 7.2 increases only linearly with the number of computational steps needed to evaluate f. For rational functions, the computational complexity is essentially the same as that for computing a gradient using automatic differentiation as discussed in Section 7.5.

Suppose that a given function is defined in terms of subfunctions and that slopes can be determined for some of the subfunctions but can only be bounded for others. We can expand such a function using slopes when possible and using derivatives otherwise. See Zuhe and Wolfe (1990).

Suppose we wish to obtain an expansion using a method such as the automatic procedure described earlier in this section or by an analytic procedure. We begin as if we are able to obtain a slope expansion. Suppose that at some stage, we require the slope of some subfunction $f(x)$ over an interval X, but it cannot be determined. Assuming f is continuously differentiable for all $x \in X$, then

$$g(x, y) = \frac{f(y) - f(x)}{y - x} \in f'(X)$$

for $x \in X$ and $y \in X$. Therefore, we can replace the slope $g(x, X)$ by the derivative $f'(X)$ which bounds it.

When we generate a slope expansion over an interval X, the point of expansion x need not be in X. If we want to replace $g(x, X)$ by a derivative, the derivative must be evaluated over an interval containing both x and X.

The algorithms described in this book use expansions of first or second order only. We have noted that second order expansions can be obtained by generating a first order expansion of the slope. However, second order expansions can be generated directly using a similar automatic procedure.

To do so, we need expansions for a sum, difference, product and quotient. Let $h(x, y)$ denote the second order slope; i.e., the slope of the slope. Let

$$f_i(y) = f_i(x) + (y - x)f_i'(x) + (y - x)^2 h_i(x, y) \ (i = 1, 2)$$

For addition or subtraction,

$$\begin{aligned} f(y) &= f_1(y) \pm f_2(y) \\ &= f_1(x) \pm f_2(x) + (y-x)\left[f_1'(x) \pm f_2'(x)\right] \\ &\quad + (y-x)^2\left[h_1(x,y) \pm h_2(x,y)\right]. \end{aligned}$$

For multiplication,

$$\begin{aligned} f(y) &= f_1(y) f_2(y) \\ &= f_1(x) f_2(x) + (y-x)\left[f_1(y) f_2'(x) + f_1'(x) f_2(x)\right] \\ &\quad + (y-x)^2\left[f_1'(y) h_2(x,y) + f_2(x) h_1(x,y)\right]. \end{aligned}$$

For division,

$$\begin{aligned} f(y) &= \frac{f_1(y)}{f_2(y)} \\ &= \frac{f_1(x)}{f_2(x)} + (y-x)\frac{f_1'(x) f_2(x) - f_1(x) f_2'(x)}{f_2(x) f_2(y)} \\ &\quad + (y-x)^2 \frac{f_2(x) h_1(x,y) - f_1(x) h_2(x,y)}{f_2(x) f_2(y)}. \end{aligned}$$

7.13 EQUIVALENT EXPANSIONS

A salient feature of a slope expansion is that its analytic form for a rational function is exact. Intervals enter only when terms are bounded. In contrast, intervals enter into a Taylor expansion to bound unknown derivative values. There are other types of expansions which are equivalent to slope expansions (and to each other) because they also use exact analytic expansions and then bound certain terms. The oldest of these is Moore's (1966) centered form (see Section 3.3) which generalizes in various ways (see Ratschek and Rokne (1984)).

Another equivalent type of expansion is the generalized interval arithmetic introduced by Hansen (1975). It has been used to speed the process of solving nonlinear equations by interval methods. See Hansen (1993).

Curiously, perhaps, these older methods did not become popular as methods of expansion and slope expansions have become more widely used. For this reason, we have discussed slopes rather than other forms. Centered forms are not as easily automated and this probably explains their limited use. However, generalized interval arithmetic is as convenient to use as slopes. Generalized interval arithmetic is very similar to slope expansions. Addition, subtraction and division are identically the same. Multiplication differs only in that second order terms are incorporated into the zero-th order term in generalized interval arithmetic and into the linear term in slope expansions.

Chapter 8

QUADRATIC EQUATIONS AND INEQUALITIES

8.1 INTRODUCTION

When solving systems of nonlinear equations and optimization problems, we frequently want to compute the real roots of a quadratic equation in which the coefficients are intervals. These roots might be finite intervals, semi-infinite intervals, or the entire real line. A naive procedure for determining the roots can be surprisingly complicated. In this chapter, we describe a procedure due to Hansen and Walster (2001) for computing such roots. We also discuss solving quadratic inequalities. These procedures are useful elsewhere in this book.

Consider the quadratic equation

$$Ax^2 + Bx + C = 0 \tag{8.1.1}$$

where $A = [\overline{A}, \underline{A}]$, $B = [\underline{B}, \ \overline{B}]$, and $C = [\underline{C}, \overline{C}]$ are intervals. The interval roots of (8.1.1) are the set of real roots x of the quadratic equation $ax^2 + bx + c = 0$ for all real $a \in A$, $b \in B$, and $c \in C$.

We can express an interval enclosure for the roots as

$$r^{\pm} \in \frac{-B \pm (B^2 - 4AC)^{1/2}}{2A}. \tag{8.1.2}$$

This interval enclosure is not sharp except in special cases described below. This is because the intervals A and B occur more than once in this expression and dependence causes loss of sharpness. It does not help to use the algebraically equivalent enclosure:

$$r^{\pm} \in \frac{2C}{-B \pm (B^2 - 4AC)^{1/2}}$$

because B and C now occur twice.

Since $x^2 \geq 0$, $Ax^2 = [\underline{A}x^2, \overline{A}x^2]$. The term Bx in (8.1.1) can be written

$$Bx = \begin{cases} [\underline{B}x, \overline{B}x] \text{ if } x \geq 0, \\ [\overline{B}x, \underline{B}x] \text{ if } x \leq 0. \end{cases}$$

Denote

$$F_1(x) = \underline{A}x^2 + \underline{B}x + \underline{C},$$

$$F_2(x) = \overline{A}x^2 + \overline{B}x + \overline{C},$$

$$F_3(x) = \underline{A}x^2 + \overline{B}x + \underline{C},$$

and

$$F_4(x) = \overline{A}x^2 + \underline{B}x + \overline{C}.$$

We can rewrite (8.1.1) as $[F_1(x), F_2(x)] = 0$ when $x \geq 0$ and as $[F_3(x), F_4(x)] = 0$ when $x \leq 0$. Denote

$$\underline{F}(x) = \begin{cases} F_1(x) \text{ if } x \geq 0, \\ F_3(x) \text{ if } x \leq 0. \end{cases} \quad \overline{F}(x) = \begin{cases} F_2(x) \text{ if } x \geq 0, \\ F_4(x) \text{ if } x \leq 0. \end{cases} \tag{8.1.3}$$

Then the quadratic function

$$F(x) = Ax^2 + Bx + C$$

can be expressed as

$$F(x) = [\underline{F}(x), \overline{F}(x)].$$

Suppose there exists a value of x such that

$$\underline{F}(x) \le 0 \le \overline{F}(x). \tag{8.1.4}$$

Then there exists $a \in A$, $b \in B$, and $c \in C$ such that $ax^2 + bx + c = 0$ for this value of x. Given such a point x, let R be the largest interval containing x such that every point in R also satisfies (8.1.4). We call R an "interval root" of $F(x)$.

In the noninterval case, there might be no real root, or there might be only one (multiple root), or there might be two disjoint roots. The same is true for the interval case. However, the interval case differs from the noninterval case in that there might be three disjoint interval roots. In the latter case, one interval root extends to $-\infty$ and another extends to $+\infty$. That is, it is really an exterior interval. We can think of an exterior interval root as two interval roots joined at projective infinity to form a single interval.

We can simplify the process of determining the interval roots by assuming that $\overline{A} > 0$. If this is not the case, we need only change the sign of $F(x)$. This can fail only if $A = 0$ in which case the equation is not quadratic.

A natural way to determine the interval roots is to find the set of points where $\underline{F}(x) \ge 0$ and the set of points where $\overline{F}(x) \le 0$ and take the intersection of these two sets.

This is surprisingly tedious. We must separately consider the cases $\underline{B} > 0$, $\overline{B} < 0$, and $\underline{B} < 0 > \overline{B}$ and (independently) consider the cases $\underline{C} > 0$, $\underline{C} < 0 < \overline{C}$, and $\overline{C} < 0$. Also we must consider the cases $\underline{A} < 0$, $\underline{A} = 0$, and $\underline{A} > 0$. There are 27 cases in all. For each possible combination of these possibilities, there are multiple sub-cases to consider. Determining an interval root in this way requires comparing the values of $\underline{F}(x)$ and $\overline{F}(x)$ at their extrema with their values at $x = 0$ and determining the shape and orientation of $\underline{F}(x)$ and $\overline{F}(x)$. For the case $\underline{A} < 0$ and $\underline{B} < 0 < \overline{B}$, there are 11 sub-cases to consider. For other cases, there are 6 sub-cases.

In our optimization algorithms, we sometimes want to find the solution set of the quadratic inequality

$$Ax^2 + Bx + C \ge 0 \tag{8.1.5}$$

We can solve this relation by rewriting it as the equation

$$Ax^2 + Bx + C = [0, +\infty]$$

so that it becomes

$$Ax^2 + Bx + [-\infty, \overline{C}] = 0.$$

This has the form of (8.1.1) and can be solved by the method that we now describe.

8.2 A PROCEDURE

We wish to know where the lower and upper real functions defining $F(x)$ are zero. These functions are determined by $F_i(x)$ $(i = 1, 2, 3, 4)$. It is easily verified that the upper function $\overline{F}(x)$ is convex. If $\underline{A} > 0$, then the lower function $\underline{F}(x)$ is also convex. If $\underline{A} < 0$, the lower function $\underline{F}(x)$ is concave for $x \leq 0$ and for $x \geq 0$, but it can have a cusp at $x = 0$.

Let us compute the real roots of each of these real functions $F_i(x)$ $(i = 1, 2, 3, 4)$ and place them in a list L. A double root is to be entered twice. If $\underline{A} = 0$, then F_1 and F_3 are linear and we have only a single root. The roots are computed using interval arithmetic to bound rounding errors. Thus, the entries in the list L are intervals.

Since we omit complex roots, it appears that these four functions can have a total of 0 to 8 roots. However, these real roots are endpoints of the interval roots and there can be no more than three interval roots. Therefore, the list L can contain no more than six real roots.

The functions $F_1(x)$ and $F_2(x)$ define bounds on $F(x)$ only when $x \geq 0$. Therefore, we drop any negative root of these functions from the list L. Also, drop any negative part of the interval bounding such a root. Similarly, drop any root (or part of root) of $F_3(x)$ or $F_4(x)$ that is positive.

The intervals remaining in L bound values of x that are either a lower or an upper endpoint of an interval root. We need only determine which they are. Before doing so, it is convenient to put the interval root endpoints $\pm\infty$ into L (if they occur).

We have assumed that $\overline{A} > 0$. Therefore, $\overline{F} > 0$ for all sufficiently large $|x|$. If $\underline{A} > 0$, then $\underline{F}(x) > 0$ for all sufficiently large $|x|$. Therefore, there is no interval root for "large" $|x|$. That is, if any root exists, it must be finite.

However, if $\underline{A} < 0$, then $\underline{F}(x) < 0$ for all sufficiently large $|x|$. Therefore, an interval root exists whose lower endpoint is $-\infty$ and an interval root exists whose upper endpoint is $+\infty$.

Similar arguments show that there exists an interval root whose left endpoint is $-\infty$ if $\underline{A} = 0$ and $\overline{B} > 0$ or if $\underline{A} = 0$ and $\overline{B} = 0$ and $\underline{C} \leq 0$. Therefore, we put a value $-\infty$ into the list L if $\underline{A} < 0$ or if $\underline{A} = 0$ and $\overline{B} > 0$ or if $\underline{A} = 0$ and $\overline{B} = 0$ and $\underline{C} \leq 0$. Similarly, we put a value $+\infty$ into L if $\underline{A} < 0$ or if $\underline{A} = 0$ and $\underline{B} > 0$ or if $\underline{A} = 0$, $\underline{B} = 0$, and $\underline{C} \leq 0$.

If L is empty, there are no interval roots. Otherwise, we name the entries in L as $S_i = [\underline{S}_i, \overline{S}_i]$ where we have ordered them so that $\underline{S}_i \leq \underline{S}_{i+1}$. Let s_i denote the exact root that is bounded by S_i.

We want the computed interval roots to contain the exact interval roots. To assure this, we replace s_i by $\underline{S}_i$ if s_i is the lower endpoint of an interval root and by $\overline{S}_i$ if s_i is an upper endpoint.

A particular case requires attention. If $\underline{C} = 0$, then $F_1(x)$ and $F_3(x)$ are both zero at $x = 0$. However, this particular zero of $\underline{F}(x)$ must be listed only once in L. Similarly, if $\overline{C} = 0$, the zero at $x = 0$ of $\overline{F}(x)$ must be listed only once.

In the next section, we list the steps to implement our procedure for computing the interval roots. Before doing so, we note that there are certain cases in which the roots can be computed more simply.

In some cases, the roots are monotonic functions of the coefficients and hence are easily computed. This is true if $AC \leq 0$. It is also true if $AC \geq 0$ provided either $B \leq 0$ or $B \geq 0$.

There are two cases in which we can compute an interval root by directly using an interval version of equation (8.1.2). It can be shown using the endpoint analysis of Section 3.5 that the interval roots can be sharply computed as $\frac{-B+(B^2-4AC)^{1/2}}{2A}$ and $\frac{2C}{-B+(B^2-4AC)^{1/2}}$ if $A \geq 0$, $B \leq 0$, $C \geq 0$, and $B^2 - 4AC \geq 0$. That is, dependency does not cause loss of sharpness.

They can be correctly computed as $\frac{-B-(B^2-4AC)^{1/2}}{2A}$ and $\frac{2C}{-B-(B^2-4AC)^{1/2}}$ if $A \geq 0$, $B \geq 0$, $C \geq 0$, and $B^2 - 4AC \geq 0$. In each of these two cases, the roots can also be determined using monotonicity.

Roots for other special cases can be easily obtained. This is the case if $B = 0$ or if $C = 0$.

Incidentally, the roots of a real quadratic $ax^2 + bx + c = 0$ are best computed as $[-b - (b^2 - 4ac)^{1/2}]/(2a)$ and $2c/[-b - (b^2 - 4ac)^{1/2}]$ if $b > 0$ and as $[-b + (b^2 - 4ac)^{1/2}]/(2a)$ and $2c/[-b + (b^2 - 4ac)^{1/2}]$ if $b < 0$. This well known procedure minimizes the effect of rounding errors. This procedure is also used in computing the real roots of the quadratic equations $F_i(x) = 0$ $(i = 1, 2, 3, 4)$ defined above.

Our method for computing roots of interval quadratics can also be extended to compute the real interval roots of a polynomial of any degree with interval coefficients.

It is a simple process to determine the range of an interval quadratic equation over an interval X. The real functions $\underline{F}(x)$ and $\overline{F}(x)$ are the lower and upper functions of the interval quadratic. The lower bound of $\underline{F}(x)$ and the upper bound of $\overline{F}(x)$ over X are easily found. These bounds define the range of the interval quadratic over X.

8.3 THE STEPS OF THE ALGORITHM

Our procedure for computing interval roots can be implemented in various ways. We have chosen the following steps.

1. Compute intervals containing the real roots of each of the real functions $F_i(x)$ $(i = 1, 2, 3, 4)$. Put the results in a list L. A double root is to be entered twice. If $\underline{C} = 0$, both $F_1(x)$ and $F_3(x)$ have a root at $x = 0$. This root is entered only once into L. If $\overline{C} = 0$, both $F_2(x)$ and $F_4(x)$ have a zero at $x = 0$. This root is entered only once into L.

2. Put a value $-\infty$ into the list L if $\underline{A} < 0$ or if $\underline{A} = 0$ and $\overline{B} > 0$ or if $\underline{A} = 0$ and $\overline{B} = 0$ and $\underline{C} \leq 0$.

3. Put a value $+\infty$ into the list L if $\underline{A} < 0$ or if $\underline{A} = 0$ and $\underline{B} > 0$ or if $\underline{A} = 0$, $\underline{B} = 0$, and $\underline{C} \leq 0$.

4. Order the (interval) entries in L so that if they are named S_i, then $\underline{S}_i \leq \underline{S}_{i+1}$. Note that entries $\pm\infty$ can be regarded as degenerate intervals.

5. Denote the number of entries in L by n. If $n = 0$, there are no interval roots. If $n = 2$, there is one interval root $[\underline{S}_1, \overline{S}_2]$. (Note that it might be $[-\infty, +\infty]$.) If $n = 4$, there are two interval roots $[\underline{S}_1, \overline{S}_2]$ and $[\underline{S}_3, \overline{S}_4]$. If $n = 6$, the interval roots are $[-\infty, \overline{S}_2]$, $[\underline{S}_3, \overline{S}_4]$, and $[\underline{S}_5, +\infty]$.

Chapter 9

NONLINEAR EQUATIONS OF ONE VARIABLE

9.1 INTRODUCTION

Consider a continuously differentiable scalar function f of a single variable x. In this chapter, we consider the problem of finding and bounding all the zeros of $f(x) = 0$ in a given finite, closed interval X_0. The only methods we consider are the interval Newton method and the variation of it in which derivatives are replaced by slopes.

Various other interval methods for this problem have been published including more general versions of the method we describe. However, the simple version of the interval Newton method has so many remarkable properties (see Section 9.6) and is so efficient that no other methods or variations are needed.

The quotation below points up the value of some of the properties of the method. Section 2.1 of the excellent book by Dennis and Schnabel (1983) is entitled "What is not possible". As we shall see, properties of the interval Newton method make this title erroneous.

Dennis and Schnabel define the functions

$$f_1(x) = x^4 - 12x^3 + 47x^2 - 60x$$
$$f_2(x) = f_1(x) + 24$$
$$f_3(x) = f_1(x) + 24.1.$$

They state that "It would be wonderful if we had a general purpose computer routine that would tell us: 'The roots of $f_1(x)$ are 0, 3, 4, and 5; the real roots of $f_2(x)$ are $x = 1$ and $x \cong 0.888$; $f_3(x)$ has no real roots' ".

They continue: "It is unlikely that there will ever be such a routine. In general the questions of existence and uniqueness — does a given problem have a solution, and is it unique? — are beyond the capabilities one can expect of algorithms that solve nonlinear problems".

As described in this chapter, the general purpose computer routine that Dennis and Schnabel say "would be wonderful" does, in fact, exist. It was used by one of the authors to solve the problems listed. It produced precisely the information requested in the above quotation, including answers to the questions of existence and uniqueness.

We derive this "wonderful" algorithm in the next section and give a version of it in Section 9.3. We now list some of its properties in informal terms. The properties are proved formally as theorems in Section 9.6

1. Every zero of f in an initial interval X_0 is always found and correctly bounded. See Theorems 9.6.2, 9.6.3, and 9.6.5 below. Noninterval methods sometimes use explicit or implicit deflation to find all zeros of a function. No special deflation steps are required in the interval algorithm. The loss of accuracy due to explicit deflation has no counterpart here.

2. If there is no zero of f in X_0, the algorithm automatically proves this fact in a finite number of iterations. See Theorems 9.6.4 and 9.6.5.

3. The existence or nonexistence of a zero of f in a given interval might (but might not) be automatically proved without extra computing. See Theorems 9.6.4 and 9.6.8.

4. Assume the interval Newton method is applied to an interval X. If $0 \notin f'(X)$, at least half of X is eliminated in one step. Thus, convergence can be reasonably rapid even when $\mathrm{w}(X)$ is large. See Theorem 9.6.6.

5. If $0 \notin f'(X)$. then, if the interval Newton method is applied iteratively beginning with X, the asymptotic rate of convergence to a zero of f in X is quadratic. See Theorem 9.6.7.

After discussing stopping criteria in Section 9.4, we list the steps of the algorithm in Section 9.5. We then state and prove several theorems about the properties of the algorithm in Section 9.6 and give a numerical example in Section 9.7. In Section 9.8, we describe a variant of the method using the slope function (discussed in Section 7.7). An illustrative example of this variant is given in Section 9.9. We close the chapter with a brief discussion of perturbed problems in Section 9.10.

9.2 THE INTERVAL NEWTON METHOD

The interval Newton method was derived by Moore (1966) in the following manner. From the mean value theorem

$$f(x) - f(x^*) = (x - x^*) f'(\xi) \tag{9.2.1}$$

where ξ is some point between x and x^*. If x^* is a zero of f, then $f(x^*) = 0$ and, from (9.2.1),

$$x^* = x - \frac{f(x)}{f'(\xi)}.$$

Let X be an interval containing both x and x^*. Since ξ is between x and x^*, it follows that $\xi \in X$. Therefore, $f'(\xi) \in f'(X)$ by Theorem 3.2.2. Hence, $x^* \in \mathrm{N}(x, X)$ where

$$\mathrm{N}(x, X) = x - \frac{f^{\mathrm{I}}(x)}{f'(X)},$$

and we use $f^{\mathbf{I}}(x)$ to denote the interval evaluation of $f(x)$ to bound rounding errors. Temporarily assume $0 \notin f'(X)$ so that $\mathrm{N}(x, X)$ is a finite interval. Since any zero of f in X is also in $\mathrm{N}(x, X)$, it is in the intersection $X \cap \mathrm{N}(x, X)$.

Using this fact, we define an algorithm for finding the zero x^*. Let X_0 be an interval containing x^*. For $n = 0, 1, 2, \cdots$, define

$$\begin{aligned} x_n &= \mathrm{m}(X_n) \\ \mathrm{N}(x_n, X_n) &= x_n - \frac{f^{\mathbf{I}}(x_n)}{f'(X_n)} \\ X_{n+1} &= X_n \cap \mathrm{N}(x_n, X_n) \end{aligned} \tag{9.2.2}$$

We call x_n the *point of expansion* for the Newton method. It is not necessary to choose x_n to be the midpoint of X_n. We require only that $x_n \in X_n$ to assure that $x^* \in \mathrm{N}(x_n, X_n)$ whenever $x^* \in X_n$. However, it is convenient and efficient to choose $x_n = \mathrm{m}(X_n)$. Later in this section, we discuss a useful result of this choice. In Section 9.3, we discuss a case in which the point of x_n is an endpoint of X_n.

In his original derivation of the interval Newton method, Moore (1966) assumed that $0 \notin f'(X_0)$. Alefeld (1968) and (independently, but much later) Hansen (1978b) extended the algorithm to include the case $0 \in f'(X_0)$. We consider this more general case in this section.

If $0 \notin f'(X_0)$ then $0 \notin f'(X_n)$ for all $n = 1, 2, \cdots$. This follows from inclusion isotonicity and the fact that $X_n \subset X_0$ for all $n = 1, 2, \ldots$. However, if $0 \in f'(X_0)$, then evaluating $\mathrm{N}(x_1, X_1)$ requires the use of extended interval arithmetic (see Chapter 4). If x^* is a multiple zero of f, then $f'(x^*) = 0$ and so $0 \in f'(X)$ for any interval X containing x^*. Even though $\mathrm{N}(x_n, X_n)$ is not finite in such a case, the intersection $X_{n+1} = X_n \cap \mathrm{N}(x_n, X_n)$ is finite.

When we evaluate $f(x_n)$, we use interval arithmetic to bound rounding errors and denote this fact by a superscript "I" on $f^{\mathbf{I}}(x_n) = [a_n, b_n]$. If $0 \in f^{\mathbf{I}}(x_n)$, then x_n is a zero of f or else is "near" to one (if one exists). If $0 \in f^{\mathbf{I}}(x_n)$ and $0 \notin f'(X_n)$, a step of the interval Newton method using the interval X_n might or might not yield a smaller interval than X_n.

Now consider the cases $0 \notin f^{\mathrm{I}}(x_n)$ and $0 \in f'(X_n)$. Denote $f'(X_n) = [c_n, d_n]$. Using extended interval arithmetic, we obtain the following results. Since $0 \notin f^{\mathrm{I}}(x_n) = [a_n, b_n]$, either $a_n > 0$ or $b_n < 0$. If $a_n > 0$, then

$$\mathrm{N}(x_n, X_n) = \begin{cases} [-\infty, q_n] \cup \{+\infty\} & \text{if } c_n = 0 \\ [p_n, +\infty] \cup \{-\infty\} & \text{if } d_n = 0 \\ [-\infty, q_n] \cup [p_n, +\infty] & \text{if } c_n < 0 < d_n \end{cases} \tag{9.2.3}$$

where

$$p_n = x_n - \frac{a_n}{c_n}$$
$$q_n = x_n - \frac{a_n}{d_n}.$$

If $b_n < 0$, then

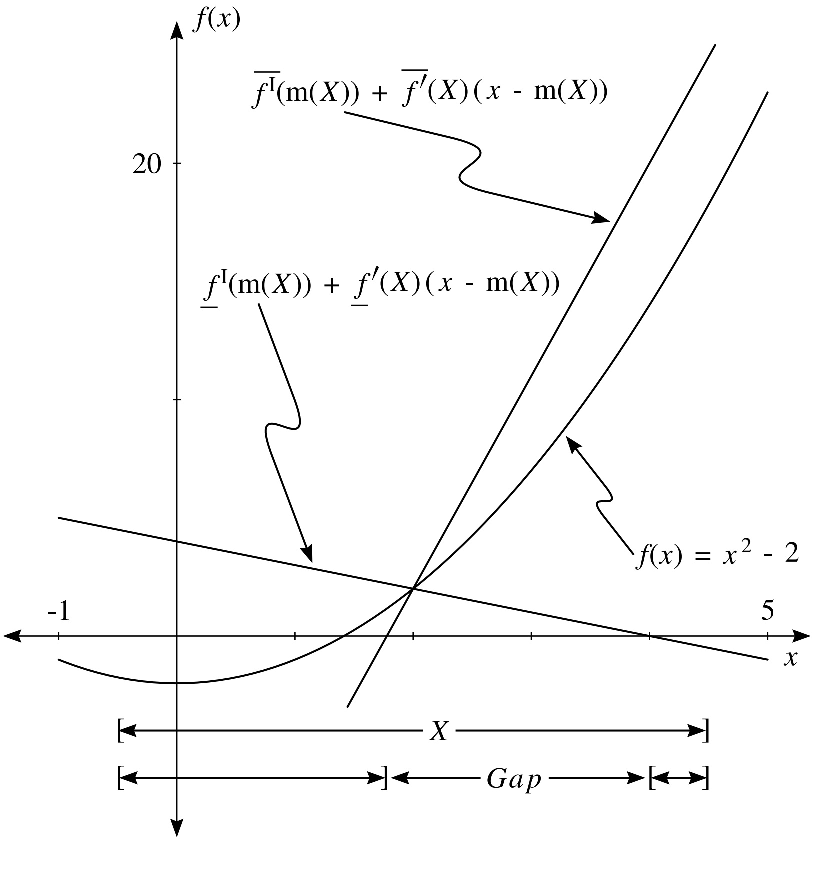
$$\mathrm{N}(x_n, X_n) = \begin{cases} [q_n', +\infty] \cup \{-\infty\} & \text{if } c_n = 0 \\ [-\infty, p_n'] \cup \{+\infty\} & \text{if } d_n = 0 \\ [-\infty, p_n'] \cup [q_n', +\infty] & \text{if } c_n < 0 < d_n \end{cases} \tag{9.2.4}$$

where

$$p_n' = x_n - \frac{b_n}{c_n}$$
$$q_n' = x_n - \frac{b_n}{d_n}.$$

The intersection $X_{n+1} = X_n \cap \mathrm{N}(x_n, X_n)$ might be a single interval, the union of two intervals, or the empty set. Figure 9.2.1 illustrates the case of the union of two intervals. Points in X_{n+1} are finite (or nonexistent if X_{n+1} is empty). Therefore, we do not actually have to use unbounded intervals in the Newton method.

One more case remains. Let X denote the interval to which a step of the Newton method is applied. Let the point of expansion x be any point in X. If both $0 \in f^{\mathrm{I}}(x)$ and $0 \in f'(X)$, then $\mathrm{N}(x, X) = [-\infty, +\infty]$, a useless result.



$f(x)$
$\overline{f}^{\mathrm{I}}(\mathrm{m}(X)) + \overline{f'}(X)(x - \mathrm{m}(X))$
20
$\underline{f}^{\mathrm{I}}(\mathrm{m}(X)) + \underline{f}'(X)(x - \mathrm{m}(X))$
$f(x) = x^2 - 2$
-1
5
x
X
Gap

Figure 9.2.1: The Two-Interval Newton Iteration.

This can occur when X is a narrow interval containing a multiple zero x^* of f because, in this case, $f(x^*) = 0$ and $f'(x^*) = 0$. Therefore, $f'(X)$ must contain zero; and x must be near x^* so it is not unlikely that $0 \in f^{\mathbf{I}}(x)$.

But, it can happen that both $0 \in f^{\mathbf{I}}(x)$ and $0 \in f'(X)$ when X is wide and x accidentally is a zero or near a zero of f. A wide interval X is likely to contain a stationary point of f, in which case $0 \in f'(X)$. Since X is not a good bound for the zero of f in this case, we must assure that we continue to narrow it. We discuss a procedure for doing this in Section 9.4.

9.3 A PROCEDURE WHEN $0 \notin f'(X)$.

In a particular case, we can get what is essentially the sharpest possible (for the number system used) bounds on a zero of a function without using traditional stopping criteria. In this section, we discuss this special case and a procedure for it. In Section 9.4, we discuss other criteria for terminating an interval algorithm for finding and bounding zeros of a function.

Assume we seek a zero of $f(x)$ in an interval X and that $0 \notin f'(X)$. The relation $0 \notin f'(X)$ is the condition of interest in this section. It assures that there is no more than one zero of f in X; and if there is a zero of f in X, then it is a simple one. We use the fact that inclusion isotonicity assures that $0 \notin f'(X')$ for any interval X' contained in X.

We now discuss a related topic. As previously noted, when we compute $f(x)$ using outwardly rounded arithmetic, we obtain an interval result that we denote by $f^{\mathbf{I}}(x)$. Let x^* be a zero of f. Then $0 \in f^{\mathbf{I}}(x^*)$. Because of rounding errors, it is generally true that $0 \in f^{\mathbf{I}}(x)$ for a set of values of x near x^*. We define the interval X^* to be the largest interval containing x^* such that $0 \in f^{\mathbf{I}}(x)$ for every machine number $x \in X^*$. The function f can depend on interval parameters. This can cause widening of X^*

We are satisfied if our algorithm produces an interval approximating X^* as the final bound on x^*. Actually, we can often compute a better bound on x^* than X^*. This is because the Newton step uses information about both f and f' rather than just f. However, we discuss termination as if X^* is the desired bound on x^*. We refer to X^* as the "optimal bound" on x^*.

We now return to our discussion of the special case in which $0 \notin f'(X)$. Assume this is the case and that we seek a zero of f in an interval X.

Suppose we apply an interval Newton method that is derived (as in Section 9.2) by expanding about the midpoint $x = \mathrm{m}(X)$ of X. We compute $\mathrm{N}(x, X) = x - \frac{f^{\mathbf{I}}(x)}{f'(X)}$ and we then determine a new bound $X \cap \mathrm{N}(x, X)$ on the zero.

We write $f^{\mathbf{I}}(x)$ in place of $f(x)$ to emphasize that $\mathrm{N}(x, X)$ now denotes the computed value of the theoretical function rather than the function itself. It is common practice to omit such notation.

From Theorem 9.6.6 below, if $0 \notin f^{\mathbf{I}}(x)$, then $\mathrm{w}(X') \leq \frac{1}{2}\,\mathrm{w}(X)$. That is, convergence is adequately rapid as long as $0 \notin f^{\mathbf{I}}(x)$. Therefore, we iterate the Newton step. Asymptotically, convergence will be quadratic. See Theorem 9.6.7. Eventually, either a result is empty or else $0 \in f^{\mathbf{I}}(x)$ for the current point x.

Assume that $0 \in f^{\mathbf{I}}(x)$. Then it is likely that x is in the optimal bound X^* for a zero x^* of f. We wish either to compute an approximation for X^* or else to prove that there is no zero of f in X.

Denote the point x for which we first satisfy the condition $0 \in f^{\mathbf{I}}(x)$ by $\widetilde{x}$. Suppose we have performed the Newton step when $0 \in f^{\mathbf{I}}(\widetilde{x})$. Assume (as is generally the case) that we have used a Newton method that is defined using an expansion about the center of the current interval. We now change to the following procedure in which a Newton step is sometimes computed by expanding f about an endpoint of the current interval and sometimes about its center. We denote the current interval by $X = [a, b]$ even though it changes in various steps of the procedure. The point $\widetilde{x}$ does not change.

We expand about the endpoint $x = a$ only if $0 \notin f^{\mathbf{I}}(a)$ We set a flag F_a when $0 \in f^{\mathbf{I}}(a)$. Similarly, we set a flag F_b when $0 \in f^{\mathbf{I}}(b)$. We begin with flag $F_a = 0$ and flag $F_b = 0$.

We cycle through the procedure no more than four times. The integer n counts the cycles.

0. Set $n = 0$.

1. If flag $F_a = 1$ go to step 4.

2. Evaluate $f(a)$. If $0 \in f^{\mathbf{I}}(a)$, set flag $F_a = 1$ and go to step 4.

3. Apply a Newton step in which the point of expansion is a. If the result is empty, stop.

4. If flag $F_b = 1$, go to step 7.

5. Evaluate $f(b)$. If $0 \in f^{\mathrm{I}}(b)$, set flag $F_b = 1$ and go to step 7.

6. Apply a Newton step. If $\widetilde{x} \in X$, then use b as the point of expansion. Otherwise, expand about the midpoint $\mathrm{m}(X)$. If the result is empty, stop.

7. If flag $F_a = 1$ and flag $F_b = 1$, stop.

8. Apply a Newton step in which the point of expansion is $\mathrm{m}(X)$. If the result is empty, stop.

9. Replace n by $n + 1$.

10. If $n < 4$, go to step 1. Otherwise, stop.

The procedure might stop when $n < 4$. Instead of stopping when $n = 4$, we can continue iterating until no further reduction of the interval occurs. However, this might entail an excessive number of iterations with little reduction in the width of the interval bounding the zero.

This procedure stops when the evaluation of f at each endpoint produces an interval containing zero. It also stops if a Newton step results in an empty interval. The final interval can be narrower than the optimal bound X^*.

This procedure begins only after a Newton step has been applied in which the point of expansion is a point $\widetilde{x}$ such that $0 \in f^{\mathrm{I}}(\widetilde{x})$ and $0 \in f'(X)$. When this is true, the result of this Newton step is generally a narrow interval. Therefore, the procedure can stop in one or two iterations. However, suppose an endpoint of $f'(X)$ is near zero for the interval X used when the procedure begins. Then more iterations might be required.

The point $\widetilde{x}$ is defined when $\widetilde{x} = \mathrm{m}(X)$, $0 \in f(\widetilde{x})$, and $0 \notin f'(X)$. The Newton step when this is the case will usually prove existence of a simple zero of f in X. See Theorem 9.6.8 below. This zero will be contained in the final interval produced by the above steps. See Theorem 9.6.1 below.

Even if such proof is not obtained, it is very likely that the final interval contains a zero. A possible but unlikely alternative is that there is a zero of f in an interval abutting X but not in X itself. Such a zero is bounded by the algorithm in Section 9.5.

Suppose $0 \in f'(X)$ for some interval X. If we know that X contains a single multiple zero of f, we can use the above procedure to compute an optimal bound. However, if X contains two or more distinct zeros, the procedure only returns an approximation for the smallest interval containing all of them. We therefore do not use the procedure when $0 \in f'(X)$.

9.4 STOPPING CRITERIA

Criteria for stopping iteration of an interval Newton method must assure that iteration is continued until an interval bound on a zero of f is sufficiently narrow. Also, the criteria should avoid needless iteration. We discuss these issues in this section. Throughout this section, we assume $0 \in f'(X)$ for any interval X that is a candidate for a final bound on a zero of f. Otherwise, we use the stopping procedure of Section 9.3.

A simple criterion is to terminate when the width $\text{w}(X)$ of the current interval X is small. However, this can create a difficulty. Consider a hypothetical computer that uses three significant decimal digits in its arithmetic. Suppose we require $\text{w}(X) < 0.001$ for any final interval. We consider two examples using this criterion on such a computer.

Example 9.4.1 Suppose the solution is $x^* = 123.4$ and we compute the interval $X = [123, 124]$ bounding it. In the number system used, there is no narrower interval that bounds the solution. We have the best possible result. However, the termination criterion $\text{w}(X) < 0.001$ is not satisfied because $\text{w}(X) = 1$.

Example 9.4.2 Suppose the solution is $x^* = 0.000123$ and we compute the interval $X = [-0.0004, 0.0004]$. This interval satisfies the termination criterion. However, we do not even know the sign of the solution bounded by X.

The difficulty in Example 9.4.1 can be overcome by using a relative rather than an absolute error criterion. That is, instead of requiring that $\mathrm{w}(X) < \varepsilon$ for some $\varepsilon > 0$, we can require that $\frac{\mathrm{w}(X)}{|X|} < \varepsilon$ where $|X|$ is the magnitude of X as defined in Section 3.1.

Suppose $0 \in X$. It is easily seen that in this case, $|X| \leq \mathrm{w}(X) \leq 2|X|$ so $\frac{\mathrm{w}(X)}{|X|} \geq 1$. If our termination criterion is $\frac{\mathrm{w}(X)}{|X|} < \varepsilon$ with $\varepsilon \leq 1$, then the criterion can never be satisfied by any interval containing zero. Therefore, when $0 \in X$, the difficulty in Example 9.4.2 is not resolved by using a relative criterion.

Suppose that $0 \in X$. We can use the point $x = 0$ as the point of expansion for a Newton step. When we evaluate $f(0)$ using outwardly rounded interval arithmetic, we obtain a result $f^{\mathbf{I}}(0)$. If $0 \notin f^{\mathbf{I}}(0)$, then the point $x = 0$ is not in the Newton result $\mathrm{N}(0, X)$ and can no longer be in the region of search. Therefore, we can rely upon the relative criterion for subsequent Newton steps.

It is reasonable to expand about $x = 0$ when $0 \in X$; but we do not do so. It might very well happen that only a small interval about $x = 0$ is eliminated. In this case, the difficulty concerned with relative error still occurs. If $0 \notin f^{\mathbf{I}}(0)$, the point $x = 0$ is likely to be eliminated by the Newton method without a special procedure. We consider the case $0 \in f^{\mathbf{I}}(0)$ later in this section.

We now consider the criteria we use to decide when to stop trying to reduce an interval bound on a solution. We have already mentioned the relative error criterion. We express it formally as follows.

Criterion 9.4.3 $\frac{\mathrm{w}(X)}{|X|} < \varepsilon_X$ for some given $\varepsilon_X > 0$.

It is reasonable to require that $|f(x)|$ be small for every x in an interval X accepted as a final bound on a solution. Therefore, we consider the following criterion for termination.

Criterion 9.4.4 $|f(X)| < \varepsilon_f$ for some given $\varepsilon_f > 0$.

Note that this criterion is in absolute rather than relative form because it is used when $0 \in f(X)$.

If both criteria are satisfied, we regard the interval X as an adequate bound on a zero. If desired, a user can choose either ε_X or ε_f to be large so stopping is caused by only one criterion.

Consider an interval X that contains a multiple zero of f. Then $0 \in f'(X)$. Suppose $0 \in f^{\mathbf{I}}(x)$. To apply a Newton step, we must compute $\frac{f^{\mathbf{I}}(x)}{f'(X)}$. Since zero is contained in both the numerator and denominator, the computed quotient is $[-\infty, +\infty]$. Therefore, the Newton step does not improve the bound X for the zero of f. If Criteria 9.4.3 and 9.4.4 are so stringent that X is not accepted as a final bound, the Newton algorithm splits the interval. This step could be repeated so that the optimal bound (as defined in Section 9.3) for the zero is eventually covered by a large number of subintervals. This would be wasted effort. Thus, we need a stopping criterion that can override Criteria 9.4.3 and 9.4.4 when $0 \in f^{\mathbf{I}}(x)$ and $0 \in f'(X)$.

On the other hand, X might be a wide interval and thus a poor bound on a zero. We must assure that we do not override Criteria 9.4.3 and 9.4.4 in this case.

Let $x \in X$ be the point of expansion for the Newton step. Suppose that $0 \notin f^{\mathbf{I}}(x)$ Suppose, also, that either Criterion 9.4.3 or Criterion 9.4.4 (or both) is not satisfied. Then we can say that the tolerances are not too small because there is at least one point x in X where $0 \notin f^{\mathbf{I}}(x)$. Therefore, we require that $0 \in f^{\mathbf{I}}(x)$ before we override Criteria 9.4.3 and 9.4.4. Note that it is possible that $x = 0$. This is the case we mentioned earlier in this section when discussing relative error criteria.

Now suppose that both $0 \in f^{\mathbf{I}}(x)$ and $0 \in f'(X)$. This can occur when X is either wide or narrow. It can occur when X is narrow and contains a multiple zero of f. If X is wide, it might contain a zero of f'; and the center x of X might accidentally be at or near a zero of f. In the latter case, we wish to reduce the width of the bound X. In either case, we wish to assure that the final bound on the zero approximates the optimal bound defined in Section 9.3.

Assume we are applying the interval Newton method in Section 9.5. Assume that when it is applied to a particular interval $X = [a, b]$, we find that $0 \in f^{\mathbf{I}}(x)$ and $0 \in f'(X)$. In this case, we use the following sub-procedure and then return to the main Newton algorithm.

The sub-procedure does one of three things. Steps 1 and 2 select a point of expansion for the Newton step. Step 3 decides whether an interval should be split. Step 4 decides when to accept an interval as a solution.

1. Evaluate f at the lower endpoint a of X. If $0 \notin f^{\mathbf{I}}(a)$, select the point $x = a$ as the point of expansion for the Newton step; and return to the main algorithm.

2. Evaluate f at the upper endpoint b of X. If $0 \notin f^{\mathbf{I}}(b)$, select the point $x = b$ as the point of expansion for the Newton step: and return to the main algorithm.

3. Denote $x_1 = \frac{1}{4}(3a + b)$ and $x_2 = \frac{1}{4}(a + 3b)$. If $0 \notin f^{\mathbf{I}}(x_1)$ or $0 \notin f^{\mathbf{I}}(x_2)$, split X in half and record the information that the interval X is to be split in half. Then return to the main algorithm. (Note that x_1 and x_2 are the centers of the two halves of X.)

4. Accept X as a final bound on a zero of f; and return to the main algorithm.

Note that X is accepted as a final bound in step 4 only if the computed (interval) value of f contains zero for each of the five equally spaced points $\mathrm{m}(X)$, a, b, x_1, and x_2. It is possible that X is a wide interval and each of these points is at or near a separate zero of f. However, we ignore this very unlikely possibility. Examination of output should reveal this case.

Use of higher precision interval arithmetic can help resolve the uncertainty when $0 \in f^{\mathbf{I}}(x)$ and $0 \in f'(X)$. It is possible to always determine the optimal bound for a zero as defined in Section 9.3. This can be done using bisection in conjunction with the interval Newton method.

Our overall stopping criterion is now as follows: If $0 \notin f'(X)$, we use the procedure in Section 9.3 which has its own method of stopping. If $0 \in f'(X)$ and $0 \in f^{\mathbf{I}}(x)$, we use the above procedure when Criterion 9.4.3 or 9.4.4 is not satisfied.

If $0 \notin f'(X)$, the procedure in Section 9.3 always continues until a suitable bound is obtained (or nonexistence of a zero of f in X is proved). If $0 \notin f^{\mathbf{I}}(x)$, a Newton step can always make progress because the point

x is not in the Newton result. If $0 \in f'(X)$ and $0 \in f^{\mathrm{I}}(x)$, a Newton step does not reduce X.

Note that we can set the tolerances ε_X and ε_f equal to zero. In this case, a final condition before an interval is accepted as a final bound is that $0 \in f^{\mathrm{I}}(x)$. In this case, Criteria 9.4.3 and 9.4.4 cannot be satisfied. Earlier, we mentioned the case in which $x = 0$ is a zero of f. Recall that Criterion 9.4.3 cannot be satisfied in this case if $0 \in X$ and $\varepsilon_X < 1$. This situation can also occur if a zero of f occurs near $x = 0$. No difficulty occurs if one or both criteria cannot be satisfied. Our algorithm in Section 9.5 assures that, in this case, termination occurs using either the procedure in Section 9.3 or the above procedure in this section.

Earlier in this section, we noted the possible difficulties in using an absolute error criterion for stopping an interval Newton method. However, a user might prefer such a criterion. It can replace Criterion 9.4.3 or be used in addition.

9.5 THE ALGORITHM STEPS

We now describe the steps of our interval Newton algorithm.

It is about as simple as an interval Newton method can be if it is to possess the desired convergence behavior. It is easily programmed. In Section 10.16, we describe a somewhat more complicated, but more efficient interval method for solving a nonlinear equation. It adds a procedure introduced in Section 10.3 to the interval Newton method to form a combined algorithm.

In both algorithms we assume that an initial interval X_0 and stopping tolerances ε_X and ε_f are given. The algorithm stops when interval bounds for all zeros of f in X_0 have been found.

After termination, the bounds on any simple zero generally approximate the optimal bound as defined in Section 9.3. If the tolerances ε_X and ε_f (see Section 9.4) are not chosen too small, then each multiple zero of f in X_0 is generally isolated within an interval of relative width less than ε_X. Also, we generally have $|f(x)| < \varepsilon_f$ for all points x in an interval bounding a multiple zero. When this condition is verified by the algorithm, it can be recorded for output.

Note that we can set $\varepsilon_X = \varepsilon_f = 0$. In this case, the algorithm generally finds the best possible solution(s) possible for the number system used on the computer.

In the following algorithm, the current interval is denoted by X at each step although it changes from step to step. Also x denotes $\mathrm{m}(X)$ so x changes as well. Except when using the procedure in Section 9.3 (see step 4 below) and Section 9.4 (see step 7 below), the Newton step is defined by an expansion about x.

The following steps are performed in the order given except as indicated by branching:

1. Put the initial interval X_0 into a list L of intervals to be processed.
2. If the list L is empty, stop. Otherwise, select the interval from L that has been in L for the shortest time. Denote the interval by X. Delete X from L.
3. If $0 \in f'(X)$, go to step 5.
4. Iterate the Newton method until either the result is empty or else $0 \in f(\mathrm{m}(X))$ .In the latter case, apply the procedure described in Section 9.3. If the result is empty, go to step 2. Otherwise record the solution interval that the procedure produces and go to step 2.
5. If $0 \in f^{\mathbf{I}}(x)$, go to step 7.
6. If $\frac{\mathrm{w}(X)}{|X|} < \varepsilon_X$ and $\mathrm{w}(f(X)) < \varepsilon_f$, record X as a final bound and go to step 2. Otherwise, go to step 8.
7. Use the procedure listed in Section 9.4. If that procedure prescribes a point of expansion record it; and go to step 8. If it decides that the interval X should be accepted as a solution, record X and go to step 2. If it prescribes that the interval is to be split in half, put the halves in the list L and go to step 2.
8. Apply a Newton step as given by (9.2.2) using the interval X. If a point of expansion was prescribed in step 7, use it in determining the expansion defining the Newton method. If the result is empty, go to

step 2. If the result is a single interval, go to step 9. If the result is two intervals, put them is list L and go to step 2.

9. If the Newton step reduced the width of the interval by at least half, go to step 3.

10. Split the current interval in half. Put one half in list L. Designate the other half as the current interval and go to step 3.

When the algorithm stops (see step 2), each zero of f in X_0 is in one of the intervals recorded in step 4, step 6, or step 7. Intervals might have been recorded that do not contain a zero of f. However, every zero in X_0 is in one of the output intervals. As noted in Section 9.3, the algorithm might prove the existence (using Theorem 9.6.8 below) of a zero of f in a recorded interval.

In step 2, we process the interval that has been in the list L for the shortest time. This tends to keep the list L short and conserve memory. This choice of interval is easily implemented using a stack. An alternative choice that keeps the list short is to choose the narrowest interval in L. Our choice tends to do this.

In the next chapter, we discuss a procedure that we have called "hull consistency". Before the above Newton method is used, hull consistency is applied. This can reduce the region of search for zeros of f.

The interval X_0 to which an interval Newton method is applied generally contains more than one zero of f. Steps 7 and 10 split the current interval and serve to separate different zeros into different intervals. This enables rapid convergence to each zero separately.

9.6 PROPERTIES OF THE ALGORITHM

The interval Newton method is a truly remarkable algorithm when compared to its noninterval counterparts. In this section, we present eight theorems that illustrate its reliability, efficiency, and other properties. For these theorems, we assume exact interval arithmetic is used. Relevant comments are included for the case in which practical rounded arithmetic is used.

We begin with a theorem due to Moore (1966).

Theorem 9.6.1 If there exists a zero x^* of f in X_n, then x^* is also in the interval $\mathrm{N}(x_n, X_n)$ given by (9.2.2).

The conclusion of this theorem is a motivating idea in the derivation of the interval Newton method. An examination of the derivation of the algorithm in Section 9.2, reveals that the theorem is correct.

In practice, when rounding occurs, we calculate an interval (or intervals) containing $\mathrm{N}(x_n, X_n)$. Hence, even with rounding, we never "lose" a zero. That is, the theorem is true even when rounding is present.

Theorem 9.6.2 Let an initial interval X_0 be given and assume that f and f' have a finite number of zeros in X_0. Denote the intervals in the list L at the i-th stage by $X_j^{(i)}$ $(j = 1, \cdots, N_i)$ where N_i is the number of intervals in L at stage i. Assume that at the i-th stage, a step of the interval Newton method is applied to the interval of greatest width in L. Then for arbitrary $\varepsilon > 0$, and all sufficiently large i, we have $w_i < \varepsilon$ where

$$w_i = \sum_{j=1}^{N_i} \mathrm{w}(X_j^{(i)}).$$

Note that in Theorem 9.6.2, we assume that the algorithm is applied to the widest interval in the list L. In practice, we generally apply the algorithm to the narrowest interval because this tends to minimize the number of intervals in L and conserves storage. Since the practical algorithm stops iterating on an interval while it is still finite in width, this does not affect the convergence argument. The amount of work is the same if the intervals are chosen from L in arbitrary order

In effect, this theorem says that the interval Newton algorithm always converges. The theorem is proved in Hansen (1978b). Note that with exact interval arithmetic, the interval $f^{\mathbf{I}}(x)$ obtained from evaluating $f(x)$ is the degenerate interval $[f(x), f(x)]$. Hence, if $f(x) = 0$ then x is a zero of f. Thus the difficulty that occurs in the practical algorithm (i.e., with rounding) when zero is in both $f^{\mathbf{I}}(x)$ and $f'(\dot{X})$ cannot occur with exact arithmetic. This simplifies the theorem's proof.

In the practical case, we can generally achieve "convergence" to the accuracy permitted by rounding. In Section 9.3, we discussed how to compute the optimal bound when $0 \notin f'(X)$. When $0 \in f'(X)$, this procedure can be added on after the algorithm in Section 9.5 terminates. Thus, we can compute an approximation for the optimal bound.

Theorem 9.6.3 Every discrete zero of f in X_0 is isolated and bounded to arbitrary accuracy.

Proof. From Theorem 9.6.1, no zero of f in X_0 can be "lost". From Theorem 9.6.2, the bounds become arbitrarily sharp. Thus, the theorem follows. ■

In practice, discrete zeros are isolated only if the number system provides the needed accuracy. The sharpness of the final bounds also depends on the accuracy of the number system used.

If there is no zero of f in X_0, this fact is proved by the algorithm. Theorem 9.6.4 to follow, shows how this occurs. Theorem 9.6.5 shows that it occurs within a finitely bounded number of iterations.

Theorem 9.6.4 (Moore, 1966) If $X \cap \mathrm{N}(x, X)$ is empty, then there is no zero of f in X.

Proof. If there is a zero of f in X, then it is also in $\mathrm{N}(x, X)$ by Theorem 9.6.1. Since $X \cap \mathrm{N}(x, X)$ is empty, there is no zero of f in X. ■

If rounding occurs, we compute an interval, say $\mathrm{N}'(x, X)$ containing $\mathrm{N}(x, X)$. If $X \cap \mathrm{N}'(x, X)$ is empty, then $X \cap \mathrm{N}(x, X)$ is empty. Therefore, the theorem is applicable even when rounding occurs.

Theorem 9.6.5 Assume $|f(x)| \geq \delta > 0$ for all $x \in X_0$ and $|f'(X_0)| \leq M$ for some positive number M. Then X_0 is entirely eliminated in m steps of the algorithm of Section 9.5 where

$$m \leq \frac{M\,\mathrm{w}(X_0)}{2\delta} \tag{9.6.1}$$

This theorem, proved in Hansen (1978b), says that if there is no zero of f in X_0, then this fact is proved in no more than m steps where m is bounded as in (9.6.1).

Suppose rounded interval arithmetic is used and that the conditions of this theorem on $f(x)$ and $f'(X_0)$ hold for the rounded values. Then X_0 is still eliminated in a finite number of steps. However, (9.6.1) might not be a correct bound.

Theorem 9.6.6 Assume $0 \notin f'(X_N)$ for some integer N. If f is a thin function (see Section 3.8), then $\mathrm{w}(X_{n+1}) \leq \frac{1}{2}\,\mathrm{w}(X_n)$ for all $n \geq N$. If f is a thick function, then $\mathrm{w}(X_{n+1}) \leq \frac{1}{2}\,\mathrm{w}(X_n)$ for any n for which $0 \notin f(x_n)$ and $0 \notin f'(X_n)$.

Proof. If f is a thick function and $0 \notin f(x_n)$ and $0 \notin f'(X_n)$, then $f(x_n)/f'(X_n)$ is either positive or negative. Therefore, $\mathrm{N}(x_n, X_n) < x_n$ or $\mathrm{N}(x_n, X_n) > x_n$. Since x_n is the midpoint of X_n, at least half of x_n is eliminated.

If f is a thin function, the same argument holds even if $f(x_n) = 0$. Moreover, $X_n \subset X_N$ for $n \geq N$. Therefore, by inclusion isotonicity, $f'(X_n) \subset f'(X_N)$ so $0 \notin f'(X_n)$ for all $n \geq N$. Hence, the theorem follows. ■

Note that when f is a thin function, rounding, in effect, turns it into a thick function. To invoke the theorem in the rounded case, we need only assume that the interval obtained when computing $f(x_n)$ does not contain zero.

If $0 \in f'(X_n)$, convergence is not as rapid. From (9.2.3) and (9.2.4), we see that X_{n+1} might be a single interval, the union of two intervals with a gap between them, or the empty set. In each case, we have made progress in reducing the region of search for a zero of f.

Using (9.2.3) and (9.2.4), it can be easily shown that $x_n \notin X_{n+1}$. Therefore, if X_n is a single interval, then $\mathrm{w}(X_{n+1}) \leq \frac{1}{2}\,\mathrm{w}(X_n)$. That is, substantial progress is made.

Suppose X_n is the union of two intervals and that $x_n = \mathrm{m}(X_n)$. This midpoint x_n is not in $\mathrm{N}(x_n, X_n)$. Therefore, each of the two subintervals generated in the n-th step is of width less than $\frac{1}{2}\,\mathrm{w}(X_n)$.

It might happen that $X_{n+1} = X_n$ when $0 \in f_n'(X_n)$. That is, the Newton step makes no progress. In this case, we split X_n in half. (See step 10 of the algorithm in Section 9.5.). Thus, in all cases, any new interval generated for use in a later Newton step has width less than or equal to half the width of the interval from which it is computed. This helps assure convergence as guaranteed by Theorem 9.6.2.

Theorem 9.6.7 If $0 \notin f'(X_n)$, then there exists a constant C such that $\mathrm{w}(X_{n+1}) = C[\mathrm{w}(X_n)]^2$.

This well-known theorem was first proved by Moore (1966) (see also Alefeld and Herzberger (1983)). The theorem states that if $0 \notin f'(X_n)$, then convergence is rapid asymptotically (i.e., quadratic) while Theorem 9.6.6 says that the rate can be reasonably fast (i.e., geometric) even for wide intervals.

Theorem 9.6.8 Let X be a finite interval. If $\mathrm{N}(x, X) \subset X$, there exists a simple zero of f in $\mathrm{N}(x, X)$.

This theorem was first proved by Hansen (1969a). His proof is for the case in which f is a thin function (as defined in Section 3.8). The proof contained herein follows as a special case of Theorem 9.6.9 below.

Note that evaluation of $\mathrm{N}(x, X) = x - f(x)/f'(X)$ involves division by $f'(X)$. If $0 \in f'(X)$, then $\mathrm{N}(x, X)$ is not finite and the hypothesis $\mathrm{N}(x, X) \subset X$ of the theorem cannot be satisfied. If X contains a multiple zero or more than one isolated zero of f, then $0 \in f'(X)$. Therefore, Theorem 9.6.8 can prove existence of simple zeros only.

Theorem 9.6.9 below is a generalization of Theorem 9.6.8. It is particularly useful in practice because it holds when f is a thick function (as defined in Section 3.8).

Let f depend on an interval parameter C. to emphasize this dependence, we rewrite $f(x)$ as $f(x, C)$ and $f'(X)$ as $f'(X, C)$. Assume that $f(X, C)$ is a continuously differential function of x for each $c \in C$. The function $\mathrm{N}(x, X)$ becomes

$$\mathrm{N}(x, X, C) = x - \frac{f(x, C)}{f'(X, C)}$$

To account for the parameter C, we rewrite Theorem 9.6.8 as follows:

Theorem 9.6.9 Let X be a finite interval. If $\mathrm{N}(x, X, C) \subset X$, then there exists a simple zero of $f(x, c)$ in X for each real $c \in C$.

This theorem (and the proof that follows) holds equally well when C is a vector of more than one interval parameter. We assume that C is a single parameter merely to simplify exposition.

Proof. We develop a proof by showing that $f(x, c)$ changes sign in X for each $c \in C$.

Let c be a point in C and let x and y be points in X. From the mean value theorem, for each $c \in C$,

$$f(y, c) = f(x, c) + (y - x)f'(\xi(c), c)$$

for some $\xi(c)$ between x and y. Since x and y are in X, it follows that $\xi(c) \in X$. Therefore,

$$f(y, c) \in f(x, c) + (y - x)f'(X, c) \tag{9.6.2}$$

for each $c \in C$.

Note that if $0 \in f'(X, c)$, then $\mathrm{N}(x, X, C)$ is not finite. Hence, the hypothesis $\mathrm{N}(x, X, C) \subset X$ of the theorem can be true only if $0 \notin f'(X, C)$ since X is finite. Note that the condition $0 \notin f'(X, C)$ implies that any zero of $f(x, c)$ in X must be simple for each $c \in C$.

Denote $f'(X, C) = [p, q]$. Then $0 \notin [p, q]$. Since we can change the sign of both f and f' without changing the algorithm, there is no loss of generality in assuming $f'(X, C) > 0$. Therefore, we assume that $p > 0$.

Since C is a nondegenerate interval, so is $f(x, C)$ even though x is degenerate. Denote $f(x, C) = [\underline{f}(x, C), \overline{f}(x, C)]$. Also, denote $X = [\underline{X}, \overline{X}]$ and $\mathrm{N}(x, X, C) = [\underline{N}(x, X, C), \overline{N}(x, X, C)]$.

We show that $f(\underline{X}, c) \leq 0$ and $f(\overline{X}, c) \geq 0$ for each $c \in C$, which implies that $f(x, c)$ has a zero in X for each $c \in C$. Note that the assumption $p > 0$ implies that $f(x, c)$ is monotonically increasing in X for each $c \in C$. Hence, if $\overline{f}(x, C) < 0$, then $\overline{f}(\underline{X}, C) < 0$. That is, $f(\underline{X}, c) < 0$ for each $c \in C$, as we wished to show.

Now consider the case $\overline{f}(x, C) \geq 0$. In this case, the lower endpoint of $\mathrm{N}(x, X, C)$ is

$$\underline{N}(x, X, C) = x - \frac{f(x, C)}{p}$$

By assumption, $\mathrm{N}(x, X, C) \subset X$. Therefore, the left endpoint $\underline{X}$ of X satisfies the inequality

$$\underline{X} \leq x - \frac{\overline{f}(x, C)}{p}.$$

That is,

$$\overline{f}(x, C) + (\underline{X} - x)p \leq 0$$

which implies that

$$f(x, c) + (\underline{X} - x)p \leq 0 \tag{9.6.3}$$

for each $c \in C$.

From (9.6.2),

$$f(\underline{X}, c) \in f(x, c) + (\underline{X} - x)[p, q]$$

and, hence,

$$f(\underline{X}, c) \leq f(x, c) + (\underline{X} - x)p$$

for each $c \in C$. Therefore, from (9.6.3), $f(\underline{X}, c) \leq 0$ for each $c \in C$.

We have now proved that $f(\underline{X}, c) \leq 0$ for all $c \in C$. Proof that $f(\overline{X}, c) \geq 0$ for all $c \in C$ follows in the same way. Therefore, $f(x, c)$ either is zero at an endpoint of X or changes sign in X. In either case, $f(x, c)$ has a zero in X for each $c \in C$ as stated in the theorem. ■

When rounding is present, we compute an interval, say $\mathrm{N}'(x, X, C)$, containing $\mathrm{N}(x, X, C)$. If $\mathrm{N}'(x, X, C) \subset X$, then $\mathrm{N}(x, X, C) \subset X$. Therefore, even when rounding is present, we can prove infallibly, as in Theorem 9.6.9 that a zero of f is contained in X.

The previous theorems in this section are related to the interval Newton method. The following theorem is not. However, its hypothesis can be checked using data computed for use in the interval Newton method. Therefore, it can be used when applying the method.

Theorem 9.6.10 Assume $0 \notin f'(X)$. Then if X contains a zero of f, the zero is simple (i.e., unique).

Proof. If $0 \notin f'(X)$, then $f'(x)$ is of one sign throughout X. That is, f is monotonic in X. Hence the theorem follows. ■

The computed (with rounding) interval $f'(X)$ contains the value of $f'(X)$ which can be computed with exact interval arithmetic. Therefore, the theorem is applicable using the rounded value.

9.7 A NUMERICAL EXAMPLE

We now give a simple example to illustrate the performance of the algorithm of Section 9.5 when the prescribed values of the tolerance ε_X and/or ε_f are too small. We set $\varepsilon_X = \varepsilon_f = 0$. In this case, the algorithm generally yields a solution that is slightly narrower than the "optimal bound" defined in Section 9.3. We use four decimal digit interval arithmetic.

Consider the function $f(x) = 4567(x-1)^2$, which we evaluate in the expanded form

$$f(x) = 4567x^2 - 9134x + 4567 \tag{9.7.1}$$

using Horner's rule. Let the initial interval be $X_0 = [0, 3]$.

For each of the first six steps of the algorithm, the interval value of f and/or f' does not contain zero and the intervals $X_1 = [0.7585, 0.8909]$ and $X_2 = [0.9349, 1.088]$ remain. The interval X_1 is deleted in one step. We find that $0 \in f^{\mathrm{I}}[\mathrm{m}(X_2)]$ and $0 \in f'(X_2)$.

However, $0 \notin f^{\mathrm{I}}(\underline{X}_2)$ so we use $\underline{X}_2$ as the point of expansion for the Newton step and proceed. After two more steps, we obtain the intervals $X_3 = [1.036, 1.041]$ and $X_4 = [0.9298, 1.024]$. The interval X_3 is deleted in one step. We find that $0 \notin f^{\mathrm{I}}(\underline{X}_4)$ so we proceed using $\underline{X}_4$ as the point of expansion.

The result of the next step is $X_5 = [0.9852, 1.024]$. We find that $0 \in f^{\mathrm{I}}[\mathrm{m}(X_5)]$ and $0 \in f^{\mathrm{I}}(\underline{X}_5)$. However, $0 \notin f^{\mathrm{I}}(\overline{X}_5)$ so we proceed using $\overline{X}_5$ as the point of expansion. One additional step yields $X_6 = [0.9852, 1.015]$. When f is evaluated at $\mathrm{m}(X_6)$, $\underline{X}_6$, and $\overline{X}_6$, each interval value contains zero. Therefore, we split X_6. When f is evaluated at the center of each of the subintervals of X_6, each result contains zero. Therefore, we accept X_6 as our final interval bound on the solution.

The optimal bound defined in Section 9.3 is $X^* = [0.9820, 1.015]$. Our result is somewhat sharper because the Newton method uses information about the derivative of f while the optimal bound is defined using f only.

9.8 THE SLOPE INTERVAL NEWTON METHOD

We now describe the slope interval Newton method. To obtain it, we modify the above interval Newton method by replacing the derivative f' by the slope function g discussed in Section 7.7.

From (7.7.1),

$$f(y) = f(x) + (y - x)g(x, y) \tag{9.8.1}$$

where $g(x, y)$ is the slope function. If y is a zero of f, then $f(y) = 0$ and, from (9.8.1),

$$y = x - \frac{f(x)}{g(x, y)}.$$

If y is in an interval X in which we seek a zero of f, then $y \in \mathrm{N_S}(x, X)$ where

$$\mathrm{N_S}(x, X) = x - \frac{f(x)}{g(x, X)}. \tag{9.8.2}$$

To find a zero of f in an interval X, we can use the iterative method

$$\mathrm{N_S}(x_n, X_n) = x_n - \frac{f(x_n)}{g(x_n, X_n)},$$
$$X_{n+1} = X_n \cap \mathrm{N_S}(x_n, X_n),$$

for $n = 0, 1, 2, \ldots$ where $X_0 = X$. A good choice for x_n is $\mathrm{m}(X_n)$. We call this procedure the *slope interval Newton method*. Figure 9.8.1 depicts this method.

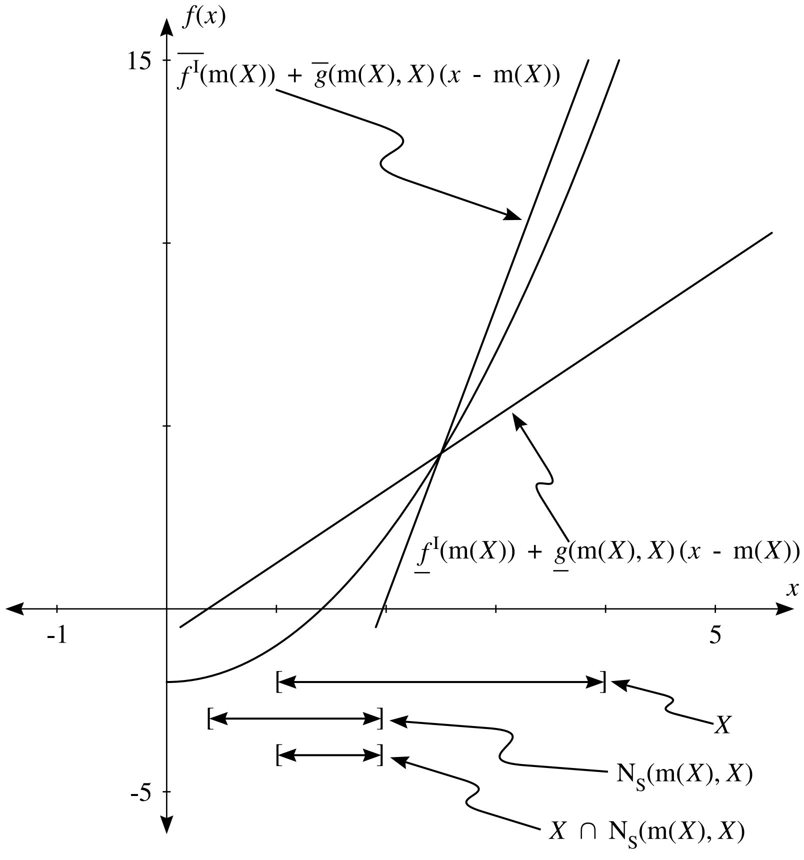

$f(x)$
15
$\overline{f^I}(\mathrm{m}(X)) + \overline{g}(\mathrm{m}(X), X)(x - \mathrm{m}(X))$
$\underline{f^I}(\mathrm{m}(X)) + \underline{g}(\mathrm{m}(X), X)(x - \mathrm{m}(X))$
x
-1
5
-5
X
$N_S(\mathrm{m}(X), X)$
$X \cap N_S(\mathrm{m}(X), X)$

Figure 9.8.1: Slope Interval Newton Method.

If we compare this relation with the relation (9.2.2) for the interval Newton method, the only apparent difference is that we have replaced $f'(X)$ by $g(x, X)$. Actually, there is another difference. To assure that any zero of f in X is also in $\mathrm{N}(x, X)$ as given by (9.2.2), it is necessary that the point x be in the interval X. For the slope interval Newton method, this is not necessary.

If $x \in X$, then $g(x, X) \subset f'(X)$; and the containment is generally strict. Therefore, the slope interval Newton method is generally more efficient than the interval Newton method given in (9.2.2).

It can be shown that Theorems 9.6.1 through 9.6.5 are true for this method using slopes. Proof of Theorem 9.6.2 requires that g have a finite number of zeros in X.

9.9 AN EXAMPLE USING THE SLOPE METHOD

We now consider a simple example to illustrate the virtue of the slope form of the interval Newton method. Consider the function

$$f(x) = x^4 + 3x^3 - 96x^2 - 388x + 480$$

discussed in Section 7.8. If we determine its slope analytically using (7.7.2), we can collect terms and write the slope as

$$\begin{aligned} g(x, X) = X^3 &+ (x + 3)X^2 \\ &+ (x^2 + 3x - 96)X + x^3 + 3x^2 - 96x - 388. \end{aligned}$$

Suppose we seek a root of $f(x)$ in the interval $X = [0, 4]$ and expand about the center $x = 2$ of X. Evaluating the slope using Horner's rule, we obtain $g(x, X) = [-904, -560]$. Since $f(x) = -640$, the slope Newton result is $[0.8571, 1.093]$ approximately.

Suppose we use Horner's method to evaluate the derivative in the standard interval Newton method. The Newton result is $[0.3505, 1.429]$ approximately. The ratio of the widths using the slope to that using derivative is 0.4034. That is, the slope result is considerably narrower.

In this example, we have used a relatively wide input interval $X = [0, 4]$. The slope method remains superior for narrower intervals. For example, if $X = [0.999, 1.003]$ and $x = 1.001$, the ratio of widths is 0.4744.

9.10 PERTURBED PROBLEMS

In some problems, there might be uncertainty about the values of certain parameters. For example, they might be measured quantities of uncertain accuracy. The function f whose zeros we seek might involve numbers that cannot be exactly expressed in the computer's number system. For example, the function might be expressed in terms of transcendental numbers such as π.

Any such parameters or numbers can be expressed as intervals that contain their true values and whose endpoints are machine-representable numbers. The "value" of a function $f(x)$ involving such intervals is itself an interval for any x. We must then ask: What do we mean by a solution to the equation $f(x) = 0$?

To answer this question, it is sufficient to consider a problem involving a single parameter p. Assume we know that p is contained in an interval P. We rewrite the problem as $f(x, P) = 0$. We define the solution to this problem to be the set $S = \{x : f(x, p) = 0\}$ for all $p \in P$.

For a given value of p, we expect the function f to have a set of discrete zeros. As p varies over P, a given zero, say x^* "smears out" over an interval, say X^*. This is precisely the situation we discussed in Section 9.3. Rounding errors made while evaluating $f(x)$ cause the resulting interval $f^{\mathrm{I}}(x)$ to contain zero when x is not a zero of f. In effect, the zeros of f are smeared out by the rounding errors.

Although the zeros of $f(x, P) = 0$ are generally discrete for a single value of p, some of the smeared zeros might overlap. This corresponds to the case in which rounding prevents us from determining whether there is a multiple zero or if there are close but separated zeros.

It makes no real difference to the interval Newton method whether values of f are intervals because of rounding in evaluating f or because f itself is a thick interval function. Therefore, there is no change in the algorithm to solve the perturbed problem.

We make rounding errors when evaluating the perturbed function $f(x, P)$. This merely widens the computed interval. An interval solution is, in effect, widened because of the rounding. As in the unperturbed case, this creates no difficulty.

If P is a wide interval, there can be considerable uncertainty as to where the boundary of the solution set lies. In Chapter 17, we discuss how this difficulty can be overcome.

One easily avoidable difficulty can occur with a perturbed problem. The interval Newton method might converge to an interval larger than a true smeared zero. To illustrate, we consider an example from Hansen and Greenberg (1983).

Let $P = [4, 9]$ and $f(x, P) = x^2 - P$. The solutions are obviously $X^* = [2, 3]$ and $[-3, -2]$. Let $X = [0.1, 4.9]$ and note that X contains only one of the smeared zeros of f. Then $x = \mathrm{m}(X) = 2.5$, $f(x) = [-2.75, 2.25]$, and $f'(X) = 2X = [0.2, 9.8]$. The Newton step produces the interval $\mathrm{N}(x, X) = [-8.75, 16.25]$. Therefore, no reduction in the original interval X occurs even though X is considerably wider than the solution $[2, 3]$ it contains.

Note that $0 \notin f'(X)$. This is the case we discussed in Section 9.3. In that discussion, we considered the case in which the computed (interval) value $f^{\mathbf{I}}(x)$ of $f(x)$ contains zero. For simplicity, we assume the only reason $f^{\mathbf{I}}(x)$ is not a degenerate interval is because of rounding. However, as noted above, there is no real difference if the value of $f^{\mathbf{I}}(x)$ is widened because of the presence of interval parameters.

Following the procedure prescribed in Section 9.3, we define three successive Newton steps by expanding about the center, the lower endpoint, and the upper endpoint of the current interval. Four of these cycles produce the interval solution $[1.9988, 3.0000]$ when recorded to five significant decimal digits. Additional iterations can increase the lower bound, thereby reducing the width of the result.

An alternative procedure for sharply bounding the solution set is given in Section 17.11.

Chapter 10

CONSISTENCIES

10.1 INTRODUCTION

Consider an equation $f(x, y) = 0$ and assume that x and y are in intervals X and Y, respectively. We can say that values of $x \in X$ and $y \in Y$ are consistent relative to the function f if for any $x \in X$ there exists $y \in Y$ such that $f(x, y) = 0$ and for any $y \in Y$, there exists $x \in X$ such that $f(x, y) = 0$. This concept obviously generalizes to more variables. It also generalizes in various other ways. See Collavizza *et al* (1999).

Suppose that for a subset of values of $x \in X$, there is no $y \in Y$ such that $f(x, y) = 0$. Then these values of x can be excluded from consideration when seeking solutions of $f(x, y) = 0$. This generalizes to functions of more variables. Suppose we are searching for a solution of a system of nonlinear functions in a given box. We can apply the concept of consistency to each equation of the system to eliminate subboxes of the given box that cannot contain the solution.

In this chapter, we consider two such procedures based on the concept of consistency. One is our version of what is called box consistency by McAllester *et al* (1995). See also Van Hentenryck, Michel, and Deville (1997) and Van Hentenryck, McAllester and Kapur (1997). To save space, we abbreviate box consistency as BC. We discuss BC in the next two sections. We discuss our version of "hull consistency" (see the above references) in Section 10.3 and discuss various aspects of it in Sections 10.4 through 10.12. We compare the two procedures in Section 10.13.

10.2 BOX CONSISTENCY

Various theoretical aspects of consistency are discussed in the above references. We restrict our discussion to those aspects needed to derive and implement our version of what is termed "box consistency". We motivate the procedure from a point of view different from that in the references. We also implement it differently. In the above references, the implementation of box consistency involves application of a one dimensional Newton method to "solve" a single equation for a single variable. This is the extent of the commonality between their procedure and ours.

Despite the differences in derivation and implementation, we refer to our procedure as the box consistency procedure or simply as box consistency. The abbreviation BC refers either to the concept or to the procedure.

Assume the solution of a given problem must satisfy the nonlinear equation

$$f(x_1, \cdots, x_n) = 0.$$

This equation can be one in a system to be solved or it can be a constraint that must be satisfied in an optimization problem. Whatever the problem, suppose that we seek a solution in a box $\mathbf{x^I}$. We use BC to eliminate subboxes of X that cannot contain a point satisfying $f(x_1, \cdots, x_n) = 0$.

If we replace all the variables except the i-th by their interval bounds (i.e., components of $\mathbf{x^I}$), we obtain an equation that we write as

$$q(x_i) = f(X_1, \cdots, X_{i-1}, x_i, X_{i+1}, \cdots, X_n) = 0. \qquad (10.2.1)$$

If $0 \notin q(x_i)$ for x_i in some subinterval X_i' of X_i, then we do not have consistency for $x_i \in X_i'$ and the subbox $(X_1, \cdots, X_{i-1}, X_i', X_{i+1}, \cdots, X_n)$ of X can be deleted.

To help motivate the BC procedure, let us now consider (10.2.1) from a different viewpoint. Note that $q(x_i) = 0$ is an equation in a single variable x_i since the X_j are fixed constant intervals for all $j \neq i$. If the intervals

X_j for all $j \neq i$ are degenerate, the zeroes of q (as a function of x_i) are isolated points. The presence of the interval constants "smears" these zeros into intervals that we call "interval zeros". If we evaluate $q(x_i)$ for some value of x_i, the resulting interval cannot contain zero unless x_i is in one of these interval zeros. The interval zeros can contain all or none of X_i. They can also form one or more subintervals of X_i. There might be gaps in X_i between interval zeros.

Let a_z denote the smallest value of x_i that is in the intersection of $X_i = [a_i, b_i]$ and the interval zeros of $q(x_i)$. If $a_z > a_i$, then a point $x_i \in X_i$ that is in the semi-open interval $[a_i, a_z)$ cannot be a component of a solution of $f(x) = 0$. Therefore, we want to know a_z so that we can delete the interval $[a_i, a_z)$. In practice, we use a one dimensional Newton method to compute an interval bound on a_z and delete values of x_i less than the lower bound on a_z.

Similarly, let b_z denote the largest value of x_i in the intersection of X_i and the interval zeros of $q(x_i)$. We want to delete the subinterval $(b_z, b_i]$ of X_i.

The widths of the interval roots of $q(x_i)$ depend on the widths of the interval bounds X_j for all $j \neq i$. Therefore, a_z and b_z change as the latter intervals change. There is little point in bounding the quantities a_z and b_z very sharply unless the bounds on other variables are relatively narrow. Therefore, we make only a modest effort to bound a_z and b_z before sharpening the bounds on the variables other than x_i (using the same process). Then we do the same process for the other variables in turn. We also use other procedures besides BC to narrow the bounds on each variable before returning to again sharpen the bounds on x_i.

Unless there is substantial progress, we apply only one Newton step to narrow the bound on a_z and one Newton step to narrow the bound on b_z. However, there are cases in which it is reasonable to use more steps. For example, suppose that f is a very complicated function and/or there are many variables and it requires 1000 arithmetic operations to compute $q(x_i)$ in the form (10.2.1) from f. Suppose, also, that only 5 arithmetic steps are needed to evaluate $q(x_i)$ and 5 steps to evaluate the derivative $q'(X_i)$. Then applying a Newton step to solve $q(x_i) = 0$ is so cheap compared to computing $q(x_i)$ that we might as well apply Newton several times.

Therefore, a user might wish to modify our algorithm (given below) for appropriate functions.

To simplify notation, we now drop the index i. We seek to bound a_z and b_z in an interval $X = [a, b]$.

Suppose we evaluate $q(a)$. If $0 \notin q(a)$, then $a < a_z$ so we use a Newton method to remove points from the lower end of X that are less than a_z. If $0 \in q(a)$, we do not. To remove points, we use the already computed value of $q(a)$ in a step of a one dimensional Newton method. Similarly, we try to reduce the upper bound on b_z only if $0 \notin q(b)$. The efforts to increase a and decrease b are treated independently.

The input interval to which the algorithm is applied is $[a, b]$. We separately try to increase a and decrease b. A new lower bound is sought in a subinterval $Y = [a, c]$ of X and a new upper bound is sought in a subinterval $Z = [d, b]$ of X. Later in this section, we describe how Y and Z are determined by choosing c and d.

It is common (good) practice to use the center of the interval as the point of expansion when deriving an interval Newton method. However, since we evaluate $q(a)$ to decide whether to even try to increase the lower bound, we define the Newton method to use this value. This saves an extra evaluation of q. Thus, the Newton result when expanding about the point a is

$$\mathrm{N}(a, Y) = a - \frac{q(a)}{\frac{\partial}{\partial x_i} q(Y)}.$$

For the interval $Y = [-4, 2]$, Figure 10.2.1 illustrates how a Newton step about the point $a = -4$ produces a (small) reduction in the width of Y. The slopes of the slanting lines in Figure 10.2.1 equal the lower and upper bounds of $\frac{\partial}{\partial x} q(Y)$.

Similarly, we use $q(b)$ to obtain a Newton result. Expanding about b, the Newton result is

$$\mathrm{N}(b, Z) = b - \frac{q(b)}{\frac{\partial}{\partial x_i} q(Z)}.$$

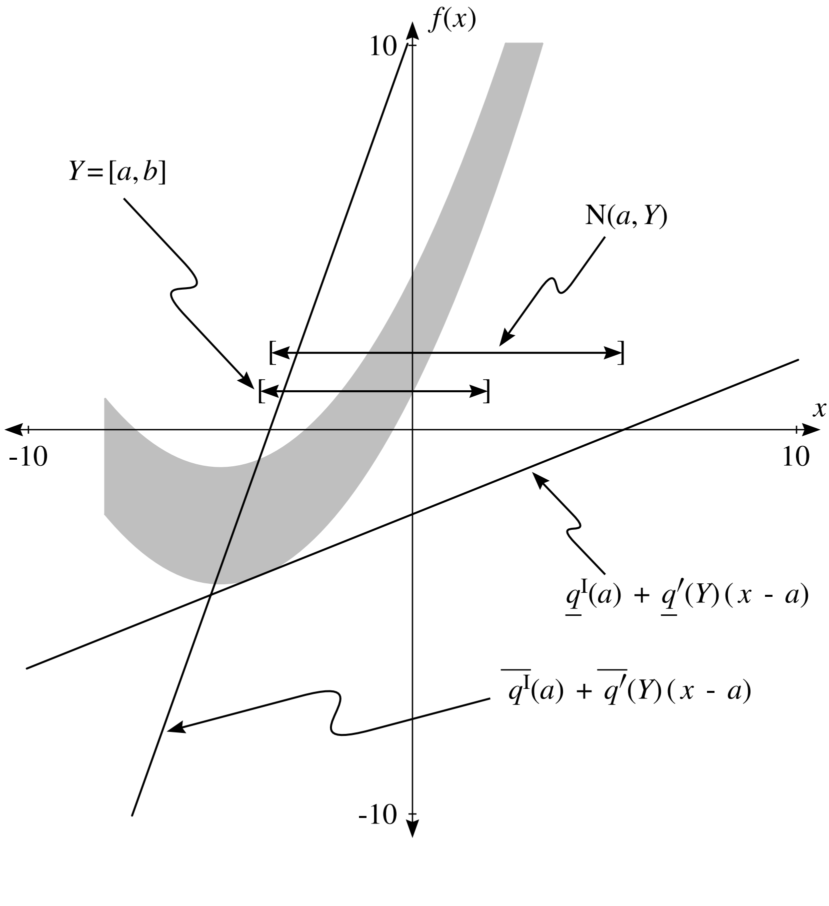
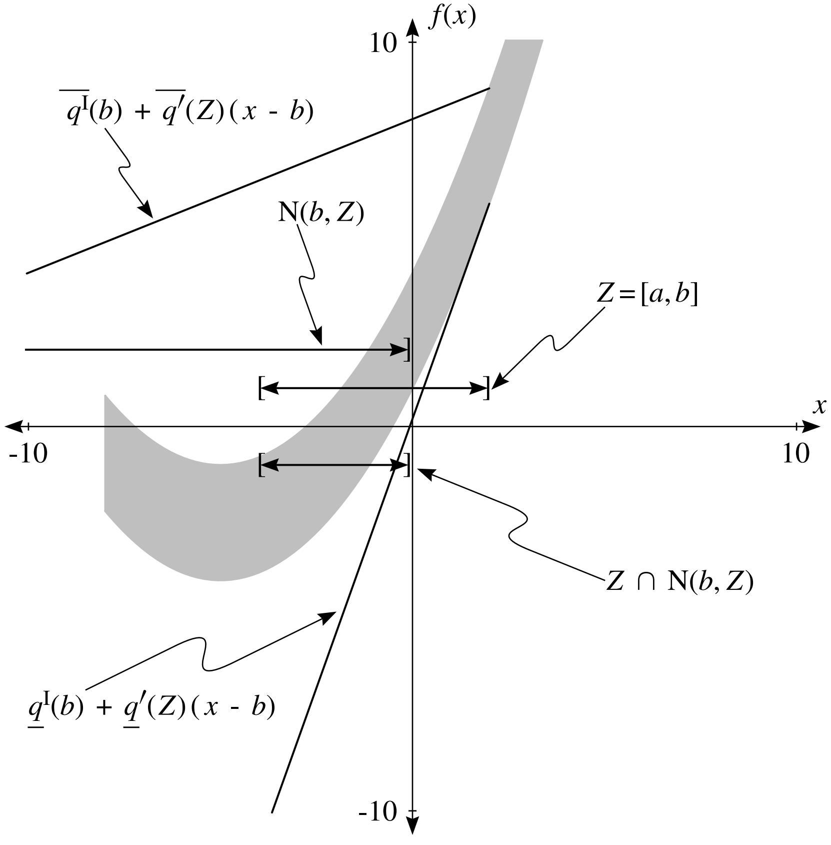

$f(x)$
10
$Y = [a, b]$
$\mathrm{N}(a, Y)$
x
-10
10
$\underline{q}^{\mathrm{I}}(a) + \underline{q}'(Y)(x - a)$
$\overline{q}^{\mathrm{I}}(a) + \overline{q'}(Y)(x - a)$
-10

Figure 10.2.1: Newton iteration with expansion about left endpoint.

For the interval $Z = [-4, 2]$, Figure 10.2.2 illustrates how the Newton step about the point $b = 2$ produces a substantial reduction in the width of Z. In our figures, $Y = Z$. In practice, this will not be the case.



$f(x)$
10
$\overline{q}^{\mathrm{I}}(b) + \overline{q'}(Z)(x - b)$
$\mathrm{N}(b, Z)$
$Z = [a, b]$
x
-10
10
$Z \cap \mathrm{N}(b, Z)$
$\underline{q}^{\mathrm{I}}(b) + \underline{q}'(Z)(x - b)$
-10

Figure 10.2.2: Newton iteration with expansion about right endpoint.

We need to choose the widths of the intervals Y and Z to which these Newton steps are applied. Our choices depend on how much progress is made in previous Newton steps.

We now describe how we choose Y. We choose Z in the same way. We choose the width of Y to be a variable fraction β of the width of X so $Y = [a, a + \beta\,\mathrm{w}(X)]$. If we choose Y too narrow, we make little progress in reducing X even if all of Y is deleted. If we choose Y too wide, the derivative of q over Y is a wide interval and again little (or no) progress is made.

Initially, we choose β relatively small. We choose β so that if all of Y is deleted, we delete a small but non-negligible part of X. If we succeed, we repeat the Newton step with a larger value of β.

Thus, we choose $\beta = \frac{1}{4}$ initially. That is, we choose $c = \frac{1}{4}(3a + b)$. If all of Y is deleted by the Newton step, we double β and repeat the step on the remaining part of X. If only a part of Y is deleted, we stop trying to increase the lower endpoint of X. We then use the same procedure to try to reduce the upper endpoint of X. That is, we apply the Newton method to an interval $Z = [d, b]$ with the point of expansion equal to b. The value of d is chosen in the same way c is chosen.

In the algorithm, we use a tolerance ε_X to decide when a given interval is sufficiently narrow to provide a final bound on a variable. We discussed such tolerances in Section 9.4. Different tolerances can be chosen for each variable.

The algorithm below lists steps to increase the lower bound on a_z. Similar steps are used to decrease the upper bound on b_z. After listing the steps, we discuss why some are chosen as they are.

1. Set $\beta = \frac{1}{4}$ and $w_0 = b - a$.

2. If $0 \in q(a)$, exit from the algorithm.

3. Denote $w = b - a$ and set $c = a + \beta w$. Define $Y = [a, c]$.

4. Compute the Newton interval $\mathrm{N}(a, Y)$ and the interval $Y' = Y \cap \mathrm{N}(a, Y)$.

5. If Y' is empty and $\beta = 1$, record the fact that all of X has been deleted and exit from the algorithm.

6. If $\mathrm{w}(Y') < \varepsilon_X$ and $\beta = 1$, record Y' and exit from the algorithm.

7. If Y' is empty and $\beta < 1$, replace a by c and replace β by 2β and go to Step 2.

8. If $\mathrm{w}(Y') < 0.5\,\mathrm{w}(Y)$ and $\beta = 1$, replace a by $\underline{Y}'$ and replace b by $\overline{Y}'$ and go to Step 2.

9. If $a < \overline{Y}'$ and $\overline{Y}' < c$, then a gap $(\overline{Y}', c)$ has been generated in the interval $[a, b]$ leaving two intervals $\mathrm{N}(a, Y)$ and $[c, b]$. Exit the algorithm and then return to apply it separately to each of the two intervals. (Note that, since $\mathrm{N}(a, y) \subset Y$, a solution to $q(x_i) = 0$ exists in $\mathrm{N}(a, y)$.)

10. Replace a by $\underline{Y}'$

11. If $b - a < 0.5w_0$, go to Step 1.

12. Record the final interval $[a, b]$ and terminate the algorithm.

This is the BC algorithm for narrowing the bounds on a given component of a box X. Note that if $0 \notin q(a)$, then $a \notin \mathrm{N}(a, Y)$. That is, progress is made in reducing X. If the progress is sufficient, we apply another Newton step. See Steps 8 and 11.

The algorithm to use BC is called by a main program that stops when the interval bounds on the variables satisfy: $\mathrm{w}(X_i) \leq \varepsilon_X$ $(i = 1, \cdots, n)$ for some $\varepsilon_X > 0$. The interval resulting from the Newton step to bound a_z might satisfy this condition. If so, we exit without trying to reduce the interval any further. See Step 6.

If $\mathrm{N}(a, Y) \subset Y$ in Step 4, we have proved that an interval zero of q exists in $\mathrm{N}(a, Y)$. See Theorem 9.6.9. Since

$$q\,(x_i) = f\,(X_1, \cdots, X_{i-1}, x_i, X_{i+1}, \cdots, X_n)$$

this proves the existence of a solution of $f(x) = 0$ for any $x_j \in X_j$ $(j = 1, \cdots, n;\ j \neq i)$ provided $x_i \in \mathrm{N}(a, Y)$. This fact might or might not be significant.

Note that BC can be applied to "solve" inequalities. Suppose that in place of the equality (10.2.1), we have an inequality $q(x_i) \leq 0$. We can replace this inequality by $q(x_i) = [-\infty, 0]$ and obtain the equation $q(x_i) + [0, +\infty] = 0$. Note that the derivative of this new function is the same as if the original relation were an equation.

We conclude our discussion of BC with comments on how it is applied. Suppose there is more than one equation to be solved. Suppose we have applied BC to one equation to try to narrow the bounds on x_i; and we now wish to try to narrow the bounds on the next variable x_{i+1}. We use a different equation to do so.

To see why a different equation is used, consider the following argument. Suppose we reduce the current box of interest by decreasing the bound on x_i when we apply BC to a given equation. The probability that an arbitrary point satisfies this equation is greater if the point is chosen randomly from the new reduced box than from the old unreduced box. Because unsatisfied equations are needed to reduce or delete a box, it is better to use a different equation the next time BC is applied to the new reduced box.

Thus, we cycle through the equations in the order in which they occur. We cycle through the variables in such a way that each equation is solved once for each variable. Having chosen an equation to be solved, we solve for the variable with smallest index that has not been used with the given equation.

We have described a simple way to order variables and equations. Further research might provide an improved procedure.

10.3 HULL CONSISTENCY

In this section, we introduce our version of a concept (and procedure) called "hull consistency". We apply it in various ways throughout this book.

We begin our discussion with a function of a single variable having a special form. Suppose we wish to solve an equation of the form

$$f(x) = x - h(x) = 0.$$

For a solution x^* of $f(x) = 0$, we have $x^* = h(x^*)$. Given an interval X, inclusion isotonicity of containment sets (lemma 4.8.8) assures that any solution in X satisfies $x^* \in h(X)$. Therefore the interval $X' = h(X)$ contains any solution in X. If $X' \cap X$ is smaller than X, this intersection provides improved bounds on any solution in X.

When f contains multiple terms, we replace x by its bound X in some terms of f. We then solve for x from the remaining terms. We abbreviate "hull consistency" as HC. We now consider a more general form of f.

Assume f involves a function g that has an easily obtained inverse. For example, g might be a power of x or e^x or even a polynomial in x. Assume, also, that we can easily solve for g from f. For example, if $f = ug - v = 0$, we obtain $g - \frac{v}{u} = 0$. If $f = \frac{u}{g+v} - w = 0$, we obtain $g - \frac{u}{w} + v = 0$. For simplicity, we assume that such manipulations have been done and that

$$f(x) = g(x) - h(x) = 0.$$

If $x^* \in X$, then containment-set inclusion isotonicity assures that $x^* \in g^{-1}[h(X)]$. In practice, the width of the interval bound $g^{-1}[h(X)]$ depends on how well we are able to overcome dependence in evaluating $h(X)$.

There are usually many choices for $g(x)$. For example, suppose we wish to use HC to narrow the interval bound X on a solution of $f(x) = ax^4 + bx + c = 0$. We can let $g(x) = bx$ and compute $X' = -\frac{aX^4+c}{b}$ or we can let $g(x) = ax^4$ and compute $X' = \pm\left(-\frac{bX+c}{a}\right)^{\frac{1}{4}}$. We can also separate x^4 into $x^2 * x^2$ and solve for one of the factors as $X' = \pm\left(-\frac{bX+c}{aX^2}\right)^{\frac{1}{2}}$. We consider a particular method for choosing g in Section 10.6.

A virtue of consistency methods is that they can work well "in the large". When we seek a solution of $f(x) = 0$, we often start the search over a large interval to assure that it contains the solution. When the solution is not where $|x|$ is large, we must somehow eliminate large values. For this purpose, HC is very useful.

As an example, suppose we seek a solution of $x^4 + x - 2 = 0$ in the interval $X = [-100, 100]$. Solving for x^4 and replacing x in the remaining terms by the interval X, we obtain $(X')^4 = 2 - [-100, 100] = [-98, 102]$. Since $(X')^4$ must be non-negative, we replace this equation by $(X')^4 = [0, 102]$ and conclude that $X' = \pm[0, 102]^{\frac{1}{4}}$ so $X' = [-3.18, 3.18]$ approximately. This is a substantial reduction of the original interval.

In Chapter 9, we discussed the interval Newton method for solving nonlinear equations. Its asymptotic convergence to a solution is usually rapid. However, generally it does not perform well when the interval in which a solution is sought is very wide. For the example just discussed, a step of the Newton method is able to delete a sub-interval of width only about 10^{-6} from the original interval $[-100, 100]$.

When we seek all the solutions of a given equation, we often begin the search in a large interval to assure that all solutions are included. Our solution procedure must eliminate the values of large magnitude. A Newton method is not efficient in doing so. The BC and HC procedures fill this need.

The Newton method and HC complement each another. One can work well when the other does not. In an algorithm using both methods to solve a system of nonlinear equations, we can emphasize use of the Newton method when the interval is narrow and emphasize use of HC (and BC) when the interval is wide.

Unfortunately, it is difficult to know when an interval is narrow or wide in this sense. In our algorithms using HC and BC, we monitor their behavior relative to that of a Newton method and emphasize use of each based on the observed behavior.

In the multidimensional case, there is an essential difference between consistency methods and a Newton method. The former are applied to one equation of a system at a time while a Newton method is applied to all equations simultaneously. This enables the Newton method to have better convergence properties "in the small". However, multidimensional Newton methods tends to make little or no progress when the box is "large".

Note that HC can be applied to inequalities. We need only replace an inequality of the form $f(x) \leq 0$ by the equation $f(x) = [-\infty, 0]$.

10.4 ANALYSIS OF HULL CONSISTENCY

Consider a general function $f(x) = g(x) - h(x) = 0$. The iterative step we are considering is $g(X') = h(X)$ from which $X' = g^{-1}[h(X)]$. If necessary, we delete[1] any values of the range of $h(X)$ that are not in the domain of g^{-1}. Therefore, we obtain $X' = g^{-1}(Z)$ where Z is the intersection of the ranges of g and h over X.

For example, suppose $f(X) = X^2 - X + 6$ and we define $g(X) = X^2$ and $h(X) = X - 6$. Let $X = [-10, 10]$. The procedural step is $(X')^2 = X - 6 = [-16, 4]$. Since $(X')^2$ must be non-negative, we replace this interval by $[0, 4]$. Solving for X', we obtain $X' = \pm[0, 2]$. In replacing the range of $h(x)$ (i.e., $[-16, 4]$) by non-negative values, we have excluded that part of the range of $h\ (x)$ that is not in the domain of $g(x) = x^2$.

Suppose that we reverse the roles of g and h and use the iterative step $h(X') = g(X)$. That is, $X' - 6 = X^2$. We obtain $X' = [6, 106]$. Intersecting this result with the interval $[-10, 10]$, of interest, we obtain $[6, 10]$. This interval excludes the set of values for which the range of $g(X)$ is not in the intersection of the domain of $h(X)$ with X.

Combining these results, we conclude that any solution of $g(x) - h(x) = 0$ that occurs in $X = [-10, 10]$ must be in both $[-2, 2]$ and $[6, 10]$. Since these intervals are disjoint, there can be no solution in $[-10, 10]$.

In practice, if we have already reduced the interval from $[-10, 10]$ to $[-2, 2]$ by solving for g, we use the narrower interval as input when solving for h.

This example illustrates the fact that it can be advantageous to solve a given equation for more than one of its terms. The order in which terms are chosen affects the efficiency. Unfortunately, it can be difficult to choose the best order.

Figure 10.4.1 illustrates a simple example of hull consistency.

The interval Newton method works rather well when solving quadratics. Nevertheless, it requires seven steps to prove that there is no zero of $x^2 - x + 6 = 0$ in $[-10, 10]$. Thus, HC is much more efficient for this example.

When $g(x)$ is a sufficiently simple function, the step of solving $g(X')$ for X' can be done sharply. However, if $f(x)$ is not a simple function, then the simplicity of g implies that $h(x) = f(x) - g(x)$ is more complicated. Therefore, when evaluating $h(X)$ in practice, dependence can prevent us

[1] Such deletions are done automatically by the compiler if cset-based interval arithmetic is used.

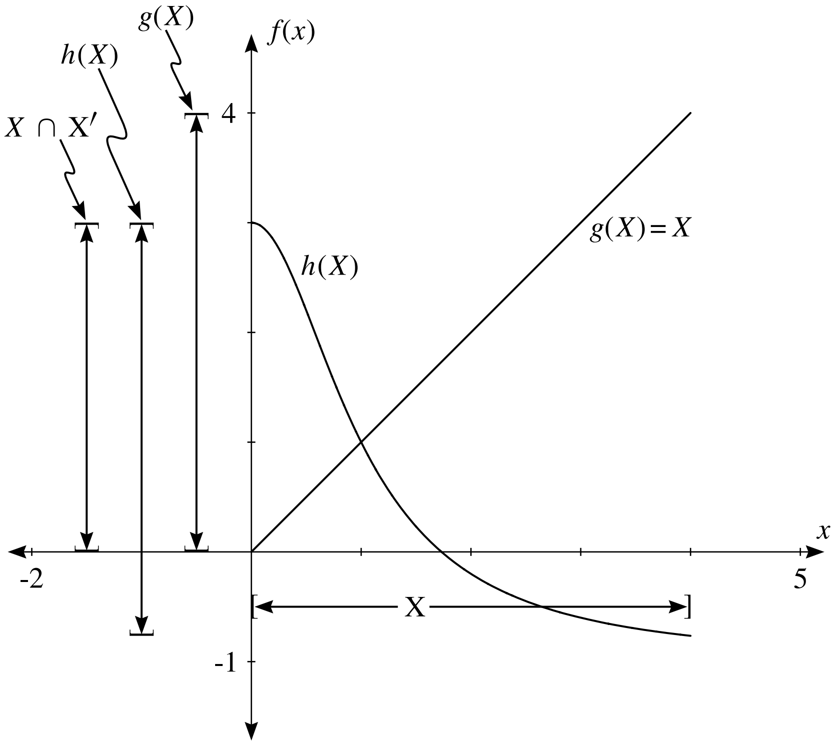


Figure 10.4.1: Hull Consistency Example

from computing sharp bounds on the range of h. In this case, we do not delete as much of X as is possible if we know the exact range of h over X.

10.5 IMPLEMENTING HULL CONSISTENCY

There are so many ways to implement HC that it is difficult to choose a procedure. We take the viewpoint that we want it to be most efficient in eliminating variable values that are large in magnitude relative to solution values. This helps us choose the implementation of HC to use.

Earlier, we noted why we want HC to eliminate variable values that are large in magnitude. It is because we initially introduce such values to assure that our region of search includes all solutions to a problem. Since a Newton method is not efficient in eliminating such values, we want HC to do so.

Let us look at the general procedure for applying HC and then focus on the case in which the width of the interval X is "large". When choosing g, we want g and $h = f - g$ to have disjoint ranges over as much of X as possible. One way to do this is to find a term of f that dominates the other terms for some portion of X. For example a given power of x dominates a power of lower degree for sufficiently large values of $|x| > 1$. This makes HC a valuable tool for reducing "large" boxes because it is often easy to find such a term.

Note that it is not always the term of highest degree that dominates in a given interval. For example, $100x^3$ dominates x^4 when $|x| < 100$.

When $|x| < 1$, a term of lower degree tends to dominate a term of higher degree. Therefore, the implementation of HC to delete small values of the variable must be different from when large values are to be deleted.

A simple procedure for choosing which term of $g(x)$ to solve is to evaluate all terms that are easily inverted and solve for the term with widest range.

Solving for the dominant term is effective in practice. However, sometimes there is a better choice. Consider the function

$$f(x) = x^4 + x^2 - x - 1 = 0.$$

It does not have a zero in the interval $X = [2, 10^8]$. If we apply HC by solving for any one term, we obtain an interval that intersects X. However, if we write the term x^2 as x times x and solve for one of the factors, we obtain

$$x = 1 - \tfrac{1}{x} - x^3.$$

Solving for X' as $1 - \frac{1}{X} - X^3$, we find that X' is negative so it has an empty intersection with X. This proves that $f(x) = 0$ does not have a solution in X.

HC can be automated so that various implementations are used. Without automation, it is simpler to program only one or a few different forms. Rather than change implementations of HC as the box size changes, we can use one of the following options.

First, we can implement HC so that it is effective for large boxes and rely on Newton's method to provide efficiency when this is not the case.

Second, we can use more than one implementation of HC to increase the likelihood that one is efficient. This has a drawback. If we solve each of a system of n equations for each of the n variables, this is n^2 procedures. Until HC is automated, adding additional procedures might not be warranted, especially because of the amount of programming involved. On the other hand, the example in Section 10.4 shows that it can be advantageous to solve a given equation for more than one occurrence of a given variable. It might be possible to automate the implementation of HC and avoid extensive programming for a given problem.

We can solve an equation for each of two occurrences of each variable. For each variable, we can solve for a term that tends to dominate when the magnitude of the variable is large, and also solve for a term that tends to dominate when the magnitude of the variable is small.

Suppose we wish to apply HC when the current interval is $X = [a, b]$. Another option is to solve for a term that dominates when $x = a$ and then solve for a term that dominates when $x = b$.

We prefer another option that we now describe. Suppose $f(x)$ is a complicated function but has several simple terms that enter additively.

That is, suppose $f(x)$ is of the form

$$f(x) = \sum_{i=1}^{m} g_i(x) - h(x) = 0 \tag{10.5.1}$$

and assume the inverse of each function $g_i(x)$ $(i = 1, \cdots, m)$ is easily determined. We might wish to use each $g_i(x)$ as the function to solve using HC. We now describe how cancellation can be used to simplify the process in a suboptimal way.

To simplify the discussion, let $m = 2$ so that

$$f(x) = g_1(x) + g_2(x) - h(x).$$

Let X be given and evaluate $g_1(X)$, $g_2(X)$, $h(X)$, and then $f(X)$. Suppose we have used g_1 as the function to solve using HC and obtained a new interval X'.

We now want to solve for g_2 as $g_2(X'') = h(X') - g_1(X')$. But if f (and hence h) is quite complicated, this is a lengthy computation. Instead of computing $h(X')$, we can use $h(X)$, which has already been computed. This saves computing at the expense of loss of sharpness. Therefore, we want to obtain

$$g_2(X'') = [f(X) - g_1(X) - g_2(X)] + g_1(X').$$

However, $f(X)$ contains $g_1(X) + g_2(X)$ and we lose sharpness if we subtract an interval from itself. Therefore, we use cancellation. That is, we replace $f(X) - g_1(X) - g_2(X)$ by $f(X) \ominus [g_1(X) + g_2(X)]$.

Thus, we obtain $g_2(X'')$ using only two additions and one cancellation. This is not really what we want because our result uses $h(X)$ rather than $h(X')$. However, it saves the effort of computing $h(X')$. This use of $h(X)$ is implicit since we actually use $g(X) - f(X)$.

If we solve for additional functions g_i $(i > 2)$ in the same way, we implicitly use X as the argument of h, X' as the argument of g_1, X'' as the argument of g_2, etc. Therefore, we obtain a narrower result if we reevaluate f each time we solve for a new term. If we reevaluate f each time, we do so m times for a function of the form (10.5.1). Using the procedure just described, the total amount of work is generally less than that to evaluate f twice.

This process produces a result that is generally much better than solving for a single term only. Therefore, we might as well solve (in this way) a given function f for every simple term in its expression.

In this procedure, we can order the terms g_i $(i = 1, ...m)$ of f in any way we like. As we discussed earlier, it can be advantageous to implement HC to solve for a dominant term when $|x|$ is large. In the procedure just described, we can choose g_1 to be such a term. That is, we can eliminate large values of a variable first.

This same procedure can be used in the multidimensional case when each summand g_i is solved for a different variable. When there is more than one variable, there is likely to be more than one equation to be solved. For a system of equations, we can cycle through the equations and variables as described for BC at the end of Section 10.2. When these functions have more than one simple term, we can cycle through the terms as well while using the procedure just described.

Often in practice, we wish to apply HC to a function that contains terms that are powers of x. In this case, we do not solve for each term separately as described above. Instead, we simultaneously use all the terms that form a polynomial.

For example, suppose the equation

$$f(x, y) = x^4 y - x^2 y^2 - 4x - 2e^x = 0$$

is to be satisfied in the box given by $X = [1, 2]$ and $Y = [1, 2]$. Replacing y by its bounding interval Y and replacing e^x by its bound $[e, e^2]$, we obtain the polynomial

$$[1, 2]x^4 - [1, 4]x^2 - 4x - [2e, 2e^2] = 0.$$

Computing the roots of this polynomial by the method of Chapter 8, we obtain the new bound $[1.6477, 2]$ on x.

In some cases, it can be useful to solve for roots of a moderately complicated function. For example, suppose the function is a multinomial. This is often the case. If we replace all the variables except one by their bounding intervals, we obtain a polynomial (with interval coefficients) in the remaining variable. Call it x_1. The interval roots of an interval polynomial can be found using the method of Chapter 8. These interval roots bound the acceptable values of x_1. Note that the interval roots need be sought only in the (assumed known) interval that bounds x_1.

In this and later chapters, we consider polynomial examples that are normally treated in the manner just described. However, we do not do so because we are demonstrating other aspects of HC.

While it is generally a good idea to replace all variables except one by their interval bounds to get a polynomial in the remaining variable, this is not always true. Consider the function

$$f(x, y) = x^2 + y^2 + x^2y^2 - 24xy + 13. \tag{10.5.2}$$

Suppose $f(x, y) = 0$ is one equation of a system that we wish to solve. Assume we seek a solution in a box given by $x \in [-a, a]$ and $y \in [-a, a]$ where a is some large number.

If we replace y by its bound $[-a, a]$ and solve the resulting quadratic equation for x, we learn only that $x \in \pm\left[\frac{13}{24a}, a\right]$, approximately. Since a is large, we have eliminated very little of the initial box.

But, suppose we write the function in the form

$$f(x, y) = x^2 + y^2 + (xy - 12)^2 - 131.$$

If we replace both x and y by their bounds except for the x^2 term and solve for x^2 (and then x). We obtain

$$x \in (131)^{1/2}[-1, 1] = 11.45[-1, 1]. \tag{10.5.3}$$

Here and in what follows we record results to four significant digits. This greatly improved bound is independent of the size of a.

This example can be used to illustrate how variations in the use of HC can change a result. Assume we have not obtained the bound (10.5.3) and we solve for the term $24xy$ from (10.5.2). Replacing all other terms by their bounds. We obtain $xy = [0.3909, a^2]$. This relation can be useful in the process of solving the system of which $f(x, y) = 0$ is a member. It shows that x and y have the same sign.

But,we can do better by writing $f(x, y)$ in the form

$$(x - y)^2 + x^2y^2 - 22xy + 13 = 0$$

and solve for $22xy$. We obtain the slightly better result $xy = [0.5909, a^2]$.

We can do still better by writing $f(x, y) = 0$ in the form

$$(x - y)^2 + (xy - 11)^2 - 108 = 0.$$

Solving for the term $(xy - 11)^2$, we obtain $xy = [0.6076, a^2]$.

10.6 CONVERGENCE

The step to solve for $g(x)$ (and then x) can be iterated. Thus we can define

$$X^{(k+1)} = g^{-1}[h(X^{(k)})] \cap X^{(k)}$$

for $k = 0, 1, 2, ...$ where $X^{(0)}$ is an initial interval in which a solution of $f(x) = 0$ is sought. This iterative procedure might or might not converge to a single point. In practice, we generally do not iterate this process to convergence. Nevertheless, consideration of its convergence can help in choosing g.

We have noted that a primary virtue of HC is its ability to delete substantial portions of wide boxes. However, it can be of value for narrow boxes as well. That is, it can be of value asymptotically when the search for a solution is in a small box. For best performance on narrow boxes, special implementation is required. We discuss this aspect in this section.

Our algorithm for solving systems of nonlinear equations (see Chapter 11) uses a multidimensional Newton method. When applied to a small box, it generates (by preconditioning) a new system of linearized equations such that each equation depends strongly on a single variable and weakly on all others. This enables HC and BC (which operate on only one equation at a time) to perform well on this new system. Since HC requires much less computing than the Newton method, we apply it because it can be profitable to do so. In this case (when the box is small), we use a special implementation of HC.

Another reason to apply HC to small boxes is that it might prove existence of a solution of a system of nonlinear equations in the multidimensional case. See Section 10.12. The possibility of doing so is enhanced when the procedure converges rapidly as a one-dimensional method.

Let us first consider convergence of HC in the noninterval case. We now show that if convergence occurs, its asymptotic rate is generally linear. We then introduce a modification of the procedure that generally converges asymptotically at a quadratic rate.

If we apply HC to the equation $f(x) = g(x) - h(x) = 0$, we solve for x' from the relation

$$g(x') = h(x). \tag{10.6.1}$$

In an iterative procedure, we repeat this step. Let x^* be a solution of $f(x) = g(x) - h(x) = 0$. Then

$$g(x^*) = h(x^*). \tag{10.6.2}$$

From (10.6.1) and (10.6.2),

$$g(x') - g(x^*) = h(x) - h(x^*). \tag{10.6.3}$$

Assume that g and h are continuously differentiable in some interval X containing x, x', and x^*. Using the mean value theorem, we can expand $g(x')$ and $h(x)$ and obtain:

$$(x' - x^*)g'(\xi) = (x - x^*)h'(\eta)$$

where $\xi \in X$ and $\eta \in X$. If $g'(\xi) \neq 0$ for all $\xi \in X$, then this equation shows that if we iterate use of (10.6.1), the asymptotic rate of convergence (if it occurs) is at least linear. That is, the error $x' - x^*$ in x' is a linear function of the error $x - x^*$ in x. However, it is generally not superlinear.

To get a higher rate of convergence, let us introduce a function $v(x)$ and write $f(x)$ as

$$f(x) = [g(x) + v(x)] - [h(x) + v(x)]$$

Instead of solving for x' using (10.6.1), we now use the iterative step

$$g(x') + v(x') = h(x) + v(x). \tag{10.6.4}$$

Using (10.6.3), we can rewrite (10.6.4) as

$$g(x') - g(x^*) + v(x') - v(x^*) = h(x) - h(x^*) + v(x) - v(x^*). \tag{10.6.5}$$

As before, assume x, x', and x^* are in an interval X. Also, assume g is continuously differentiable and that h and v are twice continuously differentiable. We expand (10.6.5) about x^*. We expand the left member to first order as a function of x' and the right member to second order as a function of x. We obtain

$$\begin{aligned}(x' - x^*)[g'(\xi) + v'(\xi)] = (x - x^*)[h'(x^*) + v'(x^*)]\\ + \frac{1}{2}(x - x^*)^2[h''(\eta) + v''(\eta)]\end{aligned}$$

where $\xi \in X$ and $\eta \in X$.

Suppose we choose v so that

$$h'(x^*) + v'(x^*) = 0. \tag{10.6.6}$$

Then

$$x' - x^* = \frac{h''(\eta) + v''(\eta)}{2[g'(\xi) + v'(\xi)]}(x - x^*)^2. \tag{10.6.7}$$

If $g'(\xi) + v'(\xi) \neq 0$ for all $\xi \in X$, then (10.6.7) shows that the procedure is quadratically convergent asymptotically as $x \to x^*$.

In practice, we do not know x^*, so we cannot use (10.6.6). Instead, we approximate (10.6.6) by replacing x^* by x. Thus, we choose v so that

$$v'(x) + h'(x) = 0. \tag{10.6.8}$$

To use the iterative step indicated by (10.6.4), we must be able to solve $g(x') + v(x')$ for x'. A simple choice for $v(x)$ that enables such a solution is $v(x) = \alpha g(x)$, where α is a constant. For this choice, we need only solve $(1 + \alpha)g(x')$ for x'. We can do so because we assume g is easily invertible. From (10.6.8), we have $\alpha = -\frac{h'(x)}{g'(x)}$ and the iterative step is

$$(1 + \alpha)g(x') = h(x) + \alpha g(x).$$

This step fails if $\alpha = -1$. This is asymptotically the case if $f'(x^*) = 0$, which implies that x^* is a multiple zero of f.

It is theoretically possible to choose v so that HC is quadratically convergent to a multiple zero of f. To achieve quadratic convergence we have to choose v so that (10.6.8) holds and also $h''(x) + v''(x) = 0$. The added condition makes it difficult to choose v; and therefore we do not consider this generalization.

Our choice of v must be such that we can invert $g(x) + v(x)$. Among the possible choices are $v(x) = \alpha[g(x)]^2$ or $v(x) = \alpha[g(x)]^{\frac{1}{2}}$. In either case, we need only solve a quadratic to obtain $g(x')$ and then solve $g(x')$ for x'.

As x approaches a solution x^*, the coefficient of $(x - x^*)^2$ in (10.6.7) approaches a value known as the asymptotic constant. We denote it by

$$C(x^*) = \frac{h''(x^*) + v''(x^*)}{2[g'(x^*) + v'(x^*)]}.$$

The asymptotic behavior of a quadratically convergent form of HC can depend strongly on how g and v are chosen. We can see this by considering $C(x^*)$.

For example, suppose we choose $v(x) = \alpha g(x)$ and use HC to find the zeros 10 and 0.001 of

$$\begin{aligned} f(x) &= (x-10)(x-1)(x-0.001)(x+1) \\ &= x^4 - 10.001x^3 - 0.99x^2 + 10.001x + 0.01. \end{aligned}$$

Let us first choose $g(x)$ so that it is large when $|x|$ is large. Let $g(x) = x^4$. Then $C(10) = 0.152$ and $C(0.001) = 1500$. That is, convergence to the large zero is much more rapid than to the small zero. Next, choose $g = 10.001x$ so that $g(x)$ dominates the other powers of x when $|x|$ is small. We find $C(10) = -0.303$ and $C(0.001) = 0.003$. Now convergence is more rapid to the smaller zero.

These differences in values of the asymptotic constant are qualitatively the same for this example if we use $v(x) = \alpha[g(x)]^2$ or $v(x) = \alpha[g(x)]^{\frac{1}{2}}$ instead of $v(x) = \alpha g(x)$. That is, if g is chosen so that HC is efficient in deleting values of x that are (say) large in magnitude, the procedure remains so no matter the form we use for v.

It can be argued that there is no need for a quadratically convergent form of HC because the interval Newton method has this property (when converging to a simple zero of f). For best performance for both small and large values of $|x|$, more than one form of HC must be used. However the Newton method requires use of a single form only.

The data needed for HC to exhibit quadratic convergence is essentially the same as for Newton's method. For Newton's method, we need to evaluate $f'(X)$ and we can compute it by separately computing $g'(X)$ and $h'(X)$. If we define $v(x) = \alpha g(x)$, then for HC, we want $\alpha = -\frac{h'(x^*)}{g'(x^*)}$. Knowing $g'(X)$ and $h'(X)$, we can approximate this value by $\alpha = -\frac{\mathrm{m}(h'(X))}{\mathrm{m}(g'(X))}$. Therefore, one can use both methods with very little extra computing.

It can be shown that the asymptotic constant for Newton's method is

$$C_{\mathrm{N}}(x^*) = \frac{f''(x^*)}{2f'(x^*)}.$$

For the above example, we find $C_\mathrm{N}(10) = -0.908$ and $C_\mathrm{N}(0.001) = 0.096$. Thus, the asymptotic performance of the Newton method can be better or worse than a particular form of HC.

10.7 CONVERGENCE IN THE INTERVAL CASE

Quadratic convergence is rather meaningless for an iterative step of a non-interval iterative procedure when the current point is far from a solution. Correspondingly, there is little or no virtue in considering the asymptotic behavior of an interval procedure when the box in which a solution is sought is large.

Therefore, there is no point in introducing the term $v(x)$ in the interval case unless the box is small. This is especially true since to determine v we must do the extra work of evaluating the derivatives of g and h.

Suppose we do want to improve convergence in the interval version of HC. Suppose we choose the function $v(x)$ occurring in (10.6.4) to have the form $v(x) = \alpha u(x)$. As noted in Section 10.6, some easy-to-implement choices for $u(x)$ are $g(x)$, $[g(x)]^2$ and $[g(x)]^{\frac{1}{2}}$. From (10.6.3), we want to choose α so that $\alpha u'(x^*) + h'(x^*) = 0$. In the interval case, we approximate the unknown x^* by $x_0 = \mathrm{m}(X)$ where X is the current interval. Thus, $\alpha = -\frac{h'(x_0)}{u'(x_0)}$.

The right member of (10.6.4) becomes $h(x) + \alpha u(x)$. Before evaluating this function with interval argument X, we rewrite its analytic form to reduce the effect of dependence. This might entail combining terms, cancelling terms, factoring, completing squares, etc.

10.8 SPLITTING

Suppose we are solving a system of nonlinear equations or an optimization problem by an interval method. We use HC as one of the procedures in the algorithm to solve such problems. When progress is slow or nonexistent, it is necessary to split the current box into subboxes. Suppose we implement HC so that it is most effective at eliminating values of variables that are large in magnitude. This influences how we split a given box.

We have noted that an interval Newton method is most effective when the width of the box is small. We can split so that both HC and Newton's method tend to be effective in one of the subboxes generated by splitting.

A natural way to split an interval such as $X = [-10, 10]$ is to split it into the two sub-intervals $[-10, 0]$ and $[0, 10]$. However, each new interval contains both small and (relatively) large values of x. A better way to split is to subdivide X into $[-10, -1]$, $[-1, 1]$, and $[1, 10]$. Assuming we have designed our HC method to perform best for large values of a variable, it should do well in the new intervals where $|x| \geq 1$. We expect the Newton method to perform better in the interval $[-1, 1]$ than in a wider interval such as $[0, 10]$. Therefore, if we split in this way, one method or the other is likely to perform well in each new interval.

This kind of splitting can have an added benefit that we illustrate by an example. Suppose we have a two-dimensional box specified by component intervals X and Y; and we wish to solve an equation of the form $f(x, y) = xy - h(x, y)$ for x as $X' = \frac{h(X,Y)}{Y}$. If $0 \in Y$, then X' is unbounded. Suppose we obtain the interval Y by splitting an interval $[-10, 10]$. If we split it into $[-10, 0]$ and $[0, 10]$, then, for Y equal to either of these subintervals, the interval X' is unbounded. But if we split $[-10, 10]$ into $[-10, -1]$, $[-1, 1]$, and $[1, 10]$, then for two of these cases, X' is bounded.

We use these considerations when discussing splitting in Section 11.8 and elsewhere. We now give the steps of a splitting procedure that is based on the above discussion. It can be used when solving either nonlinear equations or optimization problems.

For one dimensional problems, it is reasonable to split an interval into more than two parts. For multidimensional problems, we prefer to split more than one component interval rather than split a given component into more than two parts. This explains Step 2 in the procedure below.

Denote the interval to be split by $[a, b]$.

1. If $X \subseteq [-2, 2]$ and $\mathrm{m}(X) = 0$ (so that $a = -b$), split X into $\left[a, \frac{a}{1024}\right]$ and $\left[\frac{a}{1024}, b\right]$. (Note: The number 1024 is an arbitrarily chosen power of 2.)

2. If $X \nsubseteq [-2, 2]$, and $0 \in X$, split X as follows:

(a) If the problem being solved by the main program is multidimensional, split X at whichever of -1 and $+1$ is nearest the center of X. If $\mathrm{m}(X) = 0$, split at either $x = -1$ or $x = +1$.

(b) If the problem being solved is one dimensional, split X at -1 if $-1 \in X$ and also at $+1$ if $1 \in X$.

3. Otherwise, split X in half.

There is an obvious exception to this way of splitting. Suppose, $f(x, y) = xy - h(x, y)$ and we find that $h(X, Y) > 0$ so that $xy > 0$. Then x and y have the same sign. If (for example), both X and Y contain zero, we split both X and Y at 0 to delete the second and fourth quadrants where x and y have opposite signs. This method of splitting takes precedence over splitting at ± 1.

10.9 THE MULTIDIMENSIONAL CASE

In the multidimensional case, we apply HC to one equation of a system of equations and solve for one variable at a time. To do so, we replace all other variables by their interval bounds. Let a box X and an equation $f(x) = 0$ be given. As for BC (see Section 10.2), obtain

$$q(x_i) = f(X_1, \cdots, X_{i-1}, x_i, X_{i+1}, \cdots, X_n) = 0.$$

We can now solve this equation for the single variable x_i. This is the case we have been discussing. The difference is that now the equation involves interval constants.

A subset of the equations in a system of nonlinear equations often contains terms that are linear in some of the variables. In this case, we can use HC to solve for linear combinations of such variables and then solve the linear system. We can also solve for linear combinations of simple nonlinear functions.

In the multidimensional case, we can solve for a term involving more than one variable. We then have a two stage process. For example, suppose we solve for the term $\frac{1}{x+y}$ from the function

$$f(x, y) = \frac{1}{x+y} - h(x, y) = 0.$$

Let $x \in X = [1, 2]$ and $y \in Y = [0.5, 2]$. Suppose we find that $h(X, Y) = [0.5, 1]$. Then $\frac{1}{x+y} \in [0.5, 1]$ so $x + y \in [1, 2]$. Now we replace y by $Y = [0.5, 2]$ and obtain the bound $[-1, 1.5]$ on X. Intersecting this interval with the given bound $X = [1, 2]$ on x, we obtain the new bound $X' = [1, 1.5]$.

We can use X' to get a new bound on h; but this might require extensive computing if h is a complicated function; so suppose we do not. Suppose that we do, however, use this bound in our intermediate result $x+y = [1, 2]$. Solving for y as $[1, 2] - X'$, we obtain the bound $[-0.5, 1]$. Intersecting this interval with Y, we obtain the new bound $Y' = [0.5, 1]$ on y. Thus, we improve the bounds on both x and y by solving for a single term of f.

For a system of equations, we apply HC by cycling through the equations and variables as described at the end of Section 10.2. Suppose we have solved once for each variable from each equation. We can now repeat the process. In our optimization algorithms, we do so only if sufficient progress is made in a given cycle. Otherwise, we apply other procedures. If they also fail to make sufficient progress we split the box.

10.10 CHECKING FOR NONEXISTENCE

Consider a single equation $f(\mathbf{x}) = 0$ in which $\mathbf{x}$ is a vector. We often want to know if there exists a point or points in a box $\mathbf{x}^{\mathbf{I}}$ that satisfies the equation. It is common practice to check for nonexistence of such points by evaluating f over the box. If $\mathbf{0} \notin f(\mathbf{x}^{\mathbf{I}})$, then no such point exists in X. That is, f has no zero in $\mathbf{x}^{\mathbf{I}}$.

When this test fails to prove nonexistence, one usually seeks a point or points $\mathbf{x}$ in $\mathbf{x}^{\mathbf{I}}$ that satisfies $f(\mathbf{x}) = 0$ (and generally other equations as well). If we do seek such points, it is better to replace this nonexistence test by an application of HC. In so doing, we might be able to reduce $\mathbf{x}^{\mathbf{I}}$ when performing a nonexistence test that fails (to prove nonexistence).

To illustrate this fact, consider a one dimensional equation of the form $f(x) = x - h(x) = 0$. Suppose that for a given interval X, we obtain

$f(X) > 0$, which proves that there is no point $x \in X$ that satisfies $f(x) = 0$. We can express $f(X) > 0$ as $X > h(X)$. Suppose we apply HC in the form $X' = h(X)$. Then we have $X' < X$. Since any solution of $f(x) = 0$ in X must also be in X', we conclude (as for the nonexistence test) that there does not exist a solution in X. Note that this is a kind of converse of an existence proof of a solution using a fixed point theorem.

If $X' \cap X$ is not empty but is smaller than X, the latter procedure makes progress in isolating any zero of f in X. We make this progress with essentially the same amount of computing needed to evaluate $f(X)$ to perform the nonexistence test. Since HC yields the same or more information than the nonexistence test with the same amount of computing, we always use HC rather than a nonexistence test. We often use HC in this way in our optimization algorithms.

Little extra computing is needed if we use HC in the more general form in which $f(x) = g(x) - h(x)$ and we solve for $g(X')$. Only one small extra step is needed to solve $g(X')$ for X'. We have assumed that this is easy to do.

This same saving of effort can be used for inequalities. Suppose we have an inequality $f(x) \leq 0$. Instead of testing whether a box X is certainly infeasible by evaluating $f(X)$, we can solve $f(x) = [-\infty, 0]$ using HC and possibly eliminate some certainly infeasible points from X. We have occasion to use HC in this way for both equations and inequalities in various places in this book.

Consider the function

$$f(x, y) = xy - 10 = 0.$$

and assume $0 \in X$ and $0 \in Y$. To solve for x or y when using HC, we must divide by an interval containing zero. Thus, it might appear that evaluating $f(X, Y)$ is better than HC to check for nonexistence of a solution. This is not so.

Suppose $X = [-4, 6]$ and $Y = [-2, 2]$. If we evaluate $f(X, Y)$, we obtain $[-22, 2]$. Since $0 \in f(X, Y)$, we have failed to prove nonexistence. If we replace y by Y and solve for X, we obtain $X' = [-\infty, -5] \cup [5, +\infty]$. Since $X \cap X' = [5, 6]$, the equation $f(x, y) = 0$ has a solution with $y \in Y$ only if x is in this reduced interval.

It takes a little more effort to compute X' and $X \cap X'$ than to simply evaluate $f(X, Y)$. However, the extra effort reduces the interval bound on x from $[-4, 6]$ to $[5, 6]$. Even if we have to divide by an interval containing zero to apply HC, it is better to do so than to simply check for nonexistence by evaluating f.

Suppose we have applied HC to a function f over a box $\mathbf{x^I}$. If $\mathbf{x^I}$ is unchanged in doing so, we obtain the data to easily evaluate $f(\mathbf{x^I})$. If $\mathbf{x^I}$ is only slightly changed, we can get a good approximation for f evaluated over the reduced box.

This can be useful. In our optimization algorithms, we split a box when little or no progress is made in applying the algorithm to the box. This implies that we can get good approximations to the value over the box for any function to which we have applied HC. We can use such information to determine how best to split the box. We now show how to compute the approximation.

Suppose we apply HC to a function of the form $f(\mathbf{x}) = a(\mathbf{x})g(x_1) - h(\mathbf{x})$ where $\mathbf{x} = (x_1, \cdots, x_n)^T$ and we seek a new bound on x_1. In previous sections, we implicitly assume that a function of this form is rewritten by dividing through by $a(\mathbf{x})$ to isolate $g(x_1)$. When we apply HC over a box $\mathbf{x^I}$, we evaluate $a(\mathbf{x^I})$ and $h(\mathbf{x^I})$. We then determine $X_1' = g^{-1}\left(\frac{h(\mathbf{x^I})}{a(\mathbf{x^I})}\right)$ and $X_1'' = X_1 \cap X_1'$. Since g is chosen to be a simple function, we can easily evaluate $g(X_1'')$ and thus obtain the function $\widetilde{f}(\mathbf{x^I}, X_1'') = a(\mathbf{x^I})g(X_1'') - h(\mathbf{x^I})$.

Denote the new box by $\mathbf{x^{I''}} = (X_1'', X_2, \cdots, X_n)^T$. We want a value for $f(\mathbf{x^{I''}})$. Note that $a(\mathbf{x^I})$ and $h(\mathbf{x^I})$ are evaluated using X_1 rather than X_1''. Therefore, $f(\mathbf{x^{I''}}) \subset \widetilde{f}(\mathbf{x^I}, X_1'')$ because $X_1'' \subset X_1$. By assumption, either $\mathbf{x^{I''}} = \mathbf{x^I}$ or else $\mathbf{x^{I''}}$ differs very little from $\mathbf{x^I}$. Therefore, either $\widetilde{f}(\mathbf{x^I}, X_1'') = f(\mathbf{x^I})$ or else $\widetilde{f}(\mathbf{x^I}, X_1'')$ is a good approximation for $f(\mathbf{x^I})$.

Normally, we apply hull consistency to solve a given equation for each of the variables on which it depends. In practice, the step to bound $f(\mathbf{x^I})$ is done only after solving for the last of the variables.

10.11 LINEAR COMBINATIONS OF FUNCTIONS

Suppose we wish to find a solution of a system of nonlinear equations in a given box. We can apply HC to each equation of the system to try to eliminate parts of the box that cannot contain the solution. But, we can sometimes eliminate larger parts of the box if we apply HC to a linear combination of equations from the system.

The following system is trivial but illustrates the idea. Suppose we want to solve the system

$$x + y = 0,$$
$$x - y = 0$$

for $x \in X = [-1, 1]$ and $y \in Y = [-1, 1]$. Note that these functions represent the diagonals of the box. For either equation, if we choose any $x \in X$, there exists a $y \in Y$ that satisfies the equation. Therefore, HC cannot reduce the box. If we add the equations, we get $2x = 0$ and if we subtract them we get $2y = 0$. From these equations HC produces the solution $x = y = 0$.

By taking linear combinations, we have rotated (and stretched) the diagonals of the box so that they coincide with the coordinate axes. Now HC can eliminate half planes into which a given line does not enter.

Suppose we seek a solution to a general system of nonlinear equations in a box of small width. If the width is sufficiently small, the surfaces represented by the functions are closely approximated by their tangent planes. If the tangent planes at a given point are linearly independent, a linear combination of them can transform them to coincide with the coordinate axes. This enables HC to eliminate larger parts of the box than when using the original system.

Denote a system of nonlinear equations by the vector function $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots f_n(\mathbf{x}))^T$. In an interval Newton method (see Chapter 11), we use the expansion

$$\mathbf{f}(\mathbf{y}) = \mathbf{f}(\mathbf{x}) + \mathbf{J}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})(\mathbf{y} - \mathbf{x}) = 0 \tag{10.11.1}$$

where $\mathbf{J}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$ is the Jacobian of $\mathbf{f}$ evaluated over a box $\mathbf{x}^{\mathbf{I}}$ containing the points $\mathbf{x}$ and $\mathbf{y}$. See Section 7.4. If $\mathbf{x}$ is the center of $\mathbf{x}^{\mathbf{I}}$ and if $\mathbf{J}^c$ is the center of $\mathbf{J}(x, \mathbf{x}^{\mathbf{I}})$, then the equation $\mathbf{f}(\mathbf{x}) + \mathbf{J}^c(\mathbf{y} - \mathbf{x}) = \mathbf{0}$ approximates the tangent planes of the components of $\mathbf{f}$ at $\mathbf{x}$. Assume the matrix $\mathbf{J}^c$ is nonsingular and let $\mathbf{B}$ be an approximation for its inverse. Then the tangent plane of the i-th equation of $\mathbf{Bf}(\mathbf{x})$ approximates the i-th coordinate axis. In an interval Newton method, we precondition the system (10.11.1) using $\mathbf{B}$.

We can apply HC to the linear combination $\mathbf{Bf}(\mathbf{x})$ of nonlinear equations from the original system. We solve the i-th equation of $\mathbf{Bf}(\mathbf{x})$ for the i-th variable only. Before doing so, we analytically generate the function

$$[\mathbf{Bf}(\mathbf{x})]_i = \mathbf{B}_{i,1} f_1(\mathbf{x}) + \cdots + \mathbf{B}_{i,n} f_n(\mathbf{x}). \tag{10.11.2}$$

We write it in analytic form with terms collected and arranged to produce the sharpest interval values when evaluated with interval arguments. Afterward, we substitute numerical values for $\mathbf{B}_{i,j}$ $(i, j = 1, \cdots, n)$.

If the box is not small, the tangent planes can be poor approximations for the functions. In this case, this procedure might not be helpful. Moreover, a linear combination of functions is more complicated than the original component functions of $\mathbf{f}(\mathbf{x})$. Therefore, it is likely that dependence causes greater loss of sharpness when applying HC to the transformed functions. Therefore, this procedure for using linear combinations of functions is best used only when the box is small. It is for small boxes that we want to use an interval Newton method; and it is for the Newton method that we compute the matrix $\mathbf{B}$ needed for the HC procedure just described.

After a Newton step is applied, we apply HC and BC to the linear combination of nonlinear functions as described above. This involves more computation than application to the original system. If the box is small, it is for this step that the quadratically convergent form of HC (described in Section 10.6) is of value. This is because it is applied to an equation that depends strongly on only one variable and because the box is small so behavior of the procedure is approximately asymptotic.

10.12 PROVING EXISTENCE

It is possible to prove the existence of a solution of a system of nonlinear equations in a box X using HC. Let $\mathbf{x}$, $\mathbf{f}$, $\mathbf{g}$, and $\mathbf{h}$ be vectors of n components and let the components of $\mathbf{f}$ be of the form $f_i(\mathbf{x}) = g_i(\mathbf{x}) - h_i(\mathbf{x})$ $(i = 1, \cdots, n)$. Suppose we compute a new box $\mathbf{x}^{\mathbf{I}\prime}$ as $X_i' = g_i^{-1}[h_i(\mathbf{x}^\mathbf{I})]$ $(i = 1, \cdots, n)$. This is equivalent to applying HC to the equation $x_i - g_i^{-1}[h_i(\mathbf{x})]$ $(i = 1, \cdots, n)$. Therefore, we might as well assume that the original equation has the form $\mathbf{f}(x) = \mathbf{x} - \mathbf{h}(\mathbf{x}) = \mathbf{0}$. For simplicity, we do so. We now apply HC in the form

$$\mathbf{x}^{\mathbf{I}\prime} = h(\mathbf{x}^\mathbf{I}).$$

Let $\mathbf{h}$ be a continuous function of $\mathbf{x}$ for $\mathbf{x} \in \mathbf{x}^\mathbf{I}$ and let $\mathbf{h}^\mathbf{I}(\mathbf{x}^\mathbf{I})$ be a continuous, containment-set enclosure of $\mathbf{h}(\mathbf{x})$ for $\mathbf{x} \in \mathbf{x}^\mathbf{I}$.

Theorem 10.12.1 If $\mathbf{h}^\mathbf{I}(\mathbf{x}^\mathbf{I}) \subset \mathbf{x}^\mathbf{I}$, then there exists a solution of $f(\mathbf{x}) = \mathbf{x} - \mathbf{h}(\mathbf{x}) = \mathbf{0}$ in $\mathbf{x}^\mathbf{I}$.

Proof. Since $\mathbf{h}(\mathbf{x}) \in \mathbf{h}^\mathbf{I}(\mathbf{x}^\mathbf{I})$ for all $\mathbf{x} \in \mathbf{x}^\mathbf{I}$, the function $\mathbf{h}(\mathbf{x})$ maps the convex, compact set $\mathbf{x}^\mathbf{I}$ into itself. Therefore, the Brouwer fixed point theorem (see Theorem 5.3.13 of Neumaier (1990)) assures that this function has a fixed point $\mathbf{x}^*$ in the interior of $\mathbf{x}^\mathbf{I}$. That is, $\mathbf{x}^* = \mathbf{h}(\mathbf{x}^*)$ and hence $\mathbf{f}(\mathbf{x}^*) = \mathbf{0}$. ■

To apply this theorem, we evaluate each component of $\mathbf{h}$ over the same box $\mathbf{x}^\mathbf{I}$. In practice, we use a reduced component of $\mathbf{x}^\mathbf{I}$ as soon as it is computed. We can prove existence using this more efficient form.

We illustrate the procedure for a system of two equations of two variables. Assume we are able to write the equations in the form

$$f_1(x_1, x_2) = x_1 - h_1(x_1, x_2) = 0,$$
$$f_2(x_1, x_2) = x_2 - h_2(x_1, x_2) = 0.$$

We apply HC to the first equation in the form

$$X_1' = H_1(X_1, X_2).$$

Assume that $X_1' \subset X_1$. We next apply HC to the second equation in the form

$$X_2' = H_2(X_1', X_2).$$

Assume that $X_2' \subset X_2$.

Since $X_1' \subset X_1$, we conclude from Theorem 9.6.8:

Conclusion 10.12.2 For each $x_2 \in X_2$, there exists $x_1^* \in X_1$ such that $f_1(x_1^*, x_2) = 0$. Since $X_2' \subset X_2$, we conclude:

Conclusion 10.12.3 For each $x_1 \in X_1'$, there exists $x_2^* \in X_2$ such that $f_2(x_1, x_2^*) = 0$.

Since Conclusion 10.12.2 is true for each $x_2 \in X_2$ and since $x_2^* \in X_2$, it is true for x_2^*. That is, $f_1(x_1^*, x_2^*) = 0$. Since Conclusion 10.12.3 is true for each $x_1 \in X_1'$ and since $x_1^* \in X_1'$, it is true for x_1^*. That is, $f_2(x_1^*, x_2^*) = 0$. Therefore, the point (x_1^*, x_2^*) is a solution of the system. Thus, we have proved the existence of a solution in a subbox of the original box.

We have shown that it is possible to prove existence of a solution of a system of equations by applying HC to one equation at a time. This same method of proof of existence can be used when any method is applied to one equation at a time provided the method can verify existence in the one dimensional case. In particular, the one dimensional interval Newton method is such a method. See Theorem 9.6.9. We use this fact in Chapter 15.

10.13 COMPARING BOX AND HULL CONSISTENCIES

Box consistency and hull consistency differ in performance and capabilities. To apply BC to a function, the function must be continuously differentiable so that the Newton method is applicable. To apply HC, the function need not even be continuous.

HC is much faster than BC in achieving its result. Therefore, we emphasize its use and let BC play a subordinate role. However, a result from HC might not be as narrow as a result from BC. This is generally the case when HC solves for a single term. However, when HC solves for more than one term (such as a quadratic expression), the result can be sharper than would be obtained using BC. We give an example in Section 12.7. Each method has virtues and drawbacks. We use the virtues of both methods in our algorithms.

To illustrate the difference in performance of HC and BC, consider the function

$$f(x, y) = x^3 + 100x + 10y = 0.$$

Assume we wish to bound x when $x \in X = [-100, 100]$ and $Y = [-100, 100]$. Replacing y by Y, we obtain

$$x^3 + 100x + [-1000, 1000] = 0.$$

If we apply HC by solving for the term x^3, we obtain

$$x^3 \in -100X - [-1000, 1000] = [-11000, 11000]$$

and hence $x \in [-22.24, 22.24]$. If we iterate this step, the limiting interval bound on x is approximately $[-13.25, 13.25]$.

Suppose we use BC by applying one Newton step to increase the lower bound and one Newton step to decrease the upper bound. We obtain $[-66.42, 66.42]$ approximately. Thus, we perform more work than a step of HC and obtain less sharp bounds. To get bounds as good as that from one step of HC, we must apply ten Newton steps when using BC. However, if we iterate BC, the limiting interval bound is approximately $[-6.824, 6.824]$, which is narrower than the best possible HC result.

BC can usually produce bounds that are at least as narrow as those from HC. However, this requires more computing effort. In fact, it might be true only in the limit after an infinite number of BC steps.

Consider a function of the form $x_1^m - \mathbf{h}(x_2, \cdots, x_n) = 0$ where $m \geq 2$ and where x_1 does not occur in $\mathbf{h}$. To solve for x_1 using either HC or BC, we replace $x_2, \cdots, x_n$ by their interval bounds and solve for x_1. Using HC, we get the best possible result in one step. Using BC, we must iterate and generally stop with a result that is not as good as that produced by HC.

In the various algorithms in this book, we often follow a HC step with a BC step. The BC step is skipped in certain cases. Recall that to apply HC, we solve for a given variable x_i using $\mathbf{g}(x_i) = \mathbf{h}(\mathbf{x})$. If $\mathbf{h}(\mathbf{x})$ is independent of x_i, then BC cannot improve the interval bound on x_i obtained using HC. Therefore we do not use BC.

Consider a function of a single variable x whose value is dominated by a term x^m when x is large. Assume the width of the initial bound X on x is relatively large. Then a Newton step applied to X tends to reduce the width of X by a factor $1 - \frac{1}{m}$. Thus, under these circumstances, BC can be rather slow.

The speed of HC for such an example depends on the subdominant terms. It can be slow or fast or it might not make any progress at all.

Although BC might be slow in some cases, it is still of considerable value. Consider a problem in which HC makes little progress. Let us compare BC in this case with the alternative of using a multidimensional Newton method (see Section 11.2).

Suppose we are solving a system of nonlinear equations over a large box $\mathbf{x}^{\mathbf{I}}$. If the Newton method fails to make progress, we have wasted a great deal of effort. If BC fails, much less effort is wasted. To make progress for either method, we split one or more components of X. Generally, less splitting is needed for BC to make progress. This is because BC is a one dimensional procedure; and splitting need be done only in the one dimension. To improve the performance of a multidimensional Newton method, it is generally necessary to split a box in more than one dimension. Less splitting can result in considerable saving of computing effort.

A virtue of consistency methods is that they reduce the region of search for solutions to a given problem. The region might contain one or many solutions (or none). When a region has been reduced so that it contains only one solution, we generally rely on a Newton method to provide better performance. See Chapters 9 and 11. When solving for zeros of functions in one dimension, we precede the Newton procedure by an application of HC.

However, BC is omitted. It is designed merely to reduce the region of search. The Newton procedure of Section 9.5 is designed to separate isolated solutions and provide rapid convergence to each of them. If HC is appropriately implemented, it can also separate isolated solutions.

10.14 SHARPENING RANGE BOUNDS

Suppose we want to bound the range of a function $f(x)$ over an interval X. From Theorem 3.2.2, we can obtain bounds by simply evaluating the function over X using interval arithmetic. However, as noted in Section 2.4, dependence generally precludes the bounds from being sharp. In this section, we show how consistency methods can be used to sharpen such bounds.

We discuss the procedure for the case in which HC is used. However, BC can also be used. We assume the function is thin. That is, it contains no nondegenerate interval parameters. However, this need not be the case.

Denote the exact range of f over X by $f(X) = [\underline{f}(X), \overline{f}(X)]$. Suppose we evaluate $f(X)$ using interval arithmetic and, because of rounding and dependence obtain a nonsharp interval $[\underline{F}(X), \overline{F}(X)]$ bounding $f(X)$. Suppose we also evaluate f at two or more points of X. For example, we might evaluate f at the endpoints of X. Denote the smallest sampled value by f_S and the largest by f_L. Then

$$\underline{F}(X) \leq \underline{f}(X) \leq f_S \leq f_L \leq \overline{f}(X) \leq \overline{F}(X). \tag{10.14.1}$$

Inequalities of this sort are used by Jaulin, *et al* (2001) to define versions of BC.

Suppose we use HC to delete points of X where $f > f_S$. Denote the resulting interval by X'. Note that any point of X at which $f = \underline{f}(X)$ will be retained in X'. Suppose we now evaluate f over X' and obtain $[\underline{F}(X'), \overline{F}(X')]$. Then $\underline{F}(X')$ is a lower bound on $\underline{f}(X)$. This new bound will generally by sharper than $\underline{F}(X)$ since X' is generally a narrower interval than X.

A similar procedure can be used to sharpen the upper bound $\overline{F}(X)$.

As an example, consider the function

$$f(x) = x^3 - 4x^2 + 15x.$$

If we evaluate f over the interval $X = [2, 6]$ using nesting, we obtain

$$\underline{F}(X) = 6 \text{ and } \overline{F}(X) = 162.$$

Evaluating f at the endpoints of X, we obtain

$$f_S = f(2) = 22 \text{ and } f_L = f(6) = 162.$$

To apply HC to the inequality $f(x) \leq 22$, we define $g(x) = x^3$ and we write $h(x) = 4x^2 - 15x$ in the form

$$h(x) = 4(x - 1.875)^2 - 14.0625$$

to reduce dependence when evaluating h. We thus solve for X' from

$$(X')^3 - 4(X - 1.875)^2 + 14.0625 = [-\infty, 22].$$

After intersecting the result with X, we obtain $X' = [2, 4.236]$.

This interval must contain the minimum of f. Evaluating $f(X')$ using nesting, we obtain $\underline{F}(X') = 13.0567$. Thus, we have increased the lower bound on $\underline{f}(X)$ from 6 to 13.0567. The function f is monotonic in X and the exact lower bound is 22.

Using a similar procedure to obtain a subinterval X'' of X containing $\overline{f}(X)$, we obtain $\overline{F}(X'') = 162$. Since this value is actually taken on at the sampled value of f at $x = 6$, we have proved that the exact value of $\overline{f}(X)$ is 162. That is, our upper bound for $f(X)$ is sharp.

A similar procedure can be used to improve bounds on the range of a multivariable function.

10.15 USING DISCRIMINANTS

The equations to which we apply HC in practice often contain a given variable both linearly and quadratically. In this case, the function g used to define a step of HC can be a sum of these terms. To invert g we must solve a quadratic equation. In this section, we note that the discriminant of the quadratic equation can play a significant role.

The ancient formula

$$r^{\pm} = \frac{-B \pm \left(B^2 - 4AC\right)^{\frac{1}{2}}}{2A} \tag{10.15.1}$$

expresses the roots of a quadratic equation

$Ax^2 + Bx + C = 0$

in terms of the square root of the discriminant

$D = B^2 - 4AC.$

We noted in Section 8.1 that generally one should not use the explicit expression (8.1.2) to find roots of an interval quadratic equation. Instead, the method of Section 8.2 should be used. This procedure reduces loss of sharpness in computed roots resulting from dependence.

Nevertheless, (10.15.1) is useful because of the implied condition

$$B^2 - 4AC \geq 0, \tag{10.15.2}$$

which is necessary for the roots to be real. This condition can be used in various ways when solving problems in which an (appropriate) equation must be satisfied. The inequality in (10.15.2) is an example of a domain constraint discussed in Chapter 4 on page 71.

We now consider some illustrative examples. First, consider the equation

$$x^2y^2 - 20xy + x^2 + y^2 + 10 = 0. \tag{10.15.3}$$

Suppose we wish to find all solutions of this equation in the box given by $X = Y = [-100, 100]$. We can apply HC by regarding this equation as a quadratic in the product xy. Solving the quadratic by the method of Section 8.2 does not reduce the box. Similarly, if we regard the equation as a quadratic in x (or in y) we do not reduce the box.

However, regarding the equation as a quadratic in xy, the discriminant is

$D_1 = 360 - 4(x^2 + y^2).$

If we apply HC to the condition $D_1 \geq 0$, we obtain a new box given by $X' = Y' = [-9.49, 9.49]$, approximately.

We get an even better result if we regard (10.15.3) as a quadratic in y. The discriminant is

$$D_2 = -4x^4 + 356x^2 - 40.$$

The condition $D_2 \geq 0$ yields $X' = \pm[0.335, 9.43]$, approximately. The same result can be obtained for y.

In this example, we are unable to reduce the box simply by solving the quadratic equation. However, reduction is obtained using the condition that a discriminant be nonnegative.

The difficulty that prevents us from reducing the box by solving a quadratic is that the coefficients of the quadratic are dependent. It is possible to overcome this difficulty by a more sophisticated approach in which we find the extrema of the roots when they are expressed analytically using (10.15.1).

As another example, assume the equation

$$x^4 y - x^2 y^2 - 4x - 20 = 0 \tag{10.15.4}$$

must hold in some box. We can obviously regard this equation as a quadratic in y. Also, we can think of x in the linear term as a separate variable and regard the equation as a quadratic in x^2. Also, we can think of x^4 in the leading term as a separate variable and regard the equation as a quadratic in x. The relevant discriminants for these cases are

$$D_1 = x^8 - 4x^2(4x + 20) \geq 0,$$
$$D_2 = y^4 + 4y(4x + 20) \geq 0,$$
$$D_3 = 16 + 4y^2(x^4 y - 20) \geq 0.$$

Suppose (10.15.3) must be satisfied in a box given by $X = Y = [-5, 5]$. If we apply HC directly to (10.15.4), we are unable to reduce the box. However, applying HC to $D_1 \geq 0$, we can delete the gap $(-1.91, 2.20)$ from X. Applying HC to $D_3 \geq 0$, we can reduce the interval Y to $[-0.448, 5]$. Thus, use of the discriminant relations is fruitful.

10.16 NONLINEAR EQUATIONS OF ONE VARIABLE

In Section 9.5, we give the steps of an interval Newton method for solving nonlinear equations in one variable. We now give the steps of that algorithm after incorporating hull consistency. This makes the new steps somewhat more complicated than the original ones in Section 9.5. However, they are more efficient, especially when the initial interval is wide.

At any stage of the algorithm, the current interval is denoted by X even though it changes from step to step.

1. Put the initial interval X_0 into a list L of intervals to be processed.
2. If the list L is empty, stop. Otherwise, select the interval X from L that has been in L for the shortest time. For later reference (in Step 10), record this interval X by the name $X^{(1)}$.
3. Using the interval X, apply hull consistency to the equation $f(x) = 0$. If the result is empty, go to Step 2.
4. If $0 \in f'(X)$, go to Step 6.
5. Apply the procedure described in Section 9.3, which either bounds a solution in X or proves that there is no solution in X. If the result is empty, go to Step 2. Otherwise record the solution interval that the procedure produces and go to Step 2.
6. If $0 \in f^I(x)$,go to Step 8.
7. If $\frac{w(X)}{|X|} < \varepsilon_X$ and $w(f(X)) < \varepsilon_f$, record X as a final bound and go to Step 2. Otherwise, go to Step 9.
8. Use the procedure listed in Section 9.4. If that procedure prescribes a point of expansion, record it; and go to Step 9. If it decides that the interval X should be accepted as a solution, record X and go to Step 2. If it prescribes that the interval is to be split, do so using the procedure at the end of Section 10.8. Put the subintervals generated by splitting into the list L and go to Step 2.

9. Apply a Newton step as given by (9.2.2) using the interval X. If a point of expansion was prescribed in Step 8, use it in determining the expansion defining the Newton method. If the result is a single interval, go to Step 10. If the result is two intervals, put them in list L and go to Step 2.

10. If the width of X is less than half that of $X^{(1)}$ (defined in Step 3), go to Step 3.

11. Split X using the procedure given in Section 10.8. Put the resulting subintervals into list L and go to Step 2.

Chapter 11

SYSTEMS OF NONLINEAR EQUATIONS

11.1 INTRODUCTION

Let $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^n$ be a continuously differentiable function. In this chapter, we consider the problem of finding and bounding all the solution vectors of $\mathbf{f} = \mathbf{0}$ in a given box $\mathbf{x^{I}}^{(0)}$. For noninterval methods, it can sometimes be difficult to find one solution, quite difficult to find all solutions, and generally impossible to know whether all solutions have been found. In contrast, it is a straightforward problem to find all solutions in $\mathbf{x^{I}}^{(0)}$ using interval methods; and it is trivially easy to computationally determine that all solutions in $\mathbf{x^{I}}^{(0)}$ have been found. We describe such interval methods in this chapter.

If a given problem has a large number of isolated solutions, it, of course, requires a large amount of time for any method to compute them all.

The nature of interval methods is such that it appears that they must always converge globally. However, there is no proof as yet. In practice, they "fail" only in taking too much computing time. It has been proved that convergence is global for the one-dimensional case. See Theorem 9.6.2.

Watson (1986) states that "the raison d'être for homotopy methods is global convergence". Therefore, the (presumed) global convergence of interval methods is sufficient motive for their use. The reason interval methods were originally introduced, however, was to provide (as they do) guaranteed bounds on the set of all solution(s).

Interval Newton methods can be said to have a raison d'être for any of the three reasons given: They find all solutions, they (apparently) converge globally, and the computed bounds are guaranteed to be correct. We see below (especially in Section 11.15) that they have other valuable properties as well. For example, they can (despite rounding errors) prove the existence (or nonexistence) and uniqueness of a solution.

11.2 DERIVATION OF INTERVAL NEWTON METHODS

Let $\mathbf{x}$ and $\mathbf{y}$ be points in a box $\mathbf{x^I}$. Suppose we expandeach component f_i $(i = 1, \cdots, n)$ of $\mathbf{f}$ by one of the procedures given in Chapter 7. Combining the results in vector form, we have

$$\mathbf{f}\,(\mathbf{y}) \in \mathbf{f}\,(\mathbf{x}) + \mathbf{J}\left(\mathbf{x}, \mathbf{x^I}\right)(\mathbf{y} - \mathbf{x}). \tag{11.2.1}$$

We refer to $\mathbf{J}$ as the Jacobian of $\mathbf{f}$ although it need not be formed using differentiation. A more efficient method is obtained if it is formed using slopes. See Section 7.7.

If $\mathbf{y}$ is a zero of $\mathbf{f}$, then $\mathbf{f}(\mathbf{y}) = \mathbf{0}$ and we replace (11.2.1) by

$$\mathbf{f}\,(\mathbf{x}) + \mathbf{J}(\mathbf{x}, \mathbf{x^I})(\mathbf{y} - \mathbf{x}) = \mathbf{0}. \tag{11.2.2}$$

Let $\mathbf{x}$ be fixed. Define the solution set of (11.2.2) to be

$$\mathbf{s} = \left\{ \mathbf{y} \;\middle|\; \begin{array}{l} \mathbf{f}\,(\mathbf{x}) + \mathbf{J}(\mathbf{x}, \mathbf{x}')(\mathbf{y} - \mathbf{x}) = \mathbf{0} \\ \mathbf{x}' \in \mathbf{x^I}. \end{array} \right\}$$

This set contains any point $\mathbf{y} \in \mathbf{x^I}$ for which $\mathbf{f}(\mathbf{y}) = \mathbf{0}$. We could use the notation $\{\mathbf{s}\}$ to emphasize the solution set is not a point, but we choose not to do so.

The smaller the box $\mathbf{x^I}$, the smaller the set $\mathbf{s}$. The object of an interval Newton method is to reduce $\mathbf{x^I}$ until $\mathbf{s}$ is as small as desired so that a solution point $\mathbf{y} \in \mathbf{x^I}$ is tightly bounded. Note that $\mathbf{s}$ is generally not a box. (See Section 5.3).

Suppose we solve the linear equation (11.2.2) using a method such as described in Chapter 5. Let $\mathbf{y}^{\mathbf{I}}$ denote the resulting box. Then $\mathbf{y}^{\mathbf{I}}$ contains the solution set $\mathbf{s}$.

For convenience, we write the suggestive relation

$$\mathbf{f}(\mathbf{x}) + \mathbf{J}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})(\mathbf{y}^{\mathbf{I}} - \mathbf{x}) = \mathbf{0}. \tag{11.2.3}$$

We discussed in detail in Section 5.2 why this relation is only suggestive. It is because even when $\mathbf{y}^{\mathbf{I}}$ is a solution of (11.2.3), if we compute the left member of (11.2.3), the result is generally not the zero vector. The exception is when $\mathbf{x}$ is a zero of $\mathbf{f}$ and $\mathbf{y}^{\mathbf{I}} = \mathbf{x}$.

For future reference, it is desirable to have a distinctive notation for the solution of (11.2.3). In place of $\mathbf{y}^{\mathbf{I}}$, we use the notation $\mathrm{N}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$, which emphasizes the dependence on both $\mathbf{x}$ and $\mathbf{x}^{\mathbf{I}}$.

From (11.2.3), we define an iterative algorithm of the form

$$\mathbf{f}(\mathbf{x}^{(k)}) + \mathbf{J}\left(\mathbf{x}^{(k)}, \mathbf{x}^{\mathbf{I}(k)}\right)\left[\mathrm{N}\left(\mathbf{x}^{(k)}, \mathbf{x}^{\mathbf{I}(k)}\right) - \mathbf{x}^{(k)}\right] = \mathbf{0}, \tag{11.2.4a}$$

$$\mathbf{x}^{\mathbf{I}(k+1)} = \mathbf{x}^{\mathbf{I}(k)} \cap \mathrm{N}(\mathbf{x}^{(k)}, \mathbf{x}^{\mathbf{I}(k)}) \tag{11.2.4b}$$

for $k = 0, 1, 2, \cdots$ where $\mathbf{x}^{(k)}$ must be in $\mathbf{x}^{\mathbf{I}(k)}$. A good choice for $\mathbf{x}^{(k)}$ is the center $\mathrm{m}(\mathbf{x}^{\mathbf{I}(k)})$ of $\mathbf{x}^{\mathbf{I}(k)}$. However, in Section 11.4, we consider how to compute a better choice.

See Alefeld (1999), for example, for a discussion of a method in which the linear system (11.2.4a) is solved by an interval form of Gaussian elimination.

In some procedures, the components of $\mathrm{N}(\mathbf{x}^{(k)}, \mathbf{x}^{\mathbf{I}(k)})$ are computed sequentially. The intersection in (11.2.4b) is computed as soon as a new component is obtained so that components computed later are narrower intervals.

In what follows, we sometimes omit the superscripts from terms in (11.2.4).

In Section 5.6, we noted that to obtain satisfactory results when computing the solution of a system of linear equations, it is generally necessary to precondition the system. To do so in the present case, we multiply (11.2.3) by an approximate inverse $\mathbf{B}$ of the center $\mathbf{J}^c$ of $\mathbf{J}(\mathbf{x}, \mathbf{x}^\mathbf{I})$. We discuss another way to precondition in Section 11.9.

If the matrix $\mathbf{J}^c$ is singular, we can (implicitly) modify it slightly so that the modified form is nonsingular. We describe a way to do so in Section 5.11.

Other alternatives when $\mathbf{J}^c$ is singular are either to begin again with a new point $\mathbf{x}$ of expansion, or to split the box $\mathbf{x}^\mathbf{I}$ into subboxes and apply the method to each subbox separately. In either case, we hope the new $\mathbf{J}^c$ is nonsingular.

Denote

$$\mathbf{M}(\mathbf{x}, \mathbf{x}^\mathbf{I}) = \mathbf{B}\mathbf{J}(\mathbf{x}, \mathbf{x}^\mathbf{I}) \text{ and } \mathbf{r}^\mathbf{I}(\mathbf{x}) = -\mathbf{B}\mathbf{f}^\mathbf{I}(\mathbf{x})\,. \tag{11.2.5}$$

In place of (11.2.4), we can write (temporarily reintroducing superscripts)

$$\mathbf{M}(\mathbf{x}^{(k)}, \mathbf{x}^{\mathbf{I}(k)})\left[\mathrm{N}(\mathbf{x}^{(k)}, \mathbf{x}^{\mathbf{I}(k)}) - \mathbf{x}^{(k)}\right] = \mathbf{r}^\mathbf{I}\left(\mathbf{x}^{(k)}\right) \tag{11.2.6a}$$

$$\mathbf{x}^{\mathbf{I}(k+1)} = \mathbf{x}^{\mathbf{I}(k)} \cap \mathrm{N}(\mathbf{x}^{(k)}, \mathbf{x}^{\mathbf{I}(k)}) \tag{11.2.6b}$$

$(k = 0, 1, 2, \cdots)$

As before for (11.2.4b), the intersecting in (11.2.6b) is done for a given component as soon as it is computed.

We frequently make reference to the interval Newton method in which we solve (11.2.6a) using a step of the Gauss-Seidel method described in Section 5.7. We write the iteration in succinct form by dropping the superscript k and letting $\mathbf{x}'$ and $\mathbf{x}^{\mathbf{I}\prime}$ denote $\mathbf{x}^{(k+1)}$ and $\mathbf{x}^{\mathbf{I}(k+1)}$, respectively. We also replace $\mathbf{M}(\mathbf{x}, \mathbf{x}^\mathbf{I})$ by $\mathbf{M}^\mathbf{I}$. The iteration for the i-th element of $\mathrm{N}(\mathbf{x}^{(k)}, \mathbf{x}^{\mathbf{I}(k)})$ is simply denoted by

$$N_i = x_i + \frac{1}{\mathbf{M}^\mathbf{I}{}_{ii}}\left[R_i - \sum_{j=1}^{i-1}\mathbf{M}^\mathbf{I}{}_{ij}(X'_j - x_j) - \sum_{j=i+1}^{n}\mathbf{M}^\mathbf{I}{}_{ij}(X_j - x_j)\right],$$
$$X'_i = N_i \cap X_i \tag{11.2.7}$$

Note that computing N_i can be regarded as an application of hull consistency to the i-th equation of the preconditioned system.

In practice, we do not complete the application of a step of the Gauss-Seidel method if all the diagonal elements of $\mathbf{M^I}$ contain zero. Note that these elements occur in the denominator or (11.2.7). If all the diagonal elements of $\mathbf{M^I}$ contain zero, it is likely that in (11.2.7) the quantity in square brackets contains zero for every value of $i = 1, \cdots, n$. In this case, $N_i = [-\infty, \infty]$ for all $i = 1, \cdots, n$ and thus $\mathbf{x^I}' = \mathbf{x^I}$. That is, no progress is made in applying a step of the Gauss-Seidel method.

If at least one diagonal element of $\mathbf{M^I}$ does not contain zero, we apply the Gauss-Seidel method. When doing so, before other values of i, we solve for N_i for those values of i for which $0 \notin M_{ii}$. This is because X_i might be reduced for the former (but generally not for the latter) values of i. The latter might produce an exterior interval (that is, the interval $[-\infty, \infty]$ with a gap removed). We ignore such gaps during the Gauss-Seidel step unless their intersection with X_i is empty or a single finite interval. However, information about such gaps is saved for later use if a box is to be split. See Section 11.8.

We sometimes refer to the Krawczyk method; but we do not use it because the Gauss-Seidel method is more efficient. To describe it we define the matrix $\mathbf{P^I} = \mathbf{I} - \mathbf{M^I}$. The iteration is

$$\begin{aligned} K_i &= x_i + R_i + \sum_{j=1}^{i-1} \mathbf{P^I}_{ij}(X'_j - x_j) - \sum_{j=i}^{n} \mathbf{P^I}_{ij}(X_j - x_j), \\ X'_i &= K_i \cap X_i. \end{aligned} \tag{11.2.8}$$

This is actually an improved version of the Krawczyk method due to Hansen and Sengupta (1981). In the original version, intersecting is done only after all new components are computed.

11.3 VARIATIONS OF THE METHOD

The ways in which $\mathrm{N}(\mathbf{x}^{(k)}, \mathbf{x^I}^{(k)})$ is computed from (11.2.4a) or (11.2.6a) and the ways in which $\mathbf{J}$ is defined distinguishes the various interval Newton methods. In this section, we concentrate on variations of the former kind.

Some variations attempt to bound the solution set $\mathbf{s}$ as sharply as possible each time (11.2.6a) is solved. Others do not. We distinguish between methods by defining the following types:

Type I: Equation (11.2.6a) is solved as sharply as possible in each iteration.

Type II: Equation (11.2.6a) is solved to obtain bounds on its solution set; but the bounds are not sharp, in general.

Type III: A method of type I or a method of type II (or both) is used in each iteration depending on criteria designed to enhance overall efficiency.

A type I method generally uses fewer iterative steps to obtain a bound of a given tolerance on a solution point. An example of a type I method is one in which the hull method of Section 5.8 is used to solve (11.2.6a).

A type II method generally uses more steps of a simpler nature. Therefore, the work per iteration is less. Examples of type II are those using Krawczyk's method (11.2.8) or the Gauss-Seidel method (11.2.7) to solve (11.2.6a).

We discuss a type III method in Section 11.12.

When we evaluate the real vector function $\mathbf{f}(\mathbf{x})$ at the point $\mathbf{x}$, we use outwardly rounded interval arithmetic and denote this fact by writing $\mathbf{f}^{\mathbf{I}}(\mathbf{x})$. Assume that the resulting box is of "reasonably small" width. That is, assume that rounding and dependence do not produce pathologically large bounds. Assume, also, that $\mathbf{f}$ does not depend on any parameter that enters as a wide interval. Then any interval Newton method of any type produces bounds on $\mathbf{s}$ that become tight as the width of $\mathbf{x}^{\mathbf{I}}$ becomes small.

Note that the solution set $\mathbf{s}$ of (11.2.3) becomes smaller as the current box $\mathbf{x}^{\mathbf{I}}$ becomes smaller, and the box to which the interval Newton method is applied gets progressively smaller as the algorithm progresses. In the early stages in solving a system, there is little point in bounding $\mathbf{s}$ sharply.

The first interval Newton method was due to Moore (1966). It is of type I; but a single step does not achieve as much progress as later variants. Because we wish to refer to it later, we now describe Moore's original algorithm. It uses (11.2.4) rather than (11.2.6).

Let $\mathbf{x} = \mathrm{m}(\mathbf{x^I})$ and assume $\mathbf{J}(\mathbf{x}, \mathbf{x^I})$ in (11.2.4a) does not contain a singular matrix. Let $\mathbf{V}(\mathbf{x}, \mathbf{x^I})$ be an interval matrix containing the inverse of every matrix contained in $\mathbf{J}(\mathbf{x}, \mathbf{x^I})$. Then the solution of (11.2.3) is contained in the vector $\mathrm{N}_M(\mathbf{x}, \mathbf{x^I})$ where

$$\mathrm{N}_M(\mathbf{x}, \mathbf{x^I}) = \mathbf{x} - \mathbf{V}(\mathbf{x}, \mathbf{x^I})\mathbf{f^I}(\mathbf{x}) \tag{11.3.1}$$

The first procedure, which does not sharply bound $\mathbf{s}$, was introduced (independently) by both Kahan (1968) and Krawczyk (1969). It is given by (11.2.8). It is commonly called the Krawczyk method and has been studied thoroughly. For example, see Alefeld (1999), Moore (1979) and Neumaier (1990).

In Moore's method the interval matrix $\mathbf{V}(\mathbf{x}, \mathbf{x^I})$ in (11.3.1) is computed using interval Gaussian elimination. This can fail because of division by an interval containing zero. At the time of inception, the Krawczyk method was an important development because it avoided applying Gaussian elimination to an interval matrix. In fact, the Krawczyk method avoids solving any set of interval linear equations. Instead, only a real (i.e., noninterval) matrix inverse is computed. This was the motivating factor for its introduction. For a recent discussion of the Krawczyk method, see Alefeld (1999).

A minor weakness of the Krawczyk and Gauss-Seidel methods is that even if $\mathbf{x}^*$ is a zero of $\mathbf{f}$ so that $\mathbf{f}(\mathbf{x}^*) = \mathbf{0}$, the solution of (11.2.6a) does not yield a result precisely equal to the degenerate box $\mathbf{x}^*$. Instead, a nondegenerate box containing $\mathbf{x}^*$ is produced. Partly for this reason, these methods are not as rapidly convergent as some other interval Newton methods.

Hansen and Sengupta (1981) noted that the Gauss-Seidel method is more efficient than the Krawczyk method. See also Hansen and Greenberg (1983). Neumaier (1990), discusses the Gauss-Seidel and Krawczyk methods in the form in which intersecting is done only after all new components have been computed. He notes (page 177) that in this form $\mathrm{N}\left(\mathbf{x}, \mathbf{x^I}\right) \subset \mathrm{K}\left(\mathbf{x}, \mathbf{x^I}\right)$ where N_i, the elements of $\mathrm{N}\left(\mathbf{x}, \mathbf{x^I}\right)$, are given by Equation (11.2.7) and K_i, the elements of $\mathrm{K}\left(\mathbf{x}, \mathbf{x^I}\right)$, are given by (11.2.8). Therefore, between the two methods, the Gauss-Seidel method is preferred.

For variations of algorithms using the Gauss-Seidel method, See Hansen and Sengupta (1981) and Hansen and Greenberg (1983).

Shearer and Wolfe (1985a, 1985b) give an improved form of Krawczyk's method that they call the symmetric form. They solve the Krawczyk formula for new interval variables in their natural order. Before recomputing the Jacobian, they again solve the Krawczyk formula. This time, however, they solve for the variables in reverse order. This same symmetric approach can be used to solve (11.2.6a) using the Gauss-Seidel method.

The best available method for solving the preconditioned equation (11.2.6a) is the "hull method" of Section 5.8. It produces the exact hull of the solution set of (11.2.6a). However, it fails if the coefficient matrix is irregular, see Section 5.8.2. In this case, we use the Gauss-Seidel method.

11.4 AN INNER ITERATION

In this section, we describe an "inner iteration" that has been used in the past to improve the convergence of interval Newton methods. It can be used in the way described. In Section 11.4.1, we describe an alternative way to apply an inner iteration. We prefer the latter form. We use this alternative form in the algorithms given in Section 11.12 and 11.14.

The purpose of an inner iteration is to find an approximation for a solution of $\mathbf{f} = \mathbf{0}$ in the current box $\mathbf{x}^{\mathbf{I}}$. This approximation can be used as the point of expansion $\mathbf{x}$ in (11.2.1). The closer $\mathbf{x}$ is to a solution point of $\mathbf{f} = \mathbf{0}$, the smaller the solution set of (11.2.6a).

Later in this section, we describe an inner iteration that generally obtains a point $\mathbf{x} \in \mathbf{x}^{\mathbf{I}}$ where $||\mathbf{f}(\mathbf{x})||$ is smaller than at the center of $\mathbf{x}^{\mathbf{I}}$. The expectation is that by obtaining this better point of expansion, we require fewer iterations of the main algorithm. This means that fewer evaluations of the Jacobian are required. Therefore, less overall computation is required to solve the system of nonlinear equations. This has been verified by experiment. For example, see Hansen and Greenberg (1983).

In Section 11.2, we note that the first step in solving (11.2.3) is to multiply by an approximate inverse $\mathbf{B}$ of the center of the coefficient matrix $\mathbf{J}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$. Hansen and Greenberg (1983) pointed out that since $\mathbf{B}$ is available, we can use it to perform a real Newton step (or steps) to try to obtain a better approximation for a zero of $\mathbf{f}$ than the center $\mathrm{m}(\mathbf{x}^{\mathbf{I}})$ of $\mathbf{x}^{\mathbf{I}}$.

The inner iteration is

$$\mathbf{z}^{(i+1)} = \mathbf{z}^{(i)} - \mathbf{Bf}(\mathbf{z}^{(i)})\ (i = 0, 1, \cdots). \tag{11.4.1}$$

The initial point $\mathbf{z}^{(0)}$ can be chosen to be the center of the current box. At the final point, $\mathbf{f}$ is as small or smaller in norm than at $\mathbf{z}^{(0)}$. The final point is used as the point of expansion $\mathbf{x}$ in (11.2.1). This final point must be in $\mathbf{x}^{\mathbf{I}}$ so that the expansion (11.2.1) is valid for the ensuing interval Newton step using $\mathbf{x}^{\mathbf{I}}$.

The iteration is discontinued if $\mathbf{z}^{(i+1)}$ is outside $\mathbf{x}^{\mathbf{I}}$. Suppose this is the case. Let $\mathbf{z}'$ denote the point between $\mathbf{z}^{(i)}$ and $\mathbf{z}^{(i+1)}$ where the line segment joining these two points crosses the boundary of $\mathbf{x}^{\mathbf{I}}$. If $||\mathbf{f}(\mathbf{z}')|| < ||\mathbf{f}(\mathbf{z}^{(i)})||$, we choose $\mathbf{z}'$ as our approximation for a zero (and hence as our point of expansion). Otherwise,we choose $\mathbf{z}^{(i)}$. The vector norm we use in this book is

$$||\mathbf{f}|| = \max_{1 \leq j \leq n} |f_j|.$$

The inner iteration is also stopped if $||\mathbf{f}(\mathbf{z}^{(i+1)})|| > ||\mathbf{f}(\mathbf{z}^{(i)})||$. In this case, we let $\mathbf{x} = \mathbf{z}^{(i)}$. Otherwise, the iteration (11.4.1) is stopped after three steps and we set $\mathbf{x} = \mathbf{z}^{(3)}$. Further iteration might converge to a zero of $\mathbf{f}$. However, the convergence rate is only linear if $\mathbf{B}$ is fixed. Hence, it is not efficient to perform too many iterations.

It is more important to have $\mathbf{f}\,(\mathbf{x})$ small when solving (11.2.6a) using either the hull method or Gaussian elimination than using a step of the Gauss-Seidel method. This is because the former two methods yield convergence in one step of (11.2.6a) if $\mathbf{f}\,(\mathbf{x}) = \mathbf{0}$ (and exact interval arithmetic is used). A Gauss-Seidel step does not.

Therefore, a different number of steps using (11.4.1) can be used depending on which method is used to solve the linearized system (11.2.3). In the algorithm given below in this section, we have used the same upper limit (i.e., 3) on the number of inner iterations using (11.4.1).

The amount of work to do the inner iteration is small compared to that required to compute $\mathbf{J}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$ and $\mathbf{B}$ and then to solve the preconditioned linear system (11.2.6a). Therefore, the inner iteration is worth doing even when its effect on convergence is small. However, its use depends on having computed $\mathbf{B}$ for use in preconditioning.

We now list the steps of the inner iteration. All steps are done in ordinary rounded arithmetic. That is, interval arithmetic need not be used. Let the current box be $\mathbf{x}^{\mathbf{I}}$.

0. Initialize:
   (a) Set $\mathbf{z}^{(0)} = \mathrm{m}(\mathbf{x}^{\mathbf{I}})$.
   (b) Set $i = 0$.
1. Compute $\mathbf{z}^{(i+1)}$ using (11.4.1).
2. If $\mathbf{z}^{(i+1)} \in \mathbf{x}^{\mathbf{I}}$, go to Step 3. Otherwise, go to Step 5.
3. If $||\mathbf{f}(\mathbf{z}^{(i+1)})|| < ||\mathbf{f}(\mathbf{z}^{(i)})||$, go to Step 4. Otherwise, set $\mathbf{x} = \mathbf{z}^{(i)}$ and go to Step 6.
4. If $i < 3$, replace i by $i+1$ and go to Step 1. Otherwise, set $\mathbf{x} = \mathbf{z}^{(i+1)}$ and go to Step 6.
5. Let $\mathbf{z}'$ denote the point where the line segment joining $\mathbf{z}^{(i)}$ and $\mathbf{z}^{(i+1)}$ crosses the boundary of $\mathbf{x}^{\mathbf{I}}$. If $||\mathbf{f}(\mathbf{z}')|| < ||\mathbf{f}(z^{(i)})||$, set $\mathbf{x} = \mathbf{z}'$. Otherwise, set $\mathbf{x} = \mathbf{z}^{(i)}$.
6. If x_i is not in X_i for some $i = 1, \cdots, n$, replace x_i by the nearest endpoint of X_i. (Note that rounding might have been such that $\mathbf{x}$ is not in $\mathbf{x}^{\mathbf{I}}$). Return to the main program.

The first time Step 1 is used in this procedure, we have $\mathbf{z}^{(0)} = \mathrm{m}\left(\mathbf{x}^{\mathbf{I}}\right)$ and we want $\mathbf{Bf}(\mathbf{z}^{(0)})$. Note, that $-\mathbf{Bf}(\mathbf{z}^{(0)})$ is the vector $\mathbf{r}^{\mathbf{I}}$ (see Equation (11.2.5)) already computed in the algorithm in Section 11.12. That is, we need not recompute $\mathbf{f}$ and $\mathbf{Bf}$ the first time Step 1 is used.

The same kind of inner iteration can be performed in the one-dimensional interval Newton method described in Chapter 9. However, in this case, the derivative of $\mathbf{f}$ (i.e., the one-dimensional Jacobian) generally takes about the same amount of computation to evaluate as is required to evaluate $\mathbf{f}$. Hence, there is little motive to do the inner iteration.

11.4.1 A POST-NEWTON INNER ITERATION

In the previous section, we describe an inner iteration that is used *during* an interval Newton step. In this section we describe how an inner iteration can be used *after* an interval Newton step. It is designed to improve the performance of a subsequent interval Newton step rather than the current one.

If all the arguments of all the elements of the Jacobian are chosen to be intervals, then the method of the previous section is preferred to the one we now describe. This is because (in this case) changing the point of expansion does not change the Jacobian. However, in practice, one implements the Jacobian so that some arguments of some of its elements are real rather than interval. This is necessarily the case if we use a slope expansion (see Section 7.7). It is also desirable when using a Taylor expansion. (See Section 7.4.)

Suppose we compute the Jacobian by expanding about a particular point such as the center of a box. Suppose we use Jacobian data to determine a point nearer a solution of the given system. If we change the point of expansion to this new point, we change the Jacobian and it must be recomputed if it is to be used in the current interval Newton step. Therefore, the benefit of having a good point of expansion is offset by having to perform this recomputation.

In this section, we advocate an alternative method. In this method, we use an inner iteration after the interval Newton step is performed. Suppose the inner iteration using the real matrix $\mathbf{B}$ obtains a point $\mathbf{x}^B$ as an approximate solution to a system of equations. When a subsequent interval Newton step is applied to a box containing $\mathbf{x}^B$, we use $\mathbf{x}^B$ as the point of expansion over the box. Therefore, the Jacobian need be computed only once to perform a given Newton step.

This point $\mathbf{x}^B$ is generally obtained by an inner iteration in a larger box than the current one. It might seem reasonable to evaluate $\mathbf{f}$ at the center of the current box to see if this value is smaller in norm than the value at $\mathbf{x}^B$. If so, the Jacobian can be expanded about this center. We do not do this.

To use this procedure, one must store $\mathbf{x}^B$ along with the box in which it was obtained. If the box changes, we keep $\mathbf{x}^B$ only if it is contained in the new box. If the box is split, we store the point $\mathbf{x}^B$ only with the one subbox containing it.

11.5 STOPPING CRITERIA

In this section, we consider how to terminate an interval Newton algorithm. Termination procedures for interval methods are similar in some ways to those for noninterval methods. However, they are distinctly different in other ways. For a discussion of termination criteria in the noninterval case, see, for example, Dennis and Schnabel (1983).

We begin this section on stopping criteria by discussing some relevant concepts.

Suppose we evaluate $\mathbf{f}$ at a point $\mathbf{x}$ using rounded interval arithmetic. Instead of obtaining the noninterval vector $\mathbf{f}(\mathbf{x})$, outward rounding causes us to get an interval vector $\mathbf{f}^{\mathbf{I}}(\mathbf{x})$ containing $\mathbf{f}(\mathbf{x})$. Let $\mathbf{x}^*$ be a zero of $\mathbf{f}$. Because of rounding errors, there is a set of points $\mathbf{x}_s$ about $\mathbf{x}^*$ for which $\mathbf{0} \in \mathbf{f}^{\mathbf{I}}(\mathbf{x})$ for all $\mathbf{x} \in \mathbf{x}_s$. Let $\mathbf{x}^{\mathbf{I}*}$ denote the smallest box containing this set.

A reasonable convergence criterion is to require that the interval Newton method produce a box that contains $\mathbf{x}^{\mathbf{I}*}$ and is a good approximation for $\mathbf{x}^{\mathbf{I}*}$. Actually, we can generally compute a slightly better result because information is used about the Jacobian and not just $\mathbf{f}$. However, the extra accuracy might require using expansion points on the boundary of the current box that are not selected by the algorithm we give in Section 11.12. In that algorithm, we make no special effort to improve on $\mathbf{x}^{\mathbf{I}*}$. Sometimes the result is better and sometimes not. For an illustrative example in the one-dimensional case, see Section 9.7.

For simplicity, we discuss termination as if our goal is to compute a good approximation for $\mathbf{x}^{\mathbf{I}*}$. We discuss tolerances that a user might prescribe to save work by causing termination before $\mathbf{x}^{\mathbf{I}*}$ is closely approximated. If these tolerances are too small, the algorithm reverts to approximating $\mathbf{x}^{\mathbf{I}*}$ to prevent fruitless computation.

In Section 9.4, we noted that if, in the one dimensional case, an interval X contains zero, then $1 \leq \mathrm{w}(X)/|X| \leq 2$. Therefore, care has to be taken when using a relative stopping criterion. The condition $\mathrm{w}(X)/|X| \leq \varepsilon$ can never be satisfied if $\varepsilon < 1$ and $0 \in X$.

We might use a relative criterion of this form in the multidimensional case (and a user might choose to do so). However, a solution vector $\mathbf{x}^{\mathbf{I}*}$ might have a component X_i^* that is zero. If this component of a bounding box $\mathbf{x}^{\mathbf{I}}$ is the widest component of the box, a relative criterion with $\varepsilon < 1$ cannot be satisfied.

It is possible to use a procedure in which a separate criterion is used for each component of a box. A component containing zero can use (say) an absolute criterion. This has the added benefit of permitting unequal scaling of variables to be taken into account.

We take the simple approach of using an absolute criterion for the box as a whole. Thus, we define:

Criterion A:

$$\mathrm{w}(\mathbf{x}^{\mathbf{I}}) < \varepsilon_X \text{ for some } \varepsilon_X > 0. \tag{11.5.1}$$

As noted in Section 9.4, care must be taken when choosing ε_X for use in an absolute criterion of this form.

A user might want to have $||\mathbf{f}\left(\mathbf{x}^{\mathbf{I}}\right)||$ small for all $\mathbf{x}$ in a final box $\mathbf{x}^{\mathbf{I}}$ bounding a zero of $\mathbf{f}$. Therefore, we define:

Criterion B:

$$||\mathbf{f}(\mathbf{x}^{\mathbf{I}})|| < \varepsilon_f \text{ for some } \varepsilon_f > 0. \tag{11.5.2}$$

Criterion A corresponds to the noninterval criterion that two successive iterates be close together. The criterion in the noninterval case corresponding to **Criterion B** is that an approximation (of uncertain accuracy) for $||\mathbf{f}(\mathbf{x})||$ be small at a single point $\mathbf{x}$.

In each step of the interval Newton method, the current box is either substantially reduced in size[1] (see Section 11.7) or split into subboxes. Therefore, each remaining box is eventually as small as desired (subject to wordlength limitation), so it is trivially easy to satisfy **Criterion A**.

[1]We often simply write: "reduced".

The interval approach to solving systems of equations has a virtue that is unavailable in the noninterval case: in practise, it is possible to determine when ε_X has been chosen too small. If ε_X is too small, many boxes (each satisfying **Criterion A**) are required to cover the set $\mathbf{x}^{\mathbf{I}*}$. Similarly, ε_f can be chosen too small. Therefore, we wish to override **Criteria A** and **B** when they are too stringent and try to approximate $\mathbf{x}^{\mathbf{I}*}$ instead.

Note, however, that a user might wish to map the solution region by small boxes as "pixels". This can be done by allowing **Criterion A** to determine convergence.

The vector $\mathbf{f}(\mathbf{x}^{\mathbf{I}})$ is not computed by the Newton algorithm. Therefore, extra effort is required to use **Criterion B**. However, extra information results from its use. When **Criterion B** is satisfied for all final boxes, we know that $||\mathbf{f}\ (\mathbf{x})\ || < \varepsilon_f$ for all $\mathbf{x}$ in all final boxes. This information can be useful.

Actually, it is not necessary to compute $\mathbf{f}(\mathbf{x}^{\mathbf{I}})$ directly. Instead, we can reduce the effort (unless $\mathbf{f}$ is quite simple) by using the relation

$$\mathbf{f}(\mathbf{x}^{\mathbf{I}}) \subset \mathbf{f}^{\mathbf{I}}(\mathbf{x}) + \mathbf{J}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})(\mathbf{x}^{\mathbf{I}} - \mathbf{x}). \tag{11.5.3}$$

Since $\mathbf{f}^{\mathbf{I}}(\mathbf{x})$ and $\mathbf{J}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$ are computed when performing a Newton step, the only extra effort required to compute a bound on $\mathbf{f}(\mathbf{x}^{\mathbf{I}})$ is a matrix-vector multiplication and a vector addition. Moreover, when $\mathrm{w}(\mathbf{x}^{\mathbf{I}})$ is small, this method generally bounds the range of $\mathbf{f}$ over $\mathbf{x}^{\mathbf{I}}$ more sharply than directly evaluating $\mathbf{f}(\mathbf{x}^{\mathbf{I}})$.

When we use the relation in (11.5.3) to bound $\mathbf{f}(\mathbf{x}^{\mathbf{I}})$, we use the already computed $\mathbf{J}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$ to obtain a new box $\mathbf{x}^{\mathbf{I}'}$ in $\mathbf{x}^{\mathbf{I}}$. Thus, we use $\mathbf{f}(\mathbf{x}^{\mathbf{I}'}) \subset \mathbf{f}^{\mathbf{I}}(\mathbf{x}) + \mathbf{J}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})(\mathbf{x}^{\mathbf{I}} - \mathbf{x})$. Therefore, $\mathbf{J}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$ has wider elements than the correct Jacobian. As a result, the computed bound on $\mathbf{f}(\mathbf{x}^{\mathbf{I}'})$ is wider than necessary. However, if we simply evaluate $\mathbf{f}(\mathbf{x}^{\mathbf{I}'})$, then dependence causes widening so computed bounds are not sharp in either case.

To see if **Criterion B** is satisfied, another option when evaluating $\mathbf{f}(\mathbf{x}^{\mathbf{I}})$ is to use hull consistency to bound $\mathbf{f}(\mathbf{x}^{\mathbf{I}})$ as described in Section 10.10. The box might be reduced in the process.

In our termination procedure, we check whether **Criterion A** is satisfied before checking to see if **Criterion B** is satisfied. If **Criterion A** is not satisfied, we continue iterating on the current box. Hence, we often avoid evaluation of $\mathbf{f}(\mathbf{x}^{\mathbf{I}})$ because **Criterion B** is not checked in this case.

We want to have the option of stopping only when we have the best (or near best) possible result with the numerical precision used. That is, we want the option to stop when (and only when) we have a good approximation for $\mathbf{x}^{\mathbf{I}*}$. This is especially desirable when one or both of the tolerances ε_X and ε_f are chosen too small. We now discuss how this can sometimes be done.

Recall that to compute a new interval Newton iterate, we solve

$$\mathbf{f}(\mathbf{x}) + \mathbf{J}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})(\mathbf{y} - \mathbf{x}) = \mathbf{0}.$$

(See (11.2.4a).) To precondition the system, the first step in solving this equation is to multiply by an approximate inverse $\mathbf{B}$ of the center of $\mathbf{J}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$. (See Sections 5.6 and 11.2.) The resulting coefficient matrix is $\mathbf{M}(\mathbf{x}, \mathbf{x}^{\mathbf{I}}) = \mathbf{BJ}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$. We discuss another way to compute $\mathbf{M}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$ in Section 11.9.

Assume that $\mathbf{J}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$ is regular (i.e., does not contain a singular matrix). Then the inverse $\mathbf{B}$ exists and $\mathbf{M}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$ can be computed. If $w(\mathbf{x}^{\mathbf{I}})$ is small, then the interval elements of $\mathbf{M}\left(\mathbf{x}, \mathbf{x}^{\mathbf{I}}\right)$ are generally small in width and $\mathbf{M}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$ approximates the identity matrix.

For our current purpose, we are interested only in whether $\mathbf{M}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$ is regular or not. Below, we see that for another reason $\mathbf{M}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$ is checked for regularity. Therefore, no extra computation is required to make this check for our current purpose.

Note that regularity of $\mathbf{M}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$ assures that any zero of $\mathbf{f}$ in $\mathbf{x}^{\mathbf{I}}$ is simple. In this case, we are able to decide when $\mathbf{x}^{\mathbf{I}}$ approximates $\mathbf{x}^{\mathbf{I}*}$. We use

Criterion C: $\mathbf{0} \in \mathbf{f}^{\mathbf{I}}(\mathbf{x})$, $\mathbf{M}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$ is regular, and $\mathrm{N}(\mathbf{x}, \mathbf{x}^{\mathbf{I}}) \supset \mathbf{x}^{\mathbf{I}}$.

The condition $\mathbf{0} \in \mathbf{f}^{\mathbf{I}}(\mathbf{x})$ assures that $\mathbf{x}$ is at or near a zero of $\mathbf{f}$. The condition $\mathrm{N}(\mathbf{x}, \mathbf{x}^{\mathbf{I}}) \supset \mathbf{x}^{\mathbf{I}}$ indicates that the interval Newton step is unable to reduce $\mathbf{x}^{\mathbf{I}}$. When $\mathbf{M}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$ is regular, the other two conditions of **Criterion C** assure that $\mathbf{x}^{\mathbf{I}}$ approximates $\mathbf{x}^{\mathbf{I}*}$.

In our algorithm, if **Criterion C** is satisfied, we stop trying to reduce $\mathbf{x}^{\mathbf{I}}$. We do not require that **Criteria A** and **B** be satisfied in this case. If the tolerance ε_X and/or ε_f are too small and if we do not give precedence to

Criterion C, a box approximating $\mathbf{x}^{\mathbf{I}*}$ might be split into many subboxes, all of which must be output. The "solution" is their union. Letting **Criterion C** dominate the other criteria, less work is done, because the algorithm tends to return a single box approximating $\mathbf{x}^{\mathbf{I}*}$.

If $\mathbf{x}^{\mathbf{I}}$ contains a multiple zero (or more than one isolated zero) of $\mathbf{f}$, the Jacobian $\mathbf{J}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$ is irregular. Therefore, $\mathbf{M}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$ is also irregular. In this case, we need a criterion different from **Criterion C** to decide when $\mathbf{x}^{\mathbf{I}}$ approximates $\mathbf{x}^{\mathbf{I}*}$. This case is more difficult. However the "difficulty" is really only a nuisance. Consider

Condition D: $\mathbf{0} \in \mathbf{f}^{\mathbf{I}}(\mathbf{x})$, $\mathrm{N}(\mathbf{x}, \mathbf{x}^{\mathbf{I}}) \supset \mathbf{x}^{\mathbf{I}}$, and $\mathbf{M}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$ is irregular.

All we really need is a condition that assures that $\mathbf{x}^{\mathbf{I}}$ is not large when **Condition D** holds. However, a choice must be made on how to proceed. We describe our choice in this section. When $\mathbf{M}(\mathbf{x}, \mathbf{x}^{\mathbf{I}}) = \mathbf{BJ}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$ is regular, we can determine the hull of the solution set of the preconditioned system

$$\mathbf{Bf}(\mathbf{x}) + \mathbf{BJ}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})(\mathbf{y} - \mathbf{x}) = \mathbf{0}$$

by the hull method given in Section 5.8. When $\mathbf{M}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$ is irregular, this method is not applicable; and neither is Gaussian elimination. Therefore, we use the Gauss-Seidel method described in Section 5.7. A difficulty of the Gauss-Seidel method (from the perspective of termination) is that even when $\mathbf{0} \in \mathbf{f}^{\mathbf{I}}(\mathbf{x})$ and $\mathbf{x}^{\mathbf{I}}$ is large, it is possible to have $\mathrm{N}(\mathbf{x}, \mathbf{x}^{\mathbf{I}}) \supset \mathbf{x}^{\mathbf{I}}$.

Suppose that in our algorithm, we have applied hull and box consistencies to a box (say $\mathbf{y}^{\mathbf{I}}$) and have obtained the box $\mathbf{x}^{\mathbf{I}}$ to which the Newton method is applied. Suppose $\mathbf{y}^{\mathbf{I}} \neq \mathbf{x}^{\mathbf{I}}$. Then the consistency methods have reduced $\mathbf{y}^{\mathbf{I}}$. In this case, we assume $\mathbf{x}^{\mathbf{I}}$ is not as small as it can be made to be, even if the Newton method fails to reduce it.

In our algorithm, we rely upon **Criteria A** and **B** to stop iteration on a box for which **Condition D** holds. However, we now point out some alternatives that can be used, that we have not yet tried.

We can compare $||f(\mathbf{x})||$ with $||f(\mathbf{x}^{\mathbf{I}})||$ for $\mathbf{x} \in \mathbf{x}^{\mathbf{I}}$. Note that $||f(\mathbf{x}^{\mathbf{I}})||$ approaches $||f(\mathbf{x})||$ as the width of $\mathbf{x}^{\mathbf{I}}$ approaches zero with $\mathbf{x} \in \mathbf{x}^{\mathbf{I}}$.

We can attempt to find a point $\mathbf{x}$ in $\mathbf{x^I}$ for which $\mathbf{0} \notin \mathbf{f^I}(\mathbf{x})$. To do so, we can evaluate $\mathbf{f}$ at a few corners of $\mathbf{x^I}$. If such a point can be found, then $\mathbf{x^I}$ might not be as small as we want it to be.

An unsatisfactory alternative is to accept a box $\mathbf{x^I}$ when $w(\mathbf{x^I})$ is less than some relatively large tolerance (and **Condition D** is satisfied), Then the user can examine the output, reset the tolerance on $w(\mathbf{x^I})$ to a smaller value, and continue iterating.

It is unlikely that **Condition D** holds unless $\mathbf{x^I}$ approximates $\mathbf{x^{I*}}$. It is rare to have $\mathbf{0} \in \mathbf{f^I}(\mathbf{x})$ except when the smallness of $\mathbf{x^I}$ forces $\mathbf{x}$ to be near a zero of $\mathbf{f}$.

Note that boxes near a solution can satisfy the convergence criteria but not contain a zero of $\mathbf{f}$. This is rare for well conditioned simple zeros, but is common for multiple zeros. Such a "solution" box generally abuts or is near a box that actually contains a zero.

Note, also, that prescribing a value of ε_X in **Criterion A** does not assure that the location of a zero of $\mathbf{f}$ is bounded to within ε_X. First of all, it might not be possible with the numerical precision used to isolate the zero that sharply. The box $\mathbf{x^{I*}}$ (described above) containing the region of uncertainty might be of width larger than ε_X. In this case, we approximate what is achievable with the given precision; not what is requested.

Also, the algorithm might return a cluster of two or more abutting boxes each of width less than ε_X. The smallest box containing such a cluster might exceed ε_X in width. To isolate a given zero within a box of a given size, one can choose ε_X slightly smaller than the desired value. This advise is meaningless if ε_f is chosen so small that function width causes the width of a final box to be even smaller than ε_X. Moreover, a solution box's width is often smaller than ε_X in width because a Newton method tends to "overshoot the mark". Similarly, if ε_X is small and ε_f is large, $||\mathbf{f}(\mathbf{x^I})||$ is much smaller than ε_f for each final box $\mathbf{x^I}$.

11.6 THE TERMINATION PROCESS

In this section, we describe two termination procedures based on the analysis of Section 11.5. Assume the tolerances ε_X and ε_f have been specified. Let $\mathbf{x^I}$ denote the box to which a Newton step is applied. The result can be more than one box. Let $\mathbf{x^{I'}}$ denote a resulting box although it might not be the only one.

Our first process causes termination when **Criteria A** and **B** are satisfied. It is useful when the tolerances are chosen "relatively large". In this case, processing stops early and computing effort is relatively small. The second process forces the algorithm to produce the best (or near best) possible result for simple zeros.

The termination process for the first case involves the following steps:

1. If **Criteria A** and **B** of Section 11.5 are satisfied, accept $\mathbf{x^{I'}}$ as a final result. Otherwise, go to Step 2.

2. If **Criterion C** of Section 11.5 is satisfied, accept $\mathbf{x^{I'}}$ as a final result. Otherwise, go to Step 3.

3. Continue processing $\mathbf{x^{I'}}$.

The steps for the second case are as follows:

1. If **Criterion C** is satisfied, accept $\mathbf{x^{I'}}$ as a final result. Otherwise, go to Step 2.

2. If the preconditioned coefficient matrix $\mathbf{M}(\mathbf{x}, \mathbf{x^I})$ is regular, continue processing the box $\mathbf{x^{I'}}$. Otherwise, go to Step 3.

3. If **Criteria A** and **B** are satisfied, accept $\mathbf{x^{I'}}$ as a final result. Otherwise, go to Step 4.

4. Continue processing $\mathbf{x^{I'}}$.

Note that our termination processes do not take special measures when **Condition D** of Section 11.5 holds. In this case we assume that the tolerances in **Criteria A** and **B** are chosen adequately to cause termination in a suitable manner.

It might happen that two or more abutting boxes are output as final results and it is their union that contains a solution (or multiple nearly coincident solutions). However, some of these output boxes might not contain a solution.

11.7 RATE OF PROGRESS

If a box is not sufficiently reduced by a Step of our algorithm, we split the box into two or more subboxes and apply the algorithm to each subbox separately. A Newton algorithm is more efficient when applied to a small box than to a large one. This is partly because Taylor expansions are better approximations to functions over small regions. It is also partly because the effect of dependence is less for small boxes.

We need a criterion to test when a box is "sufficiently reduced" in size during a Step or steps of our algorithm. We derive a criterion for sufficient reduction in this section. The purpose of the criterion is to enable us to decide that it is not necessary to split a box. Instead, any procedure can be repeated that has sufficiently reduced the box.

Let $\mathbf{x^I}$ denote a box to which the algorithm is applied in a given step. Assume that $\mathbf{x^I}$ is not entirely deleted by the algorithm. Then either $\mathbf{x^I}$ or a subbox, say $\mathbf{x^{I}}'$, of $\mathbf{x^I}$ is returned. It might have been determined that a gap can be removed from one or more component of $\mathbf{x^{I}}'$. If so, we can choose to split $\mathbf{x^{I}}'$ by removing the gap(s). We discuss this case subsequently. For now, assume we ignore any gaps.

We could say that $\mathbf{x^I}$ is sufficiently reduced when for some $i = 1, \cdots, n$, we have

$$\mathrm{w}(x_i') < \alpha\, \mathrm{w}(x_i) \tag{11.7.1}$$

for some constant α where $0 < \alpha < 1$. But suppose x_i is the narrowest component of $\mathbf{x^I}$. Then this condition is satisfied when there is little decrease in the distance between extreme points of $\mathbf{x^I}$.

We could also require that

$$\mathrm{w}(\mathbf{x^{I}}') < \beta\, \mathrm{w}(\mathbf{x^I}) \tag{11.7.2}$$

for some constant β where $0 < \beta < 1$. In this case, we compare the widest component of $\mathbf{x^I}$ with the widest component of $\mathbf{x^{I}}'$. But even if every component of $\mathbf{x^I}$ except the widest is reduced to zero width, this criterion says that insufficient progress has been made.

We avoid these difficulties by requiring that for some $i = 1, \cdots, n$, we have

$$\mathrm{w}(x_i) - \mathrm{w}(x_i') > \gamma\, \mathrm{w}(\mathbf{x}^{\mathbf{I}}) \tag{11.7.3}$$

for some constant γ where $0 < \gamma < 1$. This assumes that at least one component of $\mathbf{x}^{\mathbf{I}}$ is reduced in width by an amount related to the widest component of $\mathbf{x}^{\mathbf{I}}$.

We choose $\gamma = 0.25$. Thus, we define

$$D = 0.25\, \mathrm{w}(\mathbf{x}^{\mathbf{I}}) - \max_{1 \le i \le n} \{\mathrm{w}(x_i) - \mathrm{w}(x_i')\} \tag{11.7.4}$$

We say that $\mathbf{x}^{\mathbf{I}}$ is *sufficiently reduced* if $D \le 0$.

When, according to the criterion, a box is not sufficiently reduced by one "pass" through our algorithm, we split the box into subboxes and apply the algorithm to each subbox separately. A pass though the main algorithm includes application of hull and box consistencies and perhaps application of the interval Newton method. In the next section, we describe how splitting is done. When a box is sufficiently reduced, we do not split it. Instead, we try to reduce it further by reapplying the algorithm.

We also need another measure of rate of progress. Our overall algorithm for solving systems of nonlinear equations uses hull consistency, box consistency, and two forms of Newton's method. We need to know whether it is better to emphasize the use of consistency methods, or a Newton method.

Early in the solution process when the current box is large, a Newton method might make little or no progress reducing the box. If hull consistency makes sufficient progress (i.e., if $D \le 0$), we might let it continue to be used without applying a Newton step. But when the box is small, a Newton method generally exhibits quadratic convergence (see Theorem 11.15.8 below) and can be much faster than hull consistency.

We wish to avoid repeated use of hull consistency when a Newton step might be more efficient. The computational effort to apply a Newton step rises faster as a function of the number n of variables than does hull consistency. Therefore, we arbitrarily limit the number of hull consistency applications to n before we apply a Newton step. Recall that n is the number of variables in the system being solved.

We also generally apply a Newton step when hull consistency (and box consistency) make insufficient progress.

On the other hand, if a Newton step makes exceptionally good progress, we repeat it. In this case, we use hull and box consistencies in only a limited way. See Section 11.9. We repeat the Newton step when it reduces the width of the box by a factor of at least 8. That is, we repeat a Newton step if

$$\mathrm{w}(\mathbf{x}^{\mathbf{I}'}) \leq 0.125\, \mathrm{w}(\mathbf{x}^{\mathbf{I}}). \tag{11.7.5}$$

We do not use this criterion form (see (11.7.2)) for determining when progress is barely sufficient. However, we use it in the present case when it indicates that progress is more than just sufficient.

11.8 SPLITTING A BOX

Our algorithm for solving a system of nonlinear equations is composed of several procedures. Suppose we have applied each of these procedures to try to find a solution in a given box and the box is not sufficiently reduced. In this case, we split the box into subboxes. We then apply the procedures to these smaller subboxes. Actually, we sometimes split when we have used some, but not all, of the procedures in the algorithm.

In this section, we consider how to split a box when the procedures for reducing a box are making insufficient progress. If a procedure has produced a gap in one or more components of the box, this provides an optional way to split. In fact, if a gap occurs, it can be worthwhile using it to split the box even when progress in reducing the box is sufficiently rapid. We discuss gaps below in this section. For now, assume no gap exists.

Suppose that we have completed the part of our algorithm that tries to reduce a box $\mathbf{x}^{\mathbf{I}}$, and that the result is $\mathbf{x}^{\mathbf{I}'}$. Assume that $D > 0$ where D is given by (11.7.4). Then $\mathbf{x}^{\mathbf{I}}$ has not been sufficiently reduced; so we wish to split $\mathbf{x}^{\mathbf{I}'}$.

Suppose we split each interval component of a box of n components into two subintervals. Then 2^n subboxes are generated. This is generally too many new boxes even for moderate values of n. Therefore, if the number n of variables is ≤ 3, we split each component of $\mathbf{x}^{\mathbf{I}'}$. If $n > 3$ we sometimes split just three components so that eight subboxes are generated; and we sometimes split so that $n + 1$ subboxes are generated.

Suppose we have made a pass through our algorithm (see Section 11.12) for solving a system of nonlinear equations; and it specifies that splitting should be done. How we split depends upon whether the Newton method was used in the given pass. This is because in applying the Newton method, we obtain information that is useful in deciding how to split.

Let us first consider the case in which the Newton method is not used. in a particular pass. Instead, we use only consistency methods. We split when the consistency methods do not make sufficient progress in reducing a box. A significant fact is that consistency methods require much less effort to apply than the Newton method. We can split so as to produce only a few new subboxes. If the splitting does not enable the next pass through the algorithm to make good progress, the original application of the consistency methods can have wasted only a relatively small effort.

We split only three interval variables defining the box. This produces eight new subboxes. We split the three widest components of the box. This can be a poor choice because it does not take scaling into account. We rely upon a case in which we have applied a Newton method before splitting to help us do splitting in a better way.

The way we split when the Newton method has been used requires some introductory discussion. We first discuss how we decide the importance of splitting a particular component of a box. We then consider how the splitting is done.

When applied to a box $\mathbf{x}^{\mathbf{I}}$, assume the algorithm produces a box $\mathbf{x}^{\mathbf{I}'}$ that we wish to split. We wish to split because $\mathbf{x}^{\mathbf{I}'}$ differs very little from $\mathbf{x}^{\mathbf{I}}$. We would like to have the Jacobian of the nonlinear system $\mathbf{f}$ evaluated over the box $\mathbf{x}^{\mathbf{I}'}$ for use in the splitting procedure. Rather than compute it, we use the Jacobian $\mathbf{J}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$ that was evaluated during the Newton process. This differs little from $\mathbf{J}(\mathbf{x}, \mathbf{x}^{\mathbf{I}'})$ because (by assumption) $\mathbf{x}^{\mathbf{I}'}$ differs little from $\mathbf{x}^{\mathbf{I}}$. Moreover, we need only a crude estimate of the Jacobian.

Knowing $\mathbf{J}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$, we can easily compute

$$T_j = \mathrm{w}(X'_j) \sum_{i=1}^{n} |\mathbf{J}_{ij}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})|. \tag{11.8.1}$$

It is clear that the system $\mathbf{f}$ depends more strongly on x_j than on x_k over $\mathbf{x}^{\mathbf{I}'}$ if $T_j > T_k$ In this case, we prefer to split the j-th component of $\mathbf{x}^{\mathbf{I}'}$ rather than the k-th. This criterion is crude because we are using a single number, T_j, to represent behavior of an entire system. The criterion is especially crude because when determining $\mathbf{J}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$, we select some arguments of elements of $\mathbf{J}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$ to be real and some to be interval. See Section 7.3. The choice of which arguments are chosen to be real can affect the value of T_j.

Having determined which components we prefer to split, we now discuss the information that we use to decide how to split.

Ratz (1994) describes a splitting strategy that produces $n + 1$ subboxes (of varying sizes) of a given n dimensional box. Using his procedure, we first split the box in half in a given component. We store one of the resulting subboxes without splitting any other component. We split the remaining subbox in half in one (different) component. We again store one portion and repeat the process using the remaining subbox. In this way a splitting is done in each dimension, but for a succession of smaller and smaller boxes.

Ratz compares this strategy with ones in which fewer subboxes are produced that all have the same size. He considers a global optimization method that uses a Gauss-Seidel step to solve the system of nonlinear equations formed by the gradient of the objective function. He gives numerical evidence to show his strategy provides better overall efficiency than splitting a few dimensions to obtain subboxes of equal sizes.

In Ratz's procedure, each successive splitting divides a component of a box in half. However, we use information about the T_j to split components into portions of differing widths. We store the smaller portion and continue to split the subbox with the larger portion of the component. We have noted that T_j is a crude measure of how much $\mathbf{f}$ changes as x_j changes over a box. We now use an even cruder measure. We assume that

$$V = \left[\sum_{i=1}^{n} T_i^2 \right]^{1/2}$$

is a measure of how much $\mathbf{f}$ can change due to the combined changes in $x_1, \cdots, x_n$ over the box. We would like to simply evaluate $\mathbf{f}$ over each generated box, but before we split, we do not know the dimensions of the boxes concerned. Functions other than V could be used, but using V is convenient.

Assume we have ordered the variables so that

$$T_j \geq T_{j+1}\,(j = 1, \cdots, n-1)\,.$$

We use the following modification of Ratz's procedure. We split X_1 into two unequal parts. We store the smaller resulting subbox and divide the component X_2 of the larger portion into two unequal parts. We repeat this process until X_{n-1} is divided. In the final step, we divide X_n into equal halves. We save both of the two subboxes.

Note that T_j $(j = 1, \cdots, n)$ and hence V changes as the box changes. We choose each splitting so that V is the same for all saved subboxes. Let w_i $(i = 1, \cdots, n)$ denote the width of the i-th component of the initial box being subdivided. Let w'_i $(i = 1, \cdots, n)$ denote the width of the i-th component of the subbox that is stored when the i-th component is split. Let $\alpha_i = w'_i/w_i$. It can be verified that V is the same for each box saved if

$$\begin{aligned}\alpha_{n-1} &= \frac{1}{2} + \frac{3T_n^2}{8T_{n-1}^2}\\ \alpha_k &= \frac{1}{2}\left[1 + \alpha_{k+1}(2 - \alpha_{k+1})\frac{T_{k+1}^2}{T_k^2}\right]\ (k = 1, \cdots, n-2)\end{aligned}$$

Each α_i, and thus w'_i $(i = 1, \cdots, n-2)$ can be found by recurring backward.

Suppose we are splitting a particular component of a box by the process we have described. Since the component is not split in half, we must decide whether to have the split occur in the lower half or the upper half of the component.

We have no information concerning how well a Newton step will perform in each new subbox. In fact, since we use the Jacobian in deciding how to split, it is reasonable to assume that a Newton step (which uses the Jacobian) will be equally effective in any one of the generated subboxes. However, the consistency methods (which do not use the Jacobian) provide a small amount of guidance.

In Section 10.8, we argue that in some cases, the effectiveness of hull consistency in reducing a component can be less if another component contains zero. Moreover, we might have implemented hull consistency so that it is more effective for boxes with large values of the variables. Therefore, when splitting a component, we choose to store the portion that has smaller values of the relevant variable. Additional subboxes will be formed using the other portion and they will be subdivided in additional splitting. If the component to be split is symmetric about zero, an arbitrary choice can be made whether to split at a negative or a positive value.

Note that we could choose not to split a component whose width is less than ε_X when there are components of width greater than ε_X. We do not make this distinction. It might be necessary to split a narrow component for the algorithm to reduce a wider one.

We now consider the case in which a gap has been produced in a component, say x_i of a box $\mathbf{x^I}$. That is, we have determined that no solution of the system $\mathbf{f}(\mathbf{x}) = \mathbf{0}$ exists when x_i is in a gap in x_i. We want to split $\mathbf{x^I}$ by deleting the gap and remove such points from consideration.

There are reasons why we might not want to split $\mathbf{x^I}$ using a particular gap. First, there might be gaps in too many components. We have noted that, whether using gaps or not, we sometimes split no more than three components of a box.

Second, using a particular gap to do the splitting might be of little value. For example, suppose the gap is quite narrow and occurs near an endpoint of the component interval of $\mathbf{x^I}$. If we split $\mathbf{x^I}$ using this gap, we merely shave off a thin slice of $\mathbf{x^I}$ as one of the two new subboxes.

Third, the gap might occur in a component of $\mathbf{x^I}$ that is already much narrower than the other interval components of $\mathbf{x^I}$. It is more important to split a component x_i for which T_i (as given by (11.8.1)) is large.

Fourth, more than one gap might have been generated in a given component of $\mathbf{x^I}$. We use only one such gap.

Consider a component of $\mathbf{x^I}$ that contains a gap. Let the component be the interval $[a, d]$ and let the gap be the open interval (b, c). Removing the gap leaves the two subintervals $[a, b]$ and $[c, d]$. We consider this gap to be suitable for use in splitting $\mathbf{x}^\mathbf{I}$ if

$$\min\{c - a, d - b\} \geq 0.2\,\mathrm{w}(\mathbf{x}^\mathbf{I}) \tag{11.8.2}$$

Note that if the width of the gap is $\geq 0.2\,\mathrm{w}(\mathbf{x}^\mathbf{I})$, then (11.8.2) is satisfied. In this case, we are willing to shave off a thin slice of $\mathbf{x}^\mathbf{I}$ because we also delete a relatively wide gap. Any gap we discuss subsequently is assumed to satisfy (11.8.2). If gaps exist that satisfy condition (11.8.2) for use in splitting, their use takes precedence over simply splitting a component of $\mathbf{x}^\mathbf{I}$ at a point. The parameter T_j given by (11.8.1) is used to select the gaps to be used. That is, we do not simply use the widest gaps.

Suppose we are using the procedure given above in this section to split in n dimensions. When a particular component is to be split and it contains a gap suitable for splitting, we split using the gap rather than the computed point. We store the smaller part and use the larger part for splitting as the procedure prescribes. This does not alter the relative sizes of subboxes formed subsequently by the procedure.

When solving an optimization problem using methods given in subsequent chapters, we sometimes use the splitting procedure described above. In this case, the matrix $\mathbf{J}(\mathbf{x}, \mathbf{x}^\mathbf{I})$ whose elements occur in (11.8.1) is the Jacobian of the gradient of an objective function. In some constrained problems, this Jacobian does not involve some of the variables defining the box to be split. In this case, we proceed as follows:

Let S_J denote the set of indices of variables that occur in the definition of $\mathbf{J}(\mathbf{x}, \mathbf{x}^\mathbf{I})$. Let S_0 denote the set of remaining variable indices.

We order all components of the box in order of decreasing width. If the Jacobian has not yet been evaluated, we split the three leading components in the list. If the Jacobian has been evaluated and the three leading components all have indices in the set S_J, we use the above procedure to split all the components with indices in S_J. If the index of at least one of the three leading components in the list is in S_0, we split only the three components leading the list.

If parallel processors are available, each can have a copy of the interval Newton algorithm. The initial box can be split before processing starts so that each processor has a box to process. When any box is split, a resulting subbox can be passed to an available processor. A processor becomes available when it has completed its job by either deleting its box or adding its final boxes to the set of possible solution boxes.

Not only does each processor tend to remain productive, but progress is enhanced because a Newton method is more efficient when applied to the smaller boxes produced by splitting. The presence or lack of available processors can affect the decision of how many new subboxes to generate by splitting.

We have specified that splitting is sometimes done in three dimensions. However, when parallel processors are used and there is some danger of a processors running out of boxes to be processed, it can be desirable to split in more dimensions to provide more new boxes.

11.9 ANALYTIC PRECONDITIONING

We noted in Section 11.2 that we precondition the system

$$\mathbf{J}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})(\mathbf{y} - \mathbf{x}) = -\mathbf{f}\,(\mathbf{x}) \tag{11.9.1}$$

by multiplying by an approximate inverse $\mathbf{B}$ of the center of $\mathbf{J}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$. This tends to make the new coefficient matrix $\mathbf{M}(\mathbf{x}, \mathbf{x}^{\mathbf{I}}) = \mathbf{BJ}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$ diagonally dominant. That is, preconditioning tends to make the i-th equation of the system depend strongly on the i-th variable and weakly on the other variables. However, if $\mathbf{x}^{\mathbf{I}}$ is a large box, the interval elements of $\mathbf{J}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$ are generally wide and $\mathbf{M}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$ need not be diagonally dominant.

If the preconditioned linear system is diagonally dominant, then the i-th equation ($i = 1, \cdots, n$) of the preconditioned nonlinear system $\mathbf{Bf}\,(\mathbf{x})$ depends strongly on the i-th variable and weakly on the other variables. We use this fact by analytically forming this system and solving the i-th analytic function $[\mathbf{Bf}\,(\mathbf{x})]_i$ for the i-th variable. When we analytically precondition $\mathbf{f}\,(\mathbf{x})$ we can reduce the effect of dependence by combining and cancelling terms.

We can write a component

$$[\mathbf{Bf}\,(\mathbf{x})]_i = \mathbf{B}_{i1} f_1\,(\mathbf{x}) + \cdots + \mathbf{B}_{in} f_n\,(\mathbf{x})$$

and/or its derivative in the best form to reduce dependence in its evaluation with interval arguments. When a particular matrix $\mathbf{B}$ is determined, its numerical elements can be substituted into this expression.

Whenever we apply a Newton step in our algorithm, we compute a preconditioning matrix $\mathbf{B}$. We use this matrix to determine $\mathbf{Bf}(\mathbf{x})$. We then apply hull and box consistencies to this preconditioned system. When doing so, we solve the i-th ($i = 1, \cdots, n$) equation for the i-th variable only.

Hansen (1997a) describes a procedure in which the analytically preconditioned function $\mathbf{Bf}(\mathbf{x})$ is expanded to obtain the matrix $\mathbf{M}(\mathbf{x}, \mathbf{x}^{\mathbf{I}}) = \mathbf{BJ}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$. This generally yields narrower interval elements of $\mathbf{M}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$ than computing $\mathbf{J}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$ and numerically multiplying by $\mathbf{B}$. This analytic preconditioning is of value if Gaussian elimination or a Gauss-Seidel step is used to solve the linearized equations.

However, we prefer the hull method of Section 5.8 over Gaussian elimination. When the hull method is used, the center of the coefficient matrix must be the identity matrix or else the interval matrix elements must be widened to achieve this condition. Unfortunately, if analytic preconditioning is used, the center is not the identity matrix. Widening the matrix elements might undo the gain obtained by analytic preconditioning. Experiments are needed to determine if it is desirable to use analytic preconditioning in conjunction with the hull method.

It is possible to use the following procedure. Compute $\mathbf{B}$ as indicated above and analytically determine $\mathbf{Bf}$. Evaluate the Jacobian of $\mathbf{Bf}$ and compute an approximate inverse $\mathbf{B}'$ of its center. Use $\mathbf{B}'$ to precondition the Jacobian of $\mathbf{Bf}$. This produces a coefficient matrix whose center is the identity matrix (if exact arithmetic is used); so the hull method can be used effectively. This involves computing an extra Jacobian and an extra preconditioner $\mathbf{B}'$. The extra computing might not be warranted.

Note that analytic preconditioning involves non-numerical computing that might not be readily available to a user. Its use can be bypassed with little effect on the performance of our algorithm. So that it is clear how to proceed without analytic preconditioning, we have inserted a note in the algorithm steps in Section 11.13 (and in other algorithms in later chapters).

Frequently, there are cases in which use of analytic preconditioning is not warranted. Consider the case in which each f_i $(i = 1, \cdots, n)$ is a function of only a few of the n variables. Each function resulting from analytic preconditioning generally involves all n variables. This increased function complexity reduces their utility. Thus, analytic preconditioning is of value if each initial function involves all or most of the variables. It is of diminishing value as there is a reduction of the number of variables upon which each function depends

11.9.1 An alternative method

We now consider an alternative to the procedure described above in this section. We believe this alternative to be preferable; but we do not have enough practical experience to decide if this is true.

We introduce the procedure as follows. Recall that the preconditioning matrix $\mathbf{B}$ is an approximate inverse of the center $\mathbf{J}^c$ of $\mathbf{J}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$. That is $\mathbf{B}\mathbf{J}^c = \mathbf{I}$. Using this fact, we can condition the system (11.9.1) by rewriting it as

$$\mathbf{M}'(\mathbf{x}, \mathbf{x}^{\mathbf{I}})\mathbf{z} = -\mathbf{f}\,(\mathbf{x}) \tag{11.9.2}$$

where $\mathbf{M}'(\mathbf{x}, \mathbf{x}^{\mathbf{I}}) = \mathbf{J}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})\mathbf{B}$ and $\mathbf{z} = \mathbf{J}^c(\mathbf{y} - \mathbf{x})$. To distinguish this result from the preconditioned case, we shall say that $\mathbf{M}'(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$ is a *postconditioned* form of $\mathbf{J}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$. To solve this new system, we first solve for $\mathbf{z}$ and then use the relation

$$\mathbf{y} = \mathbf{x} + \mathbf{B}\mathbf{z}. \tag{11.9.3}$$

This procedure does not seem to be as effective in practice as using preconditioning. However, the conditioning in (11.9.2)is done numerically. In the procedure described below, it is done analytically.

We now consider an analytically conditioned form of this procedure. Recall that (11.9.1) is obtained from

$$\mathbf{f}\,(\mathbf{y}) = \mathbf{f}\,(\mathbf{x}) + \mathbf{J}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})(\mathbf{y} - \mathbf{x})$$

by assuming that $\mathbf{y}$ is a solution of $\mathbf{f}\,(\mathbf{y}) = \mathbf{0}$. Using (11.9.3), this relation is

$$\mathbf{f}(\mathbf{x} + \mathbf{B}\mathbf{z}) = \mathbf{0}. \tag{11.9.4}$$

When $\mathbf{x}^I$ is a small box containing a solution of $\mathbf{f} = \mathbf{0}$, postconditioning causes the i-th component of equation (11.9.2) to depend strongly on z_i and weakly on the other components of $\mathbf{z}$. Since this is true for the linearized form (11.9.2), it is true for the original form (11.9.4). Therefore, we solve the i-th component of (11.9.4) for z_i $(i = 1, \cdots, n)$.

To solve, (11.9.4), we want initial bounds on $\mathbf{z}$. They can be obtained from (11.9.3) as $\mathbf{z}^I = \mathbf{B}(\mathbf{x}^I - \mathbf{x})$. After new bounds $\mathbf{z}^{I'}$ are obtained, the corresponding new bounds $\mathbf{x}^{I'}$ are obtained as $\mathbf{x}^{I'} = \mathbf{x} + \mathbf{J}^c\mathbf{z}^{I'}$. The steps of getting $\mathbf{z}^I$ from $\mathbf{x}^I$ and $\mathbf{x}^{I'}$ from $\mathbf{z}^{I'}$ each enlarge the bounding box because of the "wrapping effect". See Moore (1979) for a discussion of the wrapping effect.

Earlier in this section, we considered the preconditioned equation $\mathbf{Bf} = \mathbf{0}$. Note that each component of this vector equation is a linear combination of all the components of $\mathbf{f}$ and hence is generally quite complicated. However, a component of (11.9.4) is a single equation but involves every component of $\mathbf{z}$. In the latter case, each component is likely to be much simpler than in the former case. Therefore, the alternative method has simpler equations to be solved but its effectiveness is reduced by the wrapping effect.

11.10 THE INITIAL BOX

We have noted that the search for solutions to the system of nonlinear equations is confined to a box $\mathbf{x}^{I(0)}$. Generally, this box is specified by the user. Actually the region of search can be a set of boxes. The boxes can be disjoint or overlap. However, if they overlap, a solution at a point that is common to more than one box is separately found in each box containing it. In this case, computing effort is wasted.

If the user does not specify an initial box (or boxes), we use a "default box". Let $\overline{\mathrm{N}}$ denote the largest floating point number that can be represented in the number system used on the computer. Our default box has components $x_i^{(0)} = \left[-\overline{\mathrm{N}}, \overline{\mathrm{N}}\right]$ for all $i = 1, \cdots, n$. We assume that any finite solution outside the default box is of no interest. To find any solution outside the default box requires higher precision arithmetic.

Since we use extended interval arithmetic, it is possible to let the initial box have components $[-\infty, +\infty]$. However, we assume that any solution at $+\infty$ or $-\infty$ is uninteresting. Therefore finding one is a waste of effort.

A user can usually specify a smaller box than the default. The smaller the initial box, the faster the algorithm can solve the problem.

11.11 A LINEARIZATION TEST

In an interval Newton method, we linearize a nonlinear system of equations and solve the linear system. The linear system provides a good approximation to the nonlinear system over a box when the box is small. However, the linear system is generally a poor approximation when the box is large.

Therefore, when solving a system of nonlinear equations, it is generally more efficient to use other procedures or to split a box when the box is large. In our algorithm, we defer use of a Newton method until other procedures have reduced the size of a large box.

To determine whether a box is "large", we use a test. We use similar tests later in the book. In each case, we test whether certain functions should be linearized over a given box. We call our tests "linearization tests".

Assume we have applied a Newton method to a box $\mathbf{x^I}$; and assume it uses (if applicable) the hull method of Section 5.8 to solve the linear system. In so doing, we learn whether the preconditioned coefficient matrix $\mathbf{M^I}\left(\mathbf{x}, \mathbf{x^I}\right)$ of the linearized (and preconditioned) system is regular. This determines whether the hull method is applicable. If $\mathbf{M^I}\left(\mathbf{x}, \mathbf{x^I}\right)$ is irregular, we regard the box $\mathbf{x^I}$ as "too large".

When solving a system of nonlinear equations, we often attempt to apply a Newton step in this way. When a Newton step is applied to a box $\mathbf{x^I}$, $\mathbf{M^I}\left(\mathbf{x}, \mathbf{x^I}\right)$ might be regular or irregular. We let w_R denote the width of the largest box for which $\mathbf{M^I}\left(\mathbf{x}, \mathbf{x^I}\right)$ has been found to be regular. We let w_I denote the width of the smallest box for which $\mathbf{M^I}\left(\mathbf{x}, \mathbf{x^I}\right)$ has been found to be irregular.

In our algorithm to solve a system of nonlinear equations (see Section 11.12), we apply the Newton method to a box $\mathbf{x}^{\mathbf{I}}$ only if

$$\mathrm{w}(\mathbf{x}^{\mathbf{I}}) \leq \frac{1}{2}(w_R + w_I). \tag{11.11.1}$$

Before beginning to solve a given system in a box $\mathbf{x}^{\mathbf{I}(0)}$, we initialize by setting $w_R = 0$ and $w_I = \mathrm{w}(\mathbf{x}^{\mathbf{I}(0)})$. As the algorithm progresses and w_R and w_I change, it might happen that w_R becomes larger than w_I. This does not change our procedure. Note that (11.11.1) is not satisfied for the initial box. Therefore, other procedures must reduce the width of the initial box by at least a half before a Newton step is applied. If the initial box is already "small", this condition can be overridden.

If, in a given application of a Newton method, $\mathbf{M}^{\mathbf{I}}\left(\mathbf{x}, \mathbf{x}^{\mathbf{I}}\right)$ is irregular, we can apply the Gauss-Seidel method to the preconditioned linear system. If a Gauss-Seidel step sufficiently reduces (as defined using (11.7.4)) the box to which it is applied, we regard the box as "small" just as if $\mathbf{M}^{\mathbf{I}}\left(\mathbf{x}, \mathbf{x}^{\mathbf{I}}\right)$ were regular. We update w_R accordingly.

11.12 THE ALGORITHM STEPS

In this section, we list the steps of our interval Newton method. The algorithm contains the procedures described earlier. Note that we refer to it simply as a Newton method. However, it involves application of hull and box consistencies and two forms of Newton methods.

The selected features of the algorithm were chosen using both experimentation and theoretical considerations. There is undoubtedly room for improvement. In particular our selection of various numerical parameters was often made with too little experimental work to obtain the best values. For example, it might be better to let some parameters vary with dimensionality. Despite its shortcomings, the algorithm works well in practice.

We assume that an initial box $\mathbf{x}^{\mathbf{I}(0)}$ is given. We seek all zeros of $\mathbf{f}$ in this box. However, as discussed in Section 11.10, more than one box can be given. As the algorithm proceeds, it usually generates various subboxes of $\mathbf{x}^{\mathbf{I}(0)}$. These subboxes are stored in a list L waiting to be processed. At any given time, the list can be empty or contain several (or many) boxes.

The algorithm initially sets $w_R = 0$ and $w_I = \mathrm{w}(\mathbf{x}^{\mathbf{I}(0)})$. See Section 11.11. If more than one box is input, w_I is set equal to the maximum box width.

The steps of the algorithm are to be performed in the order given below except as indicated by branching. The current box is denoted by $\mathbf{x}^{\mathbf{I}}$ even though it changes from step to step. We assume the tolerances ε_X and ε_f discussed in Section 11.5 are given by the user.

0. Put the initial box(es) in the list L.

1. If the list L is empty, stop. Otherwise, choose the box most recently put in L to be the current box. Delete it from L.

2. For future reference, store a copy of the current box $\mathbf{x}^{\mathbf{I}}$. Call the copy $\mathbf{x}^{\mathbf{I}(1)}$. If hull consistency has been applied n times in succession without applying a Newton step, go to Step 9. (In making this count, ignore any applications of box consistency.) Otherwise, apply hull consistency to the equation $\mathbf{f}_i\,(\mathbf{x}) = \mathbf{0}$ $(i = 1, \cdots, n)$ for each variable x_j $(j = 1, \cdots, n)$. To do so, cycle through equations and variables as described at the end of Section 10.2. Use more general hull consistency methods if desired. See Section 10.5. If the result is empty, go to Step 1.

3. If $\mathbf{x}^{\mathbf{I}}$ satisfies both **Criteria A** and **B** (see (11.5.1) and (11.5.2)), record $\mathbf{x}^{\mathbf{I}}$ as a solution and go to Step 1.

4. If the box $\mathbf{x}^{\mathbf{I}(1)}$ (see Step 2) was sufficiently reduced (as defined using (11.7.4)) in Step 2, repeat Step 2.

5. Using the algorithm in Section 10.2, apply box consistency to each of the equations $\mathbf{f}_i\,(\mathbf{x}) = \mathbf{0}$ $(i = 1, \cdots, n)$ for each of the variables x_j, $(j = 1, \cdots, n)$. If the result is empty, go to Step 1.

6. Repeat Step 3.

7. If the current box $\mathbf{x^I}$ is a sufficiently reduced (as defined using (11.7.4)) version of the box $\mathbf{x}^{\mathbf{I}(1)}$ defined in Step 2, got to Step 2.

8. If $\mathrm{w}(\mathbf{x^I}) > \frac{1}{2}(w_R + w_I)$, go to Step 28.

9. If $\mathbf{x^I}$ is contained in a box for which the matrix $\mathbf{B}$ was saved in Step 19, use $\mathbf{B}$ to compute a point $\mathbf{x}^B$ as described in Section 11.4.1. (The procedure to do so uses an iterative version of $\mathbf{y} = \mathbf{x} - \mathbf{B}f(\mathbf{x})$). Then set $\mathbf{x} = \mathbf{x}^B$ for use in Step 10. Otherwise, set $\mathbf{x}$ equal to the approximate center of $\mathbf{x^I}$.

10. For later reference, denote the current box by $\mathbf{x}^{\mathbf{I}(2)}$. Compute $\mathbf{J}(\mathbf{x}, \mathbf{x^I})$ using a Taylor expansion based on (7.3.6) or else using slopes (see (7.9.1)). Use the point determined in Step 9 as the point of expansion. Compute an approximation $\mathbf{J}^c$ for the center of $\mathbf{J}(\mathbf{x}, \mathbf{x^I})$. Compute an approximate inverse $\mathbf{B}$ of $\mathbf{J}^c$. If $\mathbf{J}^c$ is singular, compute $\mathbf{B}$ as described in Section 5.11. Compute $\mathbf{M}(\mathbf{x}, \mathbf{x^I}) = \mathbf{BJ}(\mathbf{x}, \mathbf{x^I})$ and $\mathbf{r}(\mathbf{x}) = -\mathbf{Bf}(\mathbf{x})$. If $\underline{M}_{ii}(\mathbf{x}, \mathbf{x^I}) \leq 0$ for any $i = 1, \cdots, n$, then $\mathbf{M}(\mathbf{x}, \mathbf{x^I})$ is irregular; so update w_I as described in Section 11.11 and go to Step 12.

11. Compute $\mathbf{P^I} = [\underline{\mathbf{M}}(\mathbf{x}, \mathbf{x^I})]^{-1}$. If $\mathbf{P^I} \geq \mathbf{I}$, then $\mathbf{M}(\mathbf{x}, \mathbf{x^I})$ is regular (see Theorem 5.8.1). Otherwise, $\mathbf{M}(\mathbf{x}, \mathbf{x^I})$ is irregular. If $\mathbf{M}(\mathbf{x}, \mathbf{x^I})$ is regular, update w_R as described in Section 11.11 and go to Step 16. If $\mathbf{M}(\mathbf{x}, \mathbf{x^I})$ is irregular, update w_I and go to Step 12.

12. If every diagonal element of $\mathbf{M}(\mathbf{x}, \mathbf{x^I})$ contains zero, go to Step 14. Otherwise, apply one pass of the Gauss-Seidel method (11.2.7) to "solve" $\mathbf{M}(\mathbf{x}, \mathbf{x^I})(\mathbf{y} - \mathbf{x}) = \mathbf{r}(\mathbf{x})$. If the result is empty, go to Step 1.

13. Repeat Step 3.

14. If the box was sufficiently reduced (as defined using (11.7.4)) by the single pass of the Gauss-Seidel method of Step 12, update w_R as if $\mathbf{M}(\mathbf{x}, \mathbf{x^I})$ were regular for the box to which the Gauss-Seidel method was applied in Step 12 and return to Step 12. If the box was not sufficiently reduced in Step 12, go to Step 15.

15. If the current box is a sufficiently reduced (as defined using (11.7.4)) version of the box $\mathbf{x}^{\mathbf{I}(1)}$ defined in Step 2, put $\mathbf{x}$ in list L and go to Step 1. Otherwise, go to Step 28.

16. Use the hull method (see Section 5.8.2) to solve $\mathbf{M}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})(\mathbf{y} - \mathbf{x}) = \mathbf{r}(\mathbf{x})$. If the result is empty, go to Step 1.

17. Repeat Step 3.

18. If **Criterion C** of Section 11.5 is satisfied, record $\mathbf{x}^{\mathbf{I}}$ as a solution and go to Step 1.

19. Record the fact that the matrix **B** computed the last time Step 10 was applied is to be used whenever Step 9 is applied to any subbox of $\mathbf{x}^{\mathbf{I}}$.

20. If $\mathrm{w}(\mathbf{x}^{\mathbf{I}}) \leq \frac{1}{8}\,\mathrm{w}(\mathbf{x}^{\mathbf{I}(2)})$ (note $\mathbf{x}^{\mathbf{I}(2)}$ was defined in Step 9), go to Step 9.

21. **Note:** The user might wish to bypass analytic preconditioning. See the comment in Section 11.9. If so, go to Step 25.

    Additional note: This step as well as Steps 22 and Step 25 are written for the case in which the first method of analytic preconditioning described in Section 11.9 is used. If the alternative method in Section 11.9.1 is used, these steps must be altered accordingly. In either case, determine the analytically preconditioned function $\mathbf{Bf}^{\mathbf{I}}(\mathbf{x})$ as described in Section 11.9.

22. If hull consistency has been applied n times to the analytically preconditioned equation $\mathbf{Bf}(\mathbf{x}) = \mathbf{0}$ without changing **B**, go to Step 25. Otherwise, apply hull consistency to solve the i-th equation of $\mathbf{Bf}(\mathbf{x}) = \mathbf{0}$ to bound x_i for $i = 1, \cdots, n$. If the result is empty, go to Step 1.

23. Repeat Step 3.

24. If the box $\mathbf{x}^{\mathbf{I}}$ is sufficiently reduced (see (11.7.4)) in Step 16, go again to Step 22.

25. Apply box consistency to solve the i-th equation of the analytically preconditioned system $\mathbf{Bf}(\mathbf{x}) = \mathbf{0}$ to bound x_i for $i = 1, \cdots, n$. If the result is empty, go to Step 1.

26. Repeat Step 3.

27. If $\mathbf{x}^{\mathbf{I}}$ is a sufficiently reduced (see (11.7.4)) version of the box $\mathbf{x}^{\mathbf{I}(1)}$ defined in Step 2, put $\mathbf{x}^{\mathbf{I}}$ and $\mathbf{x}^{B}$ in L, and go to Step 1.

28. Split $\mathbf{x}^{\mathbf{I}}$ as described in Section 11.8. Put the boxes generated by splitting in list L. Then go to Step 1.

After termination (in Step 1), bounds on all solutions of $\mathbf{f}(\mathbf{x})$ in the initial box $\mathbf{x}^{\mathbf{I}(0)}$ have been recorded. A bounding box $\mathbf{x}^{\mathbf{I}}$ recorded in Step 3 satisfies the conditions $w(\mathbf{x}^{\mathbf{I}}) \leq \varepsilon_X$ and $||\mathbf{f}(\mathbf{x}^{\mathbf{I}})|| \leq \varepsilon_f$ specified by the user. A box $\mathbf{x}^{\mathbf{I}}$ recorded in Step 14 approximates the best possible bounds that can be computed with the number system used.

11.13 DISCUSSION OF THE ALGORITHM

In this section, we discuss why certain steps of the algorithm in Section 11.12 are as given.

In Step 1, we select the box for processing that has been in the list the shortest length of time. This tends to select the smallest box without bothering to determine box sizes. This, in turn, tends to keep the length of the list short. If, for example, we processed the largest box, it is more likely to be split and this adds boxes to the list.

After an interval Newton method has been applied, we have computed a Jacobian over a box containing $\mathbf{x}^{\mathbf{I}}$ and a matrix $\mathbf{B}$ which is the approximate inverse of the center of the Jacobian. If the resulting box $\mathbf{x}^{\mathbf{I}}$ is split we could use $\mathbf{B}$ to try to obtain an approximate solution to $\mathbf{f}(\mathbf{x})$ (as described in Section 11.4.1) for each of the new boxes generated by the splitting. However, recall that we split only if the Newton step fails to make sufficient progress. This lack of progress tends to indicate that $\mathbf{B}$ is not useful in reducing $||\mathbf{f}(\mathbf{x})||$. Therefore, we do not use $\mathbf{B}$ for this purpose in any of the generated subboxes. A user might wish to insert such a procedure.

If the matrix $\mathbf{M}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$ used in Step 16 is regular, then the hull method produces the hull of the solution set of

$$\mathbf{M}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})(\mathbf{y} - \mathbf{x}) = \mathbf{r}^{\mathbf{I}}(\mathbf{x}),$$

where $\mathbf{M}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$ and $\mathbf{r}^{\mathbf{I}}(\mathbf{x})$ are defined in (11.2.5). In this case, a repeat of Step 16 cannot improve the result without recomputing this matrix. However, if $\mathbf{M}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$ is irregular, a Gauss-Seidel step is used and it might be possible to compute a sharper solution by repeating Step 12 using the same matrix. This explains Step 14.

When the Newton method exhibits rapid convergence, there is little point in using other methods. Therefore, in Step 20, we bypass hull consistency and box consistency when the Newton method suffices.

11.14 ONE NEWTON STEP

We use an interval Newton method as part of an algorithm to solve a global optimization problem. For example, see Section 12.8. When we do so, we do not iterate the interval Newton algorithm to convergence. The reason is that it might be converging to a point that another procedure can show is not the global solution.

Therefore, we perform one "pass" of an interval Newton algorithm and then apply other procedures before doing another pass. We describe such a one-pass algorithm in this section.

This "one Newton step" algorithm does not use consistency methods because, these procedures are implemented differently in our optimization algorithms. Various steps of the algorithm of Section 11.12 are omitted because they occur in the main optimization algorithm. In particular, this special algorithm does not check for progress or convergence and does not split boxes.

When we refer to one pass of the interval Newton method, we mean the following:

1. Compute $\mathbf{J}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$ using a Taylor expansion based on (7.3.6) or else using slopes (see 7.9.1)). Compute an approximation $\mathbf{J}^c$ for the center of $\mathbf{J}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$. Compute an approximate inverse $\mathbf{B}$ of $\mathbf{J}^c$. If

$\mathbf{J}^c$ is singular, compute $\mathbf{B}$ as described in Section 5.11. Compute $\mathbf{M}(\mathbf{x}, \mathbf{x}^{\mathbf{I}}) = \mathbf{BJ}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$ and $\mathbf{r}(\mathbf{x}) = -\mathbf{Bf}(\mathbf{x})$. If $\underline{\mathbf{M}}_{ii}(\mathbf{x}, \mathbf{x}^{\mathbf{I}}) \leq 0$ for any $i = 1, \cdots, n$, then $\mathbf{M}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$ is irregular; so update w_I as described in Section 11.11 and go to Step 3.

2. Compute $\mathbf{P} = [\underline{\mathbf{M}}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})]^{-1}$. If $\mathbf{P} \geq \mathbf{I}$, then $\mathbf{M}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$ if regular (see Theorem 5.8.1). Otherwise, $\mathbf{M}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$ is irregular. If $\mathbf{M}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$ is regular, update w_R as described in Section 11.11 and go to Step 5. If $\mathbf{M}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$ is irregular, update w_I and go to Step 3.

3. If every diagonal element of $\mathbf{M}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$ contains zero, return to the main program. Otherwise, apply one pass of the Gauss-Seidel method (11.2.7) to "solve" $\mathbf{M}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})(\mathbf{y} - \mathbf{x}) = \mathbf{r}(\mathbf{x})$. If the result is empty, record this fact for use in the main program; and return to the main program.

4. If the box was sufficiently reduced (as defined using (11.7.4)) by the single pass of the Gauss-Seidel method of Step 3, update w_R as if $\mathbf{M}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$ were regular for the box to which the Gauss-Seidel method was applied in Step 3 and return to step 3. If the box was not sufficiently reduced in Step 3, record the final box for use in the main program; and return to the main program.

5. Use the hull method (see Section 5.8.2) to solve $\mathbf{M}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})(\mathbf{y} - \mathbf{x}) = \mathbf{r}(\mathbf{x})$. Record the result for use in the main program; and return to the main program.

No matter which method is used to solve the preconditioned system, it is good practice to follow with the inner iteration of Section 11.4.1 to obtain a point of expansion for a subsequent Newton step.

11.15 PROPERTIES OF INTERVAL NEWTON METHODS

Interval Newton methods for multidimensional problems have most of the properties listed in Section 9.6 for the corresponding one-dimensional algorithm. In this section, we discuss a possible exception and then state some relevant theorems. In practice, we combine use of hull and box consistencies with use of an interval Newton method. In this section on theorems we concentrate on Newton methods and assume no consistency methods are used.

Global convergence is a subject of interest in noninterval algorithms for finding the zeros of systems of nonlinear equations. Will the algorithm always converge from any initial point to some solution if one exists? In the interval case, we can bypass this question and ask the more important one: Will the algorithm always find *all* solutions in a given box.

Proof exists in the interval case for local convergence under certain conditions (see below). However, the authors know of no proof that all solutions in a given box are always found. Nevertheless, there is little doubt that an algorithm such as the one given in Section 11.12 always does so when the number of these solutions is finite. It is difficult to conceive how the truth can be otherwise. The authors are unaware of any failure in practice.

Of course, it is easy to formulate an example in which there are so many isolated solutions that it is impractical to find them all. It is obviously impossible to find all solutions when there are infinitely many of them, each of which is isolated from the others. However, for appropriate problems, interval methods can find and bound an infinite set of solutions consisting of a continuum of points. See Section 11.17.

We now consider proven properties of interval Newton methods. We begin with a theorem due to Moore (1966).

Theorem 11.15.1 If there exists a zero $\mathbf{x}^*$ of $\mathbf{f}$ in $\mathbf{x}^\mathbf{I}$, then $\mathbf{x}^* \in \mathrm{N}(\mathbf{x}, \mathbf{x}^\mathbf{I})$.

In this theorem, $\mathrm{N}\left(\mathbf{x}, \mathbf{x}^\mathbf{I}\right)$ is the box obtained (using exact interval arithmetic) by a particular interval Newton method that bounds the solution set S_y of all vectors $\mathbf{y}$ that satisfy $\mathbf{J}\left(\mathbf{x}, \mathbf{x}^\mathbf{I}\right)(\mathbf{y}-\mathbf{x}) = -\mathbf{f}^\mathbf{I}(\mathbf{x})$. See Section 11.2. The conclusion of this theorem is the motivating idea in the derivation of an interval Newton method. Examining the derivation in Section 11.2 shows that the theorem is correct.

In practice, when rounding occurs, we compute a box containing $\mathrm{N}(\mathbf{x}, \mathbf{x}^\mathbf{I})$. Hence, even with rounding, we never "lose" a zero of $\mathbf{f}$.

The following theorem is also due to Moore (1966).

Theorem 11.15.2 If $\mathbf{x}^{\mathbf{I}} \cap \mathrm{N}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$ is empty, then there is no zero of $\mathbf{f}$ in $\mathbf{x}^{\mathbf{I}}$.

Proof. If there is a zero of $\mathbf{f}$ in $\mathbf{x}^{\mathbf{I}}$, it must also be in $\mathrm{N}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$ by Theorem 11.15.1. Hence, if $\mathbf{x}^{\mathbf{I}} \cap \mathrm{N}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$ is empty, there can be no zero of $\mathbf{f}$ in $\mathbf{x}^{\mathbf{I}}$. ■

If rounding occurs, we compute a box $\mathrm{N}'(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$ containing $\mathrm{N}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$. If $\mathbf{x}^{\mathbf{I}} \cap \mathrm{N}'(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$ is empty, then $\mathbf{x}^{\mathbf{I}} \cap \mathrm{N}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$ is empty. Therefore, the theorem is applicable even in the presence of rounding.

Proofs of convergence have been given for various interval Newton methods. Theorem 11.15.3 below is for the method in which equation (11.2.2) is solved by Gaussian elimination. A more general form was proved by Alefeld (1984). See also Alefeld (1999). Krawczyk (1983) proved convergence for his method. See (11.2.8). Alefeld (1979) proved convergence for the Hansen-Sengupta version (11.2.7) of the Gauss-Seidel method. For a thorough discussion of convergence of various interval Newton methods, see Neumaier (1990).

We know of no explicit proof of convergence when the hull method of Section 5.8 is used. Note that it produces the exact hull of the preconditioned linear system

$$\mathbf{M}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})(\mathbf{y} - \mathbf{x}) = \mathbf{r}^{\mathbf{I}}(\mathbf{x}) \tag{11.15.1}$$

(see (11.2.6a) when $\mathbf{M}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$ is regular. Therefore, it produces a solution at least as sharp as when using the Gauss-Seidel method to "solve" this system. Therefore if $\mathbf{M}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$ is regular, any proof of convergence using Gauss-Seidel also proves convergence when using the hull method.

To state Alefeld's convergence theorem, we must introduce some notation relevant to Gaussian elimination. Consider a system of interval linear equations $\mathbf{A}^{\mathbf{I}}\mathbf{x} = \mathbf{b}^{\mathbf{I}}$. Suppose we are successful in trying to solve this system by Gaussian elimination. Using Alefeld's notation, we denote the computed solution by $\mathrm{IGA}(\mathbf{A}^{\mathbf{I}}, \mathbf{b}^{\mathbf{I}})$ where IGA stands for interval Gauss algorithm.

Note that success or failure when solving the system depends on the coefficient matrix $\mathbf{A}^{\mathbf{I}}$ but not on the right hand side vector $\mathbf{b}^{\mathbf{I}}$.

For the interval Newton method we are considering, we wish to solve (11.9.1) with preconditioning. Therefore, $\mathbf{A}^{\mathbf{I}} = \mathbf{J}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$ and $\mathbf{b}^{\mathbf{I}} = -\mathbf{f}^{\mathbf{I}}(\mathbf{x})$.

The order of the matrix $\mathbf{J}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$ is n. For theoretical reasons, we consider the case in which we solve $\mathbf{J}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})\mathbf{z} = \mathbf{b}^{(i)}$ for $i = 1, \cdots, n$ with $\mathbf{b}^{(i)}$ equal to the i-th column of $\mathbf{J}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$. We place the i-th solution vector as the i-th column of a matrix that we denote by $\text{IGA}\left[\mathbf{J}(\mathbf{x}, \mathbf{x}^{\mathbf{I}}), \mathbf{J}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})\right]$.

In a real sense, this matrix is the solution, say $\mathbf{z}^{\mathbf{I}}$, to the equation $\mathbf{J}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})\mathbf{z}^{\mathbf{I}} = \mathbf{J}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$. Therefore, $\text{IGA}\left[\mathbf{J}(\mathbf{x}, \mathbf{x}^{\mathbf{I}}), \mathbf{J}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})\right]$ approximates the identity matrix. It is the difference of this matrix from the identity that we use to determine whether the interval Newton method converges.

In Theorem 11.15.3 given below, we use the following notation. Let $\mathbf{A}^{\mathbf{I}}$ be an interval matrix. The real matrix $|\mathbf{A}^{\mathbf{I}}|$ is the matrix derived from $\mathbf{A}^{\mathbf{I}}$ by replacing each element by its magnitude.

Theorem 11.15.3 Let $\mathbf{f}$ be continuously differentiable. Let $\mathbf{x}^{\mathbf{I}(0)}$ be the initial box in the interval Newton algorithm given by (11.2.4) and assume $\mathbf{x}^{(0)} \in \mathbf{x}^{\mathbf{I}(0)}$. Assume Gaussian elimination can be completed successfully for the coefficient matrix $\mathbf{J}(\mathbf{x}^{(0)}, \mathbf{x}^{\mathbf{I}(0)})$. Assume $\rho(\mathbf{z}^{\mathbf{I}}) < 1$ where ρ is the spectral radius and

$$\mathbf{z}^{\mathbf{I}} = \left|\mathbf{I} - \text{IGA}\left[\mathbf{J}(\mathbf{x}^{(0)}, \mathbf{x}^{\mathbf{I}(0)}), \mathbf{J}(\mathbf{x}^{(0)}, \mathbf{x}^{\mathbf{I}(0)})\right]\right|.$$

Then the sequence defined by (11.2.4) is well defined. If $\mathbf{f}$ has a zero $\mathbf{x}^*$ in $\mathbf{x}^{\mathbf{I}(0)}$, then the limit of $\mathbf{x}^{\mathbf{I}(k)}$ as $k \to \infty$ is $\mathbf{x}^*$. If $\mathbf{f}$ does not have a zero in $\mathbf{x}^{\mathbf{I}(0)}$, then there is an integer $k' \geq 0$ such that $\text{N}(\mathbf{x}^{(k')}, \mathbf{x}^{\mathbf{I}(k')}) \cap \mathbf{x}^{\mathbf{I}(k')}$ is empty. This proves there is no zero of $\mathbf{f}$ in $\mathbf{x}^{\mathbf{I}(0)}$.

See Alefeld (1984) for a proof of this theorem. See also Alefeld (1999). In our algorithm in section 11.12, we always precondition the system (11.9.1). Alefeld's proof holds whether the system is preconditioned or not.

The following theorem shows that, under certain conditions, the volume of the current box is reduced by at least half in a step of the interval Newton method using Gaussian elimination.

Theorem 11.15.4 Let the hypothesis of Theorem 11.15.3 be satisfied. Assume that $\mathrm{m}(\mathbf{x}^{\mathrm{I}(k)})$ is not a zero of $\mathbf{f}$. If there exists a zero of $\mathbf{f}$ in $\mathbf{x}^{\mathrm{I}(0)}$, then $\mathrm{m}(\mathbf{x}^{\mathrm{I}(k)}) \notin \mathbf{x}^{\mathrm{I}(k+1)}$.

Any box that does not contain the center of $\mathbf{x}^{\mathrm{I}(k)}$ must intersect less than half of $\mathbf{x}^{\mathrm{I}(0)}$. Therefore, the theorem states, in effect, that the volume of $\mathbf{x}^{\mathrm{I}(k+1)}$ is less than half that of $\mathbf{x}^{\mathrm{I}(k)}$ for all k.

This theorem is proved by Alefeld (1984). Compare Corollary 5.2.9 of Neumaier (1990).

To discuss how interval Newton methods can prove the existence of a solution, we introduce the following proposition.

Proposition 11.15.5 If $\mathrm{N}(\mathbf{x}, \mathbf{x}^{\mathrm{I}}) \subset \mathbf{x}^{\mathrm{I}}$, then there exists a zero of $\mathbf{f}$ in $\mathrm{N}(\mathbf{x}, \mathbf{x}^{\mathrm{I}})$.

This proposition has been proved for various interval Newton methods. Recall that interval Newton methods differ in how the bound $\mathrm{N}(\mathbf{x}, \mathbf{x}^{\mathrm{I}})$ is computed from (11.2.6a). Each proof of Proposition 11.15.5 has been for a method using a specific procedure for computing $\mathrm{N}(\mathbf{x}, \mathbf{x}^{\mathrm{I}})$. The first proof of Proposition 11.15.5 was by Kahan (1968) for a method he derived and which is now called Krawczyk's method after its independent derivation by Krawczyk (1969). See (11.2.8). Proof for this method can also be found in Moore (1977, 1979). See also Alefeld (1999).

Proofs for various methods can be found in Alefeld (1999), Krawczyk (1986), Nickel (1971), Qi(1982), Shearer and Wolfe (1985a), and especially Neumaier (1990).

When the Gaussian elimination or the Gauss-Seidel method is used to compute $\mathrm{N}(\mathbf{x}, \mathbf{x}^{\mathrm{I}})$, an intersection process is used to compute each new component. See (11.2.6b). Therefore, we *always* have $\mathbf{x}^{\mathrm{I}(k+1)} \subset \mathbf{x}^{\mathrm{I}(k)}$. To invoke Proposition 11.15.5, we must assume that the box $\mathrm{N}(\mathbf{x}, \mathbf{x}^{\mathrm{I}})$ is the same one that results if no intersecting is done.

Alternatively, authors usually impose the slightly stronger condition that $\mathbf{x}^{\mathrm{I}(k+1)}$ be strictly in the interior of $\mathbf{x}^{\mathrm{I}(k)}$. Thus, we state a relevant theorem as follows.

Theorem 11.15.6 Let $\mathbf{f}$ be continuously differentiable in $\mathbf{x}^{\mathbf{I}}$ and let $\mathbf{x}$ be an interior point of $\mathbf{x}^{\mathbf{I}}$. Assume $\mathrm{N}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$ is computed from (11.2.6a) using Gaussian elimination or a Gauss-Seidel step. If $\mathrm{N}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$ is in the interior of $\mathbf{x}^{\mathbf{I}}$, then there exists a unique zero of $\mathbf{f}$ in $\mathbf{x}^{\mathbf{I}}$ (and hence in $\mathrm{N}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$).

For a proof of this theorem, see Neumaier (1985, 1990) or Alefeld (1999).

In practice, when rounding is present, we compute a box $\mathrm{N}'(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$ containing $\mathrm{N}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$. If $\mathrm{N}'(\mathbf{x}, \mathbf{x}^{\mathbf{I}}) \subset \mathbf{x}^{\mathbf{I}}$, then $\mathrm{N}(\mathbf{x}, \mathbf{x}^{\mathbf{I}}) \subset \mathbf{x}^{\mathbf{I}}$. Hence, we can invoke Theorem 11.15.6 even when rounding occurs and guarantees the existence and uniqueness of a zero of $\mathbf{f}$ in a box $\mathbf{x}^{\mathbf{I}}$.

Hansen and Walster (1990b) conjectured as follows that Proposition 11.15.5 is true for all interval Newton methods.

Conjecture 11.15.7 Let $\mathbf{s}$ denote the set of solutions $\mathbf{s}$ of

$$\mathbf{f}(\mathbf{x}) + \mathbf{J}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})(\mathbf{s} - \mathbf{x}) = \mathbf{0}.$$

(See (5.2.3)). If $\mathbf{s} \subset \mathbf{x}^{\mathbf{I}}$, then there exists a solution of $\mathbf{f} = \mathbf{0}$ in $\mathbf{s}$.

All interval Newton methods compute a box $\mathrm{N}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$ containing $\mathbf{s}$. If $\mathrm{N}(\mathbf{x}, \mathbf{x}^{\mathbf{I}}) \subset \mathbf{x}^{\mathbf{I}}$, then $\mathbf{s} \subset \mathbf{x}^{\mathbf{I}}$. Therefore, if the conjecture is true, then Proposition 11.15.5 is true for all interval Newton methods.

The following theorem shows that, despite its crudeness, the earliest interval Newton method (given by (11.3.1)) exhibits quadratic convergence. For a more thorough discussion of convergence of interval Newton methods, see Neumaier (1990). Also, see Alefeld (1999) and Frommer and Mayer (1990).

Theorem 11.15.8 Let $\mathbf{f}$ be continuously differentiable in the initial box $\mathbf{x}^{\mathbf{I}(0)}$. Assume $\mathbf{x}^{\mathbf{I}(0)}$ contains a zero $\mathbf{x}^*$ of $\mathbf{f} = \mathbf{0}$. Assume that an interval matrix $\mathbf{V}(\mathbf{x}^{(0)}, \mathbf{x}^{\mathbf{I}(0)})$ exists that contains the inverse of every matrix in $\mathbf{J}(\mathbf{x}^{(0)}, \mathbf{x}^{\mathbf{I}(0)})$. If the sequence of boxes generated by (11.3.1) converges, then

$$\mathrm{w}(\mathbf{x}^{\mathbf{I}(k+1)}) \leq \gamma \left[\mathrm{w}(\mathbf{x}^{\mathbf{I}(k)})\right]^2$$

for some constant $\gamma \geq 0$.

This theorem was proved by Alefeld and Herzberger (1974, 1983).

The theorems in this section implicitly or explicitly depend on the assumption that **f** is expressed in a Taylor series. They remain valid if the expansion is in terms of slopes.

11.16 A NUMERICAL EXAMPLE

In this section, we give a simple numerical example that we solved by the algorithm in Section 11.12. Consider the well known Broyden banded function (Broyden (1971)). It is given by

$$f_i(\mathbf{x}) = x_i(2+5x_i^2) + 1 - \sum_{j \in J_i} x_j(1+x_j) = 0 \quad (i = 1, \cdots, n)$$

where $J_i = \{j : j \neq i, \max(1, i-5) \leq j \leq \min(n, i+1)\}$. Let $n = 3$.

Let the initial box be given by $X_i = [-1, 1]$ $(i = 1, 2, 3)$. We chose $\varepsilon_X = 10^{-8}$ and $\varepsilon_f = 1$ so that termination was driven by the size of the final box.

Hansen and Greenberg (1983) solved this problem with the same initial box. Depending on the form used, their algorithm required twelve or thirteen Newton steps to satisfy the convergence criterion specified by $\varepsilon_X = 10^{-8}$. The algorithm in Section 11.12 used three Newton steps.

However, this algorithm also uses hull and box consistencies. For this example, it applied hull consistency five times to the original equations. It applied both hull and box consistencies twice to the preconditioned system. These procedures used almost as many arithmetic steps as the three Newton steps. Note that for problems in higher dimensions, these procedures require fewer arithmetic operations than a Newton step.

In this example, the box consistency steps can be omitted with almost no change in the performance of the algorithm.

No splitting of the original box or any subbox was required by the algorithm of Section 11.12 to solve the above example.

11.17 PERTURBED PROBLEMS AND SENSITIVITY ANALYSIS

Interval Newton methods can be used to solve perturbed problems and to provide sensitivity analysis. We discuss the procedures briefly in this section. For other discussions of this topic, see, for example, Neumaier (1990) and Rump (1990). See also Section 17.11.

When using an interval Newton method, noninterval quantities such as $\mathbf{f}(\mathbf{x})$ are computed as interval quantities because interval arithmetic is used to bound rounding errors. If $\mathbf{f}(\mathbf{x})$ is defined in terms of a transcendental constant such as π, it is necessary to replace the constant by an interval containing it to assure that the computed version of $\mathbf{f}(\mathbf{x})$ contains the true value.

In general, then, the function $\mathbf{f}(\mathbf{x})$ is really an interval. The interval Newton method treats it as such. In general, a "zero" of $\mathbf{f}$ is therefore a set of points and not a single point.

To simplify discussion, assume $\mathbf{f}$ depends on a single parameter p that cannot be exactly represented in the number system used on the computer. Let us replace p by an interval P containing it. We now denote the function by $\mathbf{f}(\mathbf{x}, P)$. A "zero" of $\mathbf{f}(\mathbf{x}, P)$ is the set of zeros of $\mathbf{f}(\mathbf{x}, p)$ as p varies over P.

Whether p can be exactly represented on the computer or not, we might be interested in the set of zeros of $\mathbf{f}$ as a parameter p varies over some interval P. We can compute a box containing this set. We simply replace p by P and apply an interval Newton method to solve $\mathbf{f}(\mathbf{x}, P) = \mathbf{0}$. No change in the interval Newton algorithm is necessary. It is already designed to solve this problem.

In general, the solution set is not a box. In a sense, the best the interval Newton method can do is to compute the smallest box containing the solution set. The alternative is to cover the solution set with a number of small boxes. Actually, the smallest box containing the solution cannot generally be computed because of rounding errors. However, it is contained in the computed solution.

A directly related problem is sensitivity analysis. Suppose we wish to know how much a zero of $\mathbf{f}$ changes as p varies over some interval. That is, we wish to know the size of the solution set. A partial answer is produced if we compute the smallest box containing the solution set.

We can approximate the solution set as closely as desired by subdividing P into small subintervals, solving the problem for each subinterval separately, and taking the union of the results.

As just pointed out, the computed box cannot generally be the best possible box because of rounding errors. However, we can approximate it. But to do so, we must allow the interval Newton algorithm to continue until it stops progressing. That is, we cannot use stopping **Criteria A** or **B** discussed in Section 11.5. We discussed the implications of this possibility in that section.

Previously, in this chapter, we have usually assumed that the point of expansion for the interval Newton method is the center of the current box. In the perturbed case, convergence of the method does not necessarily yield the smallest box containing the solution set. We discussed this difficulty for the one dimensional case in Section 9.10. An illustrative example for the multidimensional case is given by Hansen and Greenberg (1983).

It seems certain that the best result can be produced by using appropriate corners of the current box as the points of expansion. However, we do not know if this is true. Moreover, it might be necessary to use too many different corners for this approach to work. Nevertheless, it seems reasonable to do the following. Use the center of the current box as the point of expansion until convergence **Criterion C** is satisfied. Then perform two extra iterations of the interval Newton method. First use the lower endpoints and then the upper endpoints of the components of the current box as the point of expansion.

In Section 17.11, we give an alternative procedure for sharply bounding the smallest box containing a solution. For another approach to computing the best result, see Dinkel, Tretter, and Wong (1988). For a method that covers the solution set with boxes of a specified size, see Neumaier (1988).

11.18 OVERDETERMINED SYSTEMS

A square system of nonlinear (or linear) equations can be solved directly, or as an optimization problem in which a norm, usually L_2 (least squares), is minimized. The direct solution, when available, is preferred because it is better conditioned than the corresponding optimization. When the system is overdetermined, the only procedure has been to use optimization. Walster and Hansen (2003) have developed a procedure to directly solve overdetermined systems of nonlinear equations. This procedure can be used directly, or as part of a constrained optimization problem. It is worth noting that an empty solution set can be used to numerically prove a system of interval equations (whether linear or nonlinear) is inconsistent. This event can then be used to falsify a theory or model.

Chapter 12

UNCONSTRAINED OPTIMIZATION

12.1 INTRODUCTION

In this chapter, we consider the unconstrained optimization problem. Constrained problems are discussed in Chapters 13, 14, and 15. An interval method for multicriterion optimization is given in Ichida and Fujii (1990).

We consider the problem

$$\text{Minimize (globally) } f(\mathbf{x}) \tag{12.1.1}$$

where f is a scalar function of a vector $\mathbf{x}$ of n components. We seek the minimum value f^* of f and the point(s) $\mathbf{x}^*$ at which this minimum occurs.

In our algorithm, we often use a Taylor expansion of the objective function f. For simplicity, we assume that any used derivative exists and is continuous in the region considered.

Interval methods exist that do not require differentiability. See Ratschek and Rokne (1988) or Moore, Hansen, and Leclerc (1991). We discuss this case in Section 12.18. Such methods are slower than those that take advantage of differentiability.

In Chapter 18, we show that some problems in which f is not differentiable can be reformulated as problems in which the new objective function has continuous derivatives of arbitrary order. An alternative approach is to use subdifferentials or slopes of nondifferentiable functions. See Section 7.11.

Following our convention, the discussion below is in terms of derivatives. However, replacing derivatives by slopes yields more efficient algorithms.

A simple way to use interval analysis in optimization is merely to bound rounding errors. Its first use for this purpose was by Robinson (1973). For other early papers of this kind, see Mancini (1975) and Mancini and McCormick (1976).

However, interval analysis has had a much more profound impact on optimization than just providing a means for bounding rounding errors. Interval analysis makes it possible to solve the global optimization problem, to guarantee that the global optimum is found, and to bound its value and location. Secondarily, perhaps, it provides a means for defining and performing rigorous sensitivity analyses. See Chapter 17.

Until fairly recently, it was thought that no numerical algorithm can guarantee having found the global solution of a general nonlinear optimization problem. Various authors have flatly stated that such a guarantee is impossible. Their argument was as follows: Optimization algorithms can sample the objective function and perhaps some of its derivatives only at a finite number of distinct points. Hence, there is no way of knowing whether the function to be minimized dips to some unexpectedly small value between sample points. In fact the dip can be between closest possible points in a given floating point number system.

This is a very reasonable argument; and it is probably true that no algorithm using standard arithmetic will ever provide the desired guarantee. However, interval methods do not sample at points. They compute bounds for functions over a continuum of points, including ones that are not finitely representable. See Theorem 4.8.14.

For example, consider the function

$$f(x) = x^4 - 4x^2$$

that has minima at $x = \pm 2^{1/2}$. Suppose we evaluate f at the point $x = 1$ and over the interval [3, 4]. We obtain

$$f(1) = -3 \text{ and } f([3,4]) = [17, 220].$$

Thus, we know that $f(x) \geq 17$ for all $x \in [3, 4]$ including such transcendental points as $\pi = 3.14....$ Since $f(1) = -3$, the minimum value of f is no larger than -3. Therefore, the minimum value of f cannot occur in the interval [3, 4]. We have proved this fact using only two evaluations of f.

In general the evaluation of f at a point involves rounding. Suppose that (outward) rounding and widening of intervals from dependence occurred in our example and we somehow obtained

$$f^{\mathrm{I}}(1) = [-3.1, -2.9] \text{ and } f^{\mathrm{I}}([3,4]) = [16.9, 220.1].$$

Because we have bounded all errors, we know that $f(1) \leq -2.9$ and that $f(x) \geq 16.9$ for all $x \in [3, 4]$. Hence, as before, we know the minimum value of f is not in [3, 4]. Rounding and dependence have not prevented us from infallibly drawing this conclusion.

By eliminating subintervals that are proved to not contain the global minimum, we eventually isolate the minimum point. We describe various ways to do the elimination.

An algorithm for global optimization was introduced by Hansen (1980). The algorithm provides guaranteed bounds on the globally minimum value f^* of an objective function and on the point(s) $\mathbf{x}^*$ where it occurs. It guarantees that the global solution(s) in some given box has been found. A one dimensional version of the algorithm can be found in Hansen (1979).

In this chapter, we describe an improved version of the algorithm. The primary improvement is the introduction of hull and box consistency. Various other changes are also made.

Because of computational limitations on accuracy, our algorithm might also find "near global" minima when rounding and/or dependence prevents determination of which of two or more candidates is the true minimum. However, if the termination criteria are chosen stringently enough, our algorithm always eliminates a local minimum from consideration if f is sufficiently larger than f^* at the local minimum. Obviously, "sufficiently larger" depends on the wordlength used in the computations

In Section 12.2, we present an overview of the optimization algorithm. In Section 12.3, we discuss the initial box in which the solution is sought. Sections 12.4 through 12.13 describe various subalgorithms of the overall procedure. Termination procedures are considered in Section 12.9. The steps of the algorithm are given in Section 12.14 and discussed in Section 12.15. A numerical example is given in Section 12.16. Problems with multiple minima are discussed in Section 12.17. Section 12.18 discusses problems with nondifferentiable objective functions.

12.2 AN OVERVIEW

Our algorithm computes guaranteed bounds on the minimum value f^* of the objective function f and on the point(s) $\mathbf{x}^*$ where f takes on this minimum value. If there is more than one solution point, our algorithm never fails to find them all. It proceeds roughly as follows:

1. Begin with a box $\mathbf{x}^{\mathbf{I}(0)}$ in which the global minimum is sought. Because we restrict our search to this particular box, our problem is really constrained. We discuss this aspect in Section 12.3.

2. Delete subboxes of $\mathbf{x}^{\mathbf{I}(0)}$ that cannot contain a solution point. Use fail-safe procedures so that, despite rounding errors, the point(s) of global minimum are never deleted.

   The methods for Step 2 are as follows:

   (a) Delete subboxes of the current box wherein the gradient $\mathbf{g}$ of f is nonzero. This can be done without differentiating $\mathbf{g}$, as is described in Section 12.4. This use of monotonicity was introduced by Hansen (1980). Alternatively, consistency methods (see Chapter 10) and/or an interval Newton method (see Chapter 11) can be applied to solve the equation $\mathbf{g} = 0$. In so doing, derivatives (or slopes) of $\mathbf{g}$ are used. By narrowing bounds on points where $\mathbf{g} = 0$, application of a Newton method deletes subboxes wherein $\mathbf{g} \neq 0$.

(b) Compute upper bounds on f at various sampled points. The smallest computed upper bound $\bar{\mathrm{f}}$ is an upper bound for f^*. Then delete subboxes of the current box wherein $f > \bar{\mathrm{f}}$. See Section 12.5. The concept of generating and using an upper bound in this way was introduced by Hansen (1980).

(c) Delete subboxes of the current box wherein f is not convex. See Section 12.7. The concept of using convexity in this way was introduced by Hansen (1980).

3. Iterate Step 2 until a sufficiently small set of points remains. Since $\mathbf{x}^*$ must be in this set, its location is bounded. Then bound f over this set of points to obtain final bounds on f^*

For problems in which the objective function is not differentiable, Steps 2a and 2c cannot be used because the gradient $\mathbf{g}$ is not defined. Step 2b is always applicable.

We describe these procedures and other aspects of our algorithm in this chapter. For simpler tutorial discussions, see Hansen (1988) and Walster (1996). For a survey discussion, see Hansen and Walster (1990b).

12.3 THE INITIAL BOX

The user of our algorithm can specify a box or boxes in which the solution is sought. Any number of finite boxes can be prescribed to define the region of search. The boxes can be disjoint or overlap. However, if they overlap, a minimum at a point that is common to more than one box is separately found as a solution in each box containing it. In this case, computing effort is wasted. For simplicity, assume the search is made in a single box that we denote by $\mathbf{x}^{\mathbf{I}(0)}$.

If the user does not specify $\mathbf{x}^{\mathbf{I}(0)}$, we search in a "default box" described in Section 11.10. The smaller the initial box, the faster the algorithm can solve the problem. Therefore, it is better if the user can specify a smaller box than the default.

Since we restrict the region of search to a finite box, we have replaced the unconstrained problem by a constrained one of the form

$$\begin{aligned}&\text{Minimize (globally) } f(\mathbf{x})\\&\text{Subject to } \mathbf{x} \in \mathbf{x}^{\mathbf{I}(0)}\end{aligned} \tag{12.3.1}$$

Actually, we do not solve this constrained problem because we assume the solution occurs at a stationary point of f. We can solve this problem by using the method for inequality constrained problems to be discussed in Chapter 14. However, it is simpler to assume the box $\mathbf{x}^{\mathbf{I}(0)}$ is so large that a solution does not occur at a nonstationary point on the boundary of $\mathbf{x}^{\mathbf{I}(0)}$.

Experience has shown that, in practice, $\mathbf{x}^{\mathbf{I}(0)}$ can generally be chosen quite large without seriously degrading the efficiency of the algorithm. For noninterval algorithms, this is not necessarily the case. An algorithm that converges nicely for a given initial search point $\mathbf{x}^{(0)}$ might fail to converge from the point $10\mathbf{x}^{(0)}$ or $100\mathbf{x}^{(0)}$. See Moré, Garbow and Hillstrom (1981).

Walster, Hansen, and Sengupta (1985) compared run time on their interval algorithm as a function of the size of the initial box. They found that increasing box width by an average factor of 9.1×10^5 increased the run time by an average factor of only 4.4. For one example, however, the run time increased by a factor of 265. R. E. Moore (private communication) observed a case in which run time increased by a factor of only 2.3 when the box width increased by a factor of 4.2×10^8.

Often, a finite region of interest is known; and we can specify $\mathbf{x}^{\mathbf{I}(0)}$ with assurance that it contains the global minimum. For simplicity in what follows, we assume that this is the case.

12.4 USE OF THE GRADIENT

We assume that f is continuously differentiable. Therefore, the gradient $\mathbf{g}$ of f is zero at the global minimum. Of course, $\mathbf{g}$ is also zero at local minima, at maxima and at saddle points. Our goal is to find the zero(s) of $\mathbf{g}$ at which f is a global minimum. As we search for zeros, we attempt to discard any that are not a global minimum of f. This is done by procedures outlined in Section 12.2 and discussed fully in other sections of this chapter. Generally, we discard boxes containing nonglobal minimum points before we spend the effort to bound such minima very accurately.

Note that the condition $\mathbf{g} = \mathbf{0}$ does not have a counterpart expressed in terms of slopes. This is one of the few cases where derivatives, rather than slopes, must be used.

Hansen (1980) noted that we can use a simple monotonicity test that can enable us to prove nonexistence of a stationary point in a box. In such a case, the box can be deleted. We now describe this test and then note that it is better replaced by a step of a hull consistency procedure.

Let $\mathbf{x}^{\mathbf{I}}$ be a subbox of the initial box $\mathbf{x}^{\mathbf{I}(0)}$. (In Section 12.14, we discuss how $\mathbf{x}^{\mathbf{I}}$ might be obtained.) We evaluate the components of $\mathbf{g}$ over $\mathbf{x}^{\mathbf{I}}$. That is, we evaluate $g_i(\mathbf{x}^{\mathbf{I}})$ for $i = 1, \cdots, n$. Denote the resulting interval by $[\underline{g}_i(\mathbf{x}^{\mathbf{I}}), \overline{g}_i(\mathbf{x}^{\mathbf{I}})]$. If $\underline{g}_i(\mathbf{x}^{\mathbf{I}}) > 0$ or if $\overline{g}_i(\mathbf{x}^{\mathbf{I}}) < 0$, then $g_i(\mathbf{x}) \neq 0$ for any $\mathbf{x} \in \mathbf{x}^{\mathbf{I}}$. Therefore, there is no stationary point of f in $\mathbf{x}^{\mathbf{I}}$. Therefore, $\mathbf{x}^{\mathbf{I}}$ cannot contain the global minimum of f and can be deleted.

In practice, the bounds on $g_i(\mathbf{x}^{\mathbf{I}})$ can fail to be sharp because of rounding and/or dependence. Nevertheless, the procedure remains valid.

In Section 10.10, we note that hull consistency can perform a test of this kind with essentially the same amount of computing. Moreover, it can sometimes reduce the box in the process. Therefore, hull consistency is better than simply evaluating $\mathbf{g}(\mathbf{x}^{\mathbf{I}})$. The following example illustrates this point.

A well known example in unconstrained optimization is the so-called three hump camel function

$$f(x, y) = 2x^2 - 1.05x^4 + \frac{1}{6}x^6 - xy + y^2.$$

The components of the gradient of f can be written as

$$g_1(x, y) = x[(x^2 - 2.1)^2 - 0.41] - y,$$
$$g_2(x, y) = 2y - x.$$

For the box used, we have written $g_1(x, y)$ in a form from which its interval value can be sharply computed.

Let $X = [3, 4]$ and $Y = [1.9, 142]$. If we simply evaluate $g_1(X, Y)$ and $g_2(X, Y)$, we find that each of these intervals contain zero. Therefore, we gain no information about nonexistence of a stationary point in the box.

Now suppose we use hull consistency. We write $g_1(x, y)$ in its more natural form

$$g_1(x, y) = 4x - 4.2x^3 + x^5 - y.$$

Suppose we solve for the term $4x$. We write the remaining terms containing x in factored form so that

$$4X' = Y - X^3(X^2 - 4.2).$$

We compute $X' = [-188.325, 3.1]$, which we replace by $X'' = X \cap X' = [3, 3.1]$. Thus, we have improved the interval bound on x.

We can continue to apply hull consistency to g_1 by solving for other terms; but suppose we do not. Let us now apply hull consistency to g_2. Suppose we solve for y as $2Y' = X''$. We obtain $Y' = [1.5, 1.55]$. Since $Y' \cap Y$ is empty, there is no point in the original box at which $\mathbf{g} = 0$. Therefore, the box cannot contain a minimum of f.

Simply evaluating $\mathbf{g}$ over the box gave no information. With essentially the same computational effort, hull consistency first improves the bounds on $\mathbf{x}$ and then reveals the nonexistence of a solution in the box.

12.5 AN UPPER BOUND ON THE MINIMUM

We use various procedures to find the zeros of the gradient of the objective function. We have noted that these zeros can be stationary points that are not the global minimum. Therefore, we want to avoid spending the effort to closely bound such points when they are not the desired solution. In this section we consider procedures that help in this regard.

As our algorithm proceeds, we evaluate f at various points in the original box $\mathbf{x}^{\mathrm{I}(0)}$. The computed upper bound on each such value is an upper bound for the globally minimum value f^* of f. We use the smallest bound $\overline{\mathfrak{f}}$ obtained in this way.

Let $\mathbf{x}^{\mathbf{I}}$ be a subbox of $\mathbf{x}^{\mathbf{I}(0)}$ generated by the algorithm. In any given step of our algorithm, such a box might be deleted or accepted as a final bound for a global solution point. More often, $\mathbf{x}^{\mathbf{I}}$ is reduced in size and/or split into subboxes. That is, one or more new subboxes is generated. When this occurs, we evaluate f at the center of each new box. We can also do a local search for the smallest value of f in such a box. See Section 12.6.

Suppose we evaluate f at a point $\mathbf{x}$. Because of rounding, the result is generally an interval $[\underline{f^{\mathbf{I}}}(\mathbf{x}), \overline{f^{\mathbf{I}}}(\mathbf{x})]$. Despite rounding errors in the evaluation, we know without question that $\overline{f^{\mathbf{I}}}(\mathbf{x})$ is an upper bound for $f(\mathbf{x})$ and hence for f^*. Let $\bar{\mathfrak{f}}$ denote the smallest such upper bound obtained for various points sampled at a given stage of the overall algorithm. This upper bound plays an important part in our algorithm.

Since $\bar{\mathfrak{f}} \geq f^*$, we can delete any point (or subbox) of $\mathbf{x}^{\mathbf{I}(0)}$ for which $f > \bar{\mathfrak{f}}$. This might serve to delete a subbox that bounds a nonoptimal stationary point of f. We describe four methods that can be applied to the relation $f(\mathbf{x}) \leq \bar{\mathfrak{f}}$ to try to delete part or all of a subbox generated by our algorithm. The concept of generating and using an upper bound in this way was introduced by Hansen (1980).

12.5.1 First Method

Let $\mathbf{x}^{\mathbf{I}}$ be a given subbox of $\mathbf{x}^{\mathbf{I}(0)}$. We can simply evaluate f over $\mathbf{x}^{\mathbf{I}}$ and obtain $[\underline{f}(\mathbf{x}^{\mathbf{I}}), \overline{f}(\mathbf{x}^{\mathbf{I}})]$. Despite rounding errors, it is certain that $\underline{f}(\mathbf{x}^{\mathbf{I}})$ is a lower bound for $f(\mathbf{x})$ for any $\mathbf{x} \in \mathbf{x}^{\mathbf{I}}$. Hence, if $\underline{f}(\mathbf{x}^{\mathbf{I}}) > \bar{\mathfrak{f}}$, then $f(\mathbf{x}) > f^*$ for every $\mathbf{x} \in \mathbf{x}^{\mathbf{I}}$. Therefore, we can delete $\mathbf{x}^{\mathbf{I}}$

However, there is a better alternative (within the first method). We can apply hull consistency to the relation $f(\mathbf{x}) \leq \bar{\mathfrak{f}}$ over the box $\mathbf{x}^{\mathbf{I}}$. As we noted in Section 10.10, this serves the same purpose with essentially the same amount of computing required to evaluate $f(\mathbf{x}^{\mathbf{I}})$. However, an added benefit is that part of $\mathbf{x}^{\mathbf{I}}$ might be deleted. In this case hull consistency is applied to the equality $f(\mathbf{x}) = [-\infty, \bar{\mathfrak{f}}]$.

The following example illustrates the utility of this procedure. Problem number 3.1 of Schwefel (1977) is to find the unconstrained minimum of

$$f(\mathbf{x}^{\mathbf{I}}) = \sum_{i=1}^{3} [(x_1 - x_i^2)^2 + (x_i - 1)^2].$$

Let the initial box be given by $X_i = [-10^6, 10^6]$ for $i = 1, 2, 3$. We use hull consistency and solve $f(\mathbf{x}) - \bar{\mathfrak{f}} \leq 0$ for each variable in two ways. We use

$$(x_1 - x_j^2)^2 \leq \bar{\mathfrak{f}} - \sum_{\substack{i=1 \\ i \neq j}}^{3} (X_1 - X_i^2)^2 - \sum_{i=1}^{3} (X_i - 1)^2$$

for $j = 1, 2, 3$ and

$$(x_i - 1)^2 \leq \bar{\mathfrak{f}} - \sum_{i=1}^{3} (X_1 - X_i^2)^2 - \sum_{\substack{i=1 \\ i \neq j}}^{3} (X_i - 1)^2$$

for $i = 1, 2, 3$.

When new bounds on both variables have been obtained in this way, we get a new bound $\overline{f}$ on f^* by evaluating f at the center of the new box. We iterate these steps. The procedure converges to the solution at (1, 1, 1) in only seven steps.

The first method (just described) applies hull consistency to $f(\mathbf{x}) \leq \bar{\mathfrak{f}}$. Note that it does not involve use of derivatives or slopes. The methods we now discuss use Taylor expansions. However, better results can be obtained using slope expansions. Because expansions are used, the following methods require more computing than the first method, which uses only hull consistency.

12.5.2 Second Method

We can apply box consistency (see Section 10.2.) to the inequality $f(\mathbf{x}) \leq \bar{\mathfrak{f}}$. To do so, we rewrite the inequality as the equation $F(\mathbf{x}) = 0$ where

$$F(\mathbf{x}) = f(\mathbf{x}) + [-\bar{\mathfrak{f}}, \infty].$$

Suppose we use box consistency to solve $F(\mathbf{x}) = 0$ for a component x_i $(i = 1, \cdots, n)$ of $\mathbf{x}$. To do so, we replace all variables except x_i by their interval bounds. Thus, we apply the one-dimensional Newton method to the function

$$q^{\mathbf{I}}(x_i) = F(X_1, \cdots, X_{i-1}, x_i, X_{i+1}, \cdots, X_n) = 0.$$

To do so, we expand $q^{\mathbf{I}}$ about an endpoint, say a_i of X_i. The Newton result is thus computed as

$$\mathrm{N}(a_i, \mathbf{x}^{\mathbf{I}}) = a_i - \frac{q^{\mathbf{I}}(a_i)}{g_i(\mathbf{x}^{\mathbf{I}})}. \tag{12.5.1}$$

However, we do not apply the Newton step if $0 \in q^{\mathbf{I}}(a_i)$. See the algorithm in Section 10.2. Therefore, assume $0 \notin q^{\mathbf{I}}(a_i)$.

In the denominator of (12.5.1), the function $g_i(\mathbf{x}^{\mathbf{I}})$ is the derivative of F with respect to x_i. That is, it is the i-th component of the gradient $\mathbf{g}$ of f. If $0 \notin g_i(\mathbf{x}^{\mathbf{I}})$ then there is no stationary point of f in $\mathbf{x}^{\mathbf{I}}$; and we delete $\mathbf{x}^{\mathbf{I}}$. Therefore, when we apply box consistency to F, we always have $0 \in g_i(\mathbf{x}^{\mathbf{I}})$.

We now know that the denominator interval in (12.5.1) contains zero but the numerator does not. Therefore, the quotient in (12.5.1) is computed as the union of two semi-infinite intervals. The endpoint a_i of X_i is in the interior of the gap between these intervals. This implies that, if the Newton step is applied, it always deletes part of X_i. However, the deleted part can be vanishingly small.

Box consistency can be effective when applied to $f(\mathbf{x}) \leq \bar{\mathfrak{f}}$ even when it is applied over a large box. We now consider methods that are usually effective only when the box is small.

12.5.3 Third Method

We first describe a method in which we use a Taylor expansion to linearize $f(\mathbf{x})$. In a sense, this method is the same as the second method using box consistency except that f is expanded with respect to all the variables. It is more effective than the second method when the box is small.

Let $\mathbf{x}$ be a given point in the current box $\mathbf{x}^{\mathbf{I}}$ and let $\mathbf{y}$ be a variable point in $\mathbf{x}^{\mathbf{I}}$. Denote

$$g^{\mathbf{I}}{}_i = g_i(X_1, \cdots, X_i, x_{i+1}, \cdots, x_n).$$

where $g^{\mathbf{I}}{}_i$ is the i-th component of the gradient of f. From (7.3.6), the first order Taylor expansion of f gives

$$f(\mathbf{y}) \in f(\mathbf{x}) + \sum_{i=1}^{n} (y_i - x_i) g^{\mathbf{I}}{}_i. \tag{12.5.2}$$

For some $j = 1, \cdots, n$, rewrite (12.5.2) as

$$f(\mathbf{y}) \in f(\mathbf{x}) + (y_j - x_j) g^{\mathbf{I}}{}_j + \sum_{\substack{i=1 \\ i \neq j}}^{n} (y_i - x_i) g^{\mathbf{I}}{}_i. \tag{12.5.3}$$

For all $i \neq j$, replace y_i by X_i in the right member of (12.5.3). Since $y_i \in X_i$,

$$f(\mathbf{y}) \in f(\mathbf{x}) + (y_j - x_j) g^{\mathbf{I}}{}_j + \sum_{\substack{i=1 \\ i \neq j}}^{n} (X_i - x_i) g^{\mathbf{I}}{}_i. \tag{12.5.4}$$

Note that $f(\mathbf{y}) > \bar{\mathfrak{f}}$ if the left endpoint of the right member of (12.5.4) exceeds $\bar{\mathfrak{f}}$.

Define $t = y_j - x_j$ and the intervals

$$U = f(\mathbf{x}) + \sum_{\substack{i=1 \\ i \neq j}}^{n} (X_i - x_i) g^{\mathbf{I}}{}_i - \bar{\mathfrak{f}}$$

and $V = g^{\mathbf{I}}{}_j$. We wish to delete points of $\mathbf{x}^{\mathbf{I}}$ where y_j is such that $U + Vt > 0$ and retain points where

$$U + Vt \leq 0. \tag{12.5.5}$$

Let T denote the set of values of t for which (12.5.5) holds. This set can be an interior or an exterior interval as given by (6.2.4).

Note that if T is the empty set, we have proved that $f(\mathbf{y}) > \bar{\mathfrak{f}}$ for all $\mathbf{y} \in \mathbf{x}^{\mathbf{I}}$ so we delete $\mathbf{x}^{\mathbf{I}}$.

Having computed T for a particular value of j, the set of retained values of y_j is $T + x_j$. Since we are interested only in values of y_j within $\mathbf{x}^{\mathbf{I}}{}_j$, we retain only the intersection

$$Y_j = \mathbf{x}^{\mathbf{I}}{}_j \cap (T + x_j).$$

If this set is empty, we delete all of $\mathbf{x}^{\mathbf{I}}$. Otherwise, we replace X_j by Y_j. When computing Y_{j+1}, we use the updated intervals $X_1, \cdots, X_j$ if and when they are narrowed. We repeat this procedure for all $j = 1, \cdots, n$.

Note that Y_j might be composed of the two intervals surrounding a gap. Suppose this is the case. Denote the two intervals by $[a_j, b_j]$ and $[c_j, d_j]$ where $b_j < c_j$. We know that the global minimum does not occur in the gap (i.e., the open interval) (b_j, c_j).

However, for the purpose of computing Y_{j+1}, we ignore this fact and replace x_j by the single interval $[a_j, d_j]$ containing both of the smaller subintervals and the gap. This simplifies the work of computing subsequent components of $\mathbf{x}$. However, we retain the information about identified gaps so that if we later split a component of the box, we can do so by removing a gap.

12.5.4 Fourth Method

We now consider a method for using $f(\mathbf{x}) \leq \bar{\mathfrak{f}}$ in which we expand f through second derivative terms. We do not use it in our unconstrained optimization algorithm. However, we do use it for the constrained case because the necessary second derivatives are sometimes computed for other purposes.

From (7.3.8), we have

$$f(\mathbf{y}) \in f(\mathbf{x}) + (\mathbf{y}-\mathbf{x})^T \mathbf{g}(\mathbf{x}) + \frac{1}{2}(\mathbf{y}-\mathbf{x})^T \mathbf{H}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})(\mathbf{y}-\mathbf{x})$$

where the element H_{ij} of the Hessian $\mathbf{H}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$ has arguments $(X_1, ...X_j, x_{j+1}, ...x_n)$. Note that $\mathbf{H}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$ is lower triangular. See Section 7.4. We choose $\mathbf{x}$ to be the center of $\mathbf{x}^{\mathbf{I}}$. Denote $z = \mathbf{y} - \mathbf{x}$. We can delete points $\mathbf{y}$ where

$$f(\mathbf{x}) + z^T \mathbf{g}(\mathbf{x}) + \frac{1}{2} z^T \mathbf{H}(\mathbf{x}, \mathbf{x}^{\mathbf{I}}) z > \bar{\mathfrak{f}}.$$

To simplify presentation, we describe the case $n = 2$. If we delete the points indicated, we retain the complementary set of points $\mathbf{y}$ where

$$f(\mathbf{x}) + z_1 g_1 + z_2 g_2 + \frac{1}{2}(z_1^2 H_{11} + z_1 z_2 H_{21} + z_2^2 H_{22}) \leq \bar{\mathfrak{f}} \quad (12.5.6)$$

We first solve (12.5.6) for z_1. Therefore, we replace z_2 by $Z_2 = X_2 - x_2$. That is, we replace y_2 by its bounding interval X_2. Thus, (12.5.6) becomes

$$A + Bz_1 + Cz_1^2 \leq 0 \quad (12.5.7)$$

where

$$A = f(\mathbf{x}) - \bar{\mathfrak{f}} + Z_2 g_2 + \frac{1}{2} Z_2^2 H_{22}$$

$$B = g_1 + \frac{1}{2} Z_2 H_{21}$$

$$C = \frac{1}{2} H_{11}.$$

Suppose we solve the quadratic relation (12.5.7) (as described in Section 8.2) for z_1 and obtain an interval Z_1' and thus the interval $X_1' = Z_1' + x_1$.

Since we are interested only in points in X_1, the solution of interest is $X_1'' = X_1 \cap X_1'$. If X_1'' is empty, there is no point in $\mathbf{x}^I$ at which f is as small as $\bar{f}$. In this case, we have proved that the global minimum of f cannot occur in $\mathbf{x}^I$.

The interval X_1' might contain a gap in which the global minimum of f cannot occur. If there is such a gap, we temporarily ignore it.

Next, we solve (12.5.6) for z_2. To do so, we replace z_1 by the interval bounding it. We use the improved result $Z_1'' = X_1'' - x_1$. We obtain the quadratic relation

$$A + Bz_2 + Cz_2^2 \leq 0 \tag{12.5.8}$$

where

$$A = f(\mathbf{x}) - \bar{f} + Z_1''g_1 + \frac{1}{2}(Z_1'')^2 H_{11}$$

$$B = g_2 + \frac{1}{2}Z_1''H_{21}$$

$$C = \frac{1}{2}H_{22}.$$

Whether we are solving (12.5.7) or (12.5.8), we must determine the solution points t of a quadratic inequality of the form

$$A + Bt + Ct^2 \leq 0 \tag{12.5.9}$$

where A, B, and C are intervals. The solution set of (12.5.9) was derived by Hansen (1980) in explicit form. Although his analysis was correct, there are errors in his listed results. Denote $A = [a_1, a_2]$. His errors are for the case $a_1 = 0$. A correct form (with some simplifications) was given in the first edition of this book. To list all the cases required almost a full page. We gave a simpler procedure in Section 8.2. See also Hansen and Walster (2001).

12.5.5 An Example

We have described four procedures for deleting points $\mathbf{x}$ where $f(\dot{\mathbf{x}}) > \bar{f}$. Generally, none of these methods can delete all the points $\mathbf{x}$ of a box $\mathbf{x}^I$

where $f(\mathbf{x}) > \bar{\mathfrak{f}}$ unless this inequality holds for all $\mathbf{x} \in \mathbf{x}^{\mathbf{I}}$. This is because the complementary set in $\mathbf{x}^{\mathbf{I}}$ (where $f(\mathbf{x}) \leq \bar{\mathfrak{f}}$) is generally not in the form of a box. Nevertheless, the relation $f(\mathbf{x}) \leq \bar{\mathfrak{f}}$ can be very useful.

As an example, consider the objective function

$$f(x) = x^2(2 + \sin \pi x).$$

It is easily shown that the derivative $f'(x)$ of this function has a zero in the interval $[n, n+1]$ for all $n = \pm 2, \pm 3, \cdots$. To see this, note that $f'(0) = 0$, $f'(\pm n) > 0$ for n even and nonzero, and $f'(\pm n) < 0$ for n odd and $n \leq 3$. Also, $f'(x) = 0$ for $x = 1.118$, approximately. Thus, $\acute{f}'$ has at least $2n+1$ zeros in the interval $[-n, n]$ for all $n = 1, 2, ...$

Suppose we have sampled the value of f at $x = 1$. Since $f(1) = 2$, we have the upper bound $\bar{\mathfrak{f}} = 2$ on f^*. We can replace the relation $x^2(2 + \sin \pi x) \leq \bar{\mathfrak{f}} = 2$ by

$$x^2(2 + \sin \pi x) = [-\infty, 2].$$

Consider the interval $X = [2, 10^{30}]$. To solve this equation using hull consistency we can replace $\sin \pi x$ by $\sin \pi X = [-1, 1]$ and solve for the factor x^2. We obtain

$$(X_1')^2 = [-\infty, 2]/[1, 3] = [-\infty, 2].$$

Since $(X_1')^2$ must be nonnegative, we replace this equation by $(X_1')^2 = [0, 2]$ from which $X_1' = [-2^{1/2}, 2^{1/2}]$.

Since X' is disjoint from X, there is no point in X where $f \leq \bar{\mathfrak{f}}$. Therefore, no point in X can be the global minimum. We have proved this with one simple application of hull consistency. In so doing, we have proved that the $10^{30} - 2$ local extrema in X are not global minima.

Note that of the four methods described above, only the first can be effective for this example. This is because the other methods use interval bounds on derivatives; and these bounds contain zero for almost any subinterval of X of unit width.

12.6 UPDATING THE UPPER BOUND

In Section 12.5, we discussed how we can use an upper bound $\bar{f}$ on the global minimum f^* to delete points that cannot be global minimum points. We noted that whenever we generate a new subbox of the initial box $\mathbf{x}^{I(0)}$, we evaluate f at its center and use the value to update $\bar{f}$. However, we can also search for a point where $f < \bar{f}$ and use such a point to reduce $\bar{f}$. In this section, we discuss two ways that this can be done efficiently.

Suppose that before starting to solve an unconstrained optimization problem by our algorithm, we know a value of $\bar{f}$ that is near or equal to the global minimum f^*. In this case, the solution is found more rapidly than when no such initial value is known.

Walster, Hansen and Sengupta (1985) observed that obtaining a good approximation $\bar{f}$ to f^* early in the process of solving an optimization problem did not greatly improve the efficiency of their algorithm. However, they solved the relation $f \leq \bar{f}$ by expanding f in Taylor series through first or second derivative terms as in Section 12.5.4. As we note throughout this book, using Taylor expansions in interval algorithms is generally effective only when the box is small. Therefore, expanding f to solve $f \leq \bar{f}$ is not fruitful in the early stage of the solution process when the current box is large. That is, having a good upper bound $\bar{f}$ early in the solution process was not very helpful in their algorithm.

Using hull and box consistency changes the situation. If a value of $\bar{f}$ near f^* is available, consistency methods can effectively reduce a box even when the box is large. Therefore, it is now more important to have a good approximation to f^* early in the solution process. Sampling a value of f at the center of each new generated box serves to produce good values of $\bar{f}$ fairly quickly. This is particularly true because of the way we choose which box to process. See Section 12.12.

Because it is so helpful to have a small value of $\bar{f}$, we also use other procedures to reduce $\bar{f}$. In one such procedure, we do a line search for a point where f is small. We now describe this procedure.

Suppose we evaluate the gradient $\mathbf{g}(\mathbf{x})$ of $f(\mathbf{x})$ at a point $\mathbf{x}$. Note that

f decreases (locally) in the negative gradient direction from $\mathbf{x}$. A simple procedure for finding a point where f is small is to search along this half-line. Let $\mathbf{x}$ be the center of the current box. Define the half-line of points $\mathbf{y}(\alpha) = \mathbf{x} - \alpha\mathbf{g}(\mathbf{x})$ where $\alpha \geq 0$. We now use a standard procedure for finding an approximate minimum of the objective function f on this half-line.

We first restrict our region of search by determining the value α' such that $\mathbf{y}(\alpha') = \mathbf{x} - \alpha'\mathbf{g}$ is on the boundary of the current box $\mathbf{x}^{\mathbf{I}}$. We search between $\mathbf{x}$ and $\mathbf{x}' = \mathbf{y}(\alpha')$. We use the following procedure. Each application of the procedure requires an evaluation of f.

Procedure: If $f(\mathbf{x}') < f(\mathbf{x})$, replace $\mathbf{x}$ by $(\mathbf{x}+\mathbf{x}')/2$. Otherwise, replace $\mathbf{x}'$ by $(\mathbf{x} + \mathbf{x}')/2$.

We apply this procedure eight times. We then use the smaller of the final values of $f(\mathbf{x})$ and $f(\mathbf{x}')$ to update $\bar{\mathfrak{f}}$.

Much of this procedure can be done using rounded real arithmetic. The evaluation of $\mathbf{g}(\mathbf{x})$ and the computations to do the search steps need only be approximate. However, the final evaluation of f used to update $\bar{\mathfrak{f}}$ must be done using interval arithmetic.

The other procedure that we use to update $\bar{\mathfrak{f}}$ uses data that are expensive to compute. Therefore, we apply the procedure only when the data are available because they have been computed for another purpose. The relevant data are computed when we apply a Newton method to bound a zero of the gradient $\mathbf{g}(\mathbf{x})$. When we do so, we compute the Jacobian $\mathbf{J}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$ and an approximate inverse $\mathbf{B}$ of the center of $\mathbf{J}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$. See Section 11.2. We also evaluate $\mathbf{g}$ at the center $\mathbf{x}$ of the current box $\mathbf{x}^{\mathbf{I}}$. The data we require consist of $\mathbf{g}(\mathbf{x})$ and $\mathbf{B}$.

If the Newton method succeeds in obtaining a new box bounding the solution of $\mathbf{g} = 0$, we evaluate f at the center of the box and use the result to update $\bar{\mathfrak{f}}$. Regardless of whether the Newton step is successful or not, we use the data already computed to obtain an additional test point at which to evaluate $f(\mathbf{x})$. This test point is the point $\mathbf{y} = \mathbf{x} - \mathbf{B}\mathbf{g}(\mathbf{x})$. It doesn't matter whether $\mathbf{y}$ is in the current box or not. In fact, $\mathbf{y}$ need not even be in the initial box $\mathbf{x}^{\mathbf{I}(0)}$. Any point can be used to update $\bar{\mathfrak{f}}$. This procedure can be iterated. Steps of a method for doing so are given in Section 11.4.

We used a similar procedure in Section 11.4 to get an approximation for a solution to a system of nonlinear equations.

We can iterate this step starting from the point $\mathbf{y}$ and use the same value of $\mathbf{B}$. However, this requires recomputing $\mathbf{g}(\mathbf{y})$; and therefore we choose not to do so.

Note that the computation of $\mathbf{y}$ from the quantities $\mathbf{x}$, $\mathbf{B}$, and $\mathbf{g}(\mathbf{x})$ can be done using real arithmetic. Only the evaluation of $f(\mathbf{y})$ must be done using interval arithmetic so that $\bar{f}$ can be correctly updated.

12.7 CONVEXITY

If $f(\mathbf{x})$ has a minimum at $\mathbf{x}^*$, then f must be convex in some neighborhood of $\mathbf{x}^*$. Hence, the Hessian $\mathbf{H}$ of f must be positive semidefinite at $\mathbf{x}^*$. A necessary condition for this is that the diagonal elements H_{ii} $(i = 1, \cdots, n)$ of $\mathbf{H}$ be nonnegative. Note that $\mathbf{H}$ cannot be replaced by its counterpart in an expansion using slopes.

Consider a box $\mathbf{x}^I$. If $H_{ii}(\mathbf{x}^I) < 0$ for some $i = 1, \cdots, n$, then $H_{ii}(\mathbf{x}) < 0$ for all $\mathbf{x} \in \mathbf{x}^I$. Hence, $\mathbf{H}$ cannot be positive semidefinite for any point in $\mathbf{x}^I$. Therefore, f cannot have a stationary minimum in $\mathbf{x}^I$ and we can delete $\mathbf{x}^I$. The concept of using convexity in this way was introduced by Hansen (1980).

There are other conditions that $\mathbf{H}$ must satisfy to be positive semidefinite. For example, the leading principal minors of $\mathbf{H}$ of all orders $1, \cdots, n$ must be nonnegative. (This is also a sufficient condition.) We could check to see if one or more of these conditions is violated over $\mathbf{x}^I$ and, if so, delete $\mathbf{x}^I$. However, this extra effort is probably not warranted.

To use the convexity condition, we could simply evaluate $H_{ii}(\mathbf{x}^I)$. If $H_{ii}(\mathbf{x}^I) < 0$, for any $i = 1, \cdots, n$, then $\mathbf{x}^I$ cannot contain a minimum of f and we might delete $\mathbf{x}^I$. However, for essentially the same amount of effort, we can apply hull consistency to solve $H_{ii}(\mathbf{x}^I) \geq 0$. If hull consistency proves that $H_{ii}(\mathbf{x}^I) < 0$, then we can delete $\mathbf{x}^I$. Additionally, hull consistency might be able to delete part of $\mathbf{x}^I$. Simply evaluating $H_{ii}(\mathbf{x}^I)$ cannot reduce $\mathbf{x}^I$.

There is generally an extended region of points (a basin) at and near a minimum of f in which f is convex. Using the conditions $H_{ii}(\mathbf{x}) \geq 0$ does not delete such points. Compare this with solving the condition $\mathbf{g}(\mathbf{x}) = 0$ on the gradient. The latter condition deletes all but isolated points. We cannot expect the condition $H_{ii}(\mathbf{x}) \geq 0$ to be as useful.

It is possible to linearize the set of inequalities $H_{ii}(\mathbf{x}) \geq 0$ and use the procedure in Chapter 6 to solve systems of linear inequalities. However, linearization is generally worth doing only when the box over which a function is linearized is small. When the box is small in a optimization algorithm, it is probably because the box contains or is near a solution. In this case, it is probably in the region where the objective function is convex. That is, the condition $H_{ii}(\mathbf{x}) \geq 0$ does not serve to delete points. Therefore, we do not linearize this condition.

Nevertheless, it is worth applying hull consistency to $H_{ii}(\mathbf{x}) \geq 0$ over a box that is at least moderately large. Consider the one-dimensional function

$$f(x) = x^6 - 15x^4 + 27x^2 + 250.$$

This is problem #1 of Levy *et al* (1981). Its “Hessian” is

$$f''(x) = 30x^4 - 180x^2 + 54,$$

which is positive for $|x| < 0.5628$ and for $|x| > 2.384$, approximately. Therefore, hull or box consistency applied to $f''(\mathbf{x}^\mathbf{I}) \geq 0$ is not able to delete any part of an interval X if $|X| < 0.5628$ or if $|X| > 2.384$.

However, $f''(x) < 0$ in the intervals $\pm[0.5628, 2.383]$. If X intersects one of these intervals, box consistency (when iterated) can delete this intersection. Depending on how hull consistency is implemented, it can delete all or part of this intersection. For example, if $X = [-1, 2]$ and we solve

$$30x^4 - 180x^2 + 54 = [0, \infty] \tag{12.7.1}$$

for the term in the left member that is dominant over $[-1, 2]$, we have

$$180x^2 \in 30X^4 + [-\infty, 54] = [-\infty, 534].$$

Since x^2 must be nonnegative, we replace the right member by $[0, 534]$. Solving for x, we find that $x \in \pm[0, 1.723]$ or, equivalently, $[-1.723, 1.723]$.

The intersection of this interval with the original one is $[-1, 1.723]$. Thus, we have deleted part of the original interval. Iterating box consistency produces the interval $[-1, 0.5628]$ in the limit.

When applying hull consistency, we can solve (12.7.1) as a quadratic in x^2 and then solve for x. The interval zeros (as a function of x^2) of

$$30x^4 - 180x^2 + [-\infty, 54] = 0$$

are approximately $[-\infty, 0.3167]$ and $[5.683, \infty]$. Since $x^2 \geq 0$, we replace the first interval zero by $[0, 0.3167]$. The square roots of these intervals are the solution intervals $\pm[0, 0.5628]$ and $\pm[2.384, \infty]$. These are rounded versions of the exact solutions. If the initial interval is $[-a, a]$ with $a > 2.384$, the intersection of this result with $[-a, a]$ reveals that no value of x in the gap $(0.5628, 2.384)$ can be a solution of the optimization problem. Box consistency cannot prove this since the interval value of the function at either endpoint $\pm a$ contains zero.

In the early stages of our optimization algorithm, we apply hull and box consistencies to the relations $H_{ii}(\mathbf{x}) \geq 0$ $(i = 1, \cdots, n)$. We stop doing so when we expect that these relations will be of little or no use in reducing boxes obtained subsequently. We now describe our procedure.

Consider a box $\mathbf{x}^\mathbf{I}$ generated by our optimization algorithm. Suppose we find that $H_{ii}(\mathbf{x}^\mathbf{I}) \geq 0$ for all $i = 1, \cdots, n$. Then we assume that $\mathbf{x}^\mathbf{I}$ is in a basin around a minimum of the objective function. Let w_H denote the largest such box so far generated at a given stage of the algorithm. We do not use the relations $H_{ii}(\mathbf{x}^\mathbf{I}) \geq 0$ for any box of width less than w_H.

This is not a totally satisfactory procedure. If w_H is determined when seeking a solution in a large basin, we might fail to use $H_{ii}(\mathbf{x}^\mathbf{I}) \geq 0$ outside some other small basin. Moreover, the procedure can be useful in a small box which does not contain a minimum of f. A more elaborate procedure might be better. However, we use this simple one. Generally the relation $H_{ii}(\mathbf{x}^\mathbf{I}) \geq 0$ is of little use near a solution. It should not be serious if we stop using it too soon.

There is another way in which we can use the condition $H_{ii}(\mathbf{x}^\mathbf{I}) \geq 0$ that is necessary for convexity. We have noted that the gradient $\mathbf{g}$ of the objective function f is zero at a minimum of f. One procedure for finding such a point is to apply an interval Newton method to solve $\mathbf{g} = 0$. See Section 12.8.

To do so, we linearize $\mathbf{g}$ and solve

$$\mathbf{g}(\mathbf{x}) + \mathbf{J}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})(\mathbf{y} - \mathbf{x}) = 0 \qquad (12.7.2)$$

for $\mathbf{y}$. See Section 11.2. But the Jacobian $\mathbf{J}$ of $\mathbf{g}$ is the Hessian of the objective function f. In Sections 7.4 and 12.7, we noted that the diagonal elements of the Hessian must be nonnegative at a minimum That is, the diagonal elements of $\mathbf{J}^{\mathbf{I}}\left(\mathbf{x}, \mathbf{x}^{\mathbf{I}}\right)$ must be nonnegative when expanding the gradient of f.

Note that certain arguments of elements of $\mathbf{J}^{\mathbf{I}}\left(\mathbf{x}, \mathbf{x}^{\mathbf{I}}\right)$ are real (rather than interval). See Section 7.4. That is why we denote the Jacobian by $\mathbf{J}^{\mathbf{I}}\left(\mathbf{x}, \mathbf{x}^{\mathbf{I}}\right)$ rather than $\mathbf{J}^{\mathbf{I}}(\mathbf{x}^{\mathbf{I}})$. However, one element of $\mathbf{J}^{\mathbf{I}}\left(\mathbf{x}, \mathbf{x}^{\mathbf{I}}\right)$ in each of its rows must have all its arguments as intervals. The sequential expansions to obtain a row of $\mathbf{J}^{\mathbf{I}}\left(\mathbf{x}, \mathbf{x}^{\mathbf{I}}\right)$ can be ordered differently for each row so that it is the diagonal element which has intervals for all its arguments.

We can now conclude that there is no minimum of f in $\mathbf{x}^{\mathbf{I}}$ if a diagonal element of $\mathbf{J}\left(\mathbf{x}, \mathbf{x}^{\mathbf{I}}\right)$ is negative. In our minimization algorithm, this fact is useful because we sometimes check to see if any diagonal element of $\mathbf{J}\left(\mathbf{x}, \mathbf{x}^{\mathbf{I}}\right)$ is negative before we compute the remaining elements of $\mathbf{J}\left(\mathbf{x}, \mathbf{x}^{\mathbf{I}}\right)$. See Step 20 of the algorithm in Section 12.14. However, we can delete any negative part of a computed interval value of a diagonal element of $\mathbf{J}\left(\mathbf{x}, \mathbf{x}^{\mathbf{I}}\right)$. This is a valid step whether or not $\mathbf{x}^{\mathbf{I}}$ contains a minimum of f.

Note that this modification of $J_{ii}(\mathbf{x}^{\mathbf{I}})$ is not valid if $\mathbf{J}$ is obtained using slopes. This is one of the few cases in which derivatives have an advantage over slopes.

12.8 USING A NEWTON METHOD

The solution of an unconstrained optimization problem occurs where the gradient $\mathbf{g}$ of the objective function is zero. Therefore, we can apply an interval Newton method to solve $\mathbf{g}(\mathbf{x}) = 0$ over a box $\mathbf{x}^{\mathbf{I}}$ in which we seek a minimum. In a given step of our optimization algorithm, we apply a

Newton step only if the current box is not “too large”. We decide if this is the case by using the relation (11.11.1).

The gradient $\mathbf{g}$ is zero at any stationary point of the objective function f. We do not want to spend effort to sharply bound such a point if it is not the global minimum. Therefore, we do not want to iterate the Newton method to convergence when solving (12.7.2). Instead, we want to alternate a step of the method with other procedures that might prove a given stationary point is not a global minimum.

Therefore, we make only one “pass” through the Newton method before using other procedures. It is for this reason that we introduced the special interval Newton method of Section 11.14. One “pass” consists of a single application of that method.

The Jacobian can be singular at a global minimum. In this case, the Newton method is not very effective in reducing a box that contains the minimum. Our criterion for when to apply the Newton method causes it to be less frequently used in this case.

12.9 TERMINATION

Because our optimization algorithm often splits a box into subboxes, the number of stored unprocessed subboxes stored can grow. The algorithm can entirely eliminate a given subbox. This generally keeps the number of stored boxes from growing too large.

Splitting and reducing boxes eventually causes any remaining box to be “small”. We require that two conditions be satisfied before a box is deemed to be small enough to be included in the set of solution boxes. First, a box $\mathbf{x}^{\mathbf{I}}$ must satisfy a condition

$$\mathrm{w}(\mathbf{x}^{\mathbf{I}}) \leq \varepsilon_X \tag{12.9.1}$$

where ε_X is specified by the user.

Second, we require

$$\mathrm{w}[f(\mathbf{x}^{\mathbf{I}})] \leq \varepsilon_f; \tag{12.9.2}$$

that is, $\overline{f}(\mathbf{x}^I) - \underline{f}(\mathbf{x}^I) \leq \varepsilon_f$ where ε_f is also specified by the user. Condition (12.9.2) guarantees that the globally minimum value f^* of the objective function is bounded to within the tolerance ε_f. See Section 12.10.

Condition (12.9.1) can be replaced by a set of conditions $w(X_i) \leq \varepsilon_X$ $(i = 1, \cdots, n)$. Thus, scaling can be taken into account and the convergence condition can be dominated by the width of the bound on (say) a single variable. Also, conditions (12.9.1) and (12.9.2) can be replaced or augmented by relative error conditions.

If desired, a user can choose either ε_X or ε_f to be large. This enables a single criterion to control termination.

Care must be taken that ε_f is not so small that rounding errors and/or dependence preclude satisfying (12.9.2). Otherwise, the tolerances can be chosen rather arbitrarily. If ε_X is small and ε_f is large, then (12.9.2) is actually satisfied for a quantity smaller than ε_f because f does not vary much over a small box. If ε_X is large and ε_f is small, then (12.9.1) is satisfied for a quantity smaller than ε_X because (12.9.2) is not satisfied for a large box. Having two tolerances merely allows the user to specify a preference.

Termination conditions on solution boxes can essentially be dispensed with altogether if we are not interested in the point(s) $\mathbf{x}^*$ where the global minimum occurs. Suppose, instead, we are interested only in bounding f^*. In this case we can modify the procedures of Section 12.5 and delete subboxes where $f(\mathbf{x}) > \overline{\mathfrak{f}} - \varepsilon_f$ instead of $f(\mathbf{x}) > \overline{\mathfrak{f}}$. This allows more points to be deleted by the procedure than if we use the inequality $f(\mathbf{x}) > \overline{\mathfrak{f}}$. Eventually, every point of the original box is deleted.

When this occurs, the final value of $\overline{\mathfrak{f}}$ is such that

$$\overline{\mathfrak{f}} - \varepsilon_f \leq f^* \leq \overline{\mathfrak{f}}.$$

That is, f^* is bounded to the required accuracy. Note that with this approach, we can dispense with computations required to check termination conditions.

12.10 BOUNDS ON THE MINIMUM

The algorithm provides both lower and upper bounds on the global minimum f^*. After termination, the solution box (or boxes) must contain the global minimum. Suppose that some number s of boxes remain. Denote them by $\mathbf{x}^{\mathbf{I}(i)}$ $(i = 1, \cdots, s)$.

The algorithm evaluates $f(\mathbf{x}^{\mathbf{I}(i)})$ for each $i = 1, \cdots, s$. Denote the result by

$$f(\mathbf{x}^{\mathbf{I}(i)}) = [\underline{f}(\mathbf{x}^{\mathbf{I}(i)}), \overline{f}(\mathbf{x}^{\mathbf{I}(i)})].$$

Denote

$$\underline{F} = \min_{1 \leq i \leq s} \underline{f}(\mathbf{x}^{\mathbf{I}(i)}).$$

The algorithm also evaluates f at (an approximation for) the center of each box to update the upper bound $\bar{\mathfrak{f}}$ on f^*. A box $\mathbf{x}^{\mathbf{I}(i)}$ is deleted if $\underline{f}(\mathbf{x}^{\mathbf{I}(i)}) > \bar{\mathfrak{f}}$. Therefore,

$$\underline{f}(\mathbf{x}^{\mathbf{I}(i)}) \leq \bar{\mathfrak{f}} \leq \overline{f}(\mathbf{x}^{\mathbf{I}(i)}) \tag{12.10.1}$$

for all $i = 1, \cdots, s$.

Since the global minimum must be in one of the final boxes,

$$\underline{F} \leq f^*. \tag{12.10.2}$$

Let j be an index such that $\underline{f}(\mathbf{x}^{\mathbf{I}(j)}) = \underline{F}$. Letting $i = j$ in (12.10.1), we obtain

$$f^* \leq \bar{\mathfrak{f}} \leq \overline{f}(\mathbf{x}^{\mathbf{I}(j)}). \tag{12.10.3}$$

From the termination condition (12.9.2),

$$\overline{f}(\mathbf{x}^{\mathbf{I}(j)}) - \underline{f}(\mathbf{x}^{\mathbf{I}(j)}) \leq \varepsilon_f. \tag{12.10.4}$$

From (12.10.2), (12.10.3), and (12.10.4),

$$\underline{F} \leq f^* \leq \underline{F} + \varepsilon_f. \tag{12.10.5}$$

Thus, the global minimum is bounded to within ε_f. In general, the upper bound $\overline{\mathfrak{f}}$ on f^* is smaller than $\underline{F} + \varepsilon_f$.

From (12.10.1) and (12.10.4), we conclude that

$$\underline{F} \leq \overline{\mathfrak{f}} \leq \underline{F} + \varepsilon_f.$$

From this relation and (12.10.5), we see that $\overline{\mathfrak{f}}$ and f^* are in the same interval of width ε_f. Therefore, using (12.10.3)

$$f^* \leq \overline{\mathfrak{f}} \leq f^* + \varepsilon_f. \tag{12.10.6}$$

That is, the upper bound $\overline{\mathfrak{f}}$ differs from the global minimum by no more than ε_f.

Note that $f^* \leq \overline{\mathfrak{f}}$. This might be a sharper upper bound on f^* than that given by (12.10.5).

From (12.10.1), (12.10.4), and (12.10.6), we conclude that $f(\mathbf{x}) - f^* \leq 2\varepsilon_f$ for each point $\mathbf{x}$ in each final box.

The accuracy specified in the above relations is guaranteed to be correct for the results computed using our algorithm. This is because we use interval arithmetic to bound rounding errors. In contrast, noninterval algorithms generally cannot guarantee accuracy. This fact is illustrated by a published paper in which run time for a noninterval algorithm is given to obtain eight digit accuracy for the solution to a given problem. However, the reported solution is correct to only four digits.

12.11 THE LIST OF BOXES

Our optimization algorithm normally begins its search in a single given box $\mathbf{x}^{\mathrm{I}(0)}$. For simplicity, our discussion throughout this book usually assumes this to be the case. We can also begin with a set of boxes wherein we seek the global minimum. This is no disadvantage even if the region formed by the set of starting boxes is not connected. We put the initial box(es) in a list L_1 of boxes to be processed.

As the algorithm progresses, it generally divides $\mathbf{x}^{\mathbf{I}(0)}$ into subboxes. This might be done by direct splitting of a box or by removing a gap in a component of a box that is proved to not contain the global minimum. Such a gap might be generated by hull consistency, box consistency, or by an interval Newton method.

Any box $\mathbf{x}^{\mathbf{I}}$ that satisfies the termination criteria

$$\mathrm{w}(\mathbf{x}^{\mathbf{I}}) \leq \varepsilon_{\mathbf{X}} \tag{12.11.1a}$$

$$\mathrm{w}[f(\mathbf{x}^{\mathbf{I}})] \leq \varepsilon_f \tag{12.11.1b}$$

(where f is the objective function) is put in a list L_2. Any box for which these criteria are not both satisfied is placed in a list L_1 of boxes yet to be processed. Assuming both ε_X and ε_f are small, boxes in L_2 are small and f varies very little over any one of them.

12.12 CHOOSING A BOX TO PROCESS

Whenever the algorithm generates a new box $\mathbf{x}^{\mathbf{I}}$, it is placed in the appropriate list as specified in the previous section. As a cycle of the main algorithm begins, the box to be processed is chosen from the list L_1. We now describe how the choice is made.

Before the choice is made, the algorithm evaluates $f(\mathbf{x}^{\mathbf{I}})$ for every box $\mathbf{x}^{\mathbf{I}}$ in L_1. Let $[\underline{f}(\mathbf{x}^{\mathbf{I}}), \overline{f}(\mathbf{x}^{\mathbf{I}})]$ denote the result. The box chosen to be processed is the one for which $\underline{f}(\mathbf{x}^{\mathbf{I}})$ is smallest.

This procedure is more likely to pick the box containing the point $\mathbf{x}^*$ of global minimum than if a box is chosen at random from the list. Therefore, we tend to get a good upper bound $\bar{\mathfrak{f}}$ for the global minimum f^* sooner than for some other choice of box. This speeds convergence because the procedures in Section 12.5 are more effective when $\bar{\mathfrak{f}}$ is smaller.

If the width $\mathrm{w}(\mathbf{x}^{\mathbf{I}})$ of a box $\mathbf{x}^{\mathbf{I}}$ is large, then $\underline{f}(\mathbf{x}^{\mathbf{I}})$ tends to be much smaller than the smallest value of $f(\mathbf{x})$ for any $\mathbf{x} \in \mathbf{x}^{\mathbf{I}}$. This is because of dependence when evaluating $f(\mathbf{x}^{\mathbf{I}})$. See Section 2.4. Therefore, when L_1

contains large boxes, the procedure tends to pick either a box in which f has some small values or else a box in need of reduction so that sharper information can be obtained about f and its derivatives. This reduction of $\mathbf{x}^I$ occurs either because the box is reduced in size by the optimization algorithm or because it is split into parts.

There are other "natural" ways to choose a box from L_1. For example, we can choose the box of largest width or the one that has been in L_1 longest. Experience has shown that the method described above is preferable.

12.13 SPLITTING A BOX

When the algorithm makes little or no progress in reducing the size of a box $\mathbf{x}^I$, it splits $\mathbf{x}^I$ into parts by dividing one or more component of $\mathbf{x}^I$ into smaller parts. We discussed splitting in Section 11.8. The reasons and procedure for doing so are similar when solving an optimization problem and a system of nonlinear equations.

Recall that, when solving systems of nonlinear equations, our procedure for splitting depended on whether Newton's method had been used in the pass though the algorithm preceding the decision to split. This is again true when solving an optimization problem.

If Newton's method has been used, we have the information available to use equation (11.8.1) as a criterion for which components of the box take priority in splitting. This criterion is used here in the same way. The equations in the system being solved are the equations $g_i = 0$ $(i = 1, \cdots, n)$ expressing that the gradient of the objective function is to be zero. The Jacobian elements in equation (11.8.1) are $J_{ij} = \partial g_i / \partial x_j$ $(i, j = 1, \cdots, n)$.

Recall that when Newton's method is used to solve nonlinear systems, we split the box into n subboxes because we have information about how to do so. If Newton's method is not used, we had no such information. We simply split the components of largest width. In the current optimization case, we do have some information when Newton's method has not been used. The width of a gradient element is a measure of how much the objective function changes over the box as a variable changes. This is not of great import unless the box contains the global minimum.

We approximate the change in f resulting from the i-th variable changing over X_i. We use the rather crude approximation

$$D_i(\mathbf{x}^\mathbf{I}) = \mathrm{w}[g_i(\mathbf{x}^\mathbf{I})]\,\mathrm{w}(X_i). \tag{12.13.1}$$

where g_i is the i-th component of the gradient of f. We split the component(s) X_i $(i = 1, \cdots, n)$ of $\mathbf{x}^\mathbf{I}$ for which $D_i(\mathbf{x}^\mathbf{I})$ is largest.

This choice provides better efficiency of the optimization algorithm than splitting the widest component of $\mathbf{x}^\mathbf{I}$. Note that it tends to be independent of variable scaling.

A virtue of this criterion is that little extra computing is required to implement it. In Section 12.4, we noted that we get an approximation for a gradient component $g_i(\mathbf{x}^\mathbf{I})$ evaluated over a box $\mathbf{x}^\mathbf{I}$ when applying hull consistency to the equation $g_i(\mathbf{x}) = 0$. We use this approximation in (12.13.1).

Hull consistency, box consistency, and the interval Newton method can each generate a gap in a component of $\mathbf{x}^\mathbf{I}$ in which the solution to the optimization problem cannot exist. Gaps generated by different procedures might overlap. If so, they are merged. When a gap is deleted, two new subboxes are generated. It is desirable to split a box using a gap because we remove part of the region of search. However, if the gap is quite narrow, little reduction in the search region is made. Therefore, we use a gap for splitting only if it is sufficiently wide to satisfy criterion (11.8.2).

If a gap in the i-th component of a box is sufficiently wide to satisfy (11.8.2), we regard it as one to be used in splitting regardless of the value of $D_i(\mathbf{x}^\mathbf{I})$ from (12.13.1). If there are gaps in more than three components of $\mathbf{x}^\mathbf{I}$, we use (12.13.1) to select the three to be used in splitting. However, instead of using $\mathrm{w}(X_i)$ in (12.13.1), we use the width of the gap in X_i.

If there are fewer than three components with gaps of sufficient width, we use those that occur, and split one or more other components as described earlier in this section.

Note that we do not split a component X_i of $\mathbf{x}^\mathbf{I}$ if it already satisfies the condition $\mathrm{w}(X_i) \leq \varepsilon_X$ and there are other components that do not. However, if all components satisfy this condition, the algorithm might need to continue splitting to satisfy the termination condition $\mathrm{w}[f(\mathbf{x}^\mathbf{I})] \leq \varepsilon_f$. Any splitting is done as described in Section 10.8.

12.14 THE ALGORITHM STEPS

In this section, we list the steps of our algorithm for solving the unconstrained optimization problem. The algorithm computes guaranteed bounds on the globally minimum value f^* of the objective function $f(\mathbf{x})$ and guaranteed bounds on the point(s) where $f(\mathbf{x})$ is a global minimum.

Assume a given box or boxes in which the solution is sought is placed in a list L_1 of boxes to be processed. Set $w_R = 0$ (see Section 11.11). If a single box $\mathbf{x}^{(0)}$ is given, set $w_I = \mathrm{w}(\mathbf{x}^{\mathbf{I}(0)})$. If more than one box is given, set w_I equal to the width of the largest one. If an upper bound on the minimum value f^* of $f(\mathbf{x})$ is known, set $\bar{\mathfrak{f}}$ equal to this value. Otherwise, set $\bar{\mathfrak{f}} = +\infty$.

A box size tolerance ε_X and a function width tolerance ε_f must be specified as described in Section 12.9 by the user.

Let w_H be defined as in Section 12.7 on page 305. The algorithm sets the initial value of $w_H = +\infty$. It also sets w_R and w_I.

We do the following steps in the order given except as indicated by branching. For each step, the current box is denoted by $\mathbf{x}^{\mathbf{I}}$ even though it might be altered in one or more steps.

1. Evaluate f at (an approximation for) the center of each initial box in L_1 and use the result to update $\bar{\mathfrak{f}}$ as described in Section 12.5.

2. For each initial box $\mathbf{x}^{\mathbf{I}}$ in L_1, evaluate $f(\mathbf{x}^{\mathbf{I}})$. Denote the result by $[\underline{f}(\mathbf{x}^{\mathbf{I}}), \overline{f}(\mathbf{x}^{\mathbf{I}})]$. Delete any box for which $\underline{f}(\mathbf{x}^{\mathbf{I}}) > \bar{\mathfrak{f}}$

3. If L_1 is empty go to Step 36. Otherwise, find the box $\mathbf{x}^{\mathbf{I}}$ in L_1 for which $\underline{f}(\mathbf{x}^{\mathbf{I}})$ is smallest. Select this box as the one to be processed next by the algorithm. For later reference denote this box by $\mathbf{x}^{\mathbf{I}(1)}$. Delete $\mathbf{x}^{\mathbf{I}(1)}$ from L_1.

4. If hull consistency has been applied (in Step 6) n times to the relation $f(\mathbf{x}) \leq \bar{\mathfrak{f}}$ without having done Step 9, go to Step 8. (The integer n is the number of variables on which f depends.)

5. Apply hull consistency to the relation $f(\mathbf{x}) \leq \bar{\mathfrak{f}}$. If the result is empty, delete $\mathbf{x^I}$ and go to Step 3.

6. If $\mathrm{w}(\mathbf{x^I}) < \varepsilon_X$ and $\mathrm{w}[f(\mathbf{x^I})] < \varepsilon_f$, put $\mathbf{x^I}$ in list L_2 and go to Step 3.

7. If the box was sufficiently reduced (as defined using (11.7.4)) in Step 5, put the result in L_1 and go to Step 3.

8. If hull consistency has been applied (in Step 9) n times to the components of the gradient without having applied a Newton step (Step 21), go to Step 20.

9. Apply hull consistency to $g_i(\mathbf{x}) = 0$ $(i = 1, \cdots, n)$ for each component $g_i(\mathbf{x})$ of the gradient of $f(\mathbf{x})$. In so doing, use the procedure described in Section 10.10 to bound g_i over the resulting box for use in Step 11. If a result is empty, delete $\mathbf{x^I}$ and go to Step 3.

10. Use the center of the bounds on $g^{\mathbf{I}}{}_i$ from Step 9 to do a line search and update $\bar{\mathfrak{f}}$ as described in Section 12.6. If $\bar{\mathfrak{f}}$ is decreased, repeat Step 5.

11. Using the bounds on $g_i(\mathbf{x^I})$ $(i = 1, \cdots, n)$ obtained in Step 9, apply the linear method of Section 12.5.3 to the relation $f(\mathbf{x}) \leq \bar{\mathfrak{f}}$. If the result is empty, go to Step 3.

12. Repeat Step 6.

13. If the current box $\mathbf{x^I}$ is a sufficiently reduced (using (11.7.4)) version of the box $\mathbf{x^{I^{(1)}}}$ defined in Step 3, put $\mathbf{x^I}$ in list L_1 and go to Step 3.

14. If $\mathrm{w}(\mathbf{x^I}) < w_H$ go to Step 18. (Note that w_H is defined in Section 12.7 on page 305.)

15. Apply hull consistency to the relation $H_{ii}(\mathbf{x^I}) \geq 0$ for $i = 1, \cdots, n$. If the result is empty, go to Step 3. If $H_{ii}(\mathbf{x^I}) \geq 0$ for all $i = 1, \cdots, n$ (which implies that the result from hull consistency is not empty), update w_H as described in Section 10.7 and go to Step 19. [Note that updating w_H is done as follows: If $w_H = +\infty$, simply replace w_H by $\mathrm{w}(\mathbf{x^I})$. Otherwise, replace w_H by the larger of w_H and $\mathrm{w}(\mathbf{x^I})$.]

16. Repeat Step 6.

17. Repeat Step 13.

18. Repeat Steps 5 through 17 using box consistency (as described in Section 10.2) in place of hull consistency. However, skip Steps 10 through 14. In Step 9, omit the process of obtaining bounds on $g_i(\mathbf{x^I})$.

19. If $\mathrm{w}(\mathbf{x^I}) > (w_I + w_R)/2$, go to Step 33. See (11.11.1).

20. Compute the Jacobian $\mathbf{J}(\mathbf{x}, \mathbf{x^I})$ of the gradient $\mathbf{g}$. Order the variables in each row of $\mathbf{J}\left(\mathbf{x}, \mathbf{x^I}\right)$ so that it is the diagonal element for which all arguments are intervals as described in Section 12.7 (and Section 7.4). If a diagonal element of $\mathbf{J}\left(\mathbf{x}, \mathbf{x^I}\right)$ is strictly negative go to Step 3. Otherwise, delete any negative part of any diagonal element of $\mathbf{J}\left(\mathbf{x}, \mathbf{x^I}\right)$. Compute an approximate inverse $\mathbf{B}$ of the approximate center of $\mathbf{J}(\mathbf{x}, \mathbf{x^I})$ and the matrix $\mathbf{M}(\mathbf{x}, \mathbf{x^I}) = \mathbf{BJ}(\mathbf{x}, \mathbf{x^I})$.

21. If the matrix $\mathbf{M}(\mathbf{x}, \mathbf{x^I})$ is regular, find the hull of the solution set of the linear system determined in Step 20 as described in Section 5.8. If $\mathbf{M}(\mathbf{x}, \mathbf{x^I})$ is irregular, apply one pass of the Gauss-Seidel method to the linear system. See Section 5.7. Update w_I or w_R as prescribed in Section 11.11. If the result of the Newton step is empty, go to Step 3. If the interval Newton step proves the existence of a solution in $\mathbf{x^I}$ (see Proposition 11.15.5), record this information.

22. Repeat Step 6.

23. If the width of the box was reduced by a factor of at least 8 in the Newton step (Step 21), go to Step 20.

24. Repeat Step 13.

25. Use the gradient value $\mathbf{g}(\mathbf{x})$ and the matrix $\mathbf{B}$ computed in Step 20 to compute the point $\mathbf{y} = \mathbf{x} - \mathbf{Bg}(\mathbf{x})$. See Section 12.6. Use the value of $f(\mathbf{y})$ to update $\bar{f}$.

26. Use the quadratic method of Section 12.5.4 to “solve” $f(\mathbf{x}) \leq \bar{f}$

27. Repeat Step 6.

28. Repeat Step 13.

29. Using the matrix $\mathbf{B}$ computed in Step 20, analytically determine the system $\mathbf{Bg}(\mathbf{x})$. Apply hull consistency to solve the i-th component of $\mathbf{Bg}(\mathbf{x})$ for the i-th variable x_i for $i = 1, \cdots, n$. If this procedure proves the existence of a solution in $\mathbf{x}^{\mathbf{I}}$ (as discussed in Section 10.12), record this information. **Note:** If the user prefers not to do analytic preconditioning, go to Step 36.

30. Repeat Step 6.

31. Repeat Step 13.

32. Apply box consistency to solve the i-th component of $\mathbf{Bg}(\mathbf{x})$ (as determined in Step 29) for the i-th variable for $i = 1, \cdots, n$.

33. Repeat Step 6.

34. Merge any overlapping gaps in components of $\mathbf{x}^{\mathbf{I}}$ if any were generated using hull consistency, box consistency, and/or the Newton method.

35. Split the box $\mathbf{x}^{\mathbf{I}}$ as prescribed Section 12.13. If gaps that satisfy (11.8.2) have been generated in any of these components, use the gaps to do the splitting. Evaluate f at the center of each new box and use the results to update $\overline{f}$. Then go to Step 3. Note that if multiple processors are use, the number of components to split might be more than three. See Section 11.8.

36. Delete any box $\mathbf{x}^{\mathbf{I}}$ from list L_2 for which $\underline{f}(\mathbf{x}^{\mathbf{I}}) > \bar{\mathfrak{f}}$. Denote the remaining boxes by $\mathbf{x}^{\mathbf{I}(1)}, \cdots, \mathbf{x}^{\mathbf{I}(s)}$ where s is the number of boxes remaining. Determine the lower bound for the global minimum f^* as $\underline{F} = \min\limits_{1 \leq i \leq s} \underline{f}(\mathbf{x}^{\mathbf{I}(i)})$.

37. Terminate.

12.15 RESULTS FROM THE ALGORITHM

After termination, $\underline{F} \leq f^* \leq \bar{\mathfrak{f}}$. Also, $f(\mathbf{x}) - f^* \leq 2\varepsilon_f$ for every point $\mathbf{x}$ in every remaining box. See Section 12.10. Also, $\mathrm{w}(\mathbf{x}^{\mathbf{I}}) \leq \varepsilon_X$ for every remaining box.

A user might want a single point $\widetilde{\mathbf{x}}$ such that

$$||\widetilde{\mathbf{x}} - \mathbf{x}^*|| \leq \varepsilon_1 \tag{12.15.1}$$

and/or

$$f(\widetilde{\mathbf{x}}) - f^* \leq \varepsilon_2 \tag{12.15.2}$$

for some ε_1 and ε_2. Recall that x^* is a point such that $f(\mathbf{x}^*) = f^*$ is the globally minimum value of the objective function f. Our algorithm might or might not determine a point $\widetilde{x}$ that fully satisfies (12.15.1) and (12.15.2).

If $\widetilde{\mathbf{x}}$ is any point in any final box, then $f(\widetilde{\mathbf{x}}) - f^* \leq 2\varepsilon_f$. See (12.10.6). Therefore, (12.15.2) can always be satisfied by choosing $\varepsilon_f = \frac{1}{2}\varepsilon_2$.

If there is only one final box $\mathbf{x}^{\mathbf{I}}$, the algorithm assures that it contains $\mathbf{x}^*$. Therefore, we can choose $\widetilde{\mathbf{x}}$ to be any point in $\mathbf{x}^{\mathbf{I}}$. Since $\mathrm{w}(\mathbf{x}^{\mathbf{I}}) \leq \varepsilon_X$, (12.15.1) is satisfied by choosing $\varepsilon_X = \varepsilon_1$. Also, $f(\widetilde{\mathbf{x}}) - f^* \leq \varepsilon_f$ for any $\widetilde{\mathbf{x}} \in \mathbf{x}^{\mathbf{I}}$ because of the termination condition (12.9.2).

If there is more than one final box, we cannot necessarily satisfy equation (12.15.1). Let $\widetilde{\mathbf{x}}$ be any point in any final box. All we can assure is that $\widetilde{\mathbf{x}}$ is no farther from $\mathbf{x}^*$ than the maximum distance from $\widetilde{\mathbf{x}}$ to any point in any final box. However, $f(\widetilde{\mathbf{x}}) - f^* \leq 2\varepsilon_f$ because this is true for every point in every final box. Decreasing ε_X and ε_f and/or using higher precision arithmetic might improve the bound on $\widetilde{\mathbf{x}} - \mathbf{x}^*$.

12.16 DISCUSSION OF THE ALGORITHM

The algorithm in Section 12.14 begins with procedures that involve the least amount of computing. We use hull consistency first because it does not require computation of derivatives (and because it is effective in reducing large boxes).

For efficiency, the best box $\mathbf{x^I}$ to process is the one for which $\underline{f}(\mathbf{x^I})$ is smallest. This tends to quickly reduce the upper bound $\bar{\mathfrak{f}}$. Because it is important to reduce $\bar{\mathfrak{f}}$ as quickly as possible, the algorithm returns to Step 3 frequently to determine the box to process next.

We stop using the relations $H_{ii}(\mathbf{x}) \geq 0$ $(i = 1, \cdots, n)$ when there is evidence that the remaining boxes are in a region where f is convex. See Step 14. Using a similar philosophy, we begin using the Newton method more often when there is evidence that it will be effective. See Steps 19 and 23.

Note that the Jacobian $\mathbf{J}(\mathbf{x}, \mathbf{x^I})$ is the Hessian of the objective function f. Therefore, knowing $\mathbf{J}(\mathbf{x}, \mathbf{x^I})$ provides the means for determining the second order Taylor expansion of f. The gradient $\mathbf{g}(\mathbf{x})$ needed for this expansion is also computed when applying the Newton method in Step 21.

Therefore, we have the data required to use the quadratic method of Section 12.5.4 to "solve" the relation $f(\mathbf{x}) \leq \bar{\mathfrak{f}}$. See Step 26. If these data are not already available, it might not be worth generating them simply to solve $f(\mathbf{x}) \leq \bar{\mathfrak{f}}$ using the quadratic method.

The algorithm avoids too many applications of hull consistency before changing procedures. Step 4 can force application of hull consistency to the gradient instead of to the inequality $f \leq \bar{\mathfrak{f}}$. Step 8 can force change from applying hull consistency to the gradient, to applying a Newton step. This takes precedence over our desire not to apply a Newton step when it might not be efficient.

We need occasional checks of efficiency of the Newton method because the current box might become so small that the Newton method exhibits quadratic convergence and thus is more efficient than hull consistency. When we force a change from hull consistency, we also force a change from box consistency. This occurs because we do not apply the latter without having previously applied the former.

Step 23 causes the Newton step to be repeated. If the Newton method is exhibiting quadratic convergence, we want to take advantage of its ability to progress rapidly.

In a given pass through Step 23, the algorithm might have proved the existence of a solution of the equation $g(\mathbf{x}) = 0$. Such a "solution" can be a stationary point of f that is not a global minimum. Suppose proof is obtained for some box $\mathbf{x^I}_E$. If $\mathbf{x^I}_E$ does not contain a global minimum, the entire box $\mathbf{x^I}_E$ might or might not be deleted by the algorithm. If L_2 (in Step 36) contains a single final box, it is the one containing a zero of g.

12.17 A NUMERICAL EXAMPLE

We now consider a numerical example. The well-known Beale function can be found as problem #5 of Moré, Garbow, and Hillstrom (1981). It also occurs as problem #2.1 of Schwefel (1981). The problem is to minimize

$$f(x, y) = [1.5 - x(1 - y)]^2 \\ + [2.25 - x(1 - y^2)]^2 + [2.625 - x(1 - y^3)]^2.$$

Van Hentenryck *et al.* (1997) solved this problem using their algorithm Numerica. Their initial box was given by $X = Y = [-10^6, 10^6]$ and the stopping criterion was given by $\varepsilon_X = 10^{-8}$. We used these same parameters and chose ε_f large so that it did not affect our stopping procedure. As a comparison criterion, we counted the number of boxes generated by splitting. Numerica generated 356 boxes. The algorithm given in Section 12.14 generated 36 boxes. This is not a definitive comparison because the computational effort per box is not compared.

Walster, Hansen, and Sengupta (1985) solved this problem beginning with the much smaller box given by $X = Y = [-4.5, 4.5]$ and obtained a bounding box of width 10^{-11}. This required 315 applications of the interval Newton method (as well as other procedures). For the much larger initial box of width 2×10^6, the algorithm of Section 12.14 needed only 18 Newton applications. Again, this is an incomplete comparison. However, it illustrates the virtue of hull and box consistency when used together with the interval Newton method.

12.18 MULTIPLE MINIMA

Interval algorithms are capable of finding the global minimum of a function with many local minima. Walster, Hansen, and Sengupta (1985) used an earlier version of our algorithm to solve various optimization problems. They report solving a ten dimensional problem in a box containing 10^{10} local minima and a single global minimum.

A given objective function can have more than one point where it takes on its globally minimum value. If they are well separated, each global minimum is isolated and separately bounded by our algorithm. Nearly coincident but distinct solution points are separately bounded only if the prescribed error tolerances are sufficiently small and word length of the computer is adequate.

A function can have a continuum of global solution points. If the error tolerances are small, then, in this case, our algorithm computes a large number of boxes covering the set of solution points. A user might want such a result in which a solution region is mapped out by small "pixels". However, it is possible to avoid such a result by bounding f^* only, but not $\mathbf{x}^*$. We can modify the procedure in Section 12.5 to eliminate points $\mathbf{x}$ for which $f(\mathbf{x}) > \bar{\mathfrak{f}} - \varepsilon_f$ rather $f(\mathbf{x}) > \bar{\mathfrak{f}}$. This causes *all* points in the initial box to be eliminated. That is, we preclude the need to determine the large set of boxes covering the continuum of solution points.

Using this option, we bound the global minimum f^* by $\bar{\mathfrak{f}} - \varepsilon_f \leq f^* \leq \bar{\mathfrak{f}}$. Since the upper bound $\bar{\mathfrak{f}}$ is obtained at some point $\bar{\mathbf{x}}$, we have a representative point where the value of f is within ε_f of f^*. Walster, Hansen and Sengupta (1985) discuss this option and provide an example of its use.

12.19 NONDIFFERENTIABLE PROBLEMS

Various procedures in the algorithm given in Section 12.14 require some degree of continuous differentiability of the objective function f. If this is not the case certain procedures must be omitted. For example, to apply box consistency or a Newton method to solve $\mathbf{g}(\mathbf{x}) = 0$ (where $\mathbf{g}$ is the gradient of f), f must be twice continuously differentiable. Exceptions might occur if **g** exists but is not differentiable. In this case, it might be possible to expand **g** in slopes rather than derivatives. See Section 7.7.

The algorithm solves the global optimization problem even if f is not differentiable. But, in this case the algorithm is slower. Applying hull consistency to the relation $f(\mathbf{x}) \leq \bar{f}$ does not require differentiability. This procedure alone (with box splitting) can solve the problem. For an example in which this procedure performs well, see Section 12.5.5.

Sometimes, a nondifferentiable objective function can be replaced by one having differentiability. See Section 18.1.

12.20 FINDING ALL STATIONARY POINTS

There are applications in which one wants to find all stationary points of a function. There are other applications in which one wants to find all local minima whether they are global or not. In this section, we discuss how our procedures can be applied to compute such results.

Note that all stationary points of a function in a box can be found by applying the procedure in Section 11.12 to solve the system of equations formed by setting to zero the components of the gradient of the given function. However, our optimization algorithm can also be used for this purpose.

In Section 12.5, we discussed how an upper bound on the global minimum can be used to delete local minima. If we omit the procedures of Section 12.5 from the algorithm of Section 12.14, the resulting algorithm finds all (global or local) minima of the objective function in the initial box.

If we wish to find all stationary points of the objective function, we can do so by omitting an additional procedure. We also omit the procedure described in Section 12.7 that deletes points of the box at which the objective function is not convex.

Chapter 13

CONSTRAINED OPTIMIZATION

13.1 INTRODUCTION

In this chapter, we consider the constrained optimization problem

$$\begin{aligned} &\text{Minimize } f(\mathbf{x}) && (13.1.1)\\ &\text{subject to } p_i(\mathbf{x}) \le 0 \;\; (i = 1, \cdots, m)\,,\\ &\qquad\qquad q_i(\mathbf{x}) = 0 \;\; (i = 1, \cdots, r)\,. \end{aligned}$$

We assume $f(\mathbf{x})$ is twice continuously differentiable and that the constraint functions $p_i(\mathbf{x})$ and $q_i(\mathbf{x})$ are continuously differentiable.

As in the unconstrained case, we assume an initial box $\mathbf{x}^{\mathbf{I}(0)}$ is given. We seek the global minimum of $f(\mathbf{x})$ in $\mathbf{x}^{\mathbf{I}(0)}$ subject to the given constraints. If an inequality constraint in (13.1.1) is of the form $a - x_i \le 0$ or $x_i - b \le 0$, then this determines a side of $\mathbf{x}^{\mathbf{I}(0)}$.

Unless a side of $\mathbf{x}^{\mathbf{I}(0)}$ is explicitly prescribed to be an inequality constraint, this particular bound is not regarded as a constraint. Instead, it merely restricts the region of search. We assume the box is sufficiently large to contain any global solution point of the problem as given by (13.1.1).

As in the unconstrained case, the initial box can generally be chosen quite large without seriously degrading the performance of the optimization algorithm. If no initial box is given, we assume each of its components (not specified by an inequality constraint) is $\left[-\overline{\mathrm{N}}, \overline{\mathrm{N}}\right]$ where $\overline{\mathrm{N}}$ is the largest finite number representable on the computer used to solve the problem.

Our approach is the same as in the unconstrained case. We delete subboxes of $\mathbf{x}^{\mathbf{I}(0)}$ that cannot contain the global solution. We stop when the bounds on the location of the solution and the bounds on the globally minimum value of f are small enough to satisfy user specified tolerances.

Our algorithms for constrained and unconstrained problems use so many of the same subroutines that we use a single program to solve both types of problem. We call those subroutines that are relevant for a particular problem. However, for pedagogical reasons, we describe separate algorithms for constrained and unconstrained problems.

Robinson (1973) was the first to use interval arithmetic to bound rounding errors in obtaining an approximate solution of (13.1.1). However, he did not otherwise use interval methods to compute the solution. He also did not attempt to find the global solution.

An important problem in interval analysis is that of bounding the range of a function over a given box. We can cast this problem as two optimization problems in which we want both the minimum and maximum of the function subject to box constraints.

Methods for bounding the range of a function over a box can be regarded as the first interval methods for global optimization. Thus, it can be said that the effort to use interval analysis to solve a global optimization began with Moore (1966). However, this approach does not include constraints.

A discussion of this special problem and methods for it can be found in Ratschek and Rokne (1984). We do not discuss these methods. One reason is that there is no effort to find the location of the global solution. Only bounds on a function over a box are sought. A more general discussion of interval methods for global optimization can be found in Ratshek and Rokne (1988). They discussed some of the easily implemented procedures derived in Hansen (1980) that suffice for finding a solution. However, they omitted other more effective (but more complicated) procedures that enhance performance.

Hansen and Sengupta (1980) first used interval methods of the kind considered herein to solve the inequality constrained problem. See also

Hansen and Walster (1992b). We discuss this special case in Chapter 14. We discuss the equality constrained case in Chapter 15. See also Hansen and Walster (1992a, b, c). For a more recent discussion of global optimization with details on software implementation of interval procedures, see Kearfott (1996).

13.2 THE JOHN CONDITIONS

In this section, we discuss the John conditions that necessarily hold at a constrained (local or global) minimum. We use these conditions in solving problem (13.1.1). We write the John conditions as

$$u_0 \nabla f(\mathbf{x}) + \sum_{i=1}^{m} u_i \nabla p_i(\mathbf{x}) + \sum_{i=1}^{r} v_i \nabla q_i(\mathbf{x}) = 0, \tag{13.2.1a}$$

$$u_i p_i(\mathbf{x}) = 0 \ \ (i = 1, \cdots, m), \tag{13.2.1b}$$

$$q_i(\mathbf{x}) = 0 \ \ (i = 1, \cdots, r), \tag{13.2.1c}$$

$$u_i \geq 0 \ \ (i = 0, \cdots, m) \tag{13.2.1d}$$

where $u_0, \cdots, u_m, v_1, \cdots, v_r$ are Lagrange multipliers.

The John conditions differ from the more commonly used Kuhn-Tucker-Karush conditions because they include the Lagrange multiplier u_0. If we set $u_0 = 1$ and omit the normalization condition (13.2.2a) or (13.2.2b) below, then we obtain the Kuhn-Tucker-Karush conditions.

When the Kuhn-Tucker-Karush conditions are used, it is assumed that the binding constraints are not linearly dependent at a minimum. That is, constraint qualifications are imposed. For definitions and discussion of these terms and concepts, see, for example, Bazaraa and Shetty (1979).

We prefer not to restrict the set of problems considered by imposing constraint qualifications. A problem with linearly dependent binding constraints can arise in practice. If so, an optimization algorithm (including ours) can fail to find the global minimum if it assumes otherwise. We avoid such failure by using the John conditions.

For a discussion of the Kuhn-Tucker-Karush conditions in an interval context, see Mohd (1990).

The John conditions do not normally include a normalization condition. Therefore, there are more variables than equations in (13.2.1a) through (13.2.1d). One is free to remove the ambiguity in whatever way desired. A normalization can be chosen arbitrarily without changing the solution x of the John conditions. For reasons given in Sections 13.3 and 13.5, we consider two separate normalizations.

The first normalization is linear; but it is not as simple as one might expect. As we discuss in the next section, we use

$$u_0 + \cdots + u_m + E_1 v_1 + \cdots + E_r v_r = 1 \tag{13.2.2a}$$

where $E_i = [1, 1 + \varepsilon_0]$ for all $i = 1, \cdots, r$. The constant ε_0 is the smallest positive machine number such that in the number system on the computer used, $1 + \varepsilon_0$ is represented as a number > 1. Actually a slightly larger value for ε_0 can be used without error and without seriously degrading sharpness.

The second normalization is

$$u_0 + \cdots + u_m + v_1^2 + \cdots + v_r^2 = 1. \tag{13.2.2b}$$

13.3 NORMALIZING LAGRANGE MULTIPLIERS

In this section, we discuss normalization of the Lagrange multipliers and explain why the linear normalization (13.2.2a) takes the given form. The normalization is due to Hansen and Walster (1990a).

If no equality constraints are present in problem (13.1.1), the normalization equation (13.2.2a) becomes

$$u_0 + \cdots + u_m = 1.$$

Since the Lagrange multipliers u_i $(i = 0, \cdots, m)$ are nonnegative (see (13.2.1d)), this assures that

$$0 \le u_i \le 1 \ \ (i = 0, \cdots, m).$$

These bounds on u_i are useful in our algorithm.

The multipliers v_i $(i = 1, \cdots, r)$ can be positive or negative. Therefore, we cannot use the normalization

$$u_0 + \cdots + u_m + v_1 + \cdots + v_r = 1$$

since the left member might be zero for the solution to a given problem.

A possible alternative normalization is

$$u_0 + \cdots + u_m + |v_1| + \cdots + |v_r| = 1. \tag{13.3.1}$$

However, we want a normalization equation that is continuously differentiable so we can apply an interval Newton method to solve the John conditions. Therefore, we reject this alternative normalization and use (13.2.2a) or (13.2.2b).

To explain why the normalization (13.2.2a) has the form it does, consider the sum

$$S = u_0 + \cdots + u_m + v_1 + \cdots + v_r. \tag{13.3.2}$$

We want to use the simple normalization $S = 1$. However, if $S = 0$ (at a solution), we have a contradiction.

In Section 13.2, we define the interval $E = [1, 1+\varepsilon_0]$ where the constant ε_0 is the smallest positive machine number such that in the number system of the computer, $1 + \varepsilon_0$ is represented as a number > 1.

Even when $S = 0$, there are numbers $e_i \in E_i = E$ $(i = 1, \cdots, r)$ such that

$$u_0 + \cdots + u_m + e_1 v_1 + \cdots + e_r v_r = 1 \tag{13.3.3}$$

after an appropriate renormalization of the Lagrange multipliers. Since this equation is contained in the interval equation (13.2.2a), the normalization (13.2.2a) is valid when $S = 0$. It is obviously valid when $S \neq 0$.

Since $E_i = E = [1, 1 + \varepsilon_0]$ for all $i = 1, \cdots, r$, one might be tempted to write (13.2.2a) in the factored form

$$u_0 + \cdots + u_m + E(v_1 + \cdots + v_r) = 1.$$

If both $u_0 + \cdots + u_m = 0$ and $v_1 + \cdots + v_r = 0$, then there is no $e \in E$ such that

$$u_0 + \cdots + u_m + e(v_1 + \cdots + v_r) = 1.$$

Therefore, we must not use the factored form. By using the non-factored form (13.2.2a), we use the fact that interval arithmetic is not distributive. This is an example of the need to carefully distinguish between interval variables that are independent and those that are not. (See Chapter 4.)

Suppose we apply our optimization algorithm to a subbox $\mathbf{x}^\mathbf{I}$ of the initial box $\mathbf{x}^{\mathbf{I}(0)}$. Suppose also that, for any solution in $\mathbf{x}^\mathbf{I}$, the interval bounds u_i^I on u_i $(i = 0, \cdots, m)$ and bounds v_i^I on v_i $(i = 1, \cdots, r)$ hold and that

$$0 \notin u_0^I + \cdots + u_m^I + v_1^I + \cdots + v_r^I.$$

Then we know that $S \neq 0$ for any solution point in $\mathbf{x}^\mathbf{I}$. Therefore, whenever considering $\mathbf{x}^\mathbf{I}$ or a subbox of $\mathbf{x}^\mathbf{I}$, we replace the normalization condition (13.2.2a) by the simpler condition $S = 1$.

The other normalization we consider is given by (13.2.2b). It has value because it immediately provides useful bounds on the Lagrange multipliers. Since $u_i \geq 0$ $(i = 0, \cdots, m)$, (13.2.2b) yields the bounds

$$\begin{aligned} 0 \leq u_i \leq 1 \;\; (i = 0, \cdots, m)\,, \\ -1 \leq v_i \leq 1 \;\; (i = 1, \cdots, r)\,. \end{aligned} \tag{13.3.4}$$

The undesirable feature of this normalization is that it is not linear. In Section 13.5, we show why this is a drawback. In Section 15.2, we discuss another advantage of the linear normalization (13.2.2a).

For convenience, we express the two normalizations (13.2.2a) and (13.2.2b) in terms of functions. Thus, we define

$$R_1(\mathbf{u}, \mathbf{v}) = u_0 + \cdots + u_m + Ev_1 + \cdots + Ev_r - 1 \tag{13.3.5}$$

and

$$R_2(\mathbf{u}, \mathbf{v}) = u_0 + \cdots + u_m + v_1^2 + \cdots + v_r^2 - 1. \tag{13.3.6}$$

We solve the John conditions using an interval Newton method. See Section 13.5. When we do so, we linearize the equations to be solved. The bounds (13.3.4) enable us to use a linearized version of (13.2.2b) that we express using $R_2(\mathbf{u}, \mathbf{v})$ as given by (13.3.6). Expanding $R_2(\mathbf{u}, \mathbf{v})$ using the Taylor expansion (7.3.6), we obtain

$$R_2(\mathbf{u}, \mathbf{v}) \in R_2(\mathbf{u}', \mathbf{v}') + (u_0 - u_0') + \cdots + (u_m - u_m') \\ + 2V_1(v_1 - v_1') + \cdots + 2V_r(v_r - v_r'), \qquad (13.3.7)$$

which is valid for all $\mathbf{u} \in \mathbf{u}^{\mathbf{I}}$ and all $\mathbf{v} \in \mathbf{v}^{\mathbf{I}}$ where $\mathbf{u}^{\mathbf{I}}$ and $\mathbf{v}^{\mathbf{I}}$ are interval vectors and V_i $(i = 1, \cdots, n)$ is the i-th component of $\mathbf{v}^{\mathbf{I}}$. The real vectors $\mathbf{u}' \in \mathbf{u}^{\mathbf{I}}$ and $\mathbf{v}' \in \mathbf{v}^{\mathbf{I}}$ are fixed. Using the bounds (13.3.4), we see that we can replace condition (13.2.2b) by its an expansion in the form

$$R_2(\mathbf{u}', \mathbf{v}') + (u_0 - u_0') + \cdots + (u_m - u_m') + [-2, 2](v_1 - v_1') + \cdots \\ + [-2, 2](v_r - v_r') = 0. \qquad (13.3.8)$$

By now, readers have observed that it is far simpler to use the normalization $u_0 = 1$ than to normalize as we have done. This can be done. It produces the Kuhn-Tucker-Karush conditions. It simplifies the formulation and slightly improves the efficiency of the algorithm. The only difficulty with using $u_0 = 1$ is that we might fail to find the global minimum in the rather rare case in which the constraints are linearly dependent at the solution. Consistent with the promise of or interval algorithms to never fail to produce bounds on the set of all problem solutions, we choose not to risk failing to find the global solution in this special circumstance. Consequently, we do not use the normalization $u_0 = 1$. Instead, we avoid this risk by performing the extra computation required to use either (13.2.2a) or (13.2.2b).

There is another simple expedient. We can write the equality constraints $q_i(\mathbf{x}) = 0$ $(i = 0, \cdots, r)$ as two inequality constraints $q_i(\mathbf{x}) \le 0$ and $-q_i(\mathbf{x}) \le 0$. Now all constraints are inequality constraints. Therefore, we can use the simple normalization

$$u_0 + \cdots + u_s = 1$$

where $s = m + 2r$ and the initial bounds are $0 \leq u_i \leq 1$ $(i = 1, \cdots, s)$. We have not used this expedient because it introduces r extra Lagrange multipliers.

13.4 USE OF CONSTRAINTS

When we solve an optimization problem, we seek a solution $\mathbf{x}^*$ in a given box $\mathbf{x}^{\mathbf{I}}$. When we do so, we apply hull consistency (see Chapter 10) to the constraints over the box $\mathbf{x}^{\mathbf{I}}$. Therefore, $\mathbf{x}^{\mathbf{I}}$ is deleted in the process of applying hull consistency if it is certainly infeasible.

Also, we might have found that $p_i(\mathbf{x}^{\mathbf{I}}) < 0$ for some inequality constraint. If so, then $p_i(\mathbf{x}) < 0$ for all $\mathbf{x} \in \mathbf{x}^{\mathbf{I}}$, which implies that this particular constraint cannot be active at any point $\mathbf{x} \in \mathbf{x}^{\mathbf{I}}$. Such a constraint can be ignored when trying to find a solution to the optimization problem in $\mathbf{x}^{\mathbf{I}}$. Whenever we discuss solving the John conditions, we assume that such inequality constraints have been omitted. However, for simplicity, we still use the letter m to denote the number of possibly active inequality constraints being considered.

13.5 SOLVING THE JOHN CONDITIONS

We now consider use of an interval Newton method to solve the portion of the John conditions given by Equations (13.2.1a) through (13.2.1c). We include either (13.2.2a) or (13.2.2b). The remaining conditions $u_i \geq 0$ $(i = 1, \cdots, m)$ are not part of this computation since they are not equations. They are used after the equations are solved. Our discussion parallels that of Hansen and Walster (1990b).

We specify whether the normalization is given by (13.2.2a) or (13.2.2b), but only when it matters which normalization is used.

From the vectors $\mathbf{x} = (x_1, \cdots, x_n)^T$, $\mathbf{u} = (u_0, \cdots, u_m)^T$, and $\mathbf{v} = (v_1, \cdots, v_r)^T$, we define the partitioned vectors

$$\mathbf{w} = \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix}, \; \mathbf{t} = \begin{bmatrix} \mathbf{x} \\ \mathbf{w} \end{bmatrix} = \begin{bmatrix} \mathbf{x} \\ \mathbf{u} \\ \mathbf{v} \end{bmatrix}.$$

We write the John conditions (13.2.1a) through (13.2.1c) and (13.2.2) in terms of the vector $\mathbf{t}$.

To do so, we change the notation of the normalization functions (13.3.5) or (13.3.6) from $R_k\,(\mathbf{u}, \mathbf{v})$ to $R_k(\mathbf{t})$ ($k = 1$ or 2). Thus, we write the relevant John conditions (with $k = 1$ or 2) as

$$\phi(\mathbf{t}) = \begin{bmatrix} R_k(\mathbf{t}) \\ u_0 \nabla f(\mathbf{x}) + \sum\limits_{i=1}^{m} u_i \nabla p_i(\mathbf{x}) + \sum\limits_{i=1}^{r} v_i \nabla q_i(\mathbf{x}) \\ u_1 p_1(\mathbf{x}) \\ \vdots \\ u_m p_m(\mathbf{x}) \\ q_1(\mathbf{x}) \\ \vdots \\ q_r(\mathbf{x}) \end{bmatrix} \tag{13.5.1}$$

Denote $N = n + m + r + 1$. Let $\mathbf{t}^{\mathbf{I}}$ be an N-dimensional box containing the vector $\mathbf{t}$. Let $\mathbf{J}(\mathbf{t}, \mathbf{t}^{\mathbf{I}})$ denote the Jacobian of $\phi(\mathbf{t})$. As pointed out in Section 7.4, the elements of $\mathbf{J}(\mathbf{t}, \mathbf{t}^{\mathbf{I}})$ can be expressed as

$$J_{ij}(\mathbf{t}, \mathbf{t}^{\mathbf{I}}) = \frac{\partial}{\partial t_j} \phi_i(T_1, ...T_j, t_{j+1}, \cdots, t_N)^{\cdot} \tag{13.5.2}$$

for $i, j = 1, \cdots, N$, where $\phi_i\,(\mathbf{t})$ is the i-th component of $\phi\,(\mathbf{t})$. Note that $N - j$ of the arguments of $J_{ij}(\mathbf{t}, \mathbf{t}^{\mathbf{I}})$ are real.

Suppose we use the linear normalization (13.2.2a) in defining $\phi(\mathbf{t})$. That is, suppose the first component of $\phi(\mathbf{t})$ is $R_1(\mathbf{t})$ as given by (13.3.5). By choosing the Lagrange multipliers to be the arguments that are real in (13.5.2), we assure the elements of $\mathbf{J}(\mathbf{t}, \mathbf{t}^{\mathbf{I}})$ do not involve the Lagrange multipliers as intervals. Because of this fact, no initial bounds for the Lagrange multipliers are needed to solve the equation $\phi(\mathbf{t}) = \mathbf{0}$. Note that such bounds are needed only if an appropriate interval Newton method is used.

The Krawczyk and Gauss-Seidel interval Newton methods are not appropriate because they require initial bounds on all (except one) of the variables. See Section 11.2. Gaussian elimination or the hull method of Section 5.8 is appropriate.

An expansion using slopes (see Section 7.7) can also be obtained in such a way that a variable occurring linearly in the function being expanded does not occur as an interval in the "Jacobian".

Alternatively, suppose we use the nonlinear normalization (13.2.2b). Then the Lagrange multipliers occur as intervals in the Jacobian of the John conditions. Every additional interval introduced into a system of equations increases the chance that the Jacobian of the system is irregular. When the Jacobian is irregular, an interval Newton method is either unable to obtain a solution or is less efficient. In this case, it is more often necessary to split the components of the box in some way. But, if we have to split the interval bounds on the Lagrange multipliers as well as those of the other variables, a great deal of extra computing might be necessary. Therefore, the linear normalization is the better choice.

When the linear normalization is used, a step of an interval Newton method provides bounds on the Lagrange multipliers. See the next section. Thereafter, any variant of an interval Newton method can be used.

Assume the Jacobian elements are computed as defined in (13.5.2). To avoid having the Lagrange multipliers occur as intervals in the Jacobian, it is essential to order the variables so that the vector variables **u** and **v** occur after **x** when defining **t**.

Consider boxes (of appropriate dimensions) $\mathbf{x}^{\mathbf{I}}$, $\mathbf{u}^{\mathbf{I}}$, and $\mathbf{v}^{\mathbf{I}}$ bounding the vectors **x**, **u**, and **v**, respectively. They define a box

$$\mathbf{t}^{\mathbf{I}} = \begin{bmatrix} \mathbf{x}^{\mathbf{I}} \\ \mathbf{u}^{\mathbf{I}} \\ \mathbf{v}^{\mathbf{I}} \end{bmatrix}.$$

We linearize $\phi^{\mathbf{I}}$ as

$$\phi^{\mathbf{I}}(\mathbf{t}_0) + \mathbf{J}(\mathbf{t}_0, \mathbf{t}^{\mathbf{I}})(\mathbf{t} - \mathbf{t}_0) = 0 \tag{13.5.3}$$

where $\mathbf{t}_0$ is a real vector in $\mathbf{t}^{\mathbf{I}}$ and the Jacobian $\mathbf{J}(\mathbf{t}_0, \mathbf{t}^{\mathbf{I}})$ is given by (13.5.2). A good choice for $\mathbf{t}_0$ is the center of $\mathbf{t}^{\mathbf{I}}$.

Assume we use the linear normalization (13.2.2a). Suppose we solve (13.5.3) using a Newton method from Chapter 11 for which Proposition 11.15.5 holds. Denote the result by $\mathbf{t}^{\mathbf{I}'}$. If $\mathbf{t}^{\mathbf{I}'} \subset \mathbf{t}^{\mathbf{I}}$, we have proof of the existence of a solution of $\phi(\mathbf{t}) = \mathbf{0}$ for $\mathbf{t} \in \mathbf{t}^{\mathbf{I}'}$. This condition $\mathbf{t}^{\mathbf{I}'} \subset \mathbf{t}^{\mathbf{I}}$ can be expressed as $\mathbf{x}^{\mathbf{I}'} \subset \mathbf{x}^{\mathbf{I}}$, $\mathbf{u}^{\mathbf{I}'} \subset \mathbf{u}^{\mathbf{I}}$, and $\mathbf{v}^{\mathbf{I}'} \subset \mathbf{v}^{\mathbf{I}}$. However, we do not define $\mathbf{u}^{\mathbf{I}}$ or $\mathbf{v}^{\mathbf{I}}$ when the linear normalization (13.2.2a) is used as indicated.

We can assume that each component of $\mathbf{v}^{\mathbf{I}}$ is $[-\infty, \infty]$ and therefore, the condition $\mathbf{v}^{\mathbf{I}'} \subset \mathbf{v}^{\mathbf{I}}$ is satisfied for any finite $\mathbf{v}^{\mathbf{I}'}$. However, the relation (13.2.1d), which is part of the John conditions, states that $u_i \geq 0$ for all $i = 1, \cdots, m$. Therefore, we must assume that $\mathbf{u}^{\mathbf{I}} \geq 0$. This implies that the condition $\mathbf{u}^{\mathbf{I}'} \geq 0$ must be satisfied to prove existence of a solution of $\phi(\mathbf{t}) = \mathbf{0}$.

On the other hand, if $\mathbf{u}^{\mathbf{I}'}_i < 0$ for any $i = 1, \cdots, m$, there can be no solution of $\phi^{\mathbf{I}}(\mathbf{x}) = \mathbf{0}$ for $\mathbf{t}$ in $\mathbf{t}^{\mathbf{I}}$. In this case, we can delete $\mathbf{x}^{\mathbf{I}}$. We obtain $\mathbf{u}^{\mathbf{I}'}$ from a Newton step no matter what normalization we use for the Lagrange multipliers

If the Jacobian of $\phi^{\mathbf{I}}$ contains a singular matrix (i.e., is not regular), then the solution set of (13.5.3) is unbounded. In this case, Gaussian elimination or the hull method of Section 5.8 fails. If we use the normalization (13.2.2b), the presence of interval bounds for the Lagrange multipliers makes it more likely that the Jacobian is irregular. This is yet another argument in favor of the linear normalization (13.2.2a). On the other hand, when the Jacobian is irregular, it is sometimes possible to improve the bounds on some of the components of $\mathbf{t}$ using a Gauss-Seidel step. This is not possible without bounds on the multipliers.

13.6 BOUNDING THE LAGRANGE MULTIPLIERS

When we use the linear normalization condition (13.2.2a), then some forms of the interval Newton method do not need initial bounds on the Lagrange multipliers. However, there are three ways in which computed bounds can be useful. First, suppose we obtain a bound U_i on a Lagrange multiplier u_i and find that $U_i < 0$. Then condition (13.2.1d) is violated for all $u_i \in U_i$; and there cannot be a solution in whatever box $\mathbf{x}^{\mathbf{I}}$ is used to compute U_i.

Such a result can be obtained whether or not there are input bounds on the Lagrange multipliers. Of course, if there are input bounds on $\mathbf{u}$ and $\mathbf{v}$, there are more such ways to prove nonexistence of a solution of the John conditions.

Second, if we choose to apply hull consistency or box consistency to the John conditions, we need bounds on the multipliers. In our algorithm, we do not apply hull or box consistency to the John conditions *per se*. However, we do apply them to constraint equations. A user can choose to apply them to the John conditions.

Although we do not need bounds on the Lagrange multipliers to apply some forms of Newton's method to solve the John conditions, we do need estimates. A third way to use bounds on the multipliers is to use the midpoint of an interval bound of a multiplier as an estimate for its value.

In this section, we show how such bounds can be computed. The procedure is relevant only when the linear normalization is used.

When we compute bounds on the Lagrange multipliers, they are valid for all points in a given box $\mathbf{x}^{\mathbf{I}}$. Suppose that, later, we have a box $\mathbf{x}^{\mathbf{I}'}$ that is a subbox of $\mathbf{x}^{\mathbf{I}}$. Then the bounds on the multipliers for points in $\mathbf{x}^{\mathbf{I}}$ are bounds for points in $\mathbf{x}^{\mathbf{I}'}$. Therefore, it is not necessary to use the procedure of this section when processing $\mathbf{x}^{\mathbf{I}'}$.

We noted in Section 13.5 that by solving $\phi(\mathbf{t}) = \mathbf{0}$ (as given by (13.5.1)) using an interval Newton method, we can compute bounds on the Lagrange multipliers. This requires that we successfully solve the linearized Equation (13.5.3) when using Gaussian elimination or the hull method. We now consider an alternative procedure due to Hansen and Walster (1990) for computing such bounds. It involves fewer equations than those in the relation $\phi(\mathbf{t}) = \mathbf{0}$.

Instead of using all of equations (13.2.1a) through (13.2.1c) and (13.2.2a) or (13.2.2b), we use only (13.2.1a) and (13.2.2a). Thus, the number of equations is reduced from $n + m + r + 1$ to $n + 1$. We assume that $m + r \leq n$.

Assume we seek a solution of the minimization problem (13.1.1) in a subbox $\mathbf{x}^{\mathbf{I}}$ of the initial box $\mathbf{x}^{\mathbf{I}(0)}$. Let $\mathbf{x}$ be a point in $\mathbf{x}^{\mathbf{I}}$. Define the $(n + 1)$

by $(m + r + 1)$ matrix

$$\mathbf{A}(\mathbf{x}) = \begin{bmatrix} 1 & 1 & \cdots & 1 & E_1 & \cdots & E_r \\ \nabla f(\mathbf{x}) & \nabla p_1(\mathbf{x}) & \cdots & \nabla p_m(\mathbf{x}) & \nabla q_1(\mathbf{x}) & \cdots & \nabla q_r(\mathbf{x}) \end{bmatrix}.$$

where E_i $(i = 1, \cdots, r)$ is defined in Section 13.2. Equations (13.2.2a) and (13.2.1a) can be written

$$\mathbf{A}(\mathbf{x})\mathbf{w} = e_1 \tag{13.6.1}$$

where $e_1 = (1, 0, \cdots, 0)^T$ is a vector of $n + 1$ components and

$$\mathbf{w} = \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix}$$

has $m + r + 1$ components.

Consider the set of vectors $\mathbf{w}$ satisfying (13.6.1) as $\mathbf{x}$ ranges over $\mathbf{x}^{\mathbf{I}}$. This set contains the vector of Lagrange multipliers that satisfy the John conditions for any $\mathbf{x} \in \mathbf{x}^{\mathbf{I}}$. We replace $\mathbf{x}$ by $\mathbf{x}^{\mathbf{I}}$ in (13.6.1) and obtain

$$\mathbf{A}(\mathbf{x}^{\mathbf{I}})\mathbf{w} = e_1. \tag{13.6.2}$$

The solution of this equation provides the desired bounds on the Lagrange multipliers.

We wish to apply Gaussian elimination to this equation to transform the (nonsquare) coefficient matrix into upper trapezoidal form. To do so, we precondition the problem by multiplying by a real transformation matrix as described for the square matrix case in Section 5.6.

Let $\mathbf{A}^c$ denote the center of $\mathbf{A}(\mathbf{x}^{\mathbf{I}})$. Using (real) Gaussian elimination with row interchanges, we determine a matrix $\mathbf{B}$ that transforms $\mathbf{A}^c$ into upper trapezoidal form. We then apply interval Gaussian elimination (without row interchanges) to the preconditioned equation

$$\mathbf{BA}(\mathbf{x}^{\mathbf{I}})\mathbf{w} = \mathbf{B}e_1$$

to transform its coefficient matrix into upper trapezoidal form.

This procedure can fail because of division by an interval containing zero. If so, we abandon the procedure. The main optimization algorithm either reduces the size of the box $\mathbf{x^I}$ or splits it into subboxes. The current procedure might then succeed when applied to one or more of the resulting subboxes.

If the elimination procedure is successful, it produces an equation that we write in partitioned form as

$$\begin{bmatrix} \mathbf{R^I} \\ \mathbf{0} \end{bmatrix} \mathbf{w} = \begin{bmatrix} \mathbf{b^I}_1 \\ \mathbf{b^I}_2 \end{bmatrix} \tag{13.6.3}$$

where $\mathbf{R^I}$ is a square upper triangular matrix of order $m+r+1$. The vectors $\mathbf{b^I}_1$ and $\mathbf{b^I}_2$ have $m+r+1$ and $n-m-r$ components, respectively. The zero block in the new coefficient matrix has $n-m-r$ rows and $m+r+1$ columns. It is absent if $m+r=n$.

Consider the case $m+r<n$. From (13.6.3)

$$\mathbf{R^I w} = \mathbf{b^I}_1 \tag{13.6.4a}$$

$$0 = \mathbf{b^I}_2. \tag{13.6.4b}$$

If $0 \notin \mathbf{b^I}_2$, then (13.6.4b) is inconsistent. This implies that there is no solution to the John conditions for any $\mathbf{x} \in \mathbf{x^I}$. Therefore, we stop this procedure and delete $\mathbf{x^I}$.

If $0 \in \mathbf{b^I}_2$, there might be a solution for some $\mathbf{x} \in \mathbf{x^I}$. If so, then $\mathbf{b^I}_2 = \mathbf{0}$ for this point and we need only consider (13.6.4a) to compute bounds on the corresponding Lagrange multipliers. This equation can be solved by back substitution for interval bounds on $\mathbf{w}$. Thus, we obtain a box $\mathbf{u^I}$ bounding $\mathbf{u}$ and a box $\mathbf{v^I}$ bounding v.

The John conditions include the conditions $u_i \geq 0$ for $i = 0, \cdots, m$. Denote the components of $\mathbf{u^I}$ by $U_i = [\underline{u}_i, \overline{u}_i]$. If $\overline{u}_i < 0$ for some $i = 0, \cdots, m$, then (13.2.1d) is violated; and there is no solution of the John conditions for any $\mathbf{x} \in \mathbf{x^I}$. Therefore, we delete $\mathbf{x^I}$.

Bounds on the Lagrange multipliers can sometimes be used to simplify the John conditions (13.2.1b). Suppose that, for some box $\mathbf{x^I}$, we compute an interval vector $\mathbf{u^I}$ bounding $\mathbf{u}$. Suppose that $\underline{u}_i > 0$ for some $i =$

$1, \cdots, m$. Then $u_i > 0$ for any solution in $\mathbf{x}^{\mathbf{I}}$, and the complementary slackness condition $u_i p_i(\mathbf{x}) = 0$ (see (13.2.1b)) can be replaced by the simpler equation $p_i(\mathbf{x}) = 0$ for any box contained in $\mathbf{x}^{\mathbf{I}}$.

13.7 FIRST NUMERICAL EXAMPLE

In this section, we give a numerical example illustrating the ideas of Section 13.6. Consider the problem

$$\text{Minimize } f(\mathbf{x}) = x_1 \tag{13.7.1}$$

$$\text{subject to } p_1(\mathbf{x}) = x_1^2 + x_2^2 - 1 \leq 0,$$

$$p_2(\mathbf{x}) = x_1^2 - x_2 \leq 0.$$

The solution is at

$$x_1^* = -\left(\frac{5^{1/2} - 1}{2}\right)^{1/2} \doteqdot -0.786,$$

$$x_2^* = \frac{5^{1/2} - 1}{2} \doteqdot 0.618.$$

Since there are no equality constraints, our normalization for the Lagrange multipliers is

$$u_0 + u_1 + u_2 = 1. \tag{13.7.2}$$

The solution values for the Lagrange multipliers are

$$u_0^* = \frac{2x_1^*}{2x_1^* - 1} \doteqdot 0.611,$$

$$u_1^* = \frac{1}{(1 - 2x_1^*)[1 + 2(x_1^*)^2]} \doteqdot 0.174,$$

$$u_2^* = 1 - u_0^* - u_1^* \doteqdot 0.215.$$

In this example, we use the normalization condition (13.7.2) to explicitly eliminate u_0. the John conditions then become

$$\begin{aligned} 1 - u_1 - u_2 + 2x_1(u_1 + u_2) &= 0, \\ 2x_2u_1 - u_2 &= 0, \\ u_1p_1(\mathbf{x}) &= 0, \\ u_2p_2(\mathbf{x}) &= 0. \end{aligned} \tag{13.7.3}$$

Consider the box $\mathbf{x^I}$ with components $X_1 = [-0.8, -0.7]$ and $X_2 = [0.6, 0.7]$ that contains the solution. In the absence of information about the Lagrange multipliers, we guess they are all equal. That is, we guess $u_0 = u_1 = u_2 = 1/3$. One step of the interval Newton method applied to equations (13.7.3) yields the bounding intervals (recorded to only three digits)

$$U_1 = [0.138, 0.201],\ U_2 = [0.179, 0.240]$$

for the Lagrange multipliers and the improved bounds

$$X_1' = [-0.799, -0.753],\ X_2' = [0.600, 0.659]$$

for the components of the solution point.

Since $X_1' \subset X_1$ and $X_2' \subset X_2$, we have proved that a solution exists in the new box X_1'. See Proposition 11.15.5. To invoke this proposition, we implicitly assume that the initial bounding interval on each Lagrange multiplier is $[0, \infty]$.

In practice, we use other subroutines (see Chapter 14) to improve bounds on the solution point and on the Lagrange multipliers. We do not simply iterate the interval Newton method. However, if we do continue iterating, three more steps bound the components of $\mathbf{x}^*$ and of $\mathbf{u}^*$ to an accuracy of 10 digits past the decimal.

If a better approximation for $\mathbf{u}^*$ is available, faster convergence occurs. See Hansen and Walster (1990a).

This example shows how, given a box $\mathbf{x^I}$, we can not only improve the bounds on a solution point in $\mathbf{x^I}$ but also compute bounds on the Lagrange multipliers for any solution to the optimization problem in $\mathbf{x}^I$. We need only very crude approximations for the Lagrange multipliers.

The method of Section 13.6 does not require an approximation for the Lagrange multipliers. Using the same initial box $\mathbf{x}^I$ and solving (13.6.2) by Gaussian elimination as described in Section 13.6, we obtain

$$U_1 = [0.160, 0.190], \ U_2 = [0.209, 0.244].$$

No new bounds on the solution point are obtained by this procedure. Therefore, we cannot improve the bounds on $\mathbf{u}^I$ by iterating.

Next, consider the box $\mathbf{x}^I$ with components $X_1 = [-0.9, -0.8]$ and $X_2 = [0.5, 0.6]$. This box does not contain a solution. Using the good approximations $u_1 = 0.174$ and $u_2 = 0.215$ (and, implicitly, $u_0 = 0.611$), one interval Newton step yields a solution box disjoint from $\mathbf{x}^I$. This proves that no solution of the optimization problem (13.7.1) exists in $\mathbf{x}^I$.

We now consider a final case for this example. This time, we do not eliminate u_0 using the normalization condition. Consider the box $\mathbf{x}^I$ with components $X_1 = [-0.7, -0.6]$ and $X_2 = [0.7, 0.8]$. This box does not contain a solution of (13.7.1). Evaluating p_2 over the box, we obtain $p_2(\mathbf{x}^I) = [-0.44, -0.21]$. Since $p_2(\mathbf{x}^I) < 0$, the constraint $p_2 \leq 0$ is not active for any point in $\mathbf{x}^I$. However, $0 \in p_1(\mathbf{x}^I)$. Dropping the inactive constraint, equation (13.6.2) becomes

$$\begin{bmatrix} 1 & 1 \\ 1 & [-1.4, -1.2] \\ 0 & [1.4, 1.6] \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}.$$

Let us omit preconditioning. Using interval Gaussian elimination to transform this coefficient matrix into upper trapezoidal form, we obtain

$$\begin{bmatrix} 1 & 1 \\ 0 & [-2.4, -2.2] \\ 0 & 0 \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ [-0.728, -0.577] \end{bmatrix}.$$

The third component of the right member does not contain zero. This proves that no solution of the optimization problem (13.7.1) exists in $\mathbf{x}^I$.

13.8 SECOND NUMERICAL EXAMPLE

For a second example, we replace the inequality constraint $p_2(\mathbf{x}) \leq 0$ in problem (13.7.1) by an equality constraint using the same function. The problem becomes

$$\begin{aligned}
&\text{Minimize } f(\mathbf{x}) = x_1 \\
&\text{subject to } p(\mathbf{x}) = x_1^2 + x_2^2 - 1 \leq 0, \\
&\qquad\qquad q(\mathbf{x}) = x_1^2 - x_2 = 0.
\end{aligned}$$

We use the linear normalization (13.2.2a) for the Lagrange multipliers.

The solution is

$$x_1^* = -(x_2^*)^{1/2} \doteq -0.786, \ x_2^* = \frac{5^{1/2} - 1}{2} \doteq 0.618$$

and the Lagrange multipliers for the solution are

$$\begin{aligned}
u_0^* &= \frac{2x_1^*}{2x_1^* - 1} \doteq 0.611, \\
u_1^* &= \frac{1}{(1 - 2x_1^*)[1 + 2x_2^*]} \doteq 0.174, \\
v_1^* &= 2x_2^* u_1^* \doteq 0.215.
\end{aligned}$$

Equation (13.5.1), which expresses (part of) the John conditions, becomes

$$\phi(\mathbf{t}) = \begin{bmatrix} u_0 + u_1 + Ev_1 - 1 \\ u_0 + 2u_1x_1 + 2v_1x_1 \\ 2u_1x_2 - v_1 \\ u_1(x_1^2 + x_2^2 - 1) \\ x_1^2 - x_2 \end{bmatrix} = 0$$

where $\mathbf{t} = (x_1, x_2, u_0, u_1, v_1)^T$.

Let the box $\mathbf{x}^I$ have components

$$X_1 = [-0.8, -0.7], X_2 = [0.6, 0.7].$$

Approximate the Lagrange multipliers by $u_0 = u_1 = v_1 = 1/3$. Linearizing $\phi(\mathbf{t})$ as in (13.5.3), and solving by interval Gaussian elimination, we obtain the interval vector

$$\mathbf{t}^{\mathbf{I}'} = \begin{bmatrix} [-0.789, -0.783] \\ [0.611, 0.622] \\ [0.611, 0.627] \\ [0.142, 0.200] \\ [0.148, 0.226] \end{bmatrix}$$

containing the vector $\mathbf{t}$ for any solution with $\mathbf{x}^* \in \mathbf{x}^{\mathbf{I}}$. The first two components of $\mathbf{t}^{\mathbf{I}'}$ are improved bounds for the solution point $\mathbf{x}^*$. The last three components of $\mathbf{t}^{\mathbf{I}'}$ are bounds for the Lagrange multipliers.

Since the last component of $\mathbf{t}^{\mathbf{I}'}$ bounds v_1, we now know that $v_1 > 0$ (for any solution with $\mathbf{x}^* \in \mathbf{x}^{\mathbf{I}}$). Hence, we can replace E by 1 in the first component of $\phi(\mathbf{t})$ for any subsequent iterations using the new box (because it is contained in $\mathbf{x}^{\mathbf{I}}$).

In this example, we started with bounds X_1 and X_2 and approximations for the Lagrange multipliers. Using (13.5.1), we computed improved bounds on $\mathbf{x}^*$ while producing bounds on the Lagrange multipliers. Iterating the procedure can produce sharper bounds on all these quantities.

Next, we consider use of the method described in Section 13.6 for the same problem using the same box $\mathbf{x}^{\mathbf{I}}$. Now, we do not need the approximations for the Lagrange multipliers. For this problem, the coefficient matrix $\mathbf{A}(\mathbf{x}^{\mathbf{I}})$ in equation (13.6.2) is square and (13.6.2) becomes

$$\begin{bmatrix} 1 & 1 & E \\ 1 & 2X_1 & 2X_1 \\ 0 & 2X_2 & -1 \end{bmatrix} \mathbf{w} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}.$$

Substituting $X_1 = [-0.8, -0.7]$ and $X_2 = [0.6, 0.7]$ into this equation and solving by Gaussian elimination (without preconditioning), we obtain

$$U_0 = [0.582, 0.618],$$

$$U_1 = [0.159, 0.190],$$

$$V_1 = [0.208, 0.245].$$

We have computed bounds on the Lagrange multipliers for any solution to the optimization problem for which $\mathbf{x}^* \in \mathbf{x}^\mathbf{I}$. We cannot iterate this step since the procedure does not provide improved bounds on $\mathbf{x}^*$.

13.9 USING CONSISTENCY

Hull consistency and box consistency can be applied to the John conditions. To do so, we need bounds on the Lagrange multipliers. We have discussed how bounds can be computed.

However, we do not apply consistency methods to the John conditions. In our optimization algorithms, we apply consistency methods to each constraint individually. See the algorithms in Sections 14.8 and 15.12. Little is gained by also applying them to the equations expressing the John conditions.

Chapter 14

INEQUALITY CONSTRAINED OPTIMIZATION

14.1 INTRODUCTION

In Chapter 13 we dealt primarily with the John conditions and the Lagrange multipliers that are introduced to provide conditions for a solution. In Chapters 14 and 15, we discuss procedures for solving constrained optimization problems.

For pedagogical reasons, we consider inequality and equality constrained problems separately. In this chapter, we discuss the optimization problem in which only inequality constraints are present. In the next chapter, we discuss the problem in which only equality constraints occur.

By separating the cases, we hope to make clear which aspects of the constrained problem are peculiar to the particular kind of constraints. There is no difficulty in combining the algorithms for the two cases into a single algorithm for problems in which both kinds of constraints occur. We do so in Chapter 16.

Suppose that a given problem has inequality constraints but no equality constraints. Then it might be possible to show that a given box is certainly strictly feasible. If so, we know that any minimum in the box is a stationary point of the objective function. See Section 14.5. Thus, we can search for a minimum in of the box using the algorithm for unconstrained optimization given in Chapter 12. This algorithm uses effective procedures that are generally not valid for a constrained problem. This is one reason for separately discussing equality and inequality constrained problems. These procedures are not available for a problem having equality constraints because the feasible region has no interior in which a box might be certainly feasible.

When there are inequality constraints only, the optimization problem (13.1.1) becomes

$$\text{Minimize } f(\mathbf{x}) \tag{14.1.1}$$
$$\text{subject to } p_i(\mathbf{x}) \leq 0 \ \ (i = 1, \cdots, m).$$

We assume that f is twice continuously differentiable and that p_i $(i = 1, \cdots, m)$ is continuously differentiable. For cases in which these conditions do not hold, see Section 14.12.

The first interval algorithm for this problem was given by Hansen and Sengupta (1980). Our present approach is similar; and we use some of the procedures from that paper. However, the algorithm we give in Section 14.8 differs in various ways.

We seek the solution to (14.1.1) in an initial box $\mathbf{x}^{\mathbf{I}(0)}$. If any constraint prescribed in the problem statement has the form $x_i \geq a_i$ or $x_i \leq b_i$ $(i = 1, \cdots, m)$, we use it to determine the appropriate endpoint of the i-th component of $\mathbf{x}^{\mathbf{I}(0)}$. We call such a constraint a *prescribed box constraint*.

If the prescribed box constraints do not fix all $2n$ sides of a box, the user of our algorithm must choose the remaining sides. Alternatively, default values can be used. An upper endpoint of a box component can be chosen to be the largest positive number representable in the number system of the computer. A lower endpoint can be the smallest such negative number. Any sides of $\mathbf{x}^{\mathbf{I}(0)}$ that are not prescribed box constraints are called *unprescribed box constraints.*

We assume the unprescribed box constraints are chosen so that $\mathbf{x}^{\mathbf{I}(0)}$ contains the global solution. Otherwise, the global solution is not found.

As we explain later in this section, if the global solution is outside $\mathbf{x}^{\mathbf{I}(0)}$, we might not even find the best solution in $\mathbf{x}^{\mathbf{I}(0)}$.

Suppose $\mathbf{x}^{\mathbf{I}(0)}$ does not contain the global minimum but contains at least one local minimum. Then our algorithm finds either the smallest of these local minima or else a point in $\mathbf{x}^{\mathbf{I}(0)}$ where f is strictly smaller than the smallest of these local minima.

To ensure that the best solution in $\mathbf{x}^{\mathbf{I}(0)}$ is found in all cases, the unprescribed constraints must be treated as if they are prescribed. We do not do so in our algorithm. Instead, we rely on the user to specify $\mathbf{x}^{\mathbf{I}(0)}$ to be sufficiently large to contain the global solution. Our philosophy is that if we do not find the global solution, we might as well be satisfied with a solution that can be slightly suboptimal in $\mathbf{x}^{\mathbf{I}(0)}$ as well.

A user who wants the solution that is global in $\mathbf{x}^{\mathbf{I}(0)}$ can simply specify all sides of $\mathbf{x}^{\mathbf{I}(0)}$ as prescribed box constraints. However, this causes introduction of an additional Lagrange multiplier in the John conditions for each constraint that is changed from unprescribed to prescribed. Therefore, the dimension of the problem is increased; and more computing is generally required to obtain the solution. See Section 14.2 or 14.8.

We now explain why the solution produced by our algorithm might not be the best one in $\mathbf{x}^{\mathbf{I}(0)}$ if the global solution is not in $\mathbf{x}^{\mathbf{I}(0)}$.

We introduced the John conditions in Section 13.2 and we specialize them for the inequality constrained problem in Section 14.2. In our algorithm, we delete subboxes of $\mathbf{x}^{\mathbf{I}(0)}$ that we prove cannot contain any point satisfying the John conditions.

Suppose the global solution in $\mathbf{x}^{\mathbf{I}(0)}$ occurs on an unprescribed box constraint. The John conditions are not satisfied at this point unless $f(\mathbf{x})$ is stationary there. Therefore, our algorithm is likely to delete the point. This might occur when a Newton method is applied to solve the John conditions. The point might also be deleted by the procedure described in Section 12.4 that deletes points where the gradient of the objective function is not zero.

Figure 14.1.1 is a simple illustration of the situation. In the figure there is a prescribed inequality constraint $-6 \leq x$, the upper bound of the starting interval is 5, and this is an unprescribed constraint. Values of x less than the sampled value can be deleted because f is greater than at the sampled value. Values of x greater than the sampled value can be deleted because none is a stationary point. The end result is that the entire starting box can be deleted. The sampled value is accepted as the minimum. To obtain the global minimum, either the constraint $x \le 5$ must be prescribed, or the upper bound of the interval X must be increased to a point where the slope of f is nonnegative.

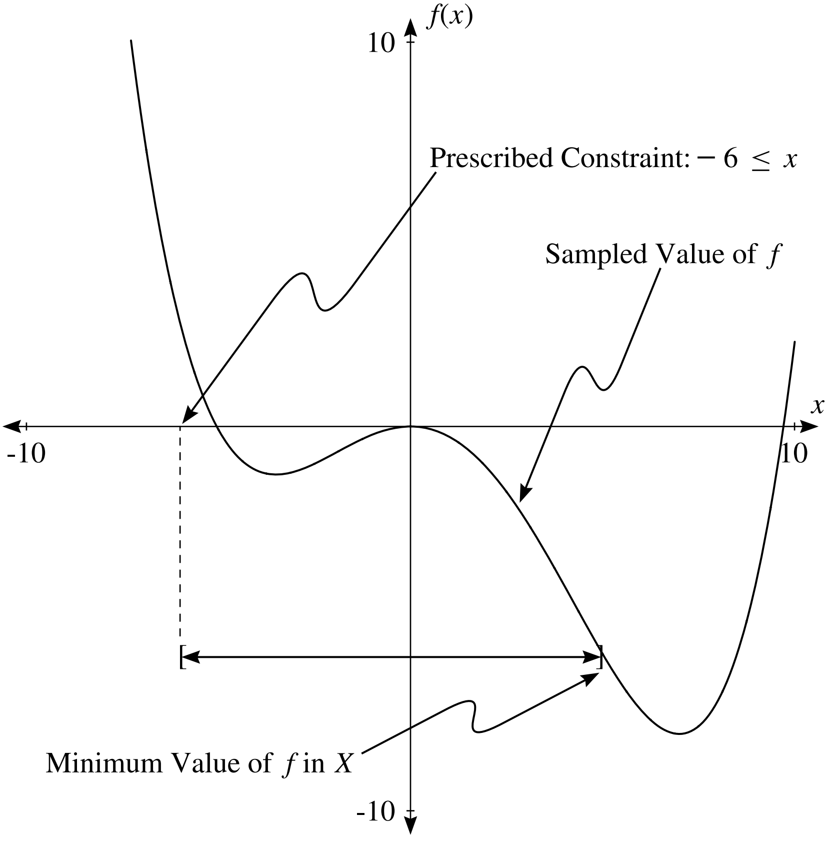


Figure 14.1.1: Deleting All Boxes in a Constrained Problem

In Section 12.5, we describe how to compute approximations for the global solution by sampling values of f at points in $\mathbf{x}^{\mathbf{I}(0)}$. In Section 14.3, we describe a similar sampling procedure for the inequality constrained case. If the global solution in $\mathbf{x}^{\mathbf{I}(0)}$ is deleted as just described, the best result from sampling is available as an approximate solution. This value can be less than the value of the objective function at any stationary point in $\mathbf{x}^{\mathbf{I}(0)}$, but it might not be the global minimum.

14.2 THE JOHN CONDITIONS

For the inequality constrained optimization problem, we normalize the Lagrange multipliers using the linear relation

$$u_0 + \cdots + u_m = 1. \tag{14.2.1}$$

See Sections 13.2 and 13.3. Therefore, the function given by (13.5.1), which expresses (part of) the John conditions, becomes

$$\phi(\mathbf{t}) = \begin{bmatrix} u_0 + \cdots + u_m - 1 \\ u_0 \nabla f(\mathbf{x}) + u_1 \nabla p_1(\mathbf{x}) + \cdots + u_m \nabla p_m(\mathbf{x}) \\ u_1 p_1(\mathbf{x}) \\ \vdots \\ u_m p_m(\mathbf{x}) \end{bmatrix}. \tag{14.2.2}$$

The remaining part of the John conditions not in (14.2.2) is that the Lagrange multipliers are nonnegative. (See (13.2.1d).) Therefore, the normalization equation (14.2.1) provides the bounds

$$0 \leq u_i \leq 1 \; (i = 0, \cdots, m). \tag{14.2.3}$$

These bounds are useful when solving the John conditions using the form of the interval Newton method in which the linearized equations are solved by the Gauss-Seidel method.

Suppose we solve the linearized John conditions by Gaussian elimination or by the "hull method" of Section 5.8. Then we do not need bounds on the Lagrange multipliers. However, we do need estimates. We can begin by letting $u_i = 1/(m + 1)$ for all $i = 0, \cdots, m$ so that (14.2.1) is satisfied. A successful step of a Newton method provides interval bounds on the Lagrange multipliers. For the next Newton step, the centers of these intervals can serve as the needed estimates.

In our algorithm, we do not iterate the Newton procedure to convergence. One reason for this is that we do not want to spend effort to get good bounds on a local (nonglobal) solution of the optimization problem. To prevent this, we use other procedures that we list in Section 14.8. Another reason is that other procedures for improving the bounds on a solution require less computing effort, and thus take precedence.

14.3 AN UPPER BOUND ON THE MINIMUM

In Section 12.5, we discussed how to obtain and use an upper bound $\bar{\mathfrak{f}}$ on the globally minimum value f^* of the objective function f. We do the same thing for the constrained case. We can delete any point $\mathbf{x}$ where $f(\mathbf{x}) > \bar{\mathfrak{f}}$. To compute $\bar{\mathfrak{f}}$, we evaluate f at various certainly feasible points obtained by the algorithm; and we set $\bar{\mathfrak{f}}$ equal to the smallest upper bound found in this way.

When constraints are present, we must assure that each point used to update $\bar{\mathfrak{f}}$ is feasible. We must assure this feasibility despite the fact that rounding makes the correct value of a constraint function uncertain. We do this by requiring that the point is certainly feasible as defined in Section 6.1.

Having computed an upper bound $\bar{\mathfrak{f}}$, we use it to delete subboxes of $\mathbf{x}^{\mathbf{I}(0)}$ in the same way described in Section 12.5. Points deleted in this way can be feasible or infeasible. The inequality $f(\mathbf{x}) \leq \bar{\mathfrak{f}}$ can also be added to the John conditions as if it were an ordinary constraint.

We try to reduce $\bar{\mathfrak{f}}$ at various stages of the algorithm. Whenever the algorithm produces a new subbox of $\mathbf{x}^{\mathbf{I}(0)}$ (see Section 14.8), we check to see if the center of the box is certainly feasible. If so, we evaluate f at this point and update $\bar{\mathfrak{f}}$.

In our algorithm for the unconstrained problem, when the new box is generated, we use a real Newton method to search for a point in the new box that enables us to reduce $\bar{f}$. This is facilitated by having a real inverse matrix to use in the Newton process. See Section 12.6.

In the constrained case, we do not have such an inverse. However, it is possible to obtain one. A procedure in our algorithm solves the John conditions in the linearized form (13.5.3). This procedure computes an approximate inverse of the center $\mathbf{J}^c$ of the Jacobian $\mathbf{J}(\mathbf{t}, \mathbf{t}^{\mathbf{I}})$ defined by (13.5.2). See Section 5.6. For our search, we want to approximate the inverse of the leading principal submatrix of $\mathbf{J}(\mathbf{t}, \mathbf{t}^{\mathbf{I}})$ of order n. This can be obtained when computing an approximate inverse of $\mathbf{J}^c$. The matrix $\mathbf{J}^c$ must be partitioned appropriately. We do not use such a procedure, so we omit the details.

There is no guarantee that such a point generated by such an interval method is feasible. Therefore, it does not seem fruitful to do all the work required to apply a Newton method when the effort might be wasted. We have not tried to do so. Moreover, we want to update $\bar{f}$ even when we are not going to try to solve the John conditions, and thus have no reason to compute a Jacobian. Therefore, we use a simpler procedure. We describe it in the next section.

14.4 A LINE SEARCH

Suppose our algorithm has generated a subbox $\mathbf{x}^{\mathbf{I}}$ of the initial box $\mathbf{x}^{\mathbf{I}(0)}$. (See Section 14.8 to see how this might be done.) Let $\mathbf{x}$ denote the center of $\mathbf{x}^{\mathbf{I}}$. Under appropriate conditions (to be given), we search along a half-line beginning at x for an approximate solution point of the optimization problem (14.1.1).

If we have already found a certainly feasible point in $\mathbf{x}^{\mathbf{I}(0)}$, we have a finite upper bound $\bar{f}$ on the globally minimum value f^*. (See Section 14.3.) Otherwise, $\bar{f} = +\infty$. We perform the line search to try to reduce $\bar{f}$ only if $f(\mathbf{x}) < \bar{f}$. This decision is made regardless of whether $\mathbf{x}$ is a certainly feasible point or not.

Consider the half-line extending in the negative gradient direction from $\mathbf{x}$. If $\mathbf{x}$ is certainly feasible, we search on the segment of this half-line between $\mathbf{x}$ and the point, say $\mathbf{x}'$, where the half-line crosses the boundary of $\mathbf{x}^{\mathbf{I}}$. If the gradient is not zero at $\mathbf{x}$, then f decreases (locally) in the direction of $\mathbf{x}'$ from $\mathbf{x}$. If we can find a certainly feasible point, say $\mathbf{y}$, on this line segment where $f(\mathbf{y}) < f(\mathbf{x})$, then we can reduce $\bar{\mathfrak{f}}$ since we assumed $f(\mathbf{x}) < \bar{\mathfrak{f}}$ and hence $f(\mathbf{y}) < \bar{\mathfrak{f}}$.

If $\mathbf{x}$ is not certainly feasible, then it is possible that no point on the line segment joining $\mathbf{x}$ and $\mathbf{x}'$ is certainly feasible. To enhance the possibility of finding a certainly feasible point in $\mathbf{x}^{\mathbf{I}}$, we search in a direction in which we know there exists at least one certainly feasible point. The point $\bar{\mathbf{x}}$ at which we computed the current value of $\bar{\mathfrak{f}}$ is such a point, and is the one at which f has the smallest currently known feasible value. A user might have provided a finite upper bound $\overline{f}$ without knowing a point $\bar{\mathbf{x}}$ where $f(\bar{\mathbf{x}}) = \overline{f}$. If no point $\bar{\mathbf{x}}$ is known and $\mathbf{x}$ is not certainly feasible, we do not do a line search.

We search in this direction even though the value of f at $\bar{\mathbf{x}}$ is larger than at $\mathbf{x}$. We can take the positive view that we are not searching from $\mathbf{x}$ toward larger values of f, but we are searching from the point $\bar{\mathbf{x}}$ toward the point $\mathbf{x}$ where we know f is smaller. The search is restricted to the box $\mathbf{x}^{\mathbf{I}}$ even though the point $\bar{\mathbf{x}}$ need not be in $\mathbf{x}^{\mathbf{I}}$.

Certain conditions must be met before we do the line search. Given a box $\mathbf{x}^{\mathbf{I}}$, we determine its center $\mathbf{x}$ and compute $f^{\mathbf{I}}(\mathbf{x})$. Because of rounding, we obtain an interval $[\underline{f^{\mathbf{I}}}(\mathbf{x}), \overline{f^{\mathbf{I}}}(\mathbf{x})]$. If $\overline{f^{\mathbf{I}}}(\mathbf{x}) < \bar{\mathfrak{f}}$, we do the line search. Otherwise, we do not.

Also, if $\mathbf{x}$ in not certainly feasible and $\bar{\mathfrak{f}} = +\infty$, we do not do the search. The condition $\bar{\mathfrak{f}} = +\infty$ indicates that no certainly feasible point has yet been found. Therefore, we do not know a preferred direction in which to search.

The point $\mathbf{x}$ need only be an approximation for the center of $\mathbf{x}^{\mathbf{I}}$. Therefore, it can be computed using rounded real arithmetic.

We now list the steps of such a line search. Unless otherwise specified by branching, the steps are done in the order given. The initialization step contains the tests to decide if the procedure is to be used or not.

In the procedure, $\mathbf{y}$ always denotes the certainly feasible point where f takes the smallest value yet found. We do not use interval arithmetic in the algorithm except in step 11 and when evaluating the constraints to decide if a point is certainly feasible.

0. Initialize:

   (a) If $f(\mathbf{x}) \geq \bar{f}$ terminate.

   (b) If $\mathbf{x}$ is not certainly feasible and no certainly feasible point is yet known, terminate.

   (c) Set $n = 0$.

1. If $\mathbf{x}$ is certainly feasible, go to Step 3.

2. If a feasible point $\overline{\mathbf{x}}$ is known where $f(\overline{\mathbf{x}}) = \bar{f} < +\infty$, set $\mathbf{y} = \overline{\mathbf{x}}$ and $\mathbf{y}' = \mathbf{x}$. If no such point $\overline{\mathbf{x}}$ is known, go to Step 12. Note that $\overline{\mathbf{x}}$ is the point where $f(\overline{\mathbf{x}}) = \bar{f}$.

3. Determine the positive constant c such that $\mathbf{x}' = \mathbf{x} - cg(\mathbf{x})$ is on the boundary of $\mathbf{x}^{\mathbf{I}}$ where g is the gradient of f. That is, $c = \min\limits_{1 \leq i \leq n} \frac{w(X_i)}{2|g_i(\mathbf{x})|}$.

4. Set $\mathbf{y} = \mathbf{x}$ and $\mathbf{y}' = \mathbf{x}'$. If $f(\mathbf{x}') \geq f(\mathbf{x})$, go to Step 6.

5. If $\mathbf{x}'$ is certainly feasible, replace $\mathbf{y}$ by $\mathbf{x}'$ and $\mathbf{y}'$ by $\mathbf{x}$.

6. Compute $\mathbf{y}'' = \frac{1}{2}(\mathbf{y} + \mathbf{y}')$. Replace n by $n + 1$.

7. If $f(\mathbf{y}'') \geq \max\{f(\mathbf{y}), f(\mathbf{y}')\}$, go to Step 11.

8. If $f(\mathbf{y}'') \geq f(\mathbf{y})$, replace $\mathbf{y}'$ by $\mathbf{y}''$ and go to Step 10.

9. If $\mathbf{y}''$ is certainly feasible, replace $\mathbf{y}'$ by $\mathbf{y}$ and then replace $\mathbf{y}$ by $\mathbf{y}''$.

10. If $n < 4$, go to Step 6.

11. Evaluate $f(\mathbf{y})$ in interval arithmetic getting $[\underline{f^{\mathbf{I}}}(\mathbf{y}), \overline{f^{\mathbf{I}}}(\mathbf{y})]$. Replace $\bar{f}$ by $\min\{\bar{f}, \overline{f^{\mathbf{I}}}(\mathbf{y})\}$.

12. Terminate.

Note that since the center $\mathbf{x}$ of $\mathbf{x^I}$ is computed only approximately, it might be necessary to adjust $\mathbf{x}'$ in Step 3 so that $\mathbf{x}' \in \mathbf{x^I}$. Actually, all that is necessary is that $\mathbf{x}'$ be in the initial box in which the optimization algorithm is solved.

To do this line search, the constraint functions are evaluated up to five times. See Steps 5 and 9. If there are a large number of constraints, this number can be reduced to save computation. This can be done by reducing the bound on n in Step 10. If there are few constraints, more iterations can be used to try to compute a better bound on $\bar{f}$. The procedure can also be modified so the number of iterations depends on progress in reducing $\bar{f}$.

Note that a user might know an upper bound $\bar{f} < +\infty$ on the global minimum, but might not know a point $\bar{\mathbf{x}}$ where f takes this value. Also, a procedure in Chapter 16 uses this search algorithm when $\bar{f} < +\infty$ but $\bar{\mathbf{x}}$ is not known. In this case, a search cannot be made in the direction of $\bar{\mathbf{x}}$. See Step 2 of the algorithm.

14.5 CERTAINLY STRICT FEASIBILITY

Consider a certainly strictly feasible (as defined in Section 6.1) subbox $\mathbf{x^I}$ of the initial box $\mathbf{x^{I(0)}}$. If a minimum of f occurs in $\mathbf{x^I}$, it must occur at a stationary point of f. When solving the constrained problem in $\mathbf{x^I}$, we can treat the problem as if it were unconstrained. Therefore, we are able to use procedures from our algorithm for the unconstrained problem that are otherwise not valid for the constrained case.

Our algorithm for inequality constrained problems is given in Section 14.8. Suppose it generates a subbox $\mathbf{x^I}$ of the initial box $\mathbf{x^{I(0)}}$. We evaluate $p_i(\mathbf{x^I})$ for $i = 1, \cdots, m$ and obtain the interval $[\underline{p}_i(\mathbf{x^I}), \overline{p}_i(\mathbf{x^I})]$. If $\overline{p}_i(\mathbf{x^I}) < 0$ for some value of i, then the i-th constraint can be ignored when considering the box $\mathbf{x^I}$ because the constraint cannot be active. If $\overline{p}_i(\mathbf{x^I}) < 0$, for all $i = 1, \cdots, m$, then $\mathbf{x^I}$ is certainly strictly feasible. That is, no constraint is active in $\mathbf{x^I}$.

In this case, we can apply two of the "eliminating procedures" of Chapter 12. First, since the gradient $\mathbf{g}$ of f must be zero at any solution in $\mathbf{x}^I$, we can use procedures designed to solve $\mathbf{g} = \mathbf{0}$. These procedures include hull consistency, box consistency, and the interval Newton method.

Second, we can use the convexity condition $H_{ii}(\mathbf{x}) \leq 0$ $(i = 1, \cdots, n)$ where H_{ii} is the i-th diagonal element of the Hessian of f. This relation can be "solved" using hull consistency, box consistency, and the method of Section 6.2.

For any optimization problem (constrained or not), we can delete points $\mathbf{x}$ where $f(\mathbf{x}) > \bar{f}$. This can be done using hull consistency or box consistency. When $\mathbf{x}^I$ is certainly strictly feasible, we can use the real Newton method of Section 12.6 to try to reduce $\bar{f}$. This procedure generally finds a smaller value of $\bar{f}$ than the line search described in Section 14.4. However, we use the real Newton method only when we intend to apply an interval Newton method to solve $\mathbf{g} = \mathbf{0}$. Otherwise, we avoid generating the real Jacobian needed to apply the real Newton Steps. Instead, we use the simpler line search procedure of Section 14.4.

Note that when we evaluate $p_i(\mathbf{x}^I)$ for some $i = 1, \cdots, m$, and find that $\overline{p}_i(\mathbf{x}^I) < 0$, we identify a constraint (the i-th) that is without question not active for any solution in $\mathbf{x}^I$. This is the converse of what is often done in noninterval optimization procedures when the attempt is made to identify active constraints. See, for example, Burke (1990).

14.6 USING THE CONSTRAINTS

Our optimization algorithm uses the inequality constraints to delete points that are not feasible. One way is to apply hull consistency to each constraint separately. As noted previously, we can apply hull consistency to an inequality $p_i(\mathbf{x}) \leq 0$ by writing it as the equality $p_i(\mathbf{x}) = [-\infty, 0]$. We can also apply box consistency to the latter form.

We also use linearized forms of the constraints that enables us to apply them as a system rather than one at a time. This provides a better procedure for eliminating parts of boxes that are sufficiently small that linearization provides a good approximation to the constraint functions. This resembles application of an interval Newton method to solve a system of nonlinear equations. We discuss linearization in the remainder of this section.

Assume that we want to linearize the constraints over $\mathbf{x}^{\mathbf{I}}$ and "solve" them by the method of Chapter 6. Before doing so, suppose we evaluate $p_i(\mathbf{x}^{\mathbf{I}})$ and obtain $[\underline{p}_i(\mathbf{x}^{\mathbf{I}}), \overline{p}_i(\mathbf{x}^{\mathbf{I}})]$ for $i = 1, \cdots, m$. If $\underline{p}_i(\mathbf{x}^{\mathbf{I}}) > 0$ for any i, then $\mathbf{x}^{\mathbf{I}}$ is certainly infeasible and can be deleted. Therefore, we can safely assume that $\underline{p}_i(\mathbf{x}^{\mathbf{I}}) \leq 0$. At the other extreme, suppose $\overline{p}_i(\mathbf{x}^{\mathbf{I}}) \leq 0$. In this case, the constraint cannot serve to delete any points of $\mathbf{x}^{\mathbf{I}}$; and we omit it from current considerations. Therefore, we assume $\underline{p}_i(\mathbf{x}^{\mathbf{I}}) \leq 0 < \overline{p}_i(\mathbf{x}^{\mathbf{I}})$.

In Section 6.4, we defined

$$s_i = \overline{p}_i(\mathbf{x}^{\mathbf{I}})/[\overline{p}_i(\mathbf{x}^{\mathbf{I}}) - \underline{p}_i(\mathbf{x}^{\mathbf{I}})] \ \ (i = 1, \cdots, m). \tag{14.6.1}$$

From the above discussion, we see that the i-th constraint is not violated in $\mathbf{x}^{\mathbf{I}}$ if $s_i \leq 0$, and there is no feasible point in $\mathbf{x}^{\mathbf{I}}$ if $s_i > 1$. For small values of s_i the i-th constraint is only "slightly violated" by points in $\mathbf{x}^{\mathbf{I}}$ and is likely to delete only a small part of $\mathbf{x}^{\mathbf{I}}$. Therefore, to reduce effort, we use the i-th constraint only if

$$s_i > 0.25 \tag{14.6.2}$$

We linearize the constraints for which (14.6.2) is satisfied as described in Section 6.3. We then "solve" the set of linear constraints over $\mathbf{x}^{\mathbf{I}}$ as described in Chapter 6. This process generally deletes all or part of $\mathbf{x}^{\mathbf{I}}$.

There is an additional constraint that can possibly be used. If $\overline{\mathfrak{f}} < +\infty$, we can also include the inequality $f(\mathbf{x}) - \overline{\mathfrak{f}} \leq 0$. This inequality must hold for any point $\mathbf{x}$ that is a candidate for a global solution point. We include this inequality in the set to be linearized and solved as if it is just another constraint whenever

$$\frac{\overline{f}(\mathbf{x}^{\mathbf{I}}) - \overline{\mathfrak{f}}}{\overline{f}(\mathbf{x}^{\mathbf{I}}) - \underline{f}(\mathbf{x}^{\mathbf{I}})} > 0.25. \tag{14.6.3}$$

When we linearize the constraints over a box $\mathbf{x}^{\mathbf{I}}$ as described in Section 6.3, the resulting coefficient matrix is an interval matrix. Assume that $\mathbf{x}^{\mathbf{I}}$

contains a local (or global) solution of the minimization problem (14.1.1). Assume, also that the inequality $f(\mathbf{x}) - \bar{\mathfrak{f}} \leq 0$ has been introduced as an explicit constraint. Then the coefficient matrix contains a real (noninterval) matrix whose rows are linearly dependent.

The existence of such a singular real matrix is assured by the John conditions. They specify (in part) that some linear combination of the gradients of the active constraints at a solution is opposite in direction to the gradient of the objective function. This linear dependence causes no difficulty with our algorithm.

Earlier in this section we stated that we evaluate $p_i(\mathbf{x}^{\mathbf{I}})$ $(i = 1, \cdots, m)$ to determine which constraints to linearize according to the criterion (14.6.2). Rather than simply evaluating $p_i(\mathbf{x}^{\mathbf{I}})$, we actually do something somewhat different.

Assume we have applied hull consistency to the system of constraints. When doing so, we solve each constraint for each variable. Assume that the last time we used the i-th constraint, we solved it for x_j for some $j = 1, \cdots, m$. To do so, we write $p_i(\mathbf{x}) \leq 0$ as $a(\mathbf{x})g(x_j) - h(\mathbf{x}) = [-\infty, 0]$ and compute

$$X_j' = g^{-1}\{(h(\mathbf{x}^{\mathbf{I}}) + [-\infty, 0])/a(\mathbf{x}^{\mathbf{I}})\}$$

In so doing, we obtain $a(\mathbf{x}^{\mathbf{I}})$ and $h(\mathbf{x}^{\mathbf{I}})$. Denote $X_j'' = X_j \cap X_j'$. The function g is chosen to be simple. Therefore, we can easily evaluate $g(X_j'')$. One extra subtraction and one extra multiplication yields $a(\mathbf{x}^{\mathbf{I}})g(X_j'') - h(\mathbf{x}^{\mathbf{I}})$. This is an adequate approximation to $p_i(\mathbf{x}^{\mathbf{I}})$ for the purpose of determining s_i as defined in (14.6.1). Therefore, we save the work of evaluating $p_i(\mathbf{x}^{\mathbf{I}})$ $(i = 1, \cdots, m)$.

Suppose that hull consistency is applied to another constraint after it was last applied to the i-th. If so, the box can change and the computed value for $p_i(\mathbf{x}^{\mathbf{I}})$ is not for the final box. However, when we cycle through the constraints and through the variables when applying hull consistency, most of the change in the box occurs in the early stages. Therefore, we expect the value of $p_i(\mathbf{x}^{\mathbf{I}})$ (computed as described) to be a reasonable approximation for the value p_i over the final box.

Note that if $p_i(\mathbf{x}^{\mathbf{I}}) < 0$, hull consistency obtains $\mathbf{x}^{\mathbf{I}''} = \mathbf{x}^{\mathbf{I}}$. Therefore, when we evaluate $a(\mathbf{x}^{\mathbf{I}})g(X_j'') - h(\mathbf{x}^{\mathbf{I}})$, we obtain $p_i(\mathbf{x}^{\mathbf{I}})$ and learn that $p_i(\mathbf{x}^{\mathbf{I}}) < 0$. This can be useful information. For example, see Step 11 of the algorithm in Section 14.8.

In Section 11.9, we analytically precondition a system of nonlinear equations. This is done to make the i-th preconditioned equation strongly dependent on the i-th variable. We then solve the i-th preconditioned equation for the i-th variable using both hull consistency and box consistency. We (sometimes) apply a similar process when solving systems of nonlinear inequalities. We now describe this process.

Assume we have linearized the system of constraints and obtained a linear system $\mathbf{A}^{\mathbf{I}}\mathbf{x} \leq \mathbf{b}^{\mathbf{I}}$. Let $\mathbf{A}^{\mathbf{I}}$ be composed of m rows and n columns. We described in Section 6.3 how we can generate a preconditioning matrix $\mathbf{B}^{\mathbf{I}}$ by operating on the center $\mathbf{A}^c$ of $\mathbf{A}^{\mathbf{I}}$. The elements of $\mathbf{B}$ must be nonnegative to avoid reversing the sense of any inequality. We can analytically precondition the system $\mathbf{p}(\mathbf{x}) \leq 0$ by multiplying by $\mathbf{B}$ and then solve the system $\mathbf{Bp}(\mathbf{x}) \leq 0$ using consistency methods.

To determine $\mathbf{B}$, we use Gaussian elimination to zero elements of $\mathbf{A}^c$ in positions (i, j) with $i \neq j$ for $i = 1, \cdots, m$ and $j = 1, \cdots, r'$. The number r' cannot generally exceed m and might be less than $m - 1$ because the elements of $\mathbf{B}$ must be nonnegative. If $r' < n/2$, we assume that $\mathbf{Bp}(\mathbf{x})$ is not sufficiently different from $\mathbf{p}(\mathbf{x})$ and we therefore do not generate $\mathbf{Bp}(\mathbf{x})$ and do not apply the consistency methods.

Assume $r' \geq n/2$. Then we apply the consistency methods to solve the i-th inequality of $\mathbf{Bp}(\mathbf{x}) \leq 0$ for x_i for $i = 1, \cdots, r'$. When determining $\mathbf{B}$, we also generate r' inequalities corresponding to the "secondary pivot rows" (see Section 6.5). We also solve the i-th of these inequalities from $\mathbf{Bp}(\mathbf{x}) \leq 0$ for the i-th variable.

A user might want to assure that all the efforts to delete points which do not satisfy the constraints are sufficiently successful. To assure this, we assume that the user specifies a tolerance ε_p. The algorithm assures that $p_i(\mathbf{x}) \leq \varepsilon_p$ $(i = 1, \cdots, m)$ for all x in every box remaining after the minimization algorithm terminates. If there is no desire to assure this condition, the tolerance ε_p can be set to ∞.

The procedure to linearize and solve the system of inequality constraints involves a substantial amount of work. We now discuss whether it is worth doing. Assume the constraints are functions of n variables; and we linearize the vector $\mathbf{p}(\mathbf{x}) \leq \mathbf{0}$ of m constraints over a box about a point $\mathbf{x}_0$ in the box. We obtain a linear system of the form

$$\mathbf{p}(\mathbf{x}_0) + \mathbf{J}(\mathbf{y} - \mathbf{x}_0) \leq \mathbf{0}$$

where the matrix $\mathbf{J}$ is m by n. We discussed the possible choices of real versus interval arguments of $\mathbf{J}$ in Section 7.3. We discussed how to solve such a linear system of inequalities with interval coefficients in Chapter 6.

In practice, each constraint usually depends on only a few of the n variables. Therefore, $\mathbf{J}$ is sparse. If we perform elimination on the linear system, we can only eliminate m variables. In the process we expect "fill-in" to make the final matrix dense. If so, each equation of the final system involves all $n - m$ of the remaining variables. To "solve" them, we can apply hull consistency.

For the sake of argument, assume each constraint is a function of s variables (although different constraints are functions of a different set of s variables). If $n - m > s$, we are less likely to obtain information from a transformed linear inequality of $n - m$ variables than from an original nonlinear constraint of s variables. Therefore, there is little point in linearizing and solving the original nonlinear system when $n - m \geq s$.

Let us redefine s to be the average number of variables upon which a given inequality constraint depends. To use the procedure, we must linearize the system of constraints, compute and apply the preconditioning matrix, do the elimination in the interval system, and solve the transformed system. See Chapter 6. This is a substantial amount of work. Moreover, we might not be able to complete the last phase of elimination in the interval system because, at some stage, there might not be a positive multiplier to do an elimination step. Therefore, we might end up with more than $n - m$ variables in each inequality after elimination.

Because of the work to apply the procedure, we do not use it unless $n - m$ is substantially less than s. We have arbitrarily chosen a condition.

We use the procedure only if

$$n - m \leq s/2. \tag{14.6.4}$$

14.7 USING TAYLOR EXPANSIONS

To apply a step of a Newton method requires quite a lot of computing. When solving the John conditions, the interval Jacobian of $\phi(\mathbf{t})$ (as given by (14.2.2)) can contain a singular real matrix (i.e., be irregular). If so, the Newton step (using the Gauss-Seidel method) tends to make little progress. We want to apply such a Newton step only when the Jacobian is regular. In Section 11.11, we described a linearization test (11.11.1) for deciding whether or not to apply a Newton step in an unconstrained problem. A Newton step is bypassed if the criterion indicates that the step is likely to be unsuccessful.

We now repeat our discussion of criterion (11.11.1) and then discuss similar criteria for deciding whether to apply other procedures that involve linearization.

Assume that the Newton method has previously been applied to the equation $\mathbf{g}(\mathbf{x}) = \mathbf{0}$ where $\mathbf{g}(\mathbf{x})$ is the gradient of the objective function. Let $w_I^{\mathbf{g}}$ denote the width of the smallest box for which the preconditioned Jacobian (computed in the Newton method) has been found to be irregular. Let $w_R^{\mathbf{g}}$ denote the width of the largest box for which the preconditioned Jacobian was found to be regular. (We have introduced a superscript $\mathbf{g}$ to indicate that the Newton method was applied to the *gradient* of the objective function.) We apply the Newton method to a given box $\mathbf{x^I}$ if $\mathrm{w}(\mathbf{x^I}) \leq (w_I^{\mathbf{g}} + w_R^{\mathbf{g}})/2$.

For the constrained case, we use the same criterion for deciding whether to apply the Newton method to the John conditions. That is, we use the size of boxes for which the preconditioned Jacobian of the John conditions is regular or irregular. Now we use corresponding parameters $w_I^{\mathbf{J}}$ and $w_R^{\mathbf{J}}$ defined in the same way as $w_I^{\mathbf{g}}$ and $w_R^{\mathbf{g}}$, respectively. Thus, we apply the Newton method to the John conditions only if

$$\mathrm{w}(\mathbf{x^I}) \leq (w_I^{\mathbf{J}} + w_R^{\mathbf{J}})/2. \tag{14.7.1}$$

We want to use a similar criterion to decide whether to "solve" the system of constraint inequalities by linearizing them as in Chapter 6.Now, however, we do not have a square coefficient matrix to test for regularity.

We solve the constraints over a given box only if we expect to make progress in reducing the box. A linearized version of a constraint function over a given box is generally a good approximation for the function only if the box is "small". Therefore linearizing and solving the constraints tends to be useful only if the box is "small". Instead of using regularity of a matrix as a criterion for linearizing, we linearize the constraints over a given box only if progress was previously made when doing so for a box of similar size.

Suppose we solve the constraints over a box $\mathbf{x^I}$. In so doing, the box might or might not be sufficiently reduced as defined using (11.7.4). Let w_S^p denote the largest width of any sufficiently-reduced box. Let w_I^p denote the smallest width of any box that was not sufficiently reduced. In Chapter 11, we described how we linearize a system of nonlinear constraints and use Gaussian elimination to solve the derived linear system. (See also Section 14.5.) We use this method to solve the inequality constraints only if the width of the current box is $\leq (w_S^p + w_I^p)/2$.

Initially, we set $w_S^p = 0$ and $w_I^p = \mathrm{w}(\mathbf{x^{I(0)}})$ where $\mathbf{x^{I(0)}}$ is the initial box in which the optimization problem is solved. After applying the procedure, we replace w_S^p by $\max\left(w_S^p, \mathrm{w}(\mathbf{x^I})\right)$ if $\mathbf{x^I}$ was sufficiently reduced (as defined using (11.7.4) by the procedure. If the procedure has not sufficiently reduced $\mathbf{x^I}$, we replace w_I^p by $\min\left(w_I^p, \mathrm{w}(\mathbf{x^I})\right)$.

We can solve the inequality $f(\mathbf{x}) \leq \bar{\mathfrak{f}}$ by linearizing and using the method of Section 6.2. We sometimes do so by including this inequality in the set of constraint inequalities as described in Section 14.6. However, we sometimes solve this inequality by itself using linearization. In this case, we decide whether to do so in the same way just described for the system of constraints. Now, however, we use separate width parameters w_S^f and w_I^f to make our decision regarding the inequality $f(\mathbf{x}) \leq \bar{\mathfrak{f}}$.

The tests we have described in this section all serve to decide whether to linearize particular nonlinear functions. Thus, we call them "linearization tests". Note that there are four of them. They concern linearization of the gradient of the objective function, of the John conditions, of the inequality constraints, and of the inequality $f(\mathbf{x}) \leq \bar{f}$.

We use another similar test to decide whether to expand the relation $f(\mathbf{x}) \leq \bar{f}$ through second order (quadratic) terms. For this test, we use parameters w_S^S and w_I^S corresponding to w_S^f and w_I^f, respectively.

14.8 THE ALGORITHM STEPS

In this section, we list the steps of our algorithm for computing the global solution to the inequality constrained problem (14.1.1).

Generally, we seek a solution in a single box specified by the user. However, any number of boxes can by specified. The boxes can be disjoint or overlap. However, if they overlap, a minimum at a point that is common to more than one box is separately found as a solution in each box containing it. In this case, computing effort is wasted. If the user does not specify an initial box or boxes, we use a default box as described in Section 12.3. The algorithm finds the global minimum in the set of points formed by the set of boxes. We assume these initial boxes are placed in a list L_1 of boxes to be processed.

Suppose the user of our algorithm knows a point $\bar{\mathbf{x}}$ that is guaranteed to be feasible. If so, we use this point to compute an initial upper bound $\bar{f}$ on the global minimum f^*. If $\bar{\mathbf{x}}$ cannot be represented exactly on the computer, we input a representable interval vector $\mathbf{x^I}$ containing $\bar{\mathbf{x}}$. We evaluate $f(\mathbf{x^I})$ and obtain $[\underline{f}(\mathbf{x^I}), \overline{f}(\mathbf{x^I})]$. Even if rounding and/or dependence are such that $\mathbf{x^I}$ cannot be numerically proven to be certainly feasible, we rely upon the user and assume that $\mathbf{x^I}$ contains a feasible point. Therefore, we set $\bar{f} = \overline{f}(\mathbf{x^I})$.

Also the user might know an upper bound $\bar{f}$ on f^* even though he might not know where (or even if) f takes on such a value. If so, we set $\bar{f}$ equal to this known bound. If the known bound is not representable on the computer, we round the value up to a larger value that is representable.

If no feasible point is known and no upper bound on f^* is known, we set $\bar{f} = +\infty$.

To initialize our algorithm, we require that the user specify a box size tolerance ε_X, a function width tolerance ε_f, a tolerance ε_p bounding values of the inequality constraints (see Section 14.6), and the initial box(es). The initial box(es) are placed in the list L_1.

The algorithm initializes the parameters needed to perform the linearization tests and the parameters to decide whether to expand the relation $f(\mathbf{x}) \leq \bar{\mathfrak{f}}$ through second order terms. (See Section 14.7.) It sets $w_R^{\mathbf{g}}$, $w_R^{\mathbf{J}}$, w_S^p, w_S^f, and w_S^S to zero and sets $w_I^{\mathbf{g}}$, $w_I^{\mathbf{J}}$, w_I^p, w_I^f, and w_I^S equal to $w(\mathbf{x}^{\mathbf{I}(0)})$. It also sets $w_H = 0$ where w_H is the width of the largest box $\mathbf{x}^\mathbf{I}$ generated by the algorithm such that $H_{ii}(\mathbf{x}^\mathbf{I}) \geq 0$ for all $i = 1, \cdots, n$. See Section 12.7. It also sets a flag $\mathcal{F}$ equal to 0. (The significance of the flag is discussed in Section 14.10.) In the algorithm, the current box is always denoted by $\mathbf{x}^\mathbf{I}$ even though it changes from step to step.

The steps of the algorithm are to be performed in the order given except as indicated by branching.

1. For each box in the list L_1, apply hull consistency to each of the inequality constraints as described in Section 10.5.

2. If $\bar{\mathfrak{f}} < +\infty$, then for each box in L_1, apply hull consistency to the inequality $f \leq \bar{\mathfrak{f}}$.

3. If L_1 is empty, go to Step 45. Otherwise, select (for the next box $\mathbf{x}^\mathbf{I}$ to be processed by the algorithm) the box in L_1 for which $\underline{f}(\mathbf{x}^\mathbf{I})$ is smallest. For later reference, denote this box by $\mathbf{x}^{\mathbf{I}(1)}$. Delete $\mathbf{x}^\mathbf{I}$ from L_1.

4. If $\mathbf{x}^\mathbf{I}$ is certainly feasible, go to 13

5. Skip this step if $\mathbf{x}^\mathbf{I}$ has not changed since Step 1. Apply hull consistency over $\mathbf{x}^\mathbf{I}$ to each constraint inequality. If $\mathbf{x}^\mathbf{I}$ is deleted, go to Step 3.

6. Compute an approximation $\mathbf{x}$ for the center $m(\mathbf{x}^\mathbf{I})$ of $\mathbf{x}^\mathbf{I}$ and an approximation $f(\mathbf{x})$ for the value of f at $\mathbf{x}$. If $f(\mathbf{x}) > \bar{\mathfrak{f}}$, go to Step 8.

7. Do the constrained line search described in Section 14.4 to try to reduce $\bar{f}$. If $\bar{f}$ is not reduced, go to Step 10.

8. Apply hull consistency to the inequality $f(\mathbf{x}) \leq \bar{f}$. If $\mathbf{x}^{\mathbf{I}}$ is deleted, go to Step 3.

9. If flag $\mathcal{F} = 0$, go to Step 9a. If flag $\mathcal{F} = 1$, go to Step 9b.

   (a) If $w(\mathbf{x}^{\mathbf{I}}) \leq \varepsilon_X$, and $w[f(\mathbf{x}^{\mathbf{I}})] \leq \varepsilon_f$, put $\mathbf{x}^{\mathbf{I}}$ in list L_2 and go to Step 10. Otherwise, go to Step 9c

   (b) If $p_i(\mathbf{x}^{\mathbf{I}}) \leq \varepsilon_p$ for all $i = 1, \cdots, m$, put $\mathbf{x}^{\mathbf{I}}$ in list L_2 and go to Step 10. Otherwise, go to Step 9c.

   (c) If $\mathbf{x}^{\mathbf{I}}$ is sufficiently reduced (as defined using (11.7.4)) relative to the box $\mathbf{x}^{\mathbf{I}(1)}$ defined in Step 3, put $\mathbf{x}^{\mathbf{I}}$ in list L_1 and go to Step 3.

10. Apply box consistency (as described in Section 10.2) to each constraint inequality. If $\bar{f} < +\infty$, also apply box consistency to the inequality $f(\mathbf{x}) \leq \bar{f}$. If $\mathbf{x}^{\mathbf{I}}$ is deleted, go to Step 3.

11. If $\overline{p}_i(\mathbf{x}^{\mathbf{I}}) \geq 0$ for any $i = 1, \cdots, m$ (i.e., if $\mathbf{x}^{\mathbf{I}}$ is not certainly strictly feasible), go to Step 28.

12. Apply hull consistency to $g_i = 0$ for $i = 1, \cdots, n$ where $\mathbf{g}$ is the gradient of the objective function f. See Section 12.4. If the result for any $i = 1, \cdots, n$ is empty, go to Step 3.

13. Evaluate f at the center of $\mathbf{x}^{\mathbf{I}}$. That is $F\left(\mathrm{m}\left(\mathbf{x}^{\mathbf{I}}\right)\right)$. Use the result to update $\bar{f}$.

14. If $\bar{f} < +\infty$, apply hull consistency to the relation $f(x) \leq \bar{f}$. If the result is empty, go to Step 3.

15. If $\mathrm{w}(\mathbf{x}^{\mathbf{I}}) \leq w_H$, go to Step 20. Otherwise, apply hull consistency to the relation $H_{ii}(\mathbf{x}) \geq 0$ for $i = 1, \cdots, n$ where H_{ii} is an element of the Hessian of f. See Section 12.7. If the result is empty, go to Step 3.

16. Repeat Step 9.

17. Apply box consistency to $g_i = 0$ for $i = 1, \cdots, n$. If the result is empty, go to Step 3.

18. Apply box consistency to $H_{ii}(\mathbf{x}) \geq 0$ for $i = 1, \cdots, n$. If the result is empty, go to Step 3.

19. Repeat Step 9.

20. If $\mathrm{w}(\mathbf{x}^{\mathbf{I}}) > (w_I^{\mathbf{g}} + w_R^{\mathbf{g}})/2$ (see Section 14.7), go to Step 28.

21. Generate the interval Jacobian $\mathbf{J}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$ of the gradient $\mathbf{g}$ and compute the approximate inverse $\mathbf{B}$ of the center of $\mathbf{J}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$. See Section 5.6. Compute $\mathbf{M}\left(\mathbf{x}, \mathbf{x}^{\mathbf{I}}\right) = \mathbf{BJ}\left(\mathbf{x}, \mathbf{x}^{\mathbf{I}}\right)$ and $\mathbf{r}^{\mathbf{I}}(\mathbf{x}) = -\mathbf{Bf}^{\mathbf{I}}(\mathbf{x})$ .Update $w_I^{\mathbf{g}}$ and $w_R^{\mathbf{g}}$. (See Section 14.7.) Apply one step of an interval Newton method to solve $\mathbf{g} = \mathbf{0}$. If the result is empty, go to Step 3.

22. Repeat Step 9.

23. The user might wish to bypass use of analytic preconditioning (see Section 11.9). If so, go to Step 27. To apply analytic preconditioning, use the matrix $\mathbf{B}$ found in Step 21 to obtain $\mathbf{Bg}$ in analytic form. Apply hull consistency to solve the i-th equation of $\mathbf{Bg} = \mathbf{0}$ for the i-th variable x_i for $i = 1, \cdots, n$. If the result is empty, go to Step 3.

24. Repeat Step 9.

25. Use box consistency to solve the i-th equation of $\mathbf{Bg}$ (as obtained in Step 23). for the i-th variable for $i = 1, \cdots, n$. If the result is empty, go to Step 3.

26. Repeat Step 9.

27. Use the matrix $\mathbf{B}$ found in Step 21 in the search method of Section 12.6 to try to reduce the upper bound $\bar{f}$.

28. Compute an approximation $\mathbf{x}$ for the center $\mathrm{m}(\mathbf{x}^{\mathbf{I}})$ of $\mathbf{x}^{\mathbf{I}}$ and an approximate value of $f(\mathbf{x})$. If $f(\mathbf{x}) > \bar{f}$, go to Step 30.

29. Skip this step and go to Step 36 if $\mathbf{x^I}$ is the same box for which a line search was done in Step 7. Otherwise, do the line search described in Section 14.4 to try to reduce $\bar{f}$. If $\bar{f}$ is not reduced, go to Step 36.

30. If $\mathrm{w}(\mathbf{x^I}) > (w_I^f + w_S^f)/2$ (see Section 14.7), go to Step 36.

31. Use the linear method of Section 12.5.3 to try to reduce $\mathbf{x^I}$ using the inequality $f(\mathbf{x}) \leq \bar{f}$. Update w_I^f and w_S^f. If $\mathbf{x^I}$ is deleted, go to Step 3. Otherwise, if this application of the linear method does not sufficiently reduce (as defined using (11.7.4)) the box considered in Step 30, go to Step 35.

32. Repeat Step 9.

33. If $\mathrm{w}(\mathbf{x^I}) > (w_I^S + w_S^S)/2$ (see Section 14.7.) go to Step 36.

34. Use the quadratic method of Section 12.5.4, to try to reduce $\mathbf{x^I}$ using the inequality $f(\mathbf{x}) \leq \bar{f}$. Update w_I^S and w_S^S (see Section 14.7.) If $\mathbf{x^I}$ is deleted, go to Step 3.

35. Repeat Step 9.

36. If $\mathrm{w}(\mathbf{x^I}) > (w_I^p + w_S^p)/2$ (see Section 14.7), go to Step 43.

37. If inequality (14.6.4) is not satisfied, go to Step 43. Otherwise, using the selection process of Section 14.6, choose the constraints to be solved in linearized form by the method in Chapter 6. Add to this set the inequality $f(\mathbf{x}) \leq \bar{f}$ if (14.6.3) is satisfied. If no inequalities pass the selection tests, go to Step 43. Otherwise, linearize the resulting set of inequalities using the expansion given in Section 7.3. (See also Section 6.2). Solve the resulting set of linear inequalities by the method of Chapter 6. Update w_I^p and w_S^p. If the solution set is empty, go to Step 3.

38. Repeat Step 9.

39. In Step 37, the procedure of Chapter 6 generates a preconditioning matrix $\mathbf{B}$. Also, in Step 37, the procedure in Section 14.6 determines

an integer r'. If $r' \geq n/2$, use $\mathbf{B}$ to analytically precondition the set of inequalities that were selected for use in Step 37. Use hull consistency to solve each of the $2r'$ inequalities described in Section 14.6. In so doing, each inequality is solved for the same (single) variable for which the linearized inequality was solved in Step 37.

40. Repeat Step 9.

41. Use box consistency to solve the same inequalities for the same variables as in Step 39.

42. Repeat Step 9.

43. If $\mathrm{w}(\mathbf{x^I}) > (w_I^{\mathbf{J}} + w_R^{\mathbf{J}})/2$ go to Step 46.

44. Modify the John conditions by omitting those constraints p_i for which $p_i(\mathbf{x^I}) < 0$ (since they are not active in $\mathbf{x^I}$). Apply one pass of the interval Newton method of Section 11.14 to the (modified) John conditions. Update $w_I^{\mathbf{J}}$ and $w_R^{\mathbf{J}}$. If the result is empty, go to Step 3.

45. Repeat Step 9.

46. In various previous steps, gaps might have been generated in components of $\mathbf{x^I}$. If so, merge any of these gaps that overlap. Use the procedure described in Section 11.8 to split $\mathbf{x^I}$. Note that the vector of functions use in defining the Jacobian in Section 11.8 is now the gradient. Note also that when the Newton method has been used, the Jacobian elements needed in (11.8.1), will have been determined in Step 21 or Step 44.

    Put the generated subboxes in L_1 and go to Step 3.

47. If flag $\mathcal{F} = 1$, go to Step 51. Otherwise, set $\mathcal{F} = 1$.

48. For each box $\mathbf{x^I}$ in list L_2, do the following: If $p_i(\mathbf{x^I}) \not\leq \varepsilon_p$ for any $i = 1, \cdots, m$, put $\mathbf{x^I}$ in list L_1.

49. If any box was put in list L_1 in Step 48, go to Step 3.

50. If $\bar{\mathfrak{f}} < +\infty$, apply hull consistency to $f(\mathbf{x}) \leq \bar{\mathfrak{f}}$ for each box in the list L_2. Denote those that remain by $\mathbf{x}^{\mathrm{I}(1)},...,\mathbf{x}^{\mathrm{I}(s)}$ where s is the number of boxes remaining. Determine

$$\underline{F} = \min_{1 \leq i \leq s} \underline{f}(\mathbf{x}^{\mathrm{I}(i)}) \text{ and } \overline{F} = \max_{1 \leq i \leq s} \overline{f}(\mathbf{x}^{\mathrm{I}(i)}).$$

51. Terminate.

14.9 RESULTS FROM THE ALGORITHM

After termination, $\mathrm{w}(\mathbf{x}^{\mathrm{I}}) < \varepsilon_X$ and $\mathrm{w}[f(\mathbf{x}^{\mathrm{I}})] < \varepsilon_f$ for each remaining box $\mathbf{x}^{\mathrm{I}}$. Also, $\underline{F} \leq f(\mathbf{x}) \leq \overline{F}$ for every point $\mathbf{x}$ in all remaining boxes. If, after termination, $\bar{\mathfrak{f}} < +\infty$, we know there is a feasible point in the initial box(es). Therefore, we know that

$$\underline{F} \leq f^* \leq \min\{\bar{\mathfrak{f}}, \overline{F}\}.$$

If, after termination, $\bar{\mathfrak{f}} = +\infty$, then we have not found a certainly feasible point. There might or might not be one in $\mathbf{x}^{\mathrm{I}(0)}$. However, we know that if a feasible point does exist in $\mathbf{x}^{\mathrm{I}(0)}$, then

$$\underline{F} \leq f^* \leq \overline{F}.$$

Suppose a feasible point exists. If our algorithm fails to find a certainly feasible point, then it does not produce an upper bound $\bar{\mathfrak{f}}$ and cannot use the relation $f \leq \bar{\mathfrak{f}}$. In particular, it cannot delete local minima where $f(\mathbf{x}) > f^*$. In this case, all local minima are contained in the output boxes.

If all of the initial box $\mathbf{x}^{\mathrm{I}(0)}$ is deleted by our algorithm, then we have proved that every point in $\mathbf{x}^{\mathrm{I}(0)}$ is infeasible. Suppose that every point in $\mathbf{x}^{\mathrm{I}(0)}$ is infeasible. Our algorithm might prove this to be the case. However, we delete a subbox of $\mathbf{x}^{\mathrm{I}(0)}$ only if it is *certainly* infeasible. Rounding errors and/or dependence can prevent us from proving certain infeasibility of an infeasible subbox. Increased wordlength can reduce rounding errors and decreasing ε_X can reduce the effect of dependence by causing subboxes to eventually become smaller. However, neither effect can be removed.

Suppose $\bar{\mathrm{f}} = +\infty$ after termination and $\mathbf{x}^{\mathbf{I}(0)}$ has not been entirely eliminated. It might still be possible either to compute $\bar{\mathrm{f}} < +\infty$ or to delete all of $\mathbf{x}^{\mathbf{I}(0)}$ by reducing the values of ε_X and ε_f and continuing to apply the algorithm. To try to do so, we need only to reduce these tolerances and move the boxes from list L_2 to list L_1. We can then restart the algorithm from the beginning with or without use of increased precision.

A user might want a single feasible point $\widetilde{\mathbf{x}}$ such that

$$||\widetilde{\mathbf{x}} - \mathbf{x}^*|| \leq \varepsilon_1 \tag{14.9.1}$$

and/or

$$f(\widetilde{\mathbf{x}}) - f^* \leq \varepsilon_2 \tag{14.9.2}$$

for some ε_1 and ε_2. Recall that $\mathbf{x}^*$ is a (feasible) point such that $f(\mathbf{x}^*) = f^*$ is the globally minimum value of the objective function f. Our algorithm might or might not fully provide such a point. We distinguish four cases.

Case 1. There is only one final box $\mathbf{x}^{\mathbf{I}}$ and $\bar{\mathbf{x}} \in \mathbf{x}^{\mathbf{I}}$ and $\bar{\mathrm{f}} < +\infty$. (Recall that $\bar{\mathbf{x}}$ is the feasible point where the smallest upper bound $\bar{\mathrm{f}} = f(\bar{\mathbf{x}})$ on f^* was determined by the algorithm.)

Since $\mathbf{x}^*$ is never deleted by the algorithm, it must be in the single remaining box $\mathbf{x}^{\mathbf{I}}$. We can choose $\widetilde{\mathbf{x}} = \bar{\mathbf{x}}$. Then the stopping criteria (see Step 9 of the algorithm) assure that

$$\begin{aligned} ||\widetilde{\mathbf{x}} - \mathbf{x}^*|| &\leq \varepsilon_X \text{ and} \\ f(\widetilde{\mathbf{x}}) - f^* &\leq \varepsilon_f. \end{aligned}$$

Case 2. There is only one final box $\mathbf{x}^{\mathbf{I}}$ and $\bar{\mathbf{x}} \notin \mathbf{x}^{\mathbf{I}}$ and $\bar{\mathrm{f}} < +\infty$.

In this case, we know that a feasible point (i.e., $\overline{\mathbf{x}}$) exists in the initial box because $\bar{\mathfrak{f}} < +\infty$. Therefore, the final box $\mathbf{x}^{\mathbf{I}}$ contains a feasible point because f^* has not been deleted. If we can find a certainly feasible point in $\mathbf{x}^{\mathbf{I}}$, it will serve as the desired point $\widetilde{\mathbf{x}}$. We might be able to find such a point by using some search procedure. If so, we have case 1. However, it might not be possible to find a feasible point in $\mathbf{x}^{\mathbf{I}}$. For example, the part of the feasible region remaining in the final box when the procedure terminates might be a single feasible point that cannot be certainly feasible.

An alternative is to accept a point $\widetilde{\mathbf{x}}$ as feasible if

$$p_i(\widetilde{\mathbf{x}}) \leq \varepsilon_p \text{ for all } i = 1, \cdots, m \tag{14.9.3}$$

for some $\varepsilon_p > 0$. We do not (and should not) use (14.9.3) to delete points in the optimization algorithm. However, we can add a convergence condition of this kind to the algorithm without altering the correctness of the algorithm. The condition can be useful for the present purpose of finding a suitable point $\widetilde{\mathbf{x}}$ near $\mathbf{x}^*$. The convergence condition that we can use is

$$p_i(\mathbf{x}^{\mathbf{I}}) \leq \varepsilon_p \text{ for all } i = 1, \cdots, m. \tag{14.9.4}$$

We add this condition to the convergence conditions $w(\mathbf{x}^{\mathbf{I}}) < \varepsilon_X$ and $w[f(\mathbf{x}^{\mathbf{I}})] < \varepsilon_f$ in Step 9 of the algorithm in Section 14.8. When determining $\widetilde{\mathbf{x}}$, we assume that a final box is "feasible" if it satisfies (14.9.4). However, if we can determine a suitable point $\widetilde{\mathbf{x}}$ that is certainly feasible, we do so.

Note that rather than simply testing whether (14.9.4) is true, we should apply hull consistency to the inequalities as discussed in Section 10.10. This might reduce the box being tested whenever Step 9 is used.

Case 3. There is more than one final box and $\bar{\mathfrak{f}} < +\infty$.

In this case, $\mathbf{x}^*$ can be anywhere in any final box If $\overline{\mathbf{x}}$ is in a final box, we can let $\widetilde{\mathbf{x}} = \overline{\mathbf{x}}$. Then $f(\widetilde{\mathbf{x}}) - f^* \leq 2\varepsilon_f$. If $\overline{\mathbf{x}}$ is not in a final box, we can assure (14.9.4) holds and use the argument in **Case 2**.

Case 4. $\bar{\mathrm{f}} = +\infty$.

In this case, we do not know if there is any feasible point in the initial box in which the algorithm began its search. However, if we assure that (14.9.4) is satisfied and accept a point as feasible when it is satisfied, then any point in any final box is "feasible" and the condition $f(\widetilde{\mathbf{x}}) - f^* \leq \varepsilon_f$ is satisfied for any point $\widetilde{\mathbf{x}}$ in any final box.

14.10 DISCUSSION OF THE ALGORITHM

It is possible that, for a given problem, the feasible region does not have an interior. In this case, the algorithm will probably not find a certainly feasible point. As a result, the algorithm will not be able to delete local minima. In this case, we can proceed as follows.

Let $\mathbf{x}^{\mathbf{I}}$ be a final "solution" box produced by the algorithm. Evaluate the constraints over $\mathbf{x}^{\mathbf{I}}$. We will not find that $p_i(\mathbf{x}^{\mathbf{I}}) > 0$ for any $i = 1, \cdots, m$ because otherwise, the box would have been deleted by the algorithm. If $p_i(\mathbf{x}^{\mathbf{I}}) < 0$ $(i = 1, \cdots, m)$ the i-th constraint is disregarded in what follows (while we are considering the box $\mathbf{x}^{\mathbf{I}}$). The remaining constraints probably pass through $\mathbf{x}^{\mathbf{I}}$ and the stopping criteria assure that $\mathbf{x}^{\mathbf{I}}$ is small. Therefore, there is a reasonable chance that the remaining constraints have a common point in $\mathbf{x}^{\mathbf{I}}$.

We now try to prove that there is a point in $\mathbf{x}^{\mathbf{I}}$ satisfying the remaining constraints written as equalities. A procedure for doing so is given in Sections 15.4 through 15.6. If this procedure is successful, it proves existence of a solution in a box $\mathbf{x}^{\mathbf{I}\prime}$ contained in $\mathbf{x}^{\mathbf{I}}$. Now $\overline{f}(\mathbf{x}^{\mathbf{I}\prime})$ is an upper bound on the global minimum. If we do this process for each of the boxes that remains after the optimization algorithm terminates, we are likely to obtain an upper bound on the global minimum in at least one of the final boxes.

The stopping criteria in Step 9 require that a box $\mathbf{x}^{\mathbf{I}}$ satisfy $\mathrm{w}(\mathbf{x}^{\mathbf{I}}) \leq \varepsilon_X$, $\mathrm{w}\left(f(\mathbf{x}^{\mathbf{I}})\right) \leq \varepsilon_f$, and $p_i(\mathbf{x}^{\mathbf{I}}) \leq \varepsilon_p$ $(i = 1, \cdots, m)$. It would be possible to check that all three condition are satisfied each time Step 9 is used. However, if there are several (or many) inequality constraints, the first two conditions require less work to check than the third. A box that satisfies the first two conditions might eventually be deleted using a procedure such as that in Section 14.3. Therefore we do not use the third criterion until near the end of the algorithm. The flag $\mathcal{F}$ enables us to postpone use of the criterion. Note that when $\mathcal{F} = 1$, the first two conditions are satisfied so it is not necessary to check them in Step 9. Note that Steps 12 through 27 are essentially the same as corresponding steps in the algorithm for unconstrained optimization in Section 12.14. This is because these steps are applied to a box that is certainly feasible.

In our algorithm, we avoid using more complicated procedures until the simpler ones no longer make sufficient progress in reducing the current box. For example, we delay use of the John conditions until all other procedures have been used.

We avoid using procedures that use Taylor expansions until we have evidence that expanded forms provide sufficiently accurate approximations to functions. See steps 20, 30, 33, 36, and 43.

Inequality constraints are often simple relations of the form $x_i \leq b_i$ or $x_i \geq a_i$. Such constraints serve to determine the initial box $\mathbf{x}^{\mathbf{I}(0)}$. Therefore, they are satisfied throughout $\mathbf{x}^{\mathbf{I}(0)}$. Such constraints are omitted when applying any procedure designed to eliminate infeasible points. See Steps 1, 5, 10, and 37.

In Step 7, we use a line search to try to reduce $\bar{f}$. This involves evaluating the gradient of f. We can avoid this evaluation by simply checking if the midpoint $\mathbf{x}$ of the box is feasible and, if so, using $f(\mathbf{x})$ as a candidate value for $\bar{f}$. However, it helps to have a finite value of $\bar{f}$ early, so the line search is worth doing when $\bar{f} = +\infty$. Step 29 also uses a line search. It is less important here because a finite value of $\bar{f}$ is likely to be computed in Step 7. If there are a large number of constraints, then evaluating the gradient is not a dominant part of the work to do the line search.

Experience has shown that efficiency is enhanced if the subbox $\mathbf{x}^{\mathbf{I}}$ to be processed is chosen to be the one for which $\underline{f}(\mathbf{x}^{\mathbf{I}})$ is smallest among all candidate subboxes. This tends to cause a smaller value of $\bar{f}$ to be computed sooner. Therefore, we return to Step 3 to choose a new subbox whenever the current box has changed substantially.

Suppose we find that $p_i(\mathbf{x}^{\mathbf{I}}) \leq 0$ for some value of i and some box $\mathbf{x}^{\mathbf{I}}$. Then $p_i(\mathbf{x}^{\mathbf{I}\prime}) \leq 0$ for any $\mathbf{x}^{\mathbf{I}\prime} \subset \mathbf{x}^{\mathbf{I}}$. Therefore, we can record the fact that

$p_i(\mathbf{x}^\mathbf{I}) \leq 0$ so that we need not evaluate $p_i(\mathbf{x}^{\mathbf{I}'})$.

It is possible that the procedures in Step 21, 23, 25 or 44 prove the existence of a solution to the optimization problem. If so, the user can be informed of this fact. Such a solution can be local or global.

The splitting procedure called in Step 46 uses a measure of change in the gradient of the objective function to determine how to split a box. If the global minimum of f does not occur at a stationary point of f, the change in the gradient is not directly significant. Nevertheless, it is a useful measure in splitting.

14.11 PEELING

The inequality constrained problem can be solved in a different way. We can use peeling. To do so, we first solve an equality constrained problem. Then we solve an inequality constrained problem in which the solution must occur at a stationary point of the objective function. In this second problem, the gradient of f must be zero at a solution and the Hessian must be positive semidefinite. This allows us to use Steps 12 through 27 of the algorithm for any box regardless of whether it is feasible or not. In effect, we solve the problem as if it is unconstrained; but we use the inequality constraints to delete certainly infeasible points.

The term peeling was introduced in Kearfott (1992) and the procedure is discussed in Kearfott (1996). It is the same as a method outlined by Moore (1966). The purpose is to simplify problems in which the constraints are simple bound constraints of the form $x_i \geq a_i$ and $x_i \leq b_i$ $(i = 1, \cdots, n)$. An optimization problem is solved on each face of the box formed by these constraints with the edges remove. Other problems are solved on each edge with the corners removed and for each corner of the box. Finally, an unconstrained problem is solved to find any solution in the interior of the box. Thus 3^n simple problems in reduced dimensions are solved. The smallest solution to all these problems is the desired solution to the original problem.

This approach is efficient only for problems of very small dimension. We now describe an alternative procedure that requires solving only two optimization problems. Also, we remove the condition used by Moore and Kearfott that the constraints $p_i(\mathbf{x}) \leq 0$ be simple bounds.

In the first of our two optimization problems, the solution occurs in the interior of the feasible region. In this case, the solution occurs at a stationary point of the objective function. This fact permits application of powerful procedures that are not generally applicable for a constrained problem. The solution to the second of the two optimization problems occurs on the boundary of the feasible region. This fact enables the algorithm to delete subboxes in the interior of the feasible region and quickly narrow the region of search.

The solution to the original problem (14.1.1), is the smaller of the solutions of the two problems that we introduce. Note that the feasible region might not have an interior. In this case, the first problem has no solution.

To find a problem whose solution is in the interior of the feasible region, we can simply replace the inequality constraints in problem (14.1.1) by strict inequalities. However, there is no need to do so because it does not matter if a solution of the first problem occurs on the boundary of the feasible region rather than in the interior.

The point is that the solution now occurs at a stationary point of $f(\mathbf{x})$. Therefore, the gradient $\mathbf{g}(\mathbf{x})$ of $f(\mathbf{x})$ must be zero and the Hessian $\mathbf{H}(\mathbf{x})$ must be positive semidefinite. These facts provide powerful procedures for computing the desired minimum of $f(\mathbf{x})$. See Sections 12.4 and 12.7. Also, see Steps 12 through 25 of the algorithm in Section 14.8. Note that these conditions on $\mathbf{g}(\mathbf{x})$ and $\mathbf{H}(\mathbf{x})$ replace the more complicated John conditions.

We can express the first optimization problem as

$$\begin{aligned}&\text{minimize } f(\mathbf{x}) \qquad (14.11.1)\\ &\text{subject to: the solution point } \mathbf{x}^* \text{ is a stationary point of } f(\mathbf{x}) \text{ and}\\ &p_i(\mathbf{x}) \leq 0 \;\; (i = 1, \cdots, m)\,.\end{aligned}$$

However, there is no need to express the stationarity condition as a constraint; but it is convenient to do so.

We now derive the second new optimization problem. In an inequality constrained optimization problem, constraints $p_i(\mathbf{x})$ are imposed. These are called *prescribed constraints*. If they do not fully define the initial box $\mathbf{x}^{\mathbf{I}(0)}$, we implicitly add enough simple bound constraints to define $\mathbf{x}^{\mathbf{I}(0)}$.

Define the function

$$q(\mathbf{x}) = \prod_{i=1}^{m} p_i(\mathbf{x}).$$

where only prescribed constraints p_i are included in q. Consider the optimization problem

$$\begin{aligned} &\text{minimize } f(\mathbf{x}) \\ &\text{subject to } q(\mathbf{x}) = \mathbf{0}. \end{aligned} \tag{14.11.2}$$

The solution to this problem must occur where at least one of the constraint functions is equal to zero. If the constraints are all simple bound constraints, this solution occurs on the boundary of the feasible region (which in this case is the initial box). The problem can be solved by the algorithm given in Section 15.12.

Suppose that at least one prescribed constraint is not a simple bound constraint. Then a minimum on one constraint might not satisfy another constraint. Therefore, we formulate the problem as

$$\begin{aligned} &\text{minimize } f(\mathbf{x}) \\ &\text{subject to } p_i(\mathbf{x}) \leq 0 \;\; (i = 1, \cdots, m), \\ &\qquad\qquad q(\mathbf{x}) = \mathbf{0}. \end{aligned} \tag{14.11.3}$$

Let S denote the region satisfying all the prescribed inequality constraints $p_i(\mathbf{x}) \leq 0$. This is the feasible region of the original problem (14.1.1). Adding the constraint $q(\mathbf{x}) = \mathbf{0}$ prevents the solution of (14.11.3) from occurring in the interior of S. Therefore, the region of search is reduced more quickly for (14.11.3) than for (14.1.1).

The global minimum is the smaller of the solutions of (14.11.1) and (14.11.3). We can solve for either of these values first. However, it is best to solve (14.11.1) first. The reason is as follows. When solving either problem, the applied procedure can use an upper bound $\bar{f}$ on the global minimum found when solving the other problem. See Section 12.5. As will be seen in Chapter 15, it is more difficult to compute a bound when the problem contains equality constraints. Since (14.11.1) has no equality constraints, an upper bound $\bar{f}$ is more easily computed. Consequently, we solve (14.11.1) before solving (14.11.3).

Suppose that when solving problem (14.11.2) or (14.11.3), the current box $\mathbf{x}^\mathbf{I}$ is such that $p_i(\mathbf{x}^\mathbf{I}) < 0$ for some value of i. Then we can drop the factor $p_i(\mathbf{x})$ from the function $q(\mathbf{x})$.

A precaution is taken when generating the John condition for problem (14.11.2) or (14.11.3) to enhance the chance that the interval Jacobian does not contain a singular matrix. The John conditions for (14.11.2) are

$$\begin{aligned} \nabla f(\mathbf{x}) + v\nabla q(\mathbf{x}) &= \mathbf{0}, \\ q(\mathbf{x}) &= \mathbf{0} \end{aligned}$$

where v is a Lagrange multiplier. A column of the Jacobian of these equations generated by differentiation with respect to v is

$$\begin{bmatrix} \nabla q \\ 0 \end{bmatrix} \tag{14.11.4}$$

For simplicity, assume there are only two inequality constraints so that $q = p_1 p_2$. Then

$$\nabla q = p_1 \nabla p_2 + p_2 \nabla p_1.$$

We noted above that, for a given box $\mathbf{x}^\mathbf{I}$, a factor p_i does not occur in q if $p_i(\mathbf{x}^\mathbf{I}) < 0$ and the box is infeasible if $p_i(\mathbf{x}^\mathbf{I}) > 0$. Therefore $0 \in \nabla q(\mathbf{x}^\mathbf{I})$ and the column of the Jacobian given by (14.11.4) contains the zero vector. That is, the Jacobian is singular.

To avoid this, we can expand the John conditions in such a way that ∇q is not evaluated over the entire box. We can do so by using the sequential expansion of Section 7.3 and choosing the Lagrange multiplier v to be the last variable in the sequence. As a result, ∇q in (14.11.4) is still evaluated using interval arithmetic, but with real (noninterval) arguments. This enhances the chance that the resulting interval Jacobian does not contain a singular matrix.

This procedure is followed whenever q contains two or more factors

14.12 PILLOW FUNCTIONS

A recurring problem in interval analysis is that of bounding the range of a scalar function f of a vector $\mathbf{x} = (x_1, \cdots x_n)^T$ over a box $\mathbf{x}^\mathbf{I}$ specified by $a_i \leq x_i \leq b_i$ $(i = 1, \cdots, n)$. In particular, this problem arises as a subproblem when solving systems of equations or in constrained or unconstrained optimization. The problem is solved if we solve the two inequality constrained problems

$$\begin{aligned} &\text{minimize } f(\mathbf{x}) \\ &\text{subject to } a_i \leq x_i \leq b_i \ \ (i = 1, \cdots, n) \end{aligned} \tag{14.12.1}$$

and

$$\begin{aligned} &\text{maximize } f(\mathbf{x}) \\ &\text{subject to } a_i \leq x_i \leq b_i \ \ (i = 1, \cdots, n)\,. \end{aligned} \tag{14.12.2}$$

Frequently, we want only crude bounds on the range of f over the box $\mathbf{x}^\mathbf{I}$. Simply evaluating the function over the box provides bounds. (This follows from the fundamental theorem of interval analysis.) However, such bounds are often too far from sharp because of dependence. Beginning with Moore (1966), methods have been sought that are better than simple evaluation but less work than obtaining sharp bounds. Various methods of this kind can be found in Ratschek and Rokne (1984). Recently, methods using Bernstein polynomials have been studied. For example, see Garloff, J. and Smith, A. P. (2000).

In this section, we provide a method of the "crude bound type" which can be regarded as a simplification of peeling. It generally provides sharper bounds than most "crude bound methods"; but falls short of providing sharp bounds. Pillow functions are particularly helpful in providing an efficient method for performing *crude range tests* (CRTs). See Walster and Hansen (2004)[1]. It is not yet known how much impact on the speed of interval algorithms can be achieved using CRTs.

Given a box $\mathbf{x}^{\mathbf{I}}$ defined by $a_i \leq x_i \leq b_i$ $(i = 1, \cdots, n)$, its center is $(c_1, \cdots, c_n)$ where $c_i = (a_i + b_i)/2$ $(i = 1, \cdots, n)$; and the half-width of the i-th component is $u_i = (b_i - a_i)/2$. For any integer $m = 1, 2, \cdots$, the box $\mathbf{x}^{\mathbf{I}}$ is contained in the region specified by $p(\mathbf{x}) \leq 0$ where

$$p(\mathbf{x}) = \left(\frac{x_1 - c_1}{u_1}\right)^{2m} + \cdots + \left(\frac{x_n - c_n}{u_n}\right)^{2m} - n.$$

Note that the graph of $p(\mathbf{x}) = 0$ passes through all the corners of the box and otherwise is outside the box.

For $m = 1$, the equation $p(\mathbf{x}) = 0$ defines an ellipsoid. For higher values of m, it approximates the box more closely. Because of the vague resemblance of the graph of $p(\mathbf{x}) = 0$ to a pillow, we call $p(\mathbf{x})$ a pillow function.

Because any point $\mathbf{x} \in \mathbf{x}^{\mathbf{I}}$ satisfies $p(\mathbf{x}) \leq 0$, the range of $f(\mathbf{x})$ for $\mathbf{x} \in \mathbf{x}^{\mathbf{I}}$ is contained in the range of $f(\mathbf{x})$ as $\mathbf{x}$ varies such that $p(\mathbf{x}) \leq 0$.

[1]There is a mistake in the cited paper: In the second set of optimization problems, the objective function should have a negative sign. Text surrounding equation (25) should be corrected to read:

> By simply reversing the signs in (17) of the objective function and the functions in the inequality constraints, these optimization problems can be converted into:
>
> $$\underset{\mathbf{x} \in \mathbf{x}^{\mathbf{I}}}{\text{minimize}} - f(\mathbf{x}) \tag{25}$$
>
> $$\text{subject to} \begin{cases} \text{c)} \ -f(\mathbf{x}) \leq 0 \\ \text{d)} \ -f(\mathbf{x}) < 0 \end{cases}.$$
>
> These optimization problems are identical to:
>
> $$\underset{\mathbf{x} \in \mathbf{x}^{\mathbf{I}}}{\text{maximize}}\ f(\mathbf{x})$$
>
> $$\text{subject to} \begin{cases} \text{c)} \ f(\mathbf{x}) \geq 0 \\ \text{d)} \ f(\mathbf{x}) > 0 \end{cases}.$$

That is, we can bound the range of $f(\mathbf{x})$ over $\mathbf{x}^I$ by solving the problems

$$\begin{aligned}&\text{minimize } f(\mathbf{x})\\&\text{subject to } p(\mathbf{x}) \leq 0\end{aligned} \tag{14.12.3}$$

and

$$\begin{aligned}&\text{maximize } f(\mathbf{x})\\&\text{subject to } p(\mathbf{x}) \leq 0.\end{aligned} \tag{14.12.4}$$

Problems (14.12.1) and (14.12.2) each have $2n$ constraints; but problems (14.12.3) and (14.12.4) involve only a single constraint. The former problems can be reformulated using peeling (see Section 14.11); but the constraints in the peeling formulation are more complicated than the constraint $p(\mathbf{x}) \leq 0$. The price paid for the simplification offered by (14.12.3) and (14.12.4) is possible loss of sharpness. However, the larger the integer m is chosen, the sharper the bounds on the range because the pillow function more sharply approximates the box $\mathbf{x}^I$.

Note that the inf and sup of f inside a pillow function might actually occur in the box it surrounds. In this case, no sharpness is lost by replacing the box by the pillow.

As an example, suppose we bound the range of the function

$$f(x_1, x_2) = x_1^3 - x_1 x_2^2 + x_1^2 - x_1 x_2 - x_2^2$$

over the box given by $-5 \leq x_1 \leq 5$ and $-3 \leq x_2 \leq 3$. This function was studied by Neumaier (1988).

The range of $f(x_1, x_2)$ over the box is approximately $[-101.6, 151.1]$. For $m = 4$,

$$p(x_1, x_2) = \left(\frac{x_1}{5}\right)^8 + \left(\frac{x_2}{3}\right)^8 - 2.$$

Solving (14.12.3) and (14.12.4) by the method of this chapter, we find that the range of $f(x_1, x_2)$ for $p(x_1, x_2) \leq 0$ is approximately $[-132.4, 191.9]$. For $m = 10$, it is $[-112.0, 165.5]$. If we evaluate $f(\mathbf{x}^I)$ directly, we obtain $[-194, 210]$.

There is an added virtue to this method for getting crude bounds on the range. Suppose we are solving (14.12.3) for the minimum of f over the prescribed region. Using the method of Section 14.8 to solve this problem, we obtain sampled values of the objective function (at feasible points). We also obtain a lower bound on the global minimum over the remaining search regions at any given stage of the solution process. When these values differ by a sufficiently small amount, we can terminate the procedure and accept the current crude bounds. Thus, the procedure can be shortened so that results conform to current needs for sharpness.

Pillow functions have other uses. For example, consider problems such as the assignment problem. For an n-variable assignment problem, the constraints $x_i = 0$ or 1 ($i = 1, ..., n$) are imposed. That is, the solution must occur on a corner of the box $\mathbf{x^I}$ with sides given by $X_i = [0, 1]$. These constraints can be replaced by the two conditions such as

$$\begin{aligned} \sum_{i=1}^{n}(x_i - \frac{1}{2})^2 &= \frac{n}{4} \\ \sum_{i=1}^{n}(x_i - \frac{1}{2})^4 &= \frac{n}{16} \end{aligned}$$

These functions each pass through the corners of $\mathbf{x^I}$, but do not intersect elsewhere. The problem is now expressed in terms of continuous rather than discrete variables.

14.13 NONDIFFERENTIABLE FUNCTIONS

In this chapter, we have assumed that the constraint functions are continuously differentiable and that the objective function is twice continuously differentiable. In this section, we briefly consider how the algorithm of Section 14.8 must be altered to solve problems in which these assumptions are not satisfied.

If the objective function is not twice continuously differentiable, we cannot use the Newton method to solve $\mathbf{g} = \mathbf{0}$ nor to solve the John conditions. Also, we cannot use box consistency to solve $\mathbf{g} = \mathbf{0}$. If the objective function is not continuously differentiable, we cannot use **g** to define a line search.

If the constraints are not continuously differentiable, we cannot use box consistency to solve them. Also, we cannot linearize them to solve them as a system.

The resulting algorithm (without the indicated procedures) is, of course, not as efficient as the full algorithm. Nevertheless, it solves the inequality constrained optimization problem.

We noted in Section 7.11 that slopes can be defined for certain nondifferentiable functions. This can sometimes provide the necessary expansions when differentiability is lacking.

For an example of a nondifferentiable problem solved by interval methods, see Moore, Hansen, and Leclerc (1991).

Some nondifferentiable functions can be replaced by differentiable ones (plus constraints). See Chapter 17.

Chapter 15

EQUALITY CONSTRAINED OPTIMIZATION

15.1 INTRODUCTION

In this chapter, we discuss the global optimization problem in which the only constraints are equality constraints. In this case, problem (13.1.1) becomes

$$\begin{aligned}&\text{Minimize (globally) } f(\mathbf{x}) \qquad (15.1.1)\\ &\text{subject to } q_i(\mathbf{x}) = 0 \ \ (i = 1, \cdots, r)\,.\end{aligned}$$

We assume f is twice continuously differentiable and that q_i $(i = 1, \cdots, r)$ is continuously differentiable. For cases in which these conditions do not hold, see Section 15.15.

We assume an initial box $\mathbf{x}^{\mathbf{I}(0)}$ is given as in Section 12.3; and we seek the global solution of (15.1.1) in $\mathbf{x}^{\mathbf{I}(0)}$. Thus, in effect, we are solving a problem in which inequality constraints occur. However, we ignore these constraints when seeking a solution. The box merely serves to restrict the area of search. We assume $\mathbf{x}^{\mathbf{I}(0)}$ is sufficiently large that it contains the solution to (15.1.1) in its interior.

15.2 THE JOHN CONDITIONS

For problem (15.1.1), the function $\phi(\mathbf{t})$ given by (13.5.1) used to express (part of) the John conditions becomes

$$\phi(\mathbf{t}) = \begin{bmatrix} R(u_0, v) \\ u_0 \nabla f(\mathbf{x}) + v_1 \nabla q_1(\mathbf{x}) + \cdots + v_r \nabla q_r(\mathbf{x}) \\ q_1(\mathbf{x}) \\ \vdots \\ q_r(\mathbf{x}) \end{bmatrix} \qquad (15.2.1)$$

where

$$R(u_0, v) = u_0 + E_1 v_1 + \cdots + E_r v_r - 1 \qquad (15.2.2)$$

if the linear normalization (13.2.2a) is used and

$$R(u_0, v) = u_0 + v_1^2 + \cdots + v_r^2 - 1 \qquad (15.2.3)$$

if the quadratic normalization (13.2.2b) is used.

If the latter normalization is used, we have the initial bounds $0 \leq u_0 \leq 1$ and $-1 \leq v_i \leq 1$ $(i = 1, \cdots, r)$ for the Lagrange multipliers.

In the algorithm for the equality constrained optimization problem given in Section 15.12, we apply hull consistency and box consistency to each individual equality constraint to eliminate points of a box that are certainly infeasible. We could also apply these procedures to the component equations of

$$u_0 \nabla f(\mathbf{x}) + v_1 \nabla q_1(\mathbf{x}) + \cdots + v_r \nabla q_r(\mathbf{x}) = \mathbf{0} \qquad (15.2.4)$$

from the John conditions. However, we do not. To do so requires that we have bounds on the Lagrange multipliers. If these bounds are far from sharp, the consistency procedures are not likely to be very effective when applied to (15.2.4).

It is likely that bounds on the multipliers are reasonably sharp only when the current box $\mathbf{x}^{\mathbf{I}}$ is reasonably small. In this case, a Newton method is likely to be effective in finding any solution in $\mathbf{x}^I$. Therefore, there is no need for consistency methods.

Suppose we use the nonlinear normalization (13.2.2b) of the Lagrange multipliers when expressing the John conditions. To solve the resulting system of equations using an interval Newton method, we need bounds on the Lagrange multipliers. Since we choose not to apply consistency methods to the John conditions, we have no other need for such bounds. Therefore, we use the linear normalization (13.2.2a) so that no bounds are needed for any procedure.

Suppose we use an interval Newton method to solve the John conditions. To do so, we must solve a linearized form of (15.2.1). If we use the Gauss-Seidel method, we require initial bounds on the Lagrange multipliers. If we use Gaussian elimination or the "hull method" of Section 5.8, we do not. Using the normalization (13.2.2a) precludes the use of the Gauss-Seidel method until the Newton step has produced bounds on the multipliers.

In the initial stages of solving an equality constrained optimization problem, a box over which the John conditions are to be solved will tend to be large. As a result, the coefficient matrix of the linearized John conditions will tend to be irregular. In this case, Gaussian elimination and the hull method will fail to solve the linearized equations. One can hope to make progress by using the Gauss-Seidel method. Otherwise, it is necessary to split the box.

For the Gauss-Seidel method, we need bounds on the Lagrange multipliers. A reasonable procedure is the following. In the early stages of solving an equality constrained problem, use the nonlinear normalization (15.2.3). This provides crude bounds on the multipliers so that the Gauss-Seidel method can be used. Later in the solution process when the box is smaller (so that the hull method does not fail), we can switch to the linear normalization (15.2.2). This switch cannot cause a zero of $\phi(\mathbf{t})$ to be lost. That is, no minimum of $\phi(\mathbf{t})$ is lost.

Although we do not need bounds on the Lagrange multipliers (when using (15.2.1)), we do need estimates of their values. We can begin with $u_0 = 1$ and $v_i = 0$ $(i = 1, \cdots, r)$, for example. A successful step of a Newton method provides interval bounds on the Lagrange multipliers. For the next Newton step, the centers of these intervals can serve as the needed estimates.

In our algorithm, we do not iterate the Newton procedure to convergence. Instead, we alternate its use with other procedures. One reason for this is that we do not want to spend effort to get narrow bounds on a local (nonglobal) solution of the optimization problem. Another reason is that other procedures for improving the bounds on a solution require less computing effort, and thus take precedence.

15.3 BOUNDING THE MINIMUM

When considering the unconstrained optimization problem, we discussed (in Section 12.5) how to use an upper bound $\bar{f}$ on the globally minimum value of the objective function $f(\mathbf{x})$. This bound allows us to delete subboxes of the original box $\mathbf{x}^{I(0)}$ in which the global solution cannot occur. We also apply this procedure in the equality constrained case.

Now, it is a much more important procedure. In the unconstrained case, we have other procedures for deleting points that cannot be the solution point. We use the gradient (see Section 12.4) and nonconvexity (see Section 12.7). These procedures are not available for the equality constrained problem. In a sense, these conditions are replaced by the equality constraints. However, there are generally fewer constraint equations than gradient components (which must be zero in the unconstrained case) and no nonconvexity inequalities.

This is one reason the upper bound $\bar{f}$ is more important in the equality constrained case. Another reason is that computing narrow bounds on the location and value of the global minimum requires an upper bound $\bar{f}$ that is near the minimum solution value f^*. We discuss this issue in Section 15.14.

In an unconstrained problem, we can compute $\bar{f}$ by evaluating f at any point $\mathbf{x}$. In a constrained problem, we must prove that $\mathbf{x}$ is feasible to assure that $f(\mathbf{x})$ is an upper bound on f^*. In an inequality constrained problem, we can prove that $\mathbf{x}$ is feasible by numerically verifying that it is certainly feasible (as defined in Section 6.1).

Generally, no point can be certainly feasible for the equality constrained problem because of uncertainty caused by rounding. Consider a point $\mathbf{x}$. If we make a single rounding error in evaluating $q_i(\mathbf{x})$ for any $i = 1, \cdots, r$, then we do not know whether $q_i(\mathbf{x})$ is precisely zero even if it is. That is, we do not know whether $\mathbf{x}$ is feasible or not.

The user might know a finite upper bound for the global minimum. If so, this value can be input as the initial upper bound $\bar{\mathfrak{f}}$. Failing this, there are three ways to compute an upper bound. Each depends upon use of an existence theorem. We discuss one of the methods in this section; and we discuss the others in Sections 15.4 through 15.6.

Assume we use an interval Newton method for which Proposition 11.15.5 holds. This proposition says the following: Suppose we perform a step of the interval Newton method by applying it to a system of nonlinear equations over a box $\mathbf{x}^{\mathbf{I}}$; and it produces a new box $\mathbf{x}^{\mathbf{I}\prime}$. If $\mathbf{x}^{\mathbf{I}\prime} \subset \mathbf{x}^{\mathbf{I}}$, then there exists a solution of the system of equations in $\mathbf{x}^{\mathbf{I}\prime}$.

Suppose we apply a step of such an interval Newton method to the John conditions. Suppose that the new box produced by this step is contained in the original box. Then we have proved existence of a (simple) solution of the John conditions in the new box. Let $\mathbf{x}^{\mathbf{I}\prime}$ denote the new box.

Since we have proved the existence of a solution of the John conditions, there exists a feasible point in $\mathbf{x}^{\mathbf{I}\prime}$. If we evaluate the objective function over $\mathbf{x}^{\mathbf{I}\prime}$ and obtain $f(\mathbf{x}^{\mathbf{I}\prime}) = [\underline{f}(\mathbf{x}^{\mathbf{I}\prime}), \overline{f}(\mathbf{x}^{\mathbf{I}\prime})]$, then $\overline{f}(\mathbf{x}^{\mathbf{I}\prime})$ is an upper bound on f at this feasible point in $\mathbf{x}^{\mathbf{I}\prime}$ and, hence, is an upper bound on the global minimum f^* of f.

When attempting to prove existence in this way, it is best to use the linear normalization condition (13.2.2a) rather than a nonlinear one such as (13.2.2b). The reason is as follows. Suppose we use a nonlinear normalization. Then part of the input data to the Newton method consists of bounds on the Lagrange multipliers. To prove existence, the new computed interval bounds on the multipliers must be contained in the corresponding input intervals.

But, if the linear normalization is used, no such input bounds on the multipliers are needed. They can implicitly be assumed to be arbitrarily wide. Therefore, the multipliers play no part in proving existence. This makes it easier to obtain a proof of existence of a solution to the John conditions. Note, however, that the accuracy of the estimates of the multipliers affects the sharpness of the bounds computed (by the Newton step) for the components of $\mathbf{x}$.

Applying a Newton method to the John conditions requires quite a lot of computing. We do so only if other methods for isolating the global solution are too inefficient. We do not apply this method merely for the sole purpose of finding a bound $\bar{\mathfrak{f}}$ on f^*. However, if a bound is needed in the main program and a Newton step applied to the John conditions provides such a bound, we, of course, use it.

15.4 USING CONSTRAINTS TO BOUND THE MINIMUM

In this section, we describe two other procedures for computing an upper bound $\bar{\mathfrak{f}}$ on the global minimum f^*. Both procedures are based on one due to Hansen and Walster (1990c). For a discussion of other variations, see Kearfott (1996). As in Section 15.3, the procedures find a box guaranteed to contain a feasible point and it bounds f over the box. A guarantee that a box contains a feasible point is obtained by proving existence using Proposition 11.15.5.

Recently a method using the topological index has been developed to prove existence of a solution of a system of equations. See Kearfott and Dian (2000). We shall not discuss this subject.

Suppose we have proved that a feasible point exists in a box $\mathbf{x^I}$. We evaluate $f(\mathbf{x^I})$ getting $[\underline{f}(\mathbf{x^I}), \overline{f}(\mathbf{x^I})]$. Then $\overline{f}(\mathbf{x^I})$ is an upper bound for the global minimum f^*. Each time we obtain such an upper bound, we update our best upper bound, replacing it with $\min\left(\bar{\mathfrak{f}}, \overline{f}(\mathbf{x^I})\right)$.

In most (but not all) problems, the number r of equality constraints is less than the number n of variables. Otherwise, it is generally possible for the constraints themselves to determine the solution point(s) with no variability remaining to minimize f. Therefore, we assume $r < n$.

Suppose we wish to determine whether there is a feasible point in a given box $\mathbf{x^I}$. In Section 15.12, we describe how such a box is produced by the main program. Let $\mathbf{x}$ denote a variable point in $\mathbf{x^I}$, and let $\mathbf{c}$ be the center of $\mathbf{x^I}$. We fix $n-r$ of the variable components of $\mathbf{x}$. For now, assume we set $x_i = c_i$ for $i = r+1, \cdots, n$. In Section 15.5, we make a specific choice of which variables to fix.

Define the r-vector $\mathbf{z} = (x_1, \cdots, x_r)^T$. Consider the equation

$$\mathbf{h}(\mathbf{z}) = \mathbf{0} \tag{15.4.1}$$

where

$$h_i(\mathbf{z}) = q_i(x_1, \cdots, x_r, c_{r+1}, \cdots, c_n) \ \ (i = 1, \cdots, r)$$

and q_i is a constraint function from problem (15.1.1). This is a system of r equations in r unknowns. Let $\mathbf{z^I}$ denote the r-dimensional box with components $Z_i = X_i$ $(i = 1, \cdots, r)$.

If there exists a solution of (15.4.1) in $\mathbf{z^I}$, then it provides a feasible point for problem (15.1.1). We describe two ways in which we can try to prove the existence of a solution in $\mathbf{z^I}$.

15.4.1 First Method

One way is to apply a step of an interval Newton method to solve (15.4.1). We can use any variant of the method for which Proposition 11.15.5 is true. Let $\mathbf{z^{I\prime}}$ denote the solution box. If $\mathbf{z^{I\prime}} \subset \mathbf{z^I}$, then there exists a solution of $\mathbf{h}(\mathbf{z}) = \mathbf{0}$ in $\mathbf{z^{I\prime}}$. Therefore, there exists a feasible point in the n-dimensional box

$$\mathbf{x^{I\prime}} = (Z_1', \cdots, Z_r', c_{r+1}, \cdots, c_n)^T,$$

which is a (partially degenerate) subbox of $\mathbf{x^I}$.

If $\mathbf{z^{I\prime}} \subset \mathbf{z^I}$, the interval Newton step can be iterated to reduce the size of the box bounding the feasible point. The smaller the box, the (generally) smaller the upper bound on $\overline{f}$ we compute by evaluating f over the box. Thus, we can iterate as long as sufficient progress is made reducing the current box. However, f might be much smaller at a feasible point in a box yet to be processed. Therefore, we do not want to spend too much effort to sharply bound a given feasible point.

Suppose we have proved existence of a feasible point in a given box. If we iterate the Newton method, the feasible point is in each computed subbox. See Theorem 11.15.1. Therefore, we do not need proof of existence in subsequent iterations. Note however, that this is true only if we do not change the values of the constants $c_{r+1}, \cdots, c_n$.

Assume we have not proved that $\mathbf{z}^{\mathbf{I}}$ contains a feasible point and that $\mathbf{z}^{\mathbf{I}\prime}$ is not contained in $\mathbf{z}^{\mathbf{I}}$. If there is a feasible point in $\mathbf{z}^{\mathbf{I}}$, it must be in $\mathbf{z}^{\mathbf{I}\prime\prime} = \mathbf{z}^{\mathbf{I}} \cap \mathbf{z}^{\mathbf{I}\prime}$. If we choose to repeat the procedure we use the box

$$\mathbf{x}^{\mathbf{I}\prime} = (Z_1'', \cdots, Z_r'', c_{r+1}, \cdots, c_n).$$

We choose $\mathbf{z}^{\mathbf{I}\prime\prime}$ to have the same center and shape as $\mathbf{z}^{\mathbf{I}} \cap \mathbf{z}^{\mathbf{I}\prime}$, but we reduce the width. See below.

A large portion of the original box can often be deleted using the condition $f(\mathbf{x}) \leq \bar{f}$ (see Sections 12.5 and 14.3) even if $\bar{f}$ is not very close to f^*. Therefore, we try to prove existence of a feasible point in an early stage of the solution of the optimization problem while the current box is large. This provides a finite value for $\bar{f}$. As we point out in Section 15.14, it is important to know whether there is at least one feasible point in the original box. Therefore, we make a special effort to obtain an initial finite value of $\bar{f}$.

If there are a large number of constraints, our procedure for bounding a feasible point is costly in computing effort. Therefore, if we already have a finite value of $\bar{f}$, we make less effort to prove existence of a feasible point in a given box. Also, we do not try to prove existence of a feasible point in a given box $\mathbf{x}^{\mathbf{I}}$ unless $f[\mathrm{m}(\mathbf{x}^{\mathbf{I}})] < \bar{f}$ where $\mathrm{m}(\mathbf{x}^{\mathbf{I}})$ is the center of $\mathbf{x}^{\mathbf{I}}$.

We now consider criteria for deciding how much effort to expend in seeking a bound on a feasible point.

When we try to prove existence of a feasible point in a given box, we want the size of the box to be such that there is a high likelihood of success. If we choose it too small, it is not likely to contain a feasible point. If we choose it too large, the linearized form of the system $\mathbf{h}(\mathbf{z}) = \mathbf{0}$ given by (15.4.1) is likely to contain a singular matrix and we cannot prove existence of a feasible point. We choose a box size based on previous results. We now derive our procedure.

When we use an interval Newton method to solve the system (15.4.1), we precondition the Jacobian (see Section 5.6) to obtain a coefficient matrix that we now denote by $\mathbf{M}$. We can prove existence of a feasible point only if $\mathbf{M}$ is regular.

Suppose we have tried to prove existence in one or more boxes. Let w_R^q denote the width of the largest box for which $\mathbf{M}$ is regular. Let w_I^q denote the width of the smallest box for which $\mathbf{M}$ is irregular. Initially, set $w_R^q = 0$ and $w_I^q = \mathrm{w}(\mathbf{x}^{\mathbf{I}(0)})$ where $\mathbf{x}^{\mathbf{I}(0)}$ is the initial box in which the minimization problem is to be solved. When we try to prove existence, we choose the width of the beginning box to be $w_{ave} = \dfrac{1}{2}(w_R^q + w_I^q)$. This strikes a balance between too big and too small a box.

We now list the steps we use to update w_R^q and w_I^q and to make our decision whether to try to prove existence of a feasible point. Let $\mathrm{m}(\mathbf{x}^{\mathbf{I}})$ denote the center of the current box $\mathbf{x}^{\mathbf{I}}$. When we say we "shrink" a box, we mean we replace it by another with the same center and the same relative dimensions but of width reduced by some factor.

At any stage in the following steps, we call the current box $\mathbf{x}^{\mathbf{I}}$ even though it can change as we proceed through the steps.

1. If $f[\mathrm{m}(\mathbf{x}^{\mathbf{I}})] > \bar{\mathfrak{f}}$ terminate this procedure

2. If $\bar{\mathfrak{f}} = +\infty$, set MAXCOUNT $= 8$. If $\bar{\mathfrak{f}} < +\infty$, set MAXCOUNT $= 4$.

3. Set COUNT $= 0$.

4. If $\mathrm{w}(\mathbf{x}^{\mathbf{I}}) > w_{ave}$, shrink $\mathbf{x}^{\mathbf{I}}$ so that it has width w_{ave}.

5. If COUNT = MAXCOUNT, terminate the procedure.

6. Replace COUNT by COUNT $+ 1$.

7. Apply a Newton step to try to solve (15.4.1) for $\mathbf{z}$. (Note that the r components of $\mathbf{z}$ are generally not the first r of components of $\mathbf{x}$. See Section 15.5.) If successful (because $\mathbf{M}$ is regular), go to Step 9. **Note:** Use the hull method of Section 5.8 to solve the preconditioned equations in the Newton step.

8. Update w_I^q. Shrink $\mathbf{x^I}$ to one eighth its size. Go to Step 5.

9. Update w_R^q. Denote the result of the Newton step by $\mathbf{x^I}'$. If $\mathbf{x^I} \cap \mathbf{x^I}'$ is empty, terminate the procedure.

10. If $\mathbf{x^I}' \subset \mathbf{x^I}$ (so that we have proved existence), update $\bar{f}$.

11. If $f[\mathrm{m}(\mathbf{x^I})] < \bar{f}$, go to Step 5.

12. Terminate the procedure.

Kearfott (1996) reports that a method similar to this works well in practice.

We could use epsilon-inflation to try to prove existence of a feasible point. This procedure was introduced by Rump (1980) and discussed in detail by Mayer (1995). See, also, Kearfott (1996). A first step in the procedure is designed to prove existence of a solution of a system of equations at or near a tentative solution found by a noninterval algorithm. Thus, it is an intensive effort to prove existence of a point that is reasonably certain to exist. Our case is different. We try to prove existence in many boxes that might or might not contain a solution. Therefore we use a simpler procedure. By applying our relatively simple procedure to many boxes, we increase the likelihood of success.

15.4.2 Second Method

Our remaining method for trying to prove existence of a feasible point uses essentially the same procedure as the one we have just described. Now, however, we use hull consistency (see Chapter 10) instead of a Newton method. Proof of existence is obtained using Theorem 10.12.1.

The Newton method can be effective in proving existence because it asymptotically converges quadratically (to simple solutions). In this case, the box $\mathbf{z^I}'$ described above tends to be much smaller than $\mathbf{z^I}$. This enhances the chances that $\mathbf{z^I}' \subset \mathbf{z^I}$ as required to prove existence. Hull consistency cannot be expected to have quadratic convergence in the multidimensional case because it is a one-dimensional procedure. Normally, we make no special effort to try to prove existence using it. However, in the algorithm in Section 15.12, we apply hull consistency to a preconditioned system of equations. For this step, it can be advantageous to implement hull consistency so that it has quadratic convergence as a one dimensional process. See Section 10.6.

In the next two sections, we discuss ways to increase the likelihood of proving that a box contains a feasible point.

15.5 CHOICE OF VARIABLES

For simplicity, we assumed in the previous section that we fixed $x_i = c_i$ for the indices $i = r+1, \cdots, n$. By appropriately choosing which $n-r$ variables to fix, we can enhance the chance of being able to prove the existence of a feasible point in a given box. We now consider how this can be done.

We first note that we might be able to reduce the number of constraints by fixing appropriate variables. As an example, consider problem 61 of Hock and Schittkowski (1981). This is a problem in three variables with two constraints of the form

$$q_1(\mathbf{x}) = 3x_1 - 2x_2^2 - 7 = 0,$$
$$q_2(\mathbf{x}) = 4x_1 - x_3^2 - 11 = 0.$$

If we fix any one of the three variables, we can solve the constraint equations for the other two $\left(\text{provided } x \geq \dfrac{11}{4}\right)$. That is, we can determine a feasible point directly.

For other problems, we might be able to determine a subset of the variables by fixing one or more appropriate variables. This reduces the number of remaining constraints. However, to simplify the discussion,we assume all the constraints remain.

We now consider an illustrative example to show that care must be taken in choosing which variables to fix. Consider a two-dimensional problem in which there is a single constraint

$$q(x_1, x_2) = x_2 - 1 \tag{15.5.1}$$

that is independent of x_1. Let a box $\mathbf{x}^I$ have components $X_1 = X_2 = [-1, 1]$. If we fix the second variable (to be the midpoint of X_2), we set $x_2 = 0$. In the procedure described in Section 15.4, we try to bound a point of the form $(x_1, 0)^T$ satisfying (15.5.1). But, there is no such point.

However, suppose we fix the first variable to be the center $x_1 = 0$ of X_1. Now we wish to find a point of the form $(0, x_2)^T$ satisfying (15.5.1). This is easily done.

We now consider a procedure designed to exploit the idea implicit in this example. Consider the r by n matrix with elements

$$M_{ij}(\mathbf{x}) = \frac{\partial q_i(\mathbf{x})}{\partial x_j} \; (i = 1, \cdots, r; \; j = 1, \cdots, n). \tag{15.5.2}$$

We linearize $\mathbf{q}$ about the center $\mathbf{x}$ of a box $\mathbf{x}^I$. When we do so, some of the arguments of $M_{i,j}$ can be real as described in Section 7.3. However, for the immediate purpose, they can all be real. Denote the resulting r by n coefficient matrix by $\mathbf{M}(\mathbf{x}, \mathbf{x}^I)$ and the linearized system by $q(\mathbf{x}) + \mathbf{M}\left(\mathbf{x}, \mathbf{x}^I\right)(\mathbf{y} - \mathbf{x}) = \mathbf{0}$.

To solve such a system of linear interval equations when it is square, we precondition the system by multiplying by an approximate inverse of the center of the coefficient matrix. (See Section 5.6). We do a similar preconditioning in this nonsquare case.

Let $\mathbf{M}^c$ denote the center of the interval matrix $\mathbf{M}(\mathbf{x}, \mathbf{x}^I)$. We use Gaussian elimination with both row and column pivot searching to transform $\mathbf{M}^c$ into a matrix $\mathbf{M}'$ in upper trapezoidal form. It is the column interchanges for pivot selection that "chooses" the variables to fix. That is, $M'_{ij} = 0$ for all $i > j$ $(i = 2, \cdots, r; \; j = 1, \cdots, i - 1)$. We make a final column interchange (if necessary) so that the final element in position (r, r) has the largest magnitude of any element in row r of the final matrix.

Suppose that, in addition to the elements deliberately zeroed, the elements M'_{ij} of $\mathbf{M}'$ are zero for all $i > m$ for some $m < r$. Then the rows of $\mathbf{M}^c$ are (at least approximately) linearly dependent. In this case, we abandon our effort to find a feasible point in the current subbox of the initial box. We hope to be successful with the next subbox generated by the main algorithm.

However, if $\mathbf{M}'$ has a nonzero element M'_{ii} for all $i = 1, \cdots, r$, then we proceed. If no column interchanges are made in doing Gaussian elimination to produce $\mathbf{M}'$, then it is the last $n - r$ variables that we fix. That is, we set $x_i = \mathrm{m}(X_i)$ for $i = r + 1, \cdots, n$. If column interchanges are made, we fix the variables corresponding to the columns that are interchanged into the last $n - r$ positions.

Note that if none of the constraints depend on the variable x_j for a particular index $j = 1, \cdots, r$, then $M_{ij} = 0$ for all $i = 1, \cdots, r$. In this case, the elimination process interchanges the j-th column (all of whose elements are zero) into one of the last $n - r$ columns. This assures that x_j is fixed and set equal to $\mathrm{m}(X_j)$. That is, only the relevant variables are used when trying to find a feasible point.

If all the constraints are independent of x_j, then f cannot be. Otherwise the problem does not involve x_j. Our process for finding a feasible point (when successful) finds a box $\mathbf{x}^{\mathbf{I}}$ known to contain a feasible point. This box has $n - r$ degenerate components; and the j-th, is one of them. The (degenerate) j-th component of $\mathbf{x}^{\mathbf{I}}$ can be chosen to have any value in the j-th component of the initial box $\mathbf{x}^{\mathbf{I}(0)}$.

As pointed out in Section 15.4, if there is a feasible point in $\mathbf{x}^{\mathbf{I}}$, then $\overline{f}(\mathbf{x}^{\mathbf{I}})$ is an upper bound for f^*. We are free to try to minimize $\overline{f}(\mathbf{x}^{\mathbf{I}})$ by choosing x_j (and any other variables of which the constraints are independent) as best we can in $\mathbf{x}^{\mathbf{I}(0)}$. However, one can simply set x_j equal to the center of the j-th component of the current box.

After we have decided which variables to fix, we again do the linearization. The reason is as follow. The derivatives that determine the linearized equations are narrower after fixing some variables as point values than before. Therefore, a Newton step using the linearized equations has a better chance of proving existence of a solution. The extra effort is warranted because it is important to obtain proof of existence. Note that when we linearize this time, we want to use the kind of expansion described in Section 7.3 so that some of the arguments of the elements of the coefficient matrix are real rather than interval. We want the widths of the elements as small as possible to enhance the chance that the Newton method will be able to prove existence.

In Section 15.4.1, we considered a box $\mathbf{z}^{\mathbf{I}'}$ obtained from a Newton step applied to a box $\mathbf{z}^{\mathbf{I}}$. We might find that $\mathbf{z}^{\mathbf{I}'} \cap \mathbf{z}^{\mathbf{I}}$ is empty. This does not prove that there is no feasible point in $\mathbf{x}^{\mathbf{I}}$. It merely means that there is no feasible point in $\mathbf{x}^{\mathbf{I}}$ having the values of the fixed variables. Nevertheless, if $\mathbf{z}^{\mathbf{I}'} \cap \mathbf{z}^{\mathbf{I}}$ is empty, we abandon our effort to find a feasible point in $\mathbf{x}^{\mathbf{I}}$.

Even if there is a feasible point in $\mathbf{x}^{\mathbf{I}}$, the interval Newton method is not guaranteed to prove this is, in fact, the case. To prove existence, we must have $\mathbf{z}^{\mathbf{I}'} \subset \mathbf{z}^{\mathbf{I}}$. This condition never holds if the solution sought is a singular solution (in the variables not fixed). This possibility is reduced by the way we choose which variables to fix.

The condition $\mathbf{z}^{\mathbf{I}'} \subset \mathbf{z}^{\mathbf{I}}$ can fail to be satisfied even when the solution is simple and well conditioned. This is more likely when the solution is on or near the boundary of $\mathbf{x}^{\mathbf{I}}$. If we have not proved existence of a feasible point in a final box, we can take steps to try to avoid this difficulty by choosing a different box.

For example suppose $Z_i = [a, b]$ and $Z_i' = [c, d]$ and $a < c < b < d$. We can extend $Z_i \cap Z_i' = [c, b]$ by replacing the upper endpoint b by $b + \alpha(b - c)$ for some value of α such as 0.1. It doesn't matter whether the feasible point is in $\mathbf{z}^{\mathbf{I}}$ or not. We only require that the box be in the initial box $\mathbf{x}^{\mathbf{I}(0)}$. By extending Z_i' in this way for each $i = 1, \cdots, r$, the next iteration has a better chance of getting a new box, say $\mathbf{z}^{\mathbf{I}''}$ in the extended version of $Z_i \cap Z_i'$.

15.6 SATISFYING THE HYPOTHESIS

We must have $\mathbf{z}^{\mathbf{I}'} \subset \mathbf{z}^{\mathbf{I}}$ to prove existence of a feasible point. Therefore, it behooves us to try to assure that this condition is satisfied when possible. One way to enhance the chance of satisfying this condition is to make $\mathbf{z}^{\mathbf{I}'}$ as small as possible relative to $\mathbf{z}^{\mathbf{I}}$.

To apply an interval Newton method to solve an equation $\mathbf{h}(\mathbf{z}) = \mathbf{0}$, we linearize $\mathbf{h}(\mathbf{z})$ by expanding about a point, say $\mathbf{z}_0$ in $\mathbf{z}^{\mathbf{I}}$. The interval coefficients in the expansion are narrower if slopes, rather than derivatives, are used to obtain the expansion. This causes $\mathbf{z}^{\mathbf{I}'}$ to be smaller.

Also, the closer the expansion point $\mathbf{z}_0$ point is to a solution, the smaller $\mathbf{z}^{\mathbf{I}'}$ will be. Therefore, we insert a step into the procedure described in Section 15.4. We try to find a good approximation for a zero in $\mathbf{z}^{\mathbf{I}}$ before we apply the interval Newton method. We then choose $\mathbf{z}_0$ to be this approximate point.

The procedure we use to compute the approximation is the noninterval Newton method described in Section 11.4. In this procedure, we use an approximate inverse of the Jacobian of $\mathbf{h}$. This Jacobian is the one (in slope or derivative form) that we use next in applying the interval Newton step to try to prove the existence of a feasible point as described in Section 15.3. The approximate inverse of the center of the Jacobian is found while doing the elimination process described in Section 15.5. Note the similarity to the preconditioning process in Section 11.2.

Recall that, in Section 15.5, we used Gaussian elimination to transform the matrix $\mathbf{M}^c$ into trapezoidal form. To compute the approximate inverse, we also perform the elimination operations on the matrix that begins as the identity matrix of order r. When doing these operations (but not when operating on $\mathbf{M}^c$), we can ignore any column interchanges. When $\mathbf{M}^c$ has been transformed into trapezoidal form, we have transformed the identity matrix into a matrix we call $\mathbf{B}'$.

We can now drop the last $n - r$ columns of the modified form of $\mathbf{M}^c$. We then do the remaining elimination operations to transform the resulting matrix of order r into the identity matrix. Doing these operations on $\mathbf{B}'$ yields the matrix $\mathbf{B}$, which is the (approximate) inverse of the center of the retained submatrix of $\mathbf{M}^c$. We use this noninterval matrix $\mathbf{B}$ as described in Section 11.4 to find an approximate solution $\mathbf{z}_0$ to $\mathbf{h}(\mathbf{z}) = \mathbf{0}$.

15.7 A NUMERICAL EXAMPLE

In this section, we discuss a numerical example illustrating the ideas of the previous two sections. Problem 39 of Hock and Schittkowski (1981) is

$$\text{Minimize } f(\mathbf{x}) = -x_1$$

$$\text{subject to } q_1(\mathbf{x}) = x_1^3 - x_2 + x_3^2 = 0,$$

$$q_2(\mathbf{x}) = x_1^2 - x_2 - x_4^2 = 0.$$

(We ignore the fact that if we fix x_1 and x_2, we can solve for x_3 and x_4.) The matrix given by (15.5.2) is

$$\mathbf{M}(\mathbf{x}) = \begin{bmatrix} 3x_1^2 & -1 & 2x_3 & 0 \\ 2x_1 & -1 & 0 & -2x_4 \end{bmatrix}.$$

Suppose the current box has components $X_1 = [-1.1, -0.7]$, $X_2 = [-1.2, 1]$, $X_3 = [0, 2]$, and $X_4 = [0, 1.6]$. The center of this box is $\mathbf{x} = (-0.9, -0.1, 1, 0.8)^T$. We find the center of $\mathbf{M}(\mathbf{x}^\mathbf{I})$ to be

$$\mathbf{M}^c = \begin{bmatrix} 2.55 & -1 & 2 & 0 \\ -1.8 & -1 & 0 & -1.6 \end{bmatrix}.$$

Using Gaussian elimination to produce a zero in position (2, 1) of the matrix, we obtain (approximately)

$$\mathbf{M}' = \begin{bmatrix} 2.55 & -1 & 2 & 0 \\ 0 & -1.706 & 1.412 & -1.6 \end{bmatrix}.$$

Since no column interchanges were used, we fix the last two variables. Thus, $c_3 = \text{m}(X_3) = 1$ and $c_4 = \text{m}(X_4) = 0.8$.

We now use an interval Newton method to solve the equations

$$h_i(z_1, z_2) = q_i(z_1, z_2, c_3, c_4) \ (i = 1, 2).$$

That is,

$$h_1(\mathbf{z}) = z_1^3 - z_2 + 1 = 0, \ h_2(\mathbf{z}) = z_1^2 - z_2 - 0.64 = 0 \qquad (15.7.1)$$

The Jacobian of $\mathbf{h}(\mathbf{z})$ is

$$\mathbf{J}(\mathbf{z}) = \begin{bmatrix} 3z_1^2 & -1 \\ 2z_1 & -1 \end{bmatrix}.$$

The components of the box $\mathbf{z}^\mathbf{I}$ are $Z_1 = X_1 = [-1.1, -0.7]$ and $Z_2 = X_2 = [-1.2, 1]$. The center of $\mathbf{z}^\mathbf{I}$ has components $z_1 = -0.9$ and $z_2 = -0.1$. Using (7.3.6), we expand $\mathbf{h}$ about the center of $\mathbf{z}^\mathbf{I}$ and obtain

$$\begin{bmatrix} z_1^3 - z_2 + 1 \\ z_1^2 - z_2 - 0.64 \end{bmatrix} + \begin{bmatrix} 3Z_1^2 & -1 \\ 2Z_1 & -1 \end{bmatrix} \begin{bmatrix} z_1' - z_1 \\ z_2' - z_2 \end{bmatrix} = \mathbf{0}.$$

Substituting the appropriate numerical quantities into this equation, we obtain

$$\begin{bmatrix} 0.371 \\ 0.27 \end{bmatrix} + \begin{bmatrix} [1.47, 3.63] & -1 \\ [-2.2, -1.4] & -1 \end{bmatrix} \begin{bmatrix} z_1' + 0.9 \\ z_2' + 0.1 \end{bmatrix} = \mathbf{0}. \qquad (15.7.2)$$

The first two columns of the matrix $\mathbf{M}^c$ form the matrix

$$\begin{bmatrix} 2.55 & -1 \\ -1.8 & -1 \end{bmatrix}.$$

For use in the interval Newton method, we want an approximate inverse $\mathbf{B}$ of this matrix. The operation to get $\mathbf{M}^c$ from $\mathbf{M}$ is a first step in computing $\mathbf{B}$. Completing the process and using $\mathbf{B}$ as a preconditioner when applying the interval Newton method, we obtain

$$\mathbf{z}^{\mathbf{I}'} = \begin{bmatrix} [-0.9352, -0.9173] \\ [0.1856, 0.2380] \end{bmatrix}.$$

We indicate this result using four significant decimal digits. Higher precision was used in the computations.

Since $\mathbf{z}^{\mathbf{I}'} \subset \mathbf{z}^\mathbf{I}$, there exists (by Proposition 11.15.5) a solution of (15.7.1) in $\mathbf{z}^{\mathbf{I}'}$. That is, there is a feasible point $\mathbf{x}$ with $-0.9352 \leq x_1 \leq -0.9173$, $0.1856 \leq x_2 \leq 0.2380$, $x_3 = 1$, and $x_4 = 0.8$. The actual feasible point in $\mathbf{x}^\mathbf{I}$ with $x_3 = 1$ and $x_4 = 0.8$ is at (approximately) $x_1 = -0.9234$ and $x_2 = 0.2126$.

Since the objective function is $f(\mathbf{x}) = -x_1$, and since we know there is a feasible point with $x_1 \in [-0.9352, -0.9173]$, we now have the upper bound $\bar{f} = 0.9352$ on the global minimum f^*.

If we use the inner iteration of Section 11.4 to get a better point about which to expand $\mathbf{h}$, and if we use slopes instead of derivatives to do the expansion, we obtain $\bar{f} = 0.9234$, which is correct to four digits.

The best feasible value of f in the test box is 0.7 at $x_1 = 0.7, x_2 = 0.49$, $x_3 = (0.833)^{1/2}$, and $x_4 = 0$. We have not found this best value because we fixed x_3 and x_4 to have values at the centers of their interval bounds; and this is not where the best feasible point occurs in the box. However, the test box does not contain the solution point $\mathbf{x}^*$ so we must be satisfied with a suboptimal value of $\bar{f}$, anyway.

The global minimum for this problem is $f^* = -1$, which occurs at $\mathbf{x} = (1, 1, 0, 0)^T$. Our upper bound on f^* is far from sharp. However, when solving a problem by the algorithm given in Section 15.12, the process for getting an upper bound on f^* is applied for smaller and smaller subboxes of the initial box. Thus, the upper bound is successively improved.

15.8 USING THE UPPER BOUND

Assume we have computed an upper bound $\bar{f}$ on the global minimum as described sections 15.4 and 15.5. Alternatively, we might be given a finite upper bound by the user. In either case, we can now eliminate any (feasible or infeasible) point $\mathbf{x}$ for which $f(\mathbf{x}) > \bar{f}$.

Given a box $\mathbf{x}^I$, we can evaluate $f(\mathbf{x}^I)$ and if $f(\mathbf{x}^I) > \bar{f}$, we can eliminate $\mathbf{x}^I$. However, there is a better alternative. As noted in Section 10.10, for essentially the same amount of computing needed to evaluate $f(\mathbf{x}^I)$, we can apply hull consistency to the relation $f(\mathbf{x}) \leq \bar{f}$. This also eliminates $\mathbf{x}^I$ if $f(\mathbf{x}^I) > \bar{f}$. For the same effort, hull consistency might eliminate part of $\mathbf{x}^I$. We can also apply box consistency to this inequality.

15.9 USING THE CONSTRAINTS

The main program for solving the equality constrained optimization problem, generates a sequence of subboxes of the initial box. We can apply hull and box consistencies to each constraint equations $q_i(\mathbf{x}) = 0$ $(i = 1, \cdots, r)$ to delete infeasible points from such a subbox. We can also linearize and "solve" the constraint equations in the same way as described in Section 15.4.1. In that procedure, we try to prove existence of a feasible point in a box. Here we use the procedure to try to eliminate infeasible points from a box. To do so, we fix $n - r$ of the n variables so that we have the same number r of variables as constraints. In that procedure, the variables chosen to be fixed are given point values. Now, however, we fix them by replacing them by their interval bounds. This causes a slight change in the procedure. When we replace variables by points, it can be worth while to repeat the process of expanding the constraint equations. This is because the derivatives defining the expansion are narrowed by the replacement. If we replace variables by their interval bounds, this narrowing does not occur. Therefore, re-expanding is of no value.

Before continuing, we explain why we do not use a rather obvious alternative way of choosing which variables to fix. Suppose we have expanded the equality constraints with respect to all the variables. Then we have the information needed to (roughly) determine which variables cause a given constraint to change the most (or least) over the current box. See Section 11.8. We might choose to fix those variables that cause little change.

However, the procedure in Section 15.4.1 is designed to cause a Newton step to make a larger reduction of a box irrespective of how much change is made in the range of the constraints. We consider the former aspect to be of more value than the latter

Another topic deserves mention. Suppose we fix some variables before we linearize the constraints. Then we have fewer derivatives to determine; and we thus save some work. However, we do not linearize the constraints unless there is reason to believe that the box is so small that a linearized form will yield sharper bounds on the range over the box than the original unexpanded form. See Section 14.7. In this case, the extra effort should provide greater reduction of the box used.

We now repeat some of the discussion from Sections 15.4 and 15.5 that is relevant to the current discussion of use of constraints. From Section 7.3, we can expand a constraint in the form

$$q_i(y) = q_i(\mathbf{x}) + \sum_{j=1}^{n} J_{ij}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})(y_j - x_j) \ \ (i = 1, \cdots, r) \qquad (15.9.1)$$

where

$$J_{ij}(\mathbf{x}, \mathbf{x}^{\mathbf{I}}) = \frac{\partial}{\partial x_j} q_i(X_1, \cdots, X_i, x_{i+1}, \cdots, x_r) \qquad (15.9.2)$$

Note that, in theory, the choice of which variables are real in (15.9.2) could be related to our present question of which variables to fix. However, we do not know which variables to fix until after we have obtained the expansion (15.9.1). We do the following steps. Compare the algorithm in Section 11.12.

1. Compute a real matrix $\mathbf{J}^c$, which is the approximate center of the r by n matrix $\mathbf{J}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})$.

2. Do Gaussian elimination on $\mathbf{J}^c$ using both row and column pivoting to transform it into a form in which elements in positions (i, j) are zero for $1 \leq i \leq r$ and $1 \leq j \leq r$ except for $i = j$. In the process, do the final column pivoting so that the final element in position (r, r) is largest in magnitude among the elements in positions (r, j) for $j = r, \cdots, n$.

3. Choose the variables corresponding to those now in the final columns $r + 1, ...n$ to be those to replace by their interval bounds.

Assume we have done these three steps. For simplicity, assume it is variables with indices $r + 1, \cdots, n$ that we fix. After fixing these variables, the constraint equations become

$$Q_i + \sum_{j=1}^{r} J_{ij}(\mathbf{x}, \mathbf{x}^{\mathbf{I}})(y_j - x_j) = 0 \ \ (i = 1, \cdots, r) \qquad (15.9.3)$$

where

$$Q_i = q_i(\mathbf{x}) + \sum_{j=r+1}^{n} J_{ij}(\mathbf{x}, \mathbf{x}^{\mathrm{I}})(X_j - x_j) \;\; (i = 1, \cdots, r) \,. \qquad (15.9.4)$$

Equations (15.9.3) are r equations in r unknowns. We solve them using our standard procedure for solving square systems. The first step in doing so is to precondition the system as described in Section 5.6. Note if we apply the steps of the elimination procedure in Step 2 above to an identity matrix, we obtain the desired preconditioning matrix. The second and final step is to solve the preconditioned equations by either the hull method, Gaussian elimination, or the Gauss-Seidel method. If the preconditioned coefficient matrix is regular, we find the hull as described in Section 5.8. If it is not regular, we apply a step of the Gauss-Seidel method of Section 5.7.

To obtain equations (15.9.3), we linearized the constraint equations over a box. We noted in Section 12.8 (and elsewhere) that trying to solve such a system can be unsuccessful when the box is large. Since the processes of linearizing and solving are expensive in computing effort, we want to bypass the procedure when the box is "too large".

In Section 11.11, we described a linearization test for deciding whether to bypass such a procedure or not. We use the same procedure to decide whether to generate and solve (15.9.4). This decision is made independently of any decision of whether to use other procedures that entail linear expansions. For example, we make a separate decision whether or not to try to solve the John conditions by a Newton method.

The procedure to linearize and solve the equality constraints involves a considerable amount of work. The question arises whether it is worth the effort. In Sections 14.6 and 14.7, we discussed similar questions for systems of inequality constraints. We gave a criterion to decide whether to do the linearization procedure in various cases.

After variables have been fixed in the current procedure, we have a square coefficient matrix. The linearization test could be based on whether this matrix is regular or not (as in Section 11.11). However, suppose the number of constraints is small relative to the number of variables. Then, even when the matrix is regular, there might be little progress in reducing the box.

Therefore we use a linearization test based on whether sufficient reduction (as defined using (11.7.4)) of a box by the procedure is made or not. See Section 14.7 where the same criterion is used to decide whether to solve the inequality constraints.

15.10 INFORMATION ABOUT A SOLUTION

The quality of the information we gain about the solution of an equality constrained problem depends on how successful we are at proving the existence of feasible points in the initial box. We discuss this aspect in this section.

Suppose we are unable to prove the existence of any point at which the equality constraints are satisfied. Then we do not know if the problem has a solution. Suppose we do prove the existence of a feasible point at which f is much greater than its minimum value f^*; but we are unable to prove existence of any feasible point where f is at or near its (feasible) minimum. Then we know a solution exists; but if the algorithm terminates with several boxes remaining, we do not know which one(s) contain the global solution.

Let $\overline{\mathfrak{f}}$ denote the final value of the upper bound on f^* after termination of our algorithm. If the procedures described earlier in this chapter succeed in finding such a quantity, then it is finite. Otherwise, it retains its initial value (which is ∞).

If our algorithm deletes all of the original box $\mathbf{x}^{\mathbf{I}(0)}$, then we know there is no feasible point in $\mathbf{x}^{\mathbf{I}(0)}$. Suppose it deletes all of $\mathbf{x}^{\mathbf{I}(0)}$ except for a single box $\mathbf{x}^{\mathbf{I}}$, but we have not proven the existence of a feasible point. Then we do not know if there exists a solution; but we know that if one exists, it must be in $\mathbf{x}^{\mathbf{I}}$.

Suppose that our algorithm ends up with two boxes X_1 and X_2 that satisfy whatever convergence criteria we impose; and that $f(X_1) < f(X_2) < \overline{\mathfrak{f}} < +\infty$. Even though $f(X_1) < f(X_2)$, the global solution can be in X_2 because X_1 might not contain a feasible point. The situation is obviously more uncertain if there are more than two such boxes.

Some authors suggest that we accept a point $\widetilde{\mathbf{x}}$ as feasible if $|q_i(\widetilde{\mathbf{x}})| < \varepsilon$ for all $i = 1, \cdots, r$ (for some $\varepsilon > 0$). This can cause difficulty. Suppose we accept such a point $\widetilde{\mathbf{x}}$ as feasible when it is not, and suppose that $f(\widetilde{\mathbf{x}}) < f^*$. We could then delete the solution point $\mathbf{x}^*$ because we would have $f(\mathbf{x}^*) > f(\widetilde{\mathbf{x}})$. Therefore, we must not use the relation $f(\mathbf{x}) > f(\widetilde{\mathbf{x}})$ to eliminate a point $\mathbf{x}$.

Even if we do not prove that a feasible point exists, we can assure that the constraints are "nearly" satisfied at every point in the final box(es). Denote

$$\left|q_i(\mathbf{x}^{\mathbf{I}})\right| \leq \varepsilon_q \ \ (i = 1, \cdots, r) \tag{15.10.1}$$

In our algorithms for unconstrained or for inequality constrained optimization, we require that for any final box $\mathbf{x}^{\mathbf{I}}$, the conditions $\mathrm{w}(\mathbf{x}^{\mathbf{I}}) \leq \varepsilon_X$ and $\mathrm{w}[f(\mathbf{x}^{\mathbf{I}})] \leq \varepsilon_f$ hold. For the equality constrained problem, we add the condition that (15.10.1) hold for every final box $\mathbf{x}^{\mathbf{I}}$. However, we do not assume that $\mathbf{x}^{\mathbf{I}}$ contains a feasible point simply because (15.10.1) holds.

Despite the possible uncertainties concerning the computed solution to the equality constrained case, we have much more certainty than for non-interval algorithms. The final box(es) are always small if the tolerance ε_X is small; and any global solution is guaranteed to be in one of them. In most cases, we have very good bounds on $\mathbf{x}^*$ and f^* and they are guaranteed to be correct. In other cases, the small boxes contain both local and global solutions. A final box might not contain a minimum. Use of smaller error tolerances or higher precision arithmetic might determine this to be the case.

We have pointed out that information we gain about the global minimum depends strongly on how close the final bound $\overline{\mathrm{f}}$ is to the globally minimum value f^*. We now note that there are reasons to expect that this bound is generally quite satisfactory.

One danger in trying to prove existence of a feasible point in a given box is that the methods for doing so fail if the point is not a simple zero of the equations being solved. This is not a crucial fact because our algorithm seeks a feasible point in various different boxes and fixes different subsets of the variables. We can expect that all or most of the various feasible points sought are simple solutions of the equations to be solved.

If a box is large, a Newton step or an application of hull consistency is unlikely to prove existence of a solution of a system of equations in the box. These methods are more likely to prove existence if the box is small and the feasible point is near the center of the box. However, as our algorithm proceeds, it deletes points that are not feasible. Consequently it concentrates the search on progressively smaller boxes that are more and more likely to contain feasible points near their centers and hence increase the chance of proving existence. Thus, we are likely to prove existence of a feasible point near the global minimum point.

15.11 USING THE JOHN CONDITIONS

To apply a step of a Newton method requires quite a lot of computing. When the Jacobian of the John condition function $\phi(\mathbf{t})$ (as given by (15.2.1)) contains a singular real matrix, the Newton step can be completed only by using a Gauss-Seidel step. To do so requires bounds on the Lagrange multipliers. We are unlikely to obtain useful bounds unless the box is small. Therefore computations are wasted. In Section 11.11, we derive a criterion (11.11.1) for deciding whether or not to apply a Newton step to a system of nonlinear equations. In Section 14.7, we used the same criterion to decide whether to apply a Newton method to the John conditions for an inequality constrained problem. A Newton step is bypassed if the criterion indicates that the step is likely to be unsuccessful. We use the same criterion when solving the equality constrained problem. That is, we apply the Newton method to the John conditions over a box $\mathbf{x^I}$ only if

$$\mathrm{w}(\mathbf{x^I}) \leq \frac{1}{2}(w_I^{\mathbf{J}} + w_R^{\mathbf{J}}) \tag{15.11.1}$$

where $w_I^{\mathbf{J}}$ and $w_R^{\mathbf{J}}$ are defined as in Section 14.7 (see (14.7.1)).

We also use a Newton method when trying to prove the existence of a feasible point. See Section 15.4. However, this procedure reduces the size of the box when necessary so there is no need to decide whether the box in which the procedure is applied is large or not.

As explained in Section 14.7, we avoid using linearization when the box is so large that linearization is not effective. Thus, besides avoiding the

Newton method when the box $\mathbf{x^I}$ is large, we also avoid other linearizations. As in Section 14.7, we linearize $f(\mathbf{x}) \leq \bar{\mathfrak{f}}$ only if

$$\mathrm{w}(\mathbf{x^I}) \leq \frac{1}{2}(w_I^f + w_S^f). \tag{15.11.2}$$

We linearize the constraints only if

$$\mathrm{w}(\mathbf{x^I}) \leq \frac{1}{2}(w_I^q + w_S^q). \tag{15.11.3}$$

This relation is defined in Section 14.7 for inequality constraints. Now the constraints are equalities. Similarly, when the box is large, we avoid using the method of Section 12.5.4, which involves a Taylor expansion of the relation $f(\mathbf{x}) \leq \bar{\mathfrak{f}}$ through quadratic terms.

15.12 THE ALGORITHM STEPS

In this section, we list the steps of our algorithm for computing global solution(s) to the equality constrained optimization problem (15.1.1).

Generally, we seek a solution in a single box specified by the user. However, any number of boxes, can be specified. The boxes can be disjoint or overlap. However, if they overlap, a minimum at a point that is common to more than one box is separately found as a solution in each box containing it. In this case, computing effort is wasted.

If the user does not specify an initial box or boxes, we use a default box described in Section 11.10. We assume the box(es) are placed in a list L_1 of boxes to be processed.

Suppose the user of our algorithm knows a point $\bar{\mathbf{x}}$ that is guaranteed to be feasible. If so, we use this point to compute an initial upper bound $\bar{\mathfrak{f}}$ on the global minimum f^*. If $\bar{\mathbf{x}}$ cannot be represented exactly in the computer's number system, we input a representable box $\overline{\mathbf{x^I}}$ containing $\bar{\mathbf{x}}$. We evaluate $f(\overline{\mathbf{x^I}})$ and obtain $[\underline{f}(\overline{\mathbf{x^I}}), \overline{f}(\overline{\mathbf{x^I}})]$. We set $\bar{\mathfrak{f}} = \overline{f}(\overline{\mathbf{x^I}})$, which is guaranteed to be an upper bound on the global minimum f^*. If no feasible point is known we set $\bar{\mathfrak{f}} = +\infty$ as our upper bound for f^*.

The user might know an upper bound $\bar{f}$ on f^* even though he might not know where (or if) f takes on such a value. If so, we set $\bar{f}$ equal to this known bound. If the bound is not representable in the computer's number system, we set $\bar{f}$ equal to some machine number at least as large as the known bound.

We assume the user has specified a box size tolerance ε_X, and an objective function width tolerance ε_f and tolerance ε_q on the width of the constraint functions over a final box. At termination, the conditions $\mathrm{w}(\mathbf{x}^{\mathbf{I}}) \leq \varepsilon_X$ and $\mathrm{w}[f(\mathbf{x}^{\mathbf{I}})] \leq \varepsilon_f$ and $|q_i(\mathbf{x}^{\mathbf{I}})| \leq \varepsilon_q$ $(i = 1, \cdots, r)$ hold for each final box $\mathbf{x}^{\mathbf{I}}$.

Thus, to initialize our algorithm, the user specifies ε_X, ε_f, and ε_q, and the initial box(es). The box(es) are placed in list L_1. In addition, we specify (or compute from $\bar{\mathbf{x}}$ as just described) a bound $\bar{f}$ when one is known. The system initializes the parameters defined in Section 15.11. Let $\mathbf{x}^{\mathbf{I}(0)}$ denote the box of largest width put into the list L_1 by the initialization process. The system sets $w_R^{\mathbf{J}} = w_S^f = w_S^q = w_S^Q = 0$ and $w_I^{\mathbf{J}} = w_I^f = w_I^q = w_I^Q = \mathrm{w}(\mathbf{x}^{\mathbf{I}(0)})$. The system also sets flag $\mathcal{F}$ equal to zero.

The steps of the algorithm are performed in the order given except as indicated by branching. A box $\mathbf{x}^{\mathbf{I}}$ can be changed in a given step of the algorithm. If so, we continue to call it by the same name, $\mathbf{x}^{\mathbf{I}}$. In various steps of the algorithm, we use such a box $\mathbf{x}^{\mathbf{I}}$ to compute a new box (say $\mathbf{x}^{\mathbf{I}\prime}$). When we refer to the "result" of such a computation, we mean the intersection $\mathbf{x}^{\mathbf{I}} \cap \mathbf{x}^{\mathbf{I}\prime}$.

1. For each initial box $\mathbf{x}^{\mathbf{I}}$ in the list L_1, evaluate $f(\mathbf{x}^{\mathbf{I}})$. Denote the result by $[\underline{f}(\mathbf{x}^{\mathbf{I}}), \overline{f}(\mathbf{x}^{\mathbf{I}})]$.

2. If $\bar{f} < +\infty$, delete any box $\mathbf{x}^{\mathbf{I}}$ from L_1 for which $\underline{f}(\mathbf{x}^{\mathbf{I}}) > \bar{f}$. This can be done while applying hull consistency. See Section 10.10.

3. If L_1 is empty, go to Step 47. Otherwise, find the box $\mathbf{x}^{\mathbf{I}}$ in L_1 for which $\underline{f}(\mathbf{x}^{\mathbf{I}})$ is smallest. For later reference, call this box $\mathbf{x}^{\mathbf{I}(1)}$. This box is processed next by the algorithm. Delete $\mathbf{x}^{\mathbf{I}(1)}$ from L_1.

4. If flag $\mathcal{F} = 0$, go to Step 4(a). If flag $\mathcal{F} = 1$, go to Step 4(b).

(a) If $w(x^I) \leq \varepsilon_x$ and $w[f(x^I)] \leq \varepsilon_f$, put x^I in list L_2 and go to Step 3. Otherwise, go to the next step. Note that Step 4 is repeated elsewhere in the algorithm. If it is called in Step k, then "next step" refers to step $k+1$. When it is actually called as Step 4, "next step" is Step 5, but when it is called in Step 12, for example, the next step is Step 13.

(b) If $|q_i(x^I)| \leq \varepsilon_q$ for all $i = 1, ..., r$, put x^I in list L_2 and go to Step 3.

5. Apply hull consistency (see Chapter 10) to the constraint equations $q_i(\mathbf{x}) = 0$ for $i = 1, \cdots, r$. If it is proved that there is no point in $\mathbf{x}^\mathbf{I}$ that satisfies any one of the constraints, go to Step 3.

6. Repeat Step 4.

7. If $\mathbf{x}^{\mathbf{I}(1)}$ (as defined in Step 3) has been sufficiently reduced (as defined using (11.7.4)), put $\mathbf{x}^\mathbf{I}$ in the list L_1 and go to Step 3.

8. If $\overline{f} < +\infty$, apply hull consistency to the relation $f(x) \leq \overline{f}$. If the result is empty, go to Step 3.

9. Compute an approximate center x of x^I and an approximate value of $f(x)$. If $f(x) \geq \overline{f}$, go to Step 12.

10. For later reference call the current box $\mathbf{x}^{\mathbf{I}(2)}$. Use the procedure described in Sections 15.4 through 15.6 to try to reduce the upper bound $\overline{\mathfrak{f}}$.

11. If $\overline{f}$ was not changed in Step 10, go to Step 13. Otherwise, apply hull consistency (See Chapter 10) to the relation $f(\mathbf{x}) \leq \overline{\mathfrak{f}}$. If the result is empty, go to Step 3.

12. Repeat Step 4.

13. If $\mathbf{x}^{\mathbf{I}(1)}$ (as defined in Step 3) has been sufficiently reduced (as defined using (11.7.4)), put $\mathbf{x}^\mathbf{I}$ in L_1 and go to Step 3.

14. Apply box consistency (see Section 10.2) to the constraint equations $q_i(\mathbf{x}) = 0$ for $i = 1, \cdots, r$. If it is proved that there is no point in $\mathbf{x^I}$ that satisfies any one of the constraints, go to Step 3.

15. Compute an approximate center $\mathbf{x}$ of $\mathbf{x^I}$ and an approximate value of $f(\mathbf{x})$. If $f(\mathbf{x}) \geq \overline{f}$, go to Step 18.

16. If the current box is the same box $\mathbf{x^{I(2)}}$ defined in Step 10, go to Step 18.

17. Use the procedure described in Section 15.4 through 15.6 to try to reduce the upper bound $\bar{\mathfrak{f}}$.

18. If $f(\mathbf{x^I}) \leq \bar{\mathfrak{f}}$, go to Step 20.

19. Apply box consistency to the relation $f(\mathbf{x}) \leq \bar{\mathfrak{f}}$. If the result is empty, go to Step 3.

20. Repeat Step 4.

21. If $\mathbf{x^{I(1)}}$ (as defined in Step 3) has been sufficiently reduced, put $\mathbf{x^I}$ in the list L_1 and go to Step 3.

22. If $f[\mathrm{m}(\mathbf{x^I})] < \bar{\mathfrak{f}}$, go to Step 29.

23. If $\mathrm{w}(\mathbf{x^I}) > \frac{1}{2}(w_S^f + w_I^f)$, go to Step 27. (See Section 15.11.)

24. Denote the current box by $\mathbf{x^{I(3)}}$. Apply the linear method of Section 12.5.3 to try to reduce $\mathbf{x^{I(3)}}$ using $f(\mathbf{x}) \leq \bar{\mathfrak{f}}$. Update w_S^f and w_I^f as described in Section 14.7. If the result is empty, go to Step 3.

25. Repeat Step 4.

26. If $\mathbf{x^{I(3)}}$ (as defined in Step 24) was sufficiently reduced (as defined using (11.7.4)) in the single Step 24, go to Step 30. Otherwise, go to Step 32.

27. If $\mathrm{w}(\mathbf{x}) > \frac{1}{2}(w_S^Q + w_I^Q)$, go to Step 30. (See Section 15.11.)

28. Apply the quadratic method of Section 12.5.4 to try to reduce the current box using $f(\mathbf{x}) \leq \bar{\mathfrak{f}}$. Update w_S^Q and w_I^Q as described in Section 14.7. If the result is empty, go to Step 3.

29. Repeat Step 4.

30. If $\mathbf{x}^{\mathbf{I}(1)}$ (as defined in Step 3) has been sufficiently reduced, put $\mathbf{x}^{\mathbf{I}}$ in L_1 and go to Step 3.

31. If $\mathrm{w}(\mathbf{x}^{\mathbf{I}}) > \frac{1}{2}(w_S^q + w_I^q)$, go to Step 43. (See Section 15.11.)

32. If condition (14.6.4) is not satisfied, go to Step 40. Otherwise, do the following as described in Section 15.9. Replace $n-r$ of the variables by their interval bounds and find the preconditioning matrix $\mathbf{B}$ for the system involving the remaining r variables.

33. Precondition the linearized system. If the preconditioned coefficient matrix is regular (see Theorem 5.8.1), find the hull of the linearized system by the method of Section 5.8. If the matrix is not regular, solve the system by the Gauss-Seidel method (see Section 5.7). Update w_S^q and w_I^q as described in Section 14.7. If the result is empty, go to Step 3.

34. Repeat Step 4.

35. The user might wish to bypass analytic preconditioning (see Section 11.9). If so go to Step 40. If analytic preconditioning is to be used, analytically multiply the nonlinear system of constraint equations by the matrix $\mathbf{B}$ computed in Step 32. Do so without replacing any variables by their interval bounds (so that appropriate combinations and cancellations can be made). After the analytic multiplication is complete, replace the fixed variables (as chosen in Step 32) by their interval bounds.

36. Repeat Step 4.

37. Apply hull consistency to solve the i-th nonlinear equation of the preconditioned nonlinear system for the i-th (renamed) variable for $i = 1, \cdots, r$. If the result is empty, go to Step 3. If the existence of a feasible point is proved (see Section 10.12), use the result to update $\bar{f}$ (see Section 15.4).

38. Repeat Step 4.

39. Apply box consistency to solve the i-th nonlinear equation of the preconditioned nonlinear system for the i-th (renamed) variable for $i = 1, \cdots, r$. If the result is empty, go to Step 3.

40. Repeat Step 4.

41. If $w(\mathbf{x}^I) > \frac{1}{2}(w_R^{\mathbf{J}} + w_I^{\mathbf{J}})$, go to Step 43. (See Section 15.11.1.) Note that the vector function used to determine the Jacobian in (11.8.1) is the gradient of the objective function.

42. Apply one step of the interval Newton method of Section 11.14 for solving the John conditions (15.2.1). Update $w_R^{\mathbf{J}}$ and $w_I^{\mathbf{J}}$ as described in Section 14.7, If the result is empty, go to Step 3. If the existence of a solution of the John conditions is proved as discussed in Section 15.3, then update $\bar{f}$ (as discussed in Section 15.3).

43. If the box $\mathbf{x}^{I(1)}$ (as defined in Step 3) has been sufficiently reduced, put $\mathbf{x}^I$ in L_1 and go to Step 3.

44. Any previous step that used hull consistency, a Newton step, or a Gauss-Seidel step might have generated gaps in the interval components of $\mathbf{x}^I$. Merge any such gaps when possible. Split the box as described in Section 11.8. This might involve deleting gaps. Place the subboxes (generated by splitting) in the list L_1 and go to Step 3.

45. If the list L_2 is empty, print "There is no feasible point in $\mathbf{x}^{I(0)}$" and go to Step 52.

46. If flag $\mathcal{F} = 1$, go to Step 48. Otherwise, set $\mathcal{F} = 1$.

47. For each box $\mathbf{x}^{\mathbf{I}}$ in list L_2 do the following. If $|q_i(\mathbf{x}^{\mathbf{I}})| > \varepsilon_q$ for any $i = 1, \cdots, r$, put the box in list L_1.

48. If any box was put in list L_1 in Step 46, go to Step 3.

49. If $\bar{f} < +\infty$ and there is only one box in L_2, go to Step 52.

50. For each box $\mathbf{x}^{\mathbf{I}}$ in L_2, if $f[\mathrm{m}(\mathbf{x}^{\mathbf{I}})] < \bar{f}$, try to prove existence of a feasible point using the method described in Sections 15.4 through 15.6. Use the results to update $\bar{f}$.

51. Delete any box $\mathbf{x}^{\mathbf{I}}$ from L_2 for which $\underline{f}(\mathbf{x}^{\mathbf{I}}) > \bar{f}$.

52. Denote the remaining boxes by $\mathbf{x}^{\mathbf{I}(1)}, \cdots, \mathbf{x}^{\mathbf{I}(s)}$ where s is the number of boxes remaining. Determine

$$\underline{F} = \min_{1 \le i \le s} \underline{f}(\mathbf{x}^{\mathbf{I}(i)}) \text{ and } \overline{F} = \max_{1 \le i \le s} \overline{f}(\mathbf{x}^{\mathbf{I}(i)}).$$

53. Terminate.

15.13 RESULTS FROM THE ALGORITHM

At termination, if the list L_2 is empty, then all of the initial box $\mathbf{x}^{\mathbf{I}(0)}$ has been eliminated. This provides proof that the initial box $\mathbf{x}^{\mathbf{I}(0)}$ does not contain a feasible point.

Assume that at least one box remains in the list L_2. What we have proved in this case depends on the final value of $\bar{f}$. If $\bar{f} < +\infty$, then we know that a feasible point exists in the initial box $\mathbf{x}^{\mathbf{I}(0)}$. If $\bar{f} = +\infty$, there might or might not be a feasible point in $\mathbf{x}^{\mathbf{I}(0)}$.

Consider the case $\bar{f} < +\infty$. No matter how poor the bound $\bar{f}$ on f^*, we know that a global solution exists in $\mathbf{x}^{\mathbf{I}(0)}$; and it is in one of the remaining boxes. Also, we know that

$$\underline{F} \le f^* \le \overline{F}.$$

If only one box $\mathbf{x}^{\mathbf{I}}$ remains, then it must contain the global solution. In this case,

$$\underline{f}(\mathbf{x}^{\mathbf{I}}) \leq f^* \leq \min\{\overline{f}(\mathbf{x}^{\mathbf{I}}), \bar{\mathfrak{f}}\} \text{ and } \overline{f}(\mathbf{x}^{\mathbf{I}}) - \underline{f}(\mathbf{x}^{\mathbf{I}}) \leq \varepsilon_f.$$

Therefore,

$$f(\mathbf{x}) - f^* \leq \varepsilon_f$$

for every point $\mathbf{x}$ in the box. Also,

$$x_i^* - \underline{X}_i \leq \varepsilon_X \text{ and } \overline{X}_i - x_i^* \leq \varepsilon_X \ (i = 1, \cdots, n).$$

If more than one box remains, it is possible that one contains a local solution at which f is less than our upper bound $\bar{\mathfrak{f}}$. Also, there might be more than one global solution occurring in separate boxes. We know only that

$$\underline{F} \leq f^* \leq \min\{\bar{\mathfrak{f}}, \overline{F}\}$$

and that the global minimum point(s) are in the remaining boxes.

If the final value of $\bar{\mathfrak{f}}$ is ∞ and $\mathbf{x}^{\mathbf{I}(0)}$ is not entirely deleted, then $\mathbf{x}^{\mathbf{I}(0)}$ might or might not contain a feasible point. We do not know. It is highly probable that a solution exists since, otherwise, we expect all of $\mathbf{x}^{\mathbf{I}(0)}$ to be deleted. However, we do know that if a feasible point does exist in $\mathbf{x}^{\mathbf{I}(0)}$, then,

$$\underline{F} \leq f^* \leq \overline{F}$$

and $\mathbf{x}^*$ is somewhere in the remaining box(es). All local solutions in the initial box are contained in the final solution box(es).

It is possible that every point in the initial box $\mathbf{x}^{\mathbf{I}(0)}$ is infeasible. However, our algorithm can delete all of $\mathbf{x}^{\mathbf{I}(0)}$ (and thus prove there is no solution) only if every point in $\mathbf{x}^{\mathbf{I}(0)}$ is *proved* to be feasible (i.e., is certainly infeasible). Even if every point in $\mathbf{x}^{\mathbf{I}(0)}$ is certainly infeasible, our algorithm can still fail to delete all of $\mathbf{x}^{\mathrm{I}(0)}$. This is because it operates on boxes rather than points and this generally introduces dependence. The probability of deleting the entire box in this case is greater when the box size tolerance ε_X is smaller.

Thus, we might prove there is no feasible point in $\mathbf{x}^{\mathrm{I}(0)}$, but we do not guarantee doing so when this is, in fact, the case.

Regardless of what procedure is used to delete a part of $\mathbf{x}^{\mathrm{I}(0)}$, we know that the deleted part cannot contain a solution.

A user might want a single feasible point $\widetilde{x}$ such that

$$||\widetilde{\mathbf{x}} - \mathbf{x}^*|| \leq \varepsilon_1 \tag{15.13.1}$$

and/or

$$f(\widetilde{\mathbf{x}}) - f^* \leq \varepsilon_2 \tag{15.13.2}$$

for some ε_1 and ε_2. Recall that $\mathbf{x}^*$ is a (feasible) point such that $f(\mathbf{x}^*) = f^*$ is the globally minimum value of the objective function f.

Generally, no algorithm can assure that a single point is certainly feasible because of rounding errors in evaluating the equality constraints. Therefore, we cannot provide a point $\widetilde{x}$ that is guaranteed to be feasible. In Section 15.3, we described how we could prove that some (unknown) feasible point exists in a given box. However, we can assure that for a given point $\widetilde{x}$ the equality constraints satisfy

$$|q_i(\widetilde{\mathbf{x}})| \leq \varepsilon_q \tag{15.13.3}$$

for some $\varepsilon_q > 0$ and all $i = 1, \cdots, r$. In the algorithm, we attempt to assure this by requiring that

$$|q_i(\mathbf{x}^{\mathrm{I}})| \leq \varepsilon_q \;\; (i = 1, \cdots, r) \tag{15.13.4}$$

for an entire box $\mathbf{x^I}$. See Step 4 of the algorithm in Section 15.12. See, also, Steps 12, 20, 25, 29, 34, 38, 40, and 41.

Note that rather than simply testing whether (14.9.4) is true, we should apply hull consistency to the inequalities as discussed in Section 10.10. This might reduce the box $\mathbf{x^I}$ being tested.

In Section 15.10, we noted that a condition such as (15.13.4) should not be used to delete points. However, it can be used as a condition for convergence.

In our algorithm in Section 15.12, we assure that (15.13.1), (15.13.2), and (15.13.3) are all satisfied. However, we do not impose condition (15.13.4) until near the end of the algorithm. This is because it requires more work to apply than (15.13.1) and (15.13.2).

We now distinguish the following cases.

Case 1. There is only one final box $\mathbf{x^I}$ and $\overline{\mathfrak{f}} < +\infty$.

Since $\overline{\mathfrak{f}} < +\infty$, we know that a feasible point exists in the initial box. Since x^* is never deleted by the algorithm, it must be in the single remaining box $\mathbf{x^I}$. We can choose $\widetilde{\mathbf{x}}$ to be any point in $\mathbf{x^I}$. Then the stopping criteria of the algorithm assure that

$$\begin{aligned} ||\widetilde{\mathbf{x}} - \mathbf{x}^*|| &\leq \varepsilon_X, \\ f(\widetilde{\mathbf{x}}) - f^* &\leq \varepsilon_f, \text{ and} \\ |q_i(\widetilde{\mathbf{x}})| &\leq \varepsilon_q \ (i = 1, \cdots, r). \end{aligned}$$

Case 2. There is more than one final box and $\overline{\mathfrak{f}} < +\infty$.

Since $\overline{\mathfrak{f}} < +\infty$, we know that a feasible point exists in at least one of the final boxes; but we do not know which one(s). All or part of the box in which we proved the existence of a feasible point and obtained the final value of $\overline{\mathfrak{f}}$ might have been deleted. A given point in a final box might be far from $\mathbf{x}^*$ because $\mathbf{x}^*$ is in another box. However, any such point $\widetilde{\mathbf{x}}$ satisfies $|q_i(\widetilde{\mathbf{x}})| \leq \varepsilon_q \ (i = 1, \cdots, r)$ so it must be "almost feasible". Suppose we pick $\widetilde{x}$ to be an arbitrary point in an arbitrary final box. From Step 53 of the algorithm, we have $\underline{F} \leq f(\widetilde{\mathbf{x}}) \leq \overline{F}$.

Case 3. $\bar{f} = +\infty$.

In this case, we do not know if there is any feasible point in the initial box in which the algorithm began its search. However, if there is, then **Case 1** or **Case 2** apply.

15.14 DISCUSSION OF THE ALGORITHM

A given problem will often have several or many equality constraints $q_i(\mathbf{x}) = 0$ $(i = 1, \cdots, r)$. In our algorithm, we assure that

$$|q_i(\mathbf{x^I})| \leq \varepsilon_q \ \ (i = 1, \cdots, r) \qquad (15.14.1)$$

for every final box x^I.

We do not check that this condition is satisfied until near the end of the algorithm. We first assure that

$$\mathrm{w}(\mathbf{x^I}) \leq \varepsilon_X \text{ and } \mathrm{w}\left(f(\mathbf{x^I})\right) \leq \varepsilon_f, \qquad (15.14.2)$$

for every remaining box. We do so because it generally takes much less computing to check these condition than to check (15.14.1).

Checking is done in Step 4 (which is repeated in Steps 12, 20, 25, 29, 34, 38, 40, and 41). To check (15.14.2), we do Step 4a. When we check (15.13.4) in Step 4b, (15.14.2) is already satisfied (as verified using flag $\mathcal{F}$); so it is not necessary to check it again.

In Step 42, we complete the Newton step only if the preconditioned coefficient matrix is regular. To do so requires bounds on the Lagrange multipliers. We do not have such bounds if the linear normalization (13.2.2a) is used. If (13.2.2b) is used, instead, we can complete the Newton step by using Gauss-Seidel to "solve" the preconditioned equations. This might or might not improve the bounds on the solution.

For a point $\mathbf{x}$ to be a solution of the optimization problem (15.1.1), each equality constraint must be zero at $\mathbf{x}$. Our algorithm assures the computed (perhaps non-sharp) value of $q_i(\mathbf{x^I})$ contains zero for all $i = 1, \cdots, r$ and for each remaining box $\mathbf{x^I}$. A user might want to assure that $|q_i(\mathbf{x^I})| < \varepsilon_q$ $(i = 1, \cdots, r)$ for some $\varepsilon_q > 0$. Note that when we apply hull consistency (see Step 5 of the algorithm) to the constraints, we have the data necessary to evaluate $q_i(\mathbf{x^I})$ for the box $\mathbf{x^I}$ in use at that time. These data are used to assure that $|q_i(\mathbf{x^I})| < \varepsilon_q$ with almost no extra computing.

The evaluation of $\mathrm{m}(\mathbf{x^I})$ and $f\left(\mathrm{m}(\mathbf{x^I})\right)$ used in Steps 7, 10, 12, 15, 18, and 22 can be done in real arithmetic. Only an approximate value is needed.

Consider a function F that can be either the objective function f or a constraint function q_i $(i = 1, \cdots, r)$. When we apply hull consistency to such a function, we express it as $F(\mathbf{x}) = g(x_j) - h(\mathbf{x})$ and solve for x_j for some $j = 1, \cdots, n$. If $h(\mathbf{x})$ is independent of x_j then box consistency cannot improve on the bound for x_j computed using hull consistency. When applying box consistency, we solve a given function F for a given variable x_j only if $h(\mathbf{x})$ is a function of x_j.

Our overall strategy is to use those procedures first that require the least computing. We continue to use the simplest procedures as long as they make adequate progress. Thus, solving the John conditions is done only as a last resort because it requires more computing than the other procedures. In practice, this step might not be needed at all to solve some problems.

We prefer not to use Taylor expansions until the current box is sufficiently small that this form yields sharper bounds on the range of a function than direct evaluation. See Section 15.11. This is why we included Steps 23, 27, 31, and 41.

We use procedures that require expansions only when hull consistency and box consistency do not make sufficient progress. This implies that hull consistency is relied upon when a box is large. Therefore, hull consistency should be implemented to be efficient at reducing large boxes.

We use expansions when trying to prove existence of a feasible point. See Steps 10 and 17. However, we have designed the procedure for proving existence so that it generates a small box in which expansions are likely to be useful. See Section 15.4.1. Therefore, we never bypass the effort to prove existence of a feasible point.

Choosing which box $\mathbf{x^I}$ from the list L_1 to process can be done in many ways. Experience has shown that choosing the box for which $\underline{f}(\mathbf{x^I})$ is least is better than choosing the smallest box or basing the choice on "age". See Step 3. When a box is reduced, the value of $\underline{f}(\mathbf{x^I})$ can change. Therefore, whenever a box has been sufficiently reduced, we check to see if it is still the one with smallest $\underline{f}(\mathbf{x}^{\mathbf{I}})$.

It is helpful to have as small a value of $\bar{f}$ as possible. However, the work required to try to determine bounds on a feasible point is not trivial; and it might be wasted. Therefore, we try to reduce $\bar{f}$ only if $f[\mathrm{m}(\mathbf{x}^{\mathbf{I}})] < \bar{f}$. See Steps 10 and 15. This condition indicates the existence of point(s) $\mathbf{x} \in \mathbf{x}^{\mathbf{I}}$ where $f(\mathbf{x}) < \bar{f}$ independent of their feasibility.

The nearer the upper bound $\bar{f}$ is to the minimum f^*, the more information we get about the solution. See the comments in Section 15.10. Therefore, near the end of the algorithm, we make a last effort to reduce $\bar{f}$. See Step 49. This procedure might have already been applied to some or all of the boxes in list L_2. If so, the process should not be repeated. However, if a feasible point is not found in such a box, the procedure for proving existence of a feasible point can be modified. For example, the choice of variables to fix can be altered. See Section 15.5.

Applying hull consistency to the constraint equations can eliminate many infeasible points. Therefore, we apply it before trying to find a feasible point. We could also apply box consistency before trying to find a feasible point. However, box consistency performs a function much like that of hull consistency (with more computing). We try to reduce $\bar{f}$ to a finite value before applying box consistency to the constraints. If we do so, we can apply hull consistency to the relation $f\left(\mathbf{x}^{\mathbf{I}}\right) \leq \bar{f}$ before applying the more work intensive box consistency to any relations.

Suppose we apply a Newton step to the John conditions and succeed in computing a solution (because the preconditioned coefficient matrix is regular). Then we obtain bounds on the Lagrange multipliers. The midpoints of these interval bounds are used as estimates for Lagrange multiplier values in successive attempts to solve the John conditions. These estimates are saved for use when the Newton step is applied to a subbox of the one for which the bounds were computed. Such estimates can be used even if the current box is not a subbox of one for which estimates were originally computed.

The splitting procedure called in Step 41 uses a measure of change in the gradient of the objective function to determine how to split a box. The change in the gradient is not directly significant in an equality constrained problem as it is a change in the function in Section 11.8. Nevertheless, it is a useful measure in splitting.

15.15 NONDIFFERENTIABLE FUNCTIONS

So far in this chapter, we have assumed that the objective function and the equality constraint functions are twice continuously differentiable. We now consider how the algorithm in Section 15.12 must be altered when these assumptions do not hold.

If the constraints are not continuously differentiable, then Steps 10 and 17 of the algorithm in Section 15.12 cannot be used. That is, we cannot guarantee the existence of a feasible point as discussed in Section 15.3 and 15.4.1.

An alternative might be to assume a point $\mathbf{x}$ is feasible if all the constraints are satisfied to within some error tolerance. We discussed this possibility in Section 15.10. This would not produce *guaranteed* bounds on the solution. If the constraints are continuous, it is possible to prove existence of a feasible point using hull consistency as discussed in Section 15.4.2. If hull consistency is used for this purpose, the quadratically converging implementation should be applied in Step 39.

If the objective function is not twice continuously differentiable, we cannot apply a Newton method to solve the John conditions. Therefore, Step 44 of the algorithm cannot be used.

Dropping procedures such as Newton's method that require differentiability degrades the performance of the algorithm. However, the algorithm solves the optimization problem even when continuity is lacking. Hull consistency provides the means.

Some nondifferentiable functions can be replaced by differentiable functions (plus constraints). See Chapter 18. This can resolve the difficulty and facilitate proving existence of a feasible point.

It is always better to use expansions in slopes rather than derivatives because slopes produce sharper bounds. We noted in Section 7.11 that some nondifferentiable functions have slope expansions. This can obviate concerns regarding differentiability.

Chapter 16

THE FULL MONTY

16.1 INTRODUCTION

We discuss inequality and equality constrained optimization separately in Chapters 14 and 15. Separating the cases was done for pedagogical reasons. In this chapter, we discuss the case in which there are both inequality and equality constraints. We give an algorithm for this case is Section 16.4. Before doing so, we consider some modifications to some previously discussed algorithms. In Section 16.2, we discuss solving linear systems with interval coefficients in which some of the relations of the system are inequalities and some are equalities. In Section 16.3, we discuss an extension of the procedure in Section 15.4.1 for proving the existence of a feasible point in a given box.

16.2 LINEAR SYSTEMS WITH BOTH INEQUALITIES AND EQUATIONS

In Chapter 6, we described a method for solving linear systems of inequalities with interval coefficients. We now consider how this procedure can be extended to include equalities (or equations) as well as inequalities.

Recall that to perform a step of Gaussian elimination for the case of inequalities, the multiplier must be positive so as not to change the sense of the inequality. For equalities, this is not the case. If we use an equation as the pivotal row of the system, we can eliminate a coefficient of either an equation or of an inequality using a multiplier that is either positive or negative.

As in Chapter 6, the first phase in solving the system involves finding a preconditioning matrix. This entails elimination for the noninterval case.

Suppose there are r equalities and m inequalities in n unknowns with $r + m \leq n$. Let us write the system in matrix form in which the equalities occur first and the inequalities last. We then have a system of the form

$$\mathbf{A^I z} = \mathbf{u^I} \tag{16.2.1a}$$

$$\mathbf{B^I z} \leq \mathbf{v^I} \tag{16.2.1b}$$

where $\mathbf{A^I}$ is r by n and $\mathbf{B^I}$ is m by n. Because the linear system is generally a linearized version of a nonlinear system, the vector $\mathbf{z}$ will generally be of the form $\mathbf{z} = \mathbf{y} - \mathbf{x}$ where $\mathbf{x}$ is fixed and $\mathbf{y}$ is sought as the solution vector.

Since the inequalities can be of little help in solving the equalities, we precondition and solve the system of interval equalities first. We do so using the procedure in Section 15.9. If the procedure for solving the equalities reduces the box sufficiently (as determined using (11.7.4)), it can be repeated before solving the inequalities.

We now list the steps used to solve the combined system of relations. We describe the procedure as if the equalities and inequalities are solved simultaneously instead of one after the other. This simplifies the discussion of how the equalities are used to aid in solving the inequalities.

1. If the equalities are nonlinear, linearize them to obtain the system in (16.2.1a).

2. If the inequalities are nonlinear, linearize them to obtain the system in (16.2.1b). Denote $\mathbf{R^I} = \begin{pmatrix} \mathbf{A^I} \\ \mathbf{B^I} \end{pmatrix}$

3. Compute the approximate center $\mathbf{R}^c$ of $\mathbf{R^I}$.

4. Apply Gaussian elimination with both row and column pivoting to $\mathbf{R}^c$ to produce zeros in columns $1, \cdots, r$ except for elements in positions (i, j) with $i = j$. Select pivot elements from row $1, \cdots, r$

and columns $1, \cdots, n$. To simplify the discussion, assume that no column interchanges are needed. Do the first stage of generating a preconditioning matrix by doing the same elimination steps to an identity matrix of order n.

5. Apply Gaussian elimination as described in Section 6.3 through 6.6. to produce zeros (where possible using positive multipliers) in rows and columns $r+1, \cdots, r+m$ except for elements in positions (i, j) for $i = j$. Generate secondary pivot rows as described in Section 6.5. Denote the number of secondary pivot rows generated by m'. Select pivots from rows $r+1, \cdots, r+m$ and columns $r+1, \cdots, n$. To simplify the discussion,assume that no column interchanges are required. Complete the generation of the preconditioning matrix begun in Step 4. Denote the final preconditioning matrix by $\mathbf{P}$.

6. Replace $\mathbf{z}_{r+1}, \cdots, \mathbf{z}_n$ by their interval bounds. Note that if column interchanges were made in Step 4 or Step 5, different variables are replaced by their interval bounds.

7. Precondition the interval system by multiplying by $\mathbf{P}$ to obtain $\mathbf{PR^I}z = \mathbf{Pw^I}$. Apply an interval version of Gaussian elimination to this system without either row or column pivoting to zero the elements in positions (i, j) with $i = m+1, \cdots, m+m'$ and $j = r+1, \cdots, r+m$.

8. Solve the m preconditioned equalities as follows

   (a) Replace variables $z_{m+1}, \cdots, z_{m+r}$ by their interval bounds.

   (b) Solve the resulting m equalities in m variables by the hull method of Section 5.8. If the hull method fails, use the Gauss-Seidel method of Section 5.7.

9. Solve the $m+m'$ interval inequalities (which involve $n-m$ variables) by the method of Section 6.8.

In practice, we should solve the equalities before even linearizing the inequalities in Step 2. This produces narrower interval coefficients in the linearized inequalities. This approach requires that we save the linear operation used in Step 4 and 7 for use in eliminating variables from the linear inequalities. We omit the somewhat messy details.

When solving a system of nonlinear equalities and inequalities, we use the procedure described in this section only if the width of the current box satisfies a "linearization condition" as described in Section 14.7 and discussed in Section 15.11. We use parameters w_I^C and w_R^C defined and used similarly to $w_I^{\mathbf{g}}$ and $w_R^{\mathbf{g}}$ in Section 14.7.

16.3 EXISTENCE OF A FEASIBLE POINT

When solving a global optimization problem, an upper bound on the global minimum can be used to eliminate local minima of value larger than the upper bound. In Section 15.4.1, we discuss the problem of proving the existence of a feasible point for the equality constrained problem. It is necessary to prove existence to obtain an upper bound on the global minimum. We seek to prove existence by applying a Newton method to the constraint equalities over a box $\mathbf{x^I}$. If a Newton step applied to $\mathbf{x^I}$ produces a new box $\mathbf{x^{I\prime}}$ that is contained in $\mathbf{x^I}$, this proves that a feasible point exists in $\mathbf{x^{I\prime}}$. See Theorem 11.15.7. Therefore, $\overline{f}(\mathbf{x^{I\prime}})$ is an upper bound on the global minimum of f.

When inequality constraints also occur, we must prove that the solution point also satisfies the inequalities. When there are equality constraints, we generally do not know a single point that satisfies them. We only know that the point proven to exist by the Newton method lies in a box $\mathbf{x^{I\prime}}$. We must verify that the inequality constraints are satisfied over the entire box $\mathbf{x^{I\prime}}$. When there are no equality constraints, this verification can be done at a single point. See Section 14.3.

Sometimes it is possible to determine some or all of the components of a point that satisfies the equality constraints. This is done by fixing one or more of the variables and solving for others. We gave an example in Section 15.5. We now consider three cases. In the first case all of the variables can be determined by fixing a subset of them. In the second case, some, but not all, of the variables can be determined in this way. In the third case, no variables can be determined in this way.

Assume that we are solving the optimization problem in a given box $\mathbf{x}^{\mathbf{I}(0)}$ and denote the current subbox of interest by $\mathbf{x}^{\mathbf{I}}$.

16.3.1 Case 1

Assume that we fix a number k of the variables and are able to determine all of the others from the equality constraints. To do so, we choose these k variables to have their values at the center of $\mathbf{x}^{\mathbf{I}}$. This serves to determine a partially prescribed and partially computed point $\widetilde{\mathbf{x}}$. However, it might be necessary to make rounding errors in computing the unprescribed variables so, in practice, the "point" might have interval components. To emphasize this fact, we denote it by $\widetilde{\mathbf{x}^{\mathbf{I}}}$. If $\widetilde{\mathbf{x}^{\mathbf{I}}}$ satisfies the inequality constraints, then any point $\widetilde{\mathbf{x}} \in \widetilde{\mathbf{x}^{\mathbf{I}}}$ is a feasible point. Therefore, $\overline{f}(\widetilde{\mathbf{x}^{\mathbf{I}}})$ is an upper bound on the global minimum f^*.

Note that $\widetilde{\mathbf{x}^{\mathbf{I}}}$ might not be in the current box $\mathbf{x}^{\mathbf{I}}$. We consider this to be irrelevant. We are searching for any point that gives a good upper bound on the global minimum. The current box merely serves to begin the determination of $\widetilde{\mathbf{x}^{\mathbf{I}}}$. Note also that $\widetilde{\mathbf{x}^{\mathbf{I}}}$ might not be in the initial box $\widetilde{\mathbf{x}^{\mathbf{I}}}^{(0)}$. If not, we temporarily abandon our effort to prove the existence of a feasible point. When the main program chooses a new box, we try again.

16.3.2 Case 2

We now consider the case in which we fix a number k of the variables and determine some, but not all, of the others so that a number $s < n$ of the variables is either prescribed or computed. For simplicity, assume they are the first s components. Thus, we know the components $X_1, \cdots, X_s$ of some "point". The computed components might be intervals (to bound rounding errors) so we denote (all of) them as intervals with capital letters. If $X_i \nsubseteq X_i^{(0)}$ for some $i = 1, \cdots, s$, we abandon our effort to bound a feasible point when starting from the current box. It might happen that no point with components $X_1, \cdots, X_s$ intersects the current box. We consider this to be irrelevant.

Note that after fixing certain variables and solving for others, a subset of the equality constraints is satisfied. Our effort to prove existence of a feasible point now involves fewer variables and fewer equality constraints than originally occurred. We substitute the values of the fixed variables into the remaining constraints. We then use these constraints to try to prove existence of a point that satisfies them. We use the procedure in Sections 15.4 through 15.6. Some of the variables are fixed as just described. The procedure in Section 15.5 fixes others.

The procedure in Section 15.4.1 generates a box in which to try to prove existence. We choose this box so that its unfixed components have the same relative widths as the current box $\mathbf{x^I}$. This is reasonable because the widths of the components of the current box more likely to reflect the relative scaling of variables in the problem than an arbitrarily chosen box.

Assume the Newton step discussed in Section 15.4 proves existence of a point satisfying the equality constraints in a box $\mathbf{x^{I}}'$. If this box satisfies the inequality constraints, then $\overline{f}(\mathbf{x^{I}}')$ is an upper bound on the global minimum. If the box does not satisfy the inequality constraints, we abandon our effort to bound a feasible point when starting from the current box.

16.3.3 Case 3

In our final case, no variables are fixed and no equality constraints are satisfied before we try to prove existence of a feasible point. Now we reverse the order in which the equality and inequality constraints are used. It is generally easier to check whether the inequality constraints are satisfied than to try to prove existence of a point satisfying the equality constraints. Therefore, we first find a point satisfying the inequality constraints; and try to prove existence of a point satisfying the equality constraints in a box centered at this point. If there are few equality constraints and many inequality constraints, it might be more economical to reverse the order as in the two previous cases. We shall not do so.

When the main program generates a new box, we do a line search proceeding from the center of the box in the direction of the negative gradient of the objective function as described in Section 14.4. The purpose of the line search is to find a point in the box satisfying the inequality constraints where the objective function is smaller than at the center of the box.

We generate a new box with this point as center but otherwise as described in Section 15.4.1. We require that the box be in the initial box $\mathbf{x}^{\mathbf{I}(0)}$. Therefore, the center of the box (which is the point found by the line search) must be an interior point of $\mathbf{x}^{\mathbf{I}(0)}$. When the algorithm in Section 14.4 succeeds in finding a point satisfying the inequality constraints, the algorithm denotes it by $\mathbf{y}$. It also obtains a point, which it denotes by $\mathbf{y}'$, which might or might not satisfy the inequality constraints. If $\mathbf{y}$ is on the boundary of $\mathbf{x}^{\mathbf{I}(0)}$, then $\mathbf{y}'$ is not. In this case, $\mathbf{y}'$ can be chosen as the center of the box to be generated if $\mathbf{y}'$ satisfies the inequality constraints.

We ignore the fact that the objective function is smaller at $\mathbf{y}$ than at $\mathbf{y}'$. If $\mathbf{y}'$ does not satisfy the inequality constraints, the algorithm in Section 14.4, can be continued to obtain an interior point of $\mathbf{x}^{\mathbf{I}(0)}$ satisfying the inequality constraints. If no such point in the interior of $\mathbf{x}^{\mathbf{I}(0)}$ is found after a few steps, the procedure to prove existence of a feasible point when starting from $\mathbf{x}^{\mathbf{I}}$ can be abandoned.

Let $\mathbf{y}$ denote the point that the line search finds and that satisfies the inequality constraints. We assume the line search has been modified as just described so that $\mathbf{y}$ is an interior point of $\mathbf{x}^{\mathbf{I}(0)}$. We now do the following steps. In these steps, the box $\mathbf{z}^{\mathbf{I}}$ changes from step to step.

1. Generate a box $\mathbf{z}^{\mathbf{I}}$ with center $\mathbf{y}$ having the same relative widths of components as the current box $\mathbf{x}^{\mathbf{I}}$. Choose $\mathbf{z}^{\mathbf{I}}$ to have width $\frac{1}{2}(w_R^q + w_I^q)$ where w_R^q and w_I^q are defined in Section 15.4.1.

2. If $\mathbf{z}^{\mathbf{I}}$ extends beyond the boundaries of the initial box $\mathbf{x}^{\mathbf{I}(0)}$, shrink it as described in Section 15.4.1 until it is contained in $\mathbf{x}^{\mathbf{I}(0)}$.

3. If $\mathbf{z}^{\mathbf{I}}$ satisfies the inequality constraints, go to Step 8.

4. Set $n = 0$.

5. Replace n by $n + 1$. Shrink $\mathbf{z}^{\mathbf{I}}$ by a factor of eight.

6. If $\mathbf{z}^{\mathbf{I}}$ satisfies the inequality constraints, go to Step 8.

7. If $n < 8$, go to Step 5. Otherwise, abandon the effort to try to find a feasible point.

8. Use the procedure in Sections 15.4 through 15.6 to try to prove that there exists a point in $\mathbf{z^I}$ that satisfies the equality constraints.

Note that, for a given problem, it might not be possible to find a non-degenerate box that satisfies the inequality constraints. For example, there might be only a single point satisfying them. In Section 14.9, we discuss how to treat this situation when there are no equality constraints. We treat a subset of the inequality constraints as equalities. In the current case, we simply add the equality constraints to the set of inequality constraints that are treated as equalities. We then try to prove existence of the combined set of equalities using the procedure discussed in Sections 15.4 through 15.6.

16.4 THE ALGORITHM STEPS

We now give the steps of the algorithm for the case in which both inequality and equality constraints occur.

To initialize, we require that the user specify a box size tolerance ε_X, a function width tolerance ε_f, an inequality function width tolerance ε_p, an equality function width tolerance ε_q, and the initial box(es). Any tolerance not specified is set to $+\infty$ by the program. However, a finite value must be specified for at least one of them. The initial box(es) are put in list L_1. The algorithm provides the parameters needed to perform any linearization test. (See Section 14.7.) It sets w_S^f, w_S^S, w_S^C, w_R^q and w_R^J to zero and sets w_I^f, w_I^S, w_I^C, w_I^q and w_I^J equal to $w(\mathbf{x}^{\mathbf{I}(0)})$. It also sets the flag $\mathcal{F} = 0$.

The steps of the algorithm are to be performed in the order given except as indicated by branching. The current box is denoted by $\mathbf{x^I}$ throughout the algorithm even though it changes from step to step.

1. For each initial box $\mathbf{x^I}$ in the list L_1, evaluate $f(\mathbf{x^I})$. Denote the result by $[\underline{f}(\mathbf{x^I}), \overline{f}(\mathbf{x^I})]$.

2. If $\overline{f} < +\infty$, delete any box $\mathbf{x^I}$ from L_1 for which $\underline{f}(\mathbf{x^I}) > \overline{f}$.

3. If L_1 is empty, go to Step 47. Otherwise, find the box in L_1 for which $\underline{f}(\mathbf{x^I})$ is smallest. For later reference, call this box $\mathbf{x}^{\mathbf{I}(1)}$. This box is processed next by the algorithm. Delete $\mathbf{x}^{\mathbf{I}(1)}$ from L_1.

4. If flag $\mathcal{F} = 0$, go to Step 4a. If flag $\mathcal{F} = 1$, go to Step 4b.

   (a) If $w(\mathbf{x^I}) \leq \varepsilon_X$ and $w[f(\mathbf{x^I})] \leq \varepsilon_f$, put $\mathbf{x^I}$ in list L_2 and go to Step 3. Otherwise, go to the next step. Note that Step 4 is repeated elsewhere in the algorithm. If it is called in Step k, then "next step" refers to Step $k + 1$. For example, when it is actually called as Step 4, the next step is Step 5; but when it is called in Step 14, for example, the next step is Step 15.

   (b) If $|p_i(\mathbf{x^I})| \leq \varepsilon_p$ for all $i = 1, \cdots, m$ and $|q_i(\mathbf{x^I})| \leq \varepsilon_q$ for all $i = 1, \cdots, r$, put $\mathbf{x^I}$ in list L_2 and go to Step 3.

5. Apply hull consistency to the constraint equalities $q_i(\mathbf{x}) = 0$ for $i = 1, \cdots, r$. If it is proved that no point in $\mathbf{x^I}$ satisfies any one of the constraints, go to Step 3.

6. Apply hull consistency to the constraint inequalities $p_i(\mathbf{x}) \leq 0$ for $i = 1, \cdots, m$. If it is proved that no point in $\mathbf{x^I}$ satisfies any one of the constraints, go to Step 3.

7. Repeat Step 4.

8. If $\mathbf{x}^{\mathbf{I}(1)}$ (as defined in Step 3) has been sufficiently reduced (as defined using (11.7.4)), put $\mathbf{x^I}$ in list L_1 and go to Step 3.

9. Compute an approximation $\mathbf{x}$ for the center of $\mathbf{x^I}$ and an approximation for $f(\mathbf{x})$.

10. If $f(\mathbf{x}) \geq \overline{f}$, go to Step 13.

11. For later reference, call the current box $\mathbf{x}^{\mathbf{I}(2)}$. Use the procedure described in Section 16.3 to try to reduce the upper bound $\overline{f}$. Note: The box $\mathbf{x^I}$ could be the same one used the last time this step was used. If so, do not repeat the procedure.

12. Compute an approximate center $\mathbf{x}$ of $\mathbf{x}^{\mathbf{I}}$ and an approximate value of $f(\mathbf{x})$. If $f(\mathbf{x}) \leq \overline{f}$, go to Step 14.

13. Apply hull consistency (see Chapter 10) to the relation $f(\mathbf{x}) \leq \overline{f}$. If the result is empty, go to Step 3.

14. Repeat Step 4.

15. If $\mathbf{x}^{\mathbf{I}(1)}$ (as defined in Step 3) has been sufficiently reduced (as defined using (11.7.4)), put $\mathbf{x}^{\mathbf{I}}$ in list L_1 and go to Step 3.

16. Apply box consistency to the equality constraints $q_i(\mathbf{x}) = 0$ for $i = 1, \cdots, r$ and then to the inequality constraint $p_i(\mathbf{x}) \leq 0$ for $i = 1, \cdots, m$. If it is proved that no point in $\mathbf{x}^{\mathbf{I}}$ satisfies any one of the constraints, go to Step 3.

17. Compute an approximate center $\mathbf{x}$ of $\mathbf{x}^{\mathbf{I}}$ and an approximate value of $f(\mathbf{x})$. If $f(\mathbf{x}) \geq \overline{f}$, go to Step 20.

18. If the current box is the same box $\mathbf{x}^{\mathbf{I}(2)}$ defined in Step 11, go to Step 20.

19. Use the procedure described in Section 16.3 to try to reduce the upper bound $\overline{f}$.

20. If $f[m(\mathbf{x}^{\mathbf{I}})] \leq \overline{f}$, go to Step 22.

21. Apply box consistency to the relation $f(\mathbf{x}) \leq \overline{f}$. If the result is empty, go to Step 3.

22. Repeat Step 4.

23. If $\mathbf{x}^{\mathbf{I}(1)}$ (as defined in Step 3) has been sufficiently reduced, put $\mathbf{x}^{\mathbf{I}}$ in the list L_1 and go to Step 3.

24. Compute an approximate center $\mathbf{x}$ of $\mathbf{x}^{\mathbf{I}}$ and an approximate value of $f(\mathbf{x})$. If $f(\mathbf{x}) < \overline{f}$, go to Step 32.

25. If $w(\mathbf{x}^{\mathbf{I}}) > \frac{1}{2}(w_S^f + w_I^f)$, go to Step 29. (See Section 15.11).

26. For later reference, denote the current box by $\mathbf{x^{I}}^{(3)}$. Apply the linear method of Section 12.5.3 to try to reduce $\mathbf{x^{I}}^{(3)}$ using $f(\mathbf{x}) \leq \overline{f}$. Update w_S^f and w_I^f. If the result is empty, go to Step 3.

27. Repeat Step 4.

28. If $\mathbf{x^{I}}^{(3)}$ (defined in Step 26) was sufficiently reduced (as defined using 11.7.4), in the single Step 26, go to Step 32. Otherwise, go to Step 33.

29. If $w(\mathbf{x^I}) > \frac{1}{2}(w_S^S + w_I^S)$, go to Step 33. See Section 14.7.

30. Apply the quadratic method of Section 12.5.4 to try to reduce the current box using $f(\mathbf{x}) \leq \overline{f}$. Update w_S^S and w_I^S. If the result is empty, go to Step 3.

31. Repeat Step 4.

32. If $\mathbf{x^{I}}^{(1)}$ (as defined in Step 3) has been sufficiently reduced (as defined using (11.7.4)), put $\mathbf{x^I}$ in L_1 and go to Step 3.

33. If $w(\mathbf{x^I}) > \frac{1}{2}(w_S^C + w_I^C)$, go to Step 42. (See Section 15.11.)

34. Linearize and solve the system of equality constraints as described in Section 16.2.

35. Linearize those inequality constraints that satisfy (14.6.2). If (14.6.2) is satisfied, include the inequality $f(\mathbf{x}) \leq \overline{f}$. Solve them by the method of Section 16.2, which also uses the equality constraints. Update w_S^C and w_I^C as described in Section 16.2.

36. Repeat Step 4.

37. The user might wish to bypass analytic preconditioning (see Section 11.9). If so, go to Step 42. If analytic preconditioning is to be used, analytically multiply the nonlinear system of equality and inequality constraints by the preconditioning matrix described in Section 16.2 and computed in the combined Steps 34 and 35. Do so without

replacing any variables by their interval bounds so that appropriate combinations and cancellations can be made. After the analytic preconditioning is complete, replace variables by their interval bounds as in Steps 34 and 35.

38. Apply hull consistency to the relations derived in Step 37. Solve the equalities only for the variables that were solved for in Step 34. Solve the inequalities only for the variables that were solved for in Step 35. If the result is empty, go to Step 3.

39. Repeat Step 4.

40. Apply box consistency to the relations derived in Step 37. Solve the equalities only for the variables that were solve for in Step 34. Solve the inequalities only for the variables that were solved for in Step 35. If the result is empty, go to Step 3.

41. Repeat Step 4.

42. If $w(\mathbf{x^I}) > \frac{1}{2}(w_R^{\mathbf{J}} + w_I^{\mathbf{J}})$, go to Step 45, (where $w_R^{\mathbf{J}}$ and $w_I^{\mathbf{J}}$ are defined as in Section 14.7 (see 14.7.1)).

43. Apply one step of the interval Newton method of Section 11.14 for solving the John conditions (13.5.1). Update w_R^J and w_I^J. If the result is empty, go to Step 3. If the existence of a solution of the John conditions is proved as discussed in Section 15.3, then update $\overline{f}$ (as discussed in Section 15.3).

44. If $\mathbf{x^I}^{(1)}$ (as defined in Step 3) has been sufficiently reduced, put $\mathbf{x^I}$ in L_1 and go to Step 3.

45. Any previous step that used hull consistency, a Newton step, or a Gauss-Seidel step might have generated gaps in the interval components of $\mathbf{x^I}$. Merge any such gaps when possible. Split the box as described in Section 11.8. This might involve deleting gaps. Place the subboxes (generated by splitting) in the list L_1 and go to Step 3.

46. If the list L_2 is empty, print “There is no feasible point in $\mathbf{x}^{\mathbf{I}(0)}$" and go to Step 54.

47. If flag $\mathcal{F} = 1$, go to Step 50. Otherwise, set $\mathcal{F} = 1$.

48. For each box $\mathbf{x}^{\mathbf{I}}$ in list L_2 do the following. If $\overline{p}(\mathbf{x}^{\mathbf{I}}) > \varepsilon_p$ for any $i = 1, \cdots, m$ or if $|q_i(\mathbf{x}^{\mathbf{I}})| > \varepsilon_q$ for any $i = 1, \cdots, r$, put the box in list L_1.

49. If any box was put in list L_1 in Step 48, go to Step 3.

50. If $\overline{f} < +\infty$ and there is only one box in L_2, go to Step 54.

51. For each box $\mathbf{x}^{\mathbf{I}}$ in L_2, if $f[m(\mathbf{x}^{\mathbf{I}})] < +\infty$, try to prove existence of a feasible point using the method describe in Section 16.3. Use the results to update $\overline{f}$.

52. Delete any box $\mathbf{x}^{\mathbf{I}}$ from L_2 for which $\underline{f}(\mathbf{x}^{\mathbf{I}}) > \overline{f}$.

53. Denote the boxes in L_2 by $\mathbf{x}^{\mathbf{I}(1)}, \cdots, \mathbf{x}^{\mathbf{I}(s)}$. where s is the number of boxes in L_2. Determine

$$\underline{F} = \min_{1 \le i \le s} \underline{f}(\mathbf{x}^{\mathbf{I}(i)}) \text{ and } \overline{F} = \max_{1 \le i \le s} \overline{f}(\mathbf{x}^{\mathbf{I}(i)}).$$

54. Terminate.

What we learn about the solution to a problem depends on whether or not we obtain an upper bound on the global minimum using the procedure in Section 16.3. It also depends on whether the output of the algorithm is one box or more than one box. What we learn for a problem with both equality and inequality constraints is essentially the same as for a problem with equality constraints only. Thus, the comments following the algorithm in Section 15.12 are appropriate for the algorithm of this section. There is no need repeat them.

Chapter 17

PERTURBED PROBLEMS AND SENSITIVITY ANALYSIS

17.1 INTRODUCTION

In practice, optimization problems often involve parameters that are uncertain. In other problems, parameters can be known exactly, but it is of interest to know how much the solution to the problem changes as the parameters change. In this chapter, we discuss methods for bounding the change in the solution when parameters vary over intervals.

Perturbed problems arise, for example, when the parameters are measured quantities and the measurements are subject to error. For such problems, we assume bounds on the errors are known. That is, we assume intervals are known that contain the values of the parameters.

Small errors in data also occur because of roundoff. For example, the number π cannot be represented exactly. Instead we represent it by an interval containing its correct value and whose endpoints are machine numbers. Such small errors do not require the methods of this chapter.

In this chapter, we consider perturbed problems in which parameters are specified as intervals. The intervals can be bounds on uncertain parameters or the range over which a sensitivity analysis is desired. We discuss how the interval optimization algorithms discussed earlier can be used to compute bounds on the set of solutions that result when the parameters vary over the given intervals.

For other discussions of the use of interval analysis for perturbed problems and sensitivity analysis in optimization, see Dinkel and Tretter (1987), Dinkel, Tretter and Wong (1988), Hansen (1984), and Ratschek and Rokne (1988).

In noninterval algorithms, a sensitivity analysis might be done by linearizing about a nominal solution. The resulting approximation to the problem can be poor if large changes in the parameters are allowed. It can also be poor if small changes in a parameter produce large changes in the solution. Moreover, the cost of linearization is added in the form of both extra analysis and extra computing.

Our interval approach is different. To perform a sensitivity analysis, we replace parameters by intervals over which the parameters are chosen to vary. We then solve the problem in the same way as discussed in earlier chapters without changing the algorithms.

In Section 17.5, we show how to compute bounds on the set of solution values and solution points by solving modified problems. Some extra analysis of a simple kind is required, but no additional algorithm is needed to compute the bounds.

Consider an unperturbed problem in which the objective function and/or constraints depend on a vector $\mathbf{c}$ of parameters that are independent of the variable $\mathbf{x}$. To emphasize the dependence on $\mathbf{c}$, we write the problem as

$$\text{minimize (globally) } f(\mathbf{x}, \mathbf{c}) \tag{17.1.1}$$

$$\text{subject to} \begin{cases} p_i(\mathbf{x}, \mathbf{c}) \leq 0 \ (i = 1, ..., m) \\ q_i(\mathbf{x}, \mathbf{c}) = 0 \ (i = 1, ..., r). \end{cases}$$

To exhibit dependence of the solution on $\mathbf{c}$, we write the solution value as $f^*(\mathbf{c})$ and the solution point(s) as $\mathbf{x}^*(\mathbf{c})$. Note that $f(\mathbf{x}^*(\mathbf{c}), \mathbf{c}) = f^*(\mathbf{c})$.

In the perturbed case, we allow $\mathbf{c}$ to vary over an interval vector $\mathbf{c}^\mathbf{I}$. As $\mathbf{c}$ varies over $\mathbf{c}^\mathbf{I}$, we obtain a set of solution values

$$f^*(\mathbf{c}^\mathbf{I}) = \{f^*(\mathbf{c}) : \mathbf{c} \in \mathbf{c}^\mathbf{I}\} \tag{17.1.2}$$

and a set of solution points

$$\mathbf{x}^*(\mathbf{c}^\mathbf{I}) = \{\mathbf{x}^*(\mathbf{c}) : \mathbf{c} \in \mathbf{c}^\mathbf{I}\}. \tag{17.1.3}$$

We wish to bound $f^*(\mathbf{c}^\mathbf{I})$ and $\mathbf{x}^*(\mathbf{c}^\mathbf{I})$. The width of the interval $f^*(\mathbf{c}^\mathbf{I})$ is a measure of the sensitivity of the problem to variation of $\mathbf{c}$ over $\mathbf{c}^\mathbf{I}$.

The solution set $\mathbf{x}^*(\mathbf{c}^\mathbf{I})$ contains the global minimum for every specific (real) choice of the parameters $\mathbf{c}$ satisfying the interval bounds $\mathbf{c} \in \mathbf{c}^\mathbf{I}$. The size of the set is a measure of the sensitivity of the solution point to variation of the parameters within their interval bounds. Therefore, either or both of $f^*(\mathbf{c}^\mathbf{I})$ and $\mathbf{x}^*(\mathbf{c}^\mathbf{I})$ are of interest in sensitivity analysis.

If the intervals bounding the parameters are narrow, and if the problem is not especially sensitive to perturbation, the solution set $\mathbf{x}^*(\mathbf{c}^\mathbf{I})$ is small. Therefore, it can be covered by a small number of boxes whose width is less than the tolerance ε_X used in a termination process.

As we point out in the next section, the optimization algorithms given in previous chapters all solve perturbed problems. They produce a box or set of boxes containing the solution whether it is a single point or an extended set.

If the perturbations are large, it can require many boxes to cover the solution set, especially if the box size tolerance is small. In this case, the number of boxes (and the computing time) can be excessive.

In low dimensional problems, we might want to cover the solution set by small "pixel" boxes to obtain a kind of map of the solution region. In fact, we might want to subdivide the intervals containing the parameter to sharpen the "map" of the region. The "pixel" size of the covering boxes can be determined by specifying the box size tolerance ε_X appropriately. Dependence resulting from multiple occurrences of a parameter can cause loss of sharpness in defining the boundary of the solution set. In this case, it can be desirable to split the parameter interval into small subintervals and repeatedly solve the optimization problem using each subinterval of the parameter.

However, a primary purpose of this chapter is to show how we can bound the solution set without covering it by a large number of small boxes. Instead, we compute a single box bounding the solution set $\mathbf{x}^*(\mathbf{c}^\mathbf{I})$. In addition, we bound the interval $f^*(\mathbf{c^I})$. See Section 17.5. The procedure is the same as the one described by Hansen (1991).

Note that the solution set $\mathbf{x}^*(\mathbf{c^I})$ of a perturbed problem is generally complicated in shape. In particular, it is not a box, in general. For example, see the solution set of the linear algebraic equation, $\mathbf{A^I x} = \mathbf{b^I}$ in Figure 5.3.1. In this equation, the coefficient matrix $\mathbf{A^I}$ and the right hand side vector $\mathbf{b^I}$ are interval quantities.

The solution to this system is the same as the solution to the problem of minimizing the function

$$f(\mathbf{x}) = (\mathbf{A^I x} - \mathbf{b^I})^T(\mathbf{A^I x} - \mathbf{b^I}).$$

That is, the solution to this perturbed minimization problem is a complicated set.

The algorithms in previous chapters produce a set of boxes covering such a solution set. However, either the solution set itself or the set of boxes covering it might be inconvenient to work with, in general. If so, a simpler indication of the sensitivity is the size of the smallest box containing the solution set.

Thus, we introduce convenience and reduce the effort by computing the smallest box containing the solution set rather than covering the solution set with small boxes. In Section 17.5, we describe a procedure for bounding such a box.

17.2 THE BASIC ALGORITHMS

The global optimization algorithms described in Sections 12.14, 14.8, and 15.12 do all the essential computations in interval arithmetic. It is irrelevant (to the algorithms) whether parameters in the problem are intervals or real numbers that are treated as degenerate intervals. In this sense, all problems are perturbed problems when solved by an interval algorithm. This argument was made in detail by Hansen (1984).

For a perturbed problem, we replace each parameter to be perturbed by an interval that bounds it. The optimization algorithm solves this problem without being modified an any way.

17.3 TOLERANCES

For termination, the optimization algorithms require that the width of each output box be less than a tolerance ε_X. Also, the width of the range of the objective function over each box must not be greater than a function width tolerance ε_f. The latter tolerance assures that the final bound on the minimum value of the objective function is in error by no more than ε_f.

A question of significance is how to choose these tolerances when solving perturbed problems. If the intervals bounding the parameters are narrow and the problem is not too sensitive to parameter changes, the solution set is small and the choice of tolerances can essentially be made as if the problem is unperturbed. If the tolerances are small and the solution set is large, the solution set can be covered by a large number of small boxes. Obtaining such a result can be expensive in computer time and difficult to interpret in more than a few dimensions.

If the tolerances are too large, the solution set $\mathbf{x}^*(\mathbf{c}^\mathbf{I})$ is poorly defined and the bounds on the interval $f^*(\mathbf{c}^\mathbf{I})$ of solution values are far from sharp.

We do not have a suitable procedure for choosing the tolerances. In practice, we often use human intervention. When doing so, we choose the tolerance ε_f on the objective function be large and let the box size tolerance ε_X drive the stopping criteria. We first solve the problem with ε_X relatively large. If desired, we then continue the solution process with a smaller value of ε_X, depending on how important it is to accurately map the solution set.

When continuing with a smaller tolerance, it is not necessary to start over with the original box. We simply place the output boxes from the previous run in the list of boxes to be processed by the algorithm. We then proceed as if we are starting over. The algorithm does not repeat previous work.

A suitable stopping procedure can undoubtedly be incorporated in our algorithm to handle perturbed problems both efficiently and automatically. We have spent little effort trying to determine how to do so.

17.4 DISJOINT SOLUTION SETS

As parameter values change, the location of the global minimum can change discontinuously. This can happen when a local minimum becomes global while the global minimum becomes local. Examples are given in Sections 17.7 and 17.8.

A virtue of the interval algorithms given in the previous chapters is that such a case does not affect their behavior. Any point that is a global solution for any values of the parameters within their bounding intervals is contained in the solution set. The solution set can be composed of disjoint subsets.

17.5 SHARP BOUNDS FOR PERTURBED OPTIMIZATION PROBLEMS

Two difficulties arise in our optimization algorithm when the problem is perturbed. In this section, we describe these difficulties and then show how they can be avoided by modifying our definition of a solution.

The first difficulty is that when parameters in the objective function enter as intervals, dependence (see Section 2.4) can cause loss of sharpness in a computed value of the objective function. This, in turn, can cause the computed solution set to be larger than the true solution set.

The second difficulty has already been mentioned. If the box size tolerance is small, a large number of boxes is generally required to cover the solution set of a perturbed problem.

Rather than computing many small boxes to cover the solution set, we can simply compute the smallest box containing it. In this section, we consider how to compute sharp bounds on such a box. We also consider how to sharply bound the set of solution values $f^*(\mathbf{c^I})$ defined by (17.1.2).

By modifying the definition of a solution, we are able to replace the perturbed problem by a set of unperturbed problems. In appropriate cases (described below), this enables us to solve the modified problem as sharply as arithmetic precision permits.

We separate the solution procedure into two phases. In the first phase, we solve the original problem (17.1.1) with the real vector $\mathbf{c}$ replaced by the interval vector $\mathbf{c}^{\mathbf{I}}$ bounding it.

When doing so, we use relatively large values of the stopping tolerances ε_X and ε_f. As a result, the output is a small number of relatively large boxes covering the solution set $\mathbf{x}^*(\mathbf{c}^{\mathbf{I}})$. That is, we do not do the excessive amount of computing required to cover the solution set by a large number of small boxes. The price we pay is that the solution set $\mathbf{x}^*(\mathbf{c}^{\mathbf{I}})$ is poorly defined because the output boxes contain a relatively large set of points that are not in $\mathbf{x}^*(\mathbf{c}^{\mathbf{I}})$. In addition, the interval $f^*(\mathbf{c}^{\mathbf{I}})$ is only loosely bounded.

Let $\mathbf{x}^{\mathbf{I}'}$ denote the smallest box containing the set of boxes computed as output of the first phase. Generally, $\mathbf{x}^{\mathbf{I}'}$ is much smaller than the original box $\mathbf{x}^{\mathbf{I}(0)}$ over which the problem is to be solved.

In the second phase, we compute sharp bounds on $f^*(\mathbf{c}^{\mathbf{I}})$ and on the smallest box $\mathbf{x}^{\mathbf{I}*}(\mathbf{c}^{\mathbf{I}})$ containing $\mathbf{x}^*(\mathbf{c}^{\mathbf{I}})$. In doing so, we restrict our search to the box $\mathbf{x}^{\mathbf{I}'}$. Since $\mathbf{x}^{\mathbf{I}'}$ is relatively small, the search is rapid.

The second phase involves solving separate problems for a lower and for an upper bound on $f^*(\mathbf{c}^{\mathbf{I}})$. It also involves separate problems for each lower and each upper bound on each component of $\mathbf{x}^*(\mathbf{c}^{\mathbf{I}})$. Thus, we solve $2n+2$ problems in the second phase if we want all the components of $\mathbf{x}^*(\mathbf{c}^{\mathbf{I}})$ and both lower and upper bounds on $f^*(\mathbf{c}^{\mathbf{I}})$.

We now consider that part of the second phase in which we bound $f^*(\mathbf{c}^{\mathbf{I}})$. A lower bound on the set of values of f^* can be obtained by solving the problem

$$\underset{\mathbf{x},\mathbf{c}}{\text{Minimize}} \;\; \text{(globally)}\; f(\mathbf{x}, \mathbf{c})$$

$$\text{subject to} \left\{ \begin{array}{l} p_i(\mathbf{x}, \mathbf{c}) \le 0 \; (i = 1, ..., m) \\ q_i(\mathbf{x}, \mathbf{c}) = 0 \; (i = 1, ..., r) \\ \mathbf{c} \in \mathbf{c}^{\mathbf{I}}, \;\; \mathbf{x} \in \mathbf{x}^{\mathbf{I}'} \end{array} \right.$$

This problem differs from the original problem (17.1.1) in that the components of $\mathbf{c}$ have become independent variables. In addition, the constraint $\mathbf{c} \in \mathbf{c}^{\mathbf{I}}$ has been added. The globally minimum value of f for this problem is obviously the desired lower bound on $f^*(\mathbf{c}^I)$. Since this new problem is not a perturbed one, it can be solved as accurately as desired subject only to rounding error limitations.

As noted earlier, we solve this problem over the box $\mathbf{x}^{I'}$, which crudely bounds $\mathbf{x}^*(\mathbf{c}^I)$ and is a relatively small box. Also, the box $\mathbf{c}^I$ is relatively small in general. Therefore, while the new problem is in higher dimension than the original one (17.1.1), we can expect it to be quickly solved.

Obtaining an upper bound on $f^*(\mathbf{c}^I)$ is a more complicated problem. We want the global solution to

$$\underset{\mathbf{c}\in\mathbf{c}^I}{\text{Maximize}}\ \underset{\mathbf{x}\in\mathbf{x}^{I'}}{\text{minimum}}\, f(\mathbf{x},\mathbf{c}) \tag{17.5.1}$$

$$\text{subject to } \begin{cases} p_i(\mathbf{x},\mathbf{c}) \le 0 \ (i = 1, ..., m), \\ q_i(\mathbf{x},\mathbf{c}) = 0 \ (i = 1, ..., r). \end{cases}$$

The box $\mathbf{x}^{I'}$ computed in the first phase of our procedure contains the global solution to problem (17.1.1) for every $\mathbf{c} \in \mathbf{c}^I$. Assume that $\mathbf{x}^{I'}$ does not also contain a local (i.e., nonglobal) solution of (17.1.1) for any $\mathbf{c} \in \mathbf{c}^I$.

A solution of (17.1.1) satisfies the John conditions $\phi(\mathbf{t}) = \mathbf{0}$ where $\phi(\mathbf{t})$ is given by (13.5.1) and

$$\mathbf{t} = \begin{pmatrix} \mathbf{x} \\ \mathbf{u} \\ \mathbf{v} \end{pmatrix}$$

where $\mathbf{u}$ and $\mathbf{v}$ are vectors of Lagrange multipliers. To emphasize that our problem now depends on the parameter vector $\mathbf{c}$, we now write the John conditions as $\phi(\mathbf{t},\mathbf{c}) = \mathbf{0}$. We call a point satisfying $\phi(\mathbf{t},\mathbf{c}) = \mathbf{0}$ a *John point*.

We make use of the following assumption:

Assumption 17.5.1 For any given $\mathbf{c} \in \mathbf{c}^I$, any point in $\mathbf{x}^{I'}$ that is a John point is a global solution of (17.1.1).

In Section 17.6, we demonstrate how we can show this assumption is true using cases described therein. As we also demonstrate, Assumption

17.5.1 cannot generally be validated if the global solution point changes discontinuously as $\mathbf{c}$ varies over $\mathbf{c}^{\mathbf{I}}$.

If Assumption 17.5.1 is valid, we can compute an upper bound on $f^*(\mathbf{c}^{\mathbf{I}})$ by solving the following problem:

$$\underset{\mathbf{t},\mathbf{c}}{\text{Maximize}}\ \text{(globally)}\ f(\mathbf{x},\mathbf{c}) \tag{17.5.2}$$

$$\text{subject to}\ \begin{cases} \phi(\mathbf{t},\mathbf{c}) = \mathbf{0} \\ \mathbf{c} \in \mathbf{c}^{\mathbf{I}}. \end{cases}$$

When solving this problem, we restrict the search to points $\mathbf{x} \in \mathbf{x}^{\mathbf{I}'}$.

If Assumption 17.5.1 is not valid, the solution to this problem is for a John point that is a local (nonglobal) solution. Therefore, the solution to (17.5.2) yields an unsharp upper bound on $f^*(\mathbf{c}^{\mathbf{I}})$.

When Assumption 17.5.1 is valid, we can formulate unperturbed problems to compute the smallest box $\mathbf{x}^{\mathbf{I}*}(\mathbf{c}^{\mathbf{I}})$ containing the solution set of the perturbed problem. To compute the left endpoint of a component of $\mathbf{x}_i^{\mathbf{I}*}(\mathbf{c}^{\mathbf{I}})$ $(i = 1, ..., n)$ of $\mathbf{x}^{\mathbf{I}*}(\mathbf{c}^{\mathbf{I}})$, we solve

$$\underset{\mathbf{t},\mathbf{c}}{\text{Minimize}}\ \text{(globally)}\ x_i \tag{17.5.3}$$

$$\text{subject to}\ \begin{cases} p_i(\mathbf{x},\mathbf{c}) \le 0\ (i = 1, ..., m) \\ q_i(\mathbf{x},\mathbf{c}) = 0\ (i = 1, ..., r) \\ \phi(\mathbf{t},\mathbf{c}) = \mathbf{0} \\ \mathbf{c} \in \mathbf{c}^{\mathbf{I}}. \end{cases}$$

To compute the right endpoint of $\mathbf{x}_i^{\mathbf{I}*}(\mathbf{c}^{\mathbf{I}})$, we solve

$$\underset{\mathbf{t},\mathbf{c}}{\text{Maximize}}\ \text{(globally)}\ x_i \tag{17.5.4}$$

$$\text{subject to}\ \begin{cases} p_i(\mathbf{x},\mathbf{c}) \le 0\ (i = 1, ..., m) \\ q_i(\mathbf{x},\mathbf{c}) = 0\ (i = 1, ..., r) \\ \phi(\mathbf{t},\mathbf{c}) = \mathbf{0} \\ \mathbf{c} \in \mathbf{c}^{\mathbf{I}}. \end{cases}$$

The solutions to problems (17.5.3) and (17.5.4) yield lower and upper bounds on $\mathbf{x}_i^{\mathbf{I}*}(\mathbf{c}^{\mathbf{I}})$, respectively. However, if Assumption 17.5.1 is not valid, these bounds might not be sharp.

Note that to compute both endpoints of components $\mathbf{x}^{\mathbf{I}*}_i(\mathbf{c}^{\mathbf{I}})$ for all $i = 1, ..., n$, we must solve $2n$ problems. However, each problem is relatively easy to solve because the search for a solution is restricted to the small region $\mathbf{x}^{\mathbf{I}'}$.

Note, also, that since $\mathbf{x}^{\mathbf{I}'}$ is small, it is unlikely that there is a local (nonglobal) minimum for the initial problem (17.1.1) in $\mathbf{x}^{\mathbf{I}'}$. That is, Assumption 17.5.1 is likely to be valid for most problems in practice. However, we give examples in Sections 17.7 and 17.8 for which Assumption 17.5.1 is not valid. In Section 17.6, we show how Assumption 17.5.1 can sometimes be validated in practice.

When doing a sensitivity analysis of a particular problem, we might not want to allow all the components of $\mathbf{c}$ to vary simultaneously. Instead, we might wish to perturb various subsets of the components of $\mathbf{c}$. In this case, we can proceed as follows.

First, choose $\mathbf{c}^{\mathbf{I}}$ so that it contains all perturbations of all the "interesting" components of $\mathbf{c}$. Do the first phase of the algorithm as described above. Let $\mathbf{x}^{\mathbf{I}'}$ denote the smallest single box containing the output box(es) of the first phase.

For each perturbed problem involving subsets of $\mathbf{c}$, the box containing the perturbed parameters is contained in $\mathbf{c}^{\mathbf{I}}$. Hence, the set of output boxes for each subproblem is contained in $\mathbf{x}^{\mathbf{I}'}$. Therefore, for each subproblem, the search in the first phase can be restricted to $\mathbf{x}^{\mathbf{I}'}$. Since $\mathbf{x}^{\mathbf{I}'}$ is generally smaller than the original region of search, this can save a considerable amount to computing.

17.6 VALIDATING ASSUMPTION 17.5.1

For some problems, we can validate Assumption 17.5.1. In this section, we show how this can be done. We first note that in certain cases our validation procedure cannot be successful.

Suppose we solve a given perturbed problem. That is, we replace $\mathbf{c}$ by the box $\mathbf{c}^{\mathbf{I}}$ and solve the optimization problem using an interval method from Section 12.14, 14.8, 15.12, or 16.4. Suppose the output boxes from this phase can be separated into two (or more) subsets S and S' that are strictly disjoint. That is, no box in S touches any box in S'. Then it is possible that, as $\mathbf{c}$ varies over $\mathbf{c}^{\mathbf{I}}$, a global minimum jumps discontinuously from a point in S to a point in S' while the point in S becomes a local minimum. In such a case, the output boxes from the first phase contain John points that are not global minima for all $\mathbf{c} \in \mathbf{c}^{\mathbf{I}}$.

In the second phase of our algorithm to bound $f^*(\mathbf{c}^{\mathbf{I}})$ and $\mathbf{x}^{\mathbf{I}*}(\mathbf{c}^{\mathbf{I}})$, we solve the problems in Section 17.5. In the case we are considering, these problems do not yield sharp bounds on either $f^*(\mathbf{c}^{\mathbf{I}})$ or $\mathbf{x}^{\mathbf{I}*}(\mathbf{c}^{\mathbf{I}})$. This remains true even if the problems are solved separately over each component of the disjoint sets of boxes that are output in the first phase of our algorithm.

Suppose the output of the first phase is composed of disjoint sets of boxes. It is quite possible that the John points in all the output boxes are global minima. Unfortunately, however, there seems to be no way to determine whether this is the case or whether some of the John points are local minima.

Therefore, in this case, it seems we must use one of two options, First, we can use the second phase and accept the fact that the bounds computed for $f^*(\mathbf{c}^{\mathbf{I}})$ and $\mathbf{x}^{\mathbf{I}*}(\mathbf{c}^{\mathbf{I}})$ might not be sharp. Second, we can continue with the first phase using smaller termination tolerances and cover the solution set of the original problem (17.1.1) by a large number of small boxes.

We now consider the favorable case in which the box(es) computed in the first phase do not form strictly disjoint subsets. We are sometimes able to prove that the solutions to the problems in Section 17.5 yield sharp bounds on $f^*(\mathbf{c}^{\mathbf{I}})$ and $\mathbf{x}^{\mathbf{I}*}(\mathbf{c}^{\mathbf{I}})$. The proof relies on the following theorem.

Theorem 17.6.1 Let $\mathbf{f}(\mathbf{x})$ be a continuously differentiable vector function. Let $\mathbf{J}(\mathbf{x})$ be the Jacobian of $\mathbf{f}(\mathbf{x})$. Suppose we evaluate the Jacobian over a box $\mathbf{x}^{\mathbf{I}}$. Assume $\mathbf{J}(\mathbf{x}^{\mathbf{I}})$ does not contain a singular matrix. Then the equation $\mathbf{f}(\mathbf{x}) = \mathbf{0}$ can have no more than one solution in $\mathbf{x}^{\mathbf{I}}$.

Proof. Assume there are two solutions, $\mathbf{x}$ and $\mathbf{y}$, of $\mathbf{f}(\mathbf{x}) = \mathbf{0}$. From Section 7.3 or 7.4,

$$\mathbf{f}(\mathbf{y}) \in \mathbf{f}(\mathbf{x}) + \mathbf{J}(\mathbf{x}^{\mathbf{I}})(\mathbf{y} - \mathbf{x}).$$

Let $\mathbf{J}'$ be a matrix in $\mathbf{J}(\mathbf{x}^\mathbf{I})$ such that equality holds in this relation. Since $\mathbf{x}$ and $\mathbf{y}$ are zeros of $\mathbf{f}$,

$$\mathbf{J}'(\mathbf{y} - \mathbf{x}) = \mathbf{0}.$$

Since every matrix (including $\mathbf{J}'$) in $\mathbf{J}(\mathbf{x}^\mathbf{I})$ is nonsingular, it follows that $\mathbf{y} = \mathbf{x}$. That is, any zero of $\mathbf{f}$ in $\mathbf{x}^\mathbf{I}$ is unique. ■

We use the following corollary of this theorem.

Corollary 17.6.2 Suppose the function $\mathbf{f}$ in Theorem 17.6.1 depends on a vector $\mathbf{c}$ of parameters. Suppose we replace $\mathbf{c}$ in $\mathbf{f}$ and in $\mathbf{J}$ by a box $\mathbf{c}^\mathbf{I}$ containing $\mathbf{c}$. If $\mathbf{J}(\mathbf{x}^\mathbf{I}, \mathbf{c}^\mathbf{I})$ does not contain a singular matrix, then any zero of $\mathbf{f}(\mathbf{x}, \mathbf{c})$ in $\mathbf{x}^\mathbf{I}$ is unique for each $\mathbf{c} \in \mathbf{c}^\mathbf{I}$.

Proof. Note that $\mathbf{J}(\mathbf{x}^\mathbf{I}, \mathbf{c}) \in \mathbf{J}(\mathbf{x}^\mathbf{I}, \mathbf{c}^\mathbf{I})$ for any $\mathbf{c} \in \mathbf{c}^\mathbf{I}$. Therefore, if $\mathbf{J}(\mathbf{x}^\mathbf{I}, \mathbf{c}^\mathbf{I})$ does not contain a singular matrix, neither does $\mathbf{J}(\mathbf{x}^\mathbf{I}, \mathbf{c})$ for any $\mathbf{c} \in \mathbf{c}^\mathbf{I}$. Hence, Corollary 17.6.2 follows from Theorem 17.6.1. ■

We now consider how we can prove that $\mathbf{J}(\mathbf{x}^\mathbf{I}, \mathbf{c}^\mathbf{I})$ does not contain a singular matrix. As when preconditioning a linear system (see Section 5.6), we multiply $\mathbf{J}(\mathbf{x}^\mathbf{I}, \mathbf{c}^\mathbf{I})$ by an approximate inverse $\mathbf{B}$ of its center. We obtain $M(\mathbf{x}^\mathbf{I}, \mathbf{c}^\mathbf{I}) = \mathbf{BJ}(\mathbf{x}^\mathbf{I}, \mathbf{c}^\mathbf{I})$. If $\mathbf{M}(\mathbf{x}^\mathbf{I}, \mathbf{c}^\mathbf{I})$ does not contain a singular matrix (i.e., is regular), then neither does $\mathbf{J}(\mathbf{x}^\mathbf{I}, \mathbf{c}^\mathbf{I})$. Therefore, we can check the validity of Corollary 17.6.2 by using $\mathbf{M}(\mathbf{x}^\mathbf{I}, \mathbf{c}^\mathbf{I})$ rather than $\mathbf{J}(\mathbf{x}^\mathbf{I}, \mathbf{c}^\mathbf{I})$.

If $\mathbf{B}$ and $\mathbf{M}(\mathbf{x}^\mathbf{I}, \mathbf{c}^\mathbf{I})$ are exact, the center of $\mathbf{M}(\mathbf{x}^\mathbf{I}, \mathbf{c}^\mathbf{I})$ is the identity matrix and the off-diagonal elements are centered about zero. As a result, $\mathbf{M}(\mathbf{x}^\mathbf{I}, \mathbf{c}^\mathbf{I})$ tends to be diagonally dominant. A simple sufficient test for regularity of $\mathbf{M}(\mathbf{x}^\mathbf{I}, \mathbf{c}^\mathbf{I})$ is to check for diagonally dominance. A necessary and sufficient but more computationally intense test is given by Theorem 5.8.1. See Section 5.8.2 for a discussion of how to use Theorem 5.8.1 in practice.

If $\mathbf{M}(\mathbf{x}^\mathbf{I}, \mathbf{c}^\mathbf{I})$ is regular, then from Corollary 17.6.2, there is at most one solution of $\mathbf{f}(\mathbf{x}^\mathbf{I}, \mathbf{c}^\mathbf{I}) = \mathbf{0}$ in $\mathbf{x}^\mathbf{I}$ for each $\mathbf{c} \in \mathbf{c}^\mathbf{I}$.

We now apply this analysis to the question of whether a John point in a given box is unique. If so, we are sometimes able to prove that any John point in a box of interest is a global minimum for some value of $\mathbf{c} \in \mathbf{c}^\mathbf{I}$. We are successful if the result of the Newton step is contained in the input box. See Proposition 11.15.5.

Again let $\mathbf{x}^{\mathbf{I}'}$ denote the smallest box containing the output box(es) from the first phase of our minimization algorithm applied to a perturbed problem.

Let $\phi(\mathbf{t}, \mathbf{c})$ denote the John function given by (13.5.1) and discussed in Section 17.5. Recall that

$$\mathbf{t} = \begin{pmatrix} \mathbf{x} \\ \mathbf{u} \\ \mathbf{v} \end{pmatrix}$$

where $\mathbf{u}$ and $\mathbf{v}$ are vectors of Lagrange multipliers. The variable $\mathbf{t}$ takes the place of the variable $\mathbf{x}$ used when discussing Corollary 17.6.2. Let $\mathbf{J}(\mathbf{t}, \mathbf{c})$ denote the Jacobian of $\mathbf{f}(\mathbf{t}, \mathbf{c})$ as a function of $\mathbf{t}$. We can use this Jacobian as described above to prove that any John point in $\mathbf{x}^{\mathbf{I}'}$ is a global minimum for some $\mathbf{c} \in \mathbf{c}^{\mathbf{I}}$. Proof is obtained only if the result of the Newton step is contained in the input box. See Proposition 11.15.5.

Note that a subroutine is available for evaluating the Jacobian (and multiplying by an approximate inverse of its center) when doing the first stage of the algorithm in Section 17.5. Therefore, only a small amount of coding is needed to test the hypothesis of Corollary 17.6.2.

17.7 FIRST NUMERICAL EXAMPLE

In this and the next two section, we discuss examples that illustrate our method for solving perturbed problems.

As a first example, we consider the unconstrained minimization problem with objective function

$$f(x_1, x_2) = 12x_1^2 - 6.3x_1^4 + \mathbf{c}x_1^6 + 6x_1x_2 + 6x_2^2.$$

For $c = 1$, this is (the negative of) the so-called three hump camel function. See (for example) Dixon and Szegö (1975).

Let the coefficient (i.e., parameter) c vary over the interval $[0.9, 1]$. For all c for which $0.945 < c \leq 1$, the global minimum is the single point at the origin; and $f^* = 0$ at this point. For $c = 0.945$, there are three global minima. One is at the origin. The others are at $\pm(a, -a/2)$ where $a = (10/3)^{1/2}$.

For $c < 0.945$, there are two global minima at $\pm(b, -b/2)$ where

$$b = \left[\frac{4.2 + (17.64 - 14c)^{1/2}}{2c}\right]^{1/2}. \tag{17.7.1}$$

At these points, f takes the negative (minimal) value

$$f^* = \frac{22.05c - 18.522 + (3.5c - 4.41)(17.64 - 14c)^{1/2}}{c^2}. \tag{17.7.2}$$

The smallest value of f^* for $c \in [0.9, 1]$ occurs for $c = 0.9$ for which $f^* = -1.8589$, approximately.

Consider perturbing the problem continuously by letting c increase from an initial value of 0.9. Initially, there are two global solution points that move along (separate) straight lines in the x_1, x_2 plane until $c = 0.945$. As c passes through this value, the global minimum jumps to the origin and remains there for all $c \in [0.945, 1]$.

A traditional perturbation analysis can detect the changes in the global minimum as c increases slightly from $c = 0.9$ (say). However, an expansion about this point cannot reveal the nature of the discontinuous change in position of the global minimum as c varies at and near the value 0.945.

The algorithm in Section 12.14 solves this problem without difficulty. Unfortunately, if termination tolerances are small, the output consists of an undesirably large number of boxes.

For $\mathbf{c}^\mathbf{I} = [0.9, 1]$, the set $\mathbf{x}^*(\mathbf{c}^\mathbf{I})$ of solution points consists of three parts. One part is the origin. Another is the line segment joining the two points of the form $(y, -y/2)$ where $y = (10/3)^{1/2}$ at one endpoint and $y = \{[21 + (126)^{1/2}]/9\}^{1/2}$ at the other endpoint. The third part of the set is the reflection of this line segment in the origin.

The interval algorithm does not reveal the value of c at which the global point jumps discontinuously. However, it bounds the solution set for all $c \in [0.9, 1]$ as closely as prescribed by the tolerances.

We ran this problem with c replaced by [0.9, 1]. The initial box had components $X_1^{(0)} = X_2^{(0)} = [-2, 4]$. The function width tolerance ε_f was chosen to be large so that termination was determined by the box size tolerance ε_X. Thus, we chose $\varepsilon_f = 10^5$.

We chose ε_X to have the relatively large value 0.1 so that only a few boxes were needed to cover the solution set. As is generally desired for the first stage of the two-stage process described in Section 17.5, we obtained rather crude bounds on the solution set.

The data for this example were computed using an algorithm similar to but less sophisticated than the one in Section 12.14. The algorithm produced a set of 11 boxes covering the solution set. The smallest box covering one subset of the exact solution is

$$\begin{bmatrix} [1.825, 1.893] \\ 0.9128, 0.9462] \end{bmatrix}. \tag{17.7.3}$$

This solution set was covered by five "solution" boxes. The smallest box containing these five boxes is

$$\begin{bmatrix} [1.739, 1.896] \\ 0.856, 0.968] \end{bmatrix}.$$

The interval components of this box are too large by roughly the size of the tolerance ε_X.

The reflection (in the origin) of the exact solution set (17.7.3) is also a solution set. It was covered by five "solution" boxes in much the same way.

The third subset of the exact solution is the origin. It remains an isolated solution point as c varies. It is a global solution for all $c \in [0.945.1]$. Its location was computed precisely and is guaranteed to be exact because the bounding interval components are degenerate.

The bounds on the set of values of f^* were computed to be the interval $[-4.781, 0]$. The correct interval is $[-1.859, 0]$. Since we chose $\varepsilon_f = 10^5$, the algorithm was not configured to produce good bounds on the interval $f^*(\mathbf{c}^\mathbf{I})$. The more stringent box size tolerance $\varepsilon_X = 0.1$ kept the bounds on

$f^*(\mathbf{c}^\mathbf{I})$ from being even worse. When we choose ε_X smaller, we incidentally bound $f^*(\mathbf{c}^\mathbf{I})$ more sharply.

With a smaller value of ε_X, the algorithm produces a larger number of smaller boxes covering the solution set more accurately and produces a narrower bound on $f^*(\mathbf{c}^\mathbf{I})$.

Note that even with the loose tolerances used, the algorithm correctly computed three disjoint sets covering the three correct solution sets.

Since the set of output boxes formed strictly disjoint subsets, we must allow for the possibility that the global solution points move discontinuously as c varies. This implies that the John points in the output boxes can correspond to local (nonglobal) minima for some values of $c \in \mathbf{c}^\mathbf{I}$. For this example, we know that this is, in fact, the case. Generally, however, we do not know the nature of such a solution.

If we apply the method of Section 17.5 to bound $f^*(\mathbf{c}^\mathbf{I})$ for each of the three "solution" regions separately, we obtain the approximate bounding interval $[-1.8589, 5.4359]$. Outwardly rounded to five decimal digits, the correct interval is $[-1.8589, 0]$. To compute a reasonably accurate result, ε_X must be considerably smaller than the 0.1 value used above. Alternatively, ε_f can be chosen smaller.

17.8 SECOND NUMERICAL EXAMPLE

We now consider a second example. It adds little to our understanding of the subject of this chapter. However, it is another easily analyzed example with which to research global optimization algorithms.

Our example is an inequality constrained minimization problem. The objective function is

$$f(x_1, x_2) = 12x_1^2 - 6.3x_1^4 + cx_1^6 + 6x_1x_2 + 6x_2^2$$

as in Section 17.7. We impose the constraints

$$p_1(\mathbf{x}) = 1 - 16x_1^2 - 25x_2^2 \le 0,$$

$$p_2(\mathbf{x}) = 13x_1^3 - 145x_1 + 85x_2 - 400 \le 0,$$

$$p_3(\mathbf{x}) = x_1x_2 - 4 \le 0.$$

As in Section 17.7, we replace c by the interval [0.9, 1].

For this problem, the position of the global minimum again jumps discontinuously as c varies. The jump occurs at $c = c'$ where $c' = 0.95044391$, approximately.

For $c' \leq c < 1$, the global minimum occurs at two points on the boundary of the feasible region where $p_1(\mathbf{x}) = 0$. For $c = c'$, these two points are still global, and there are two other global minima in the interior of the feasible region. For $0.9 \leq c < c'$, only the two minima in the interior are global.

The minima in the interior are at $\pm(b, -b/2)$ where b is given by (17.7.1). The value of the objective function at these points is given by (17.7.2).

It can be shown that the minima on the boundary of the feasible region where $p_1(\mathbf{x}) = 0$ are at $\pm(x_1, x_2)$ where x_1 satisfies

$$(81.6x_1 - 126x_1^3 + 30cx_1^5)(1 - 16x_1^2)^{1/2} - 6 + 192x_1^2 = 0$$

and

$$x_2 = -(1 - 16x_1^2)/5.$$

The value of the minimum at these points depends very little on c. For $c = 0.9$, the global minimum is $f^* = 0.199035280$ and for $c = 1$, $f^* = 0.199035288$ approximately. The value of x_1 for $c = c'$ is 0.06604161 and for $c = 1$ it is 0.066041626 approximately.

The smallest boxes containing the set of points of global minimum as c varies over [0.9, 1] are (when outwardly rounded)

$$\pm \begin{bmatrix} [0.0660416, 0.0660417] \\ [-0.192896, -0.192895] \end{bmatrix}$$

and

$$\pm \begin{bmatrix} [1.81787, 1.89224] \\ [-0.946118, -0.908935] \end{bmatrix}.$$

We solved this problem by an algorithm that differs somewhat from the one given in Section 14.8. For reasons given in Section 17.3, we want ε_f to be large. We chose $\varepsilon_f = 10^5$. We again regard the computations to be the first stage of the two stage process given in Section 17.5. For such a computation, we want ε_X to be relatively large. We chose $\varepsilon_X = 0.05$.

The algorithm produced a set of 92 boxes as the solution. One isolated box is approximately

$$\begin{bmatrix} [0.0625, 0.0750] \\ [-0.2040, -0.1779] \end{bmatrix}.$$

Another isolated box is approximately the negative of this one. They contain the minima on the boundary of the feasible region.

One set of 76 contiguous boxes is isolated from all the other output boxes. The smallest box containing all of them, when outwardly rounded is

$$\mathbf{y}^{\mathbf{I}} = \begin{bmatrix} [1.7472, 1.8923] \\ [-0.9611, -0.8679] \end{bmatrix}.$$

The remaining set of 14 boxes is contained in a single box that is approximately the negative of this one.

Because of the loose tolerances used, the results do not bound the solution sets very tightly. The first output box above bounds a solution that is insensitive to perturbation of c. Therefore the computed result can be greatly improved by making the tolerance smaller.

Let $\mathbf{y}^{\mathbf{I}}$ denote the box containing the 76 contiguous output boxes as given above. This box bounds a solution that is more sensitive to c's perturbation. The width of $\mathbf{y}^{\mathbf{I}}$ is 0.1451, while the correct solution can be bounded by a box of width 0.0743. The individual boxes from which $\mathbf{y}^{\mathbf{I}}$ is determined satisfy the convergence criteria and hence are each of width ≤ 0.05. However, the size of the box covering their union is determined by the problem itself and cannot be changed by choosing a tolerance.

Since the output consists of strictly disjoint subsets, we cannot expect Assumption 17.5.1 of Section 17.5 to be satisfied. As in the example in Section 17.7, we must either be satisfied with poor bounds on $f^*(\mathbf{c}^{\mathbf{I}})$ and

$\mathbf{x}^{\mathbf{I}*}(\mathbf{c}^{\mathbf{I}})$ or else continue with the first phase of our algorithm using a smaller box size tolerance.

17.9 THIRD NUMERICAL EXAMPLE

We now consider an example of a perturbed problem that arose in practice as a chemical mixing problem. It is an equality constrained least squares problem given by

$$\begin{aligned}
\text{Minimize (globally) } f(\mathbf{x}) &= \sum_{i=1}^{18}(x_i - c_i)^2 \\
\text{subject to } x_1x_9 &= x_2x_{14} + x_3x_4 \\
x_1x_{10} &= x_2x_{15} + x_3x_5 \\
x_1x_{11} &= x_2x_{16} + x_3x_6 \\
x_1x_{12} &= x_2x_{17} + x_3x_7 \\
x_1x_{13} &= x_2x_{18} + x_3x_8 \\
x_4 + x_5 + x_6 + x_7 + x_8 &= 1 \\
x_9 + x_{10} + x_{11} + x_{12} + x_{13} &= 1 \\
x_{14} + x_{15} + x_{16} + x_{17} + x_{18} &= 1 \\
x_{14} &= 66.67x_4 \\
x_{15} &= 50x_5 \\
x_{16} &= 0.015x_6 \\
x_{17} &= 100x_7 \\
x_{18} &= 33.33x_8.
\end{aligned}$$

The parameters c_i and their uncertainties ε_i are given in the following table.

i	$c_i \pm \varepsilon_i$	i	$c_i \pm \varepsilon_i$
1	100 ± 1.11	10	0.66 ± 0.017
2	89.73 ± 1.03	11	0.114 ± 0.0046
3	10.27 ± 0.51	12	0.002 ± 0.0001
4	0.0037 ± 0.00018	13	0.004 ± 0.00012
5	0.0147 ± 0.0061	14	0.245 ± 0.0067
6	0.982 ± 0.032	15	0.734 ± 0.02
7	0 ± 0	16	0.0147 ± 0.0061
8	0.0001 ± 0	17	0.0022 ± 0.0001
9	0.22 ± 0.0066	18	0.0044 ± 0.0013

We used the constraints to eliminate variables and write the problem as an unconstrained problem in five variables. We first solved the unperturbed case. The minimum value of f was found to be $f^* = 3.07354796 \times 10^{-7} \pm 2 \times 10^{-15}$. We then solved the perturbed case. and obtained a set of 59 boxes covering the solution set. The smallest single box (call it $\mathbf{x}^{\mathbf{I}'}$) containing the 59 boxes is given in the following table. We obtained the interval [0, 2.556] bounding $f^*(\mathbf{c}^{\mathbf{I}})$.

i	X_i'	i	X_i'
1	[96.2, 103.8]	10	[0.6046, 0.7207]
2	[87.7, 91.7]	11	[0.0603, 0.1638]
3	[8.47, 12.07]	12	[−0.0174, 0.0237]
4	[0.0035, 0.00348]	13	[−0.0109, 0.0205]
5	[0.0142, 0.0152]	14	[0.2338, 0.2559]
6	[0.98142, 0.98147]	15	[0.7122, 0.7556]
7	[−0.000205, 0.000249]	16	[0.0147, 0.0148]
8	[−0.000384, 0.000645]	17	[−0.0205, 0.0249]
9	[0.1983, 0.2440]	18	[−0.0128, 0.0215]

As pointed out above, we used the equality constraints to eliminate variables and obtain an unconstrained optimization problem. Therefore, the Jacobian for the function $\phi(\mathbf{x}, \mathbf{c})$ expressing the John conditions (see Section 17.6) is just the Hessian of the objective function. As described in

Section 17.6, we verified that the Hessian (with $\mathbf{c}$ replaced by $\mathbf{c}^{\mathbf{I}}$) does not contain a singular matrix. Therefore, we know that the method described in Section 17.5 yields sharp bounds on $f^*(\mathbf{c}^{\mathbf{I}})$ and $\mathbf{x}^{\mathbf{I}*}(\mathbf{c}^{\mathbf{I}})$. We did not do the computations.

17.10 AN UPPER BOUND

In Section 12.5, and elsewhere, we discuss how we can use an upper bound $\overline{f}$ on the global minimum to improve the performance of our global optimization algorithm. In this section, we consider an artifice in which we preset $\overline{f}$ to zero in certain examples. We then give an example that shows why this is particularly helpful in the perturbed case.

For simplicity, we consider the unconstrained case. Suppose we wish to minimize $f(\mathbf{x}, \mathbf{c}^{\mathbf{I}})$ where $\mathbf{c}^{\mathbf{I}}$ is a nondegenerate interval vector. We apply the algorithm given in Section 12.14.

Assume we know that $f(\mathbf{x}, \mathbf{c})$ is nonnegative for all values of $\mathbf{x}$ and $\mathbf{c}$ of interest. Suppose we also know that $f(\mathbf{x}^*, \mathbf{c}) = \mathbf{0}$ for all $\mathbf{c} \in \mathbf{c}^{\mathbf{I}}$, where $\mathbf{x}^*$ is the point of global minimum. Then we know that $f^*(\mathbf{c}^{\mathbf{I}}) = \mathbf{0}$. Therefore, we set $\overline{f} = 0$.

Least squares problems are sometimes of this type of problem. It can be known that the squared functions are consistent and, hence, that $f^*(\mathbf{c}^{\mathbf{I}}) = 0$. For example, see Walster (1988).

As described in Section 12.5, our algorithm deletes points of an original box in which $f(\mathbf{x}, \mathbf{c}^{\mathbf{I}}) > \overline{f}$. The smaller $\overline{f}$, the more points that can be deleted in a given application of the procedure. Thus, it is advantageous to know $f^*(\mathbf{c}^{\mathbf{I}})$ when the algorithm is first applied. Often, the first value of $\overline{f}$ computed by the algorithm is much larger than $f^*(\mathbf{c}^{\mathbf{I}})$. Values of $\overline{f}$ closer to $f^*(\mathbf{c}^{\mathbf{I}})$ are computed as the algorithm proceeds.

We know $f^*(\mathbf{c}^{\mathbf{I}}) = 0$. In addition to speeding up the algorithm, this also saves the effort of repeatedly trying to improve $\overline{f}$. But, in the perturbed case, there is another advantage. When we evaluate $f(\mathbf{x}, \mathbf{c}^{\mathbf{I}})$ at some point $\mathbf{x}$, we obtain an interval. The upper endpoint of this interval is an upper bound for $f^*(\mathbf{c}^{\mathbf{I}})$. But even if we evaluate $f(\mathbf{x}, \mathbf{c}^{\mathbf{I}})$ at a global minimum point $\mathbf{x}^*$, the upper endpoint of the interval is not f^*. It is larger because the uncertainty imposed by the interval $\mathbf{c}^\mathbf{I}$ precludes the interval $f(\mathbf{x}^*, \mathbf{c}^\mathbf{I})$ from being degenerate.

Therefore, we can never compute an upper bound $\bar{f}$ equal to $f^*(\mathbf{c}^\mathbf{I})$ by evaluating $f(\mathbf{x}, \mathbf{c}^\mathbf{I})$. Knowing $f^* = 0$ and setting $\bar{f} = 0$ provides an advantage not otherwise obtainable.

We now consider an example. In Section 17.1, we pointed out that the problem of solving the set of equations $\mathbf{A}^\mathbf{I}\mathbf{x} = \mathbf{b}^\mathbf{I}$ can be recast as the least squares problem of minimizing

$$f(\mathbf{x}, \mathbf{c}^\mathbf{I}) = (\mathbf{A}^\mathbf{I}\mathbf{x} - \mathbf{b}^\mathbf{I})^T(\mathbf{A}^\mathbf{I}\mathbf{x} - \mathbf{b}^\mathbf{I}).$$

Here, the parameter vector $\mathbf{c}^\mathbf{I}$ is composed of the elements of $\mathbf{A}^\mathbf{I}$ and the components of $\mathbf{b}^\mathbf{I}$.

Consider the system given in Equation (5.3.1). The solution set is shown in Figure 5.3.1. The smallest box containing the solution set is

$$\begin{bmatrix} [-120, 90] \\ [-60, 240] \end{bmatrix}.$$

When we evaluate $f(\mathbf{x}, \mathbf{c}^\mathbf{I})$ at some point $\mathbf{x}$, we obtain an interval $[\underline{f}(\mathbf{x}, \mathbf{c}^\mathbf{I}), \overline{f}(\mathbf{x}, \mathbf{c}^\mathbf{I})]$. It can be shown that the smallest value of $\overline{f}(\mathbf{x}, \mathbf{c}^\mathbf{I})$ for any $\mathbf{x}$ is $2862900/121$, which is approximately 23660.33. Suppose we compute $\bar{f}$ equal to this best possible value we can compute.

Suppose we then delete all points $\mathbf{x}$ where $\underline{f}(\mathbf{x}, \mathbf{c}^\mathbf{I}) > \bar{f}$. The smallest box that contains the remaining points can be shown to be

$$\begin{bmatrix} [-291.98, 243.82] \\ [-277.53, 457.53] \end{bmatrix}.$$

If we rely only on the procedure that uses $\bar{f}$ to delete points, the computed solution is much larger than it is possible to compute by including other procedures. The other procedures in the optimization algorithm delete whatever remaining points they can that are outside the solution set.

If we set $\bar{f} = 0$, then deleting points where $\underline{f}(\mathbf{x}, \mathbf{c}^\mathbf{I}) > \bar{f}$ can delete all points not in the solution set. That is, the subroutine using $\bar{f}$ can contribute to the progress of the algorithm as long as points remain that are not in the solution set.

17.11 SHARP BOUNDS FOR PERTURBED SYSTEMS OF NONLINEAR EQUATIONS

We discussed perturbed systems of nonlinear equations of one variable in Section 9.10 and the multivariable case in Section 11.17. In this section, we discuss how such problems can be recast as optimization problems. We can then use the methods discussed earlier in this chapter to compute sharp bounds on the smallest box containing the solution set.

Consider a perturbed problem

$$\mathbf{f}(\mathbf{x}, \mathbf{c}^{\mathbf{I}}) = \mathbf{0} \tag{17.11.1}$$

where $\mathbf{f}$ is a vector of nonlinear functions and $\mathbf{x}$ is a vector of the same number of dimensions. The interval $\mathbf{c}^{\mathbf{I}}$ can be a scalar or vector. We replace this problem by

$$\text{Minimize } f(\mathbf{x}, \mathbf{c}^{\mathbf{I}}) \tag{17.11.2}$$

where

$$f(\mathbf{x}, \mathbf{c}^{\mathbf{I}}) = [\mathbf{f}(\mathbf{x}, \mathbf{c}^{\mathbf{I}})]^T \mathbf{f}(\mathbf{x}, \mathbf{c}^{\mathbf{I}}).$$

Pintér (1990) suggests solving unperturbed systems of nonlinear equations by recasting them as global optimization problems. He does not use interval methods. He considers more general norms than least squares.

We can apply the method described in Section 17.5 to solve (17.11.2) and thus compute sharp bounds on the smallest box containing the set of solutions of $\mathbf{f}(\mathbf{x}, \mathbf{c}^{\mathbf{I}}) = \mathbf{0}$.

It is reasonable to assume that (17.11.1) has a solution for all $\mathbf{c} \in \mathbf{c}^{\mathbf{I}}$. However, we need assume only that there exists at least one $\mathbf{c} \in \mathbf{c}^{\mathbf{I}}$ for which a solution exists. Under this assumption, the globally minimum value f^* of f is zero.

In Section 12.5, we discussed how an upper bound $\bar{f}$ on the global minimum f^* can be used in our optimization algorithm. Since we know that $f^* = 0$, we set $\bar{f} = 0$. As pointed out in Section 17.10, this improves the performance of our algorithm.

Chapter 18

MISCELLANY

18.1 NONDIFFERENTIABLE FUNCTIONS

In this chapter, we discuss some short topics that do not fit conveniently into previous chapters. We begin with a discussion of nondifferentiable objective functions.

As we have pointed out earlier, the simplest interval methods for global optimization do not require that the objective function or the constraint functions be differentiable. However, the most efficient methods require some degree of continuous differentiability. It is sometimes possible to replace problems involving nondifferentiable functions with ones having the desired smoothness.

We now consider two such reformulations from Lemaréchal (1982). We then introduce two more general reformulations.

In what follows in this section, the letter **x** denotes a vector of precisely n variables. This remains true even after we introduce additional variables x_{n+1}, x_{n+2}, etc.

Consider the unconstrained minimization problem

$$\text{Minimize } f(\mathbf{x}) = \sum_{i=1}^{m} |f_i(\mathbf{x})| \tag{18.1.1}$$

in n variables. Note that $|f_i(\mathbf{x})|$ is not differentiable where $f_i(\mathbf{x}) = 0$. Lemaréchal reformulates the problem as the following inequality constrained problem in $n + m$ variables.

$$\text{Maximize } \sum_{i=n+1}^{n+m} x_i \tag{18.1.2}$$

$$\text{subject to} \begin{cases} f_i(\mathbf{x}) \le x_{n+i} \ (i = 1, \cdots, m) \\ -f_i(\mathbf{x}) \le x_{n+i} \ (i = 1, \cdots, m). \end{cases}$$

The differentiability of this new problem is limited only by the differentiability of the functions f_i $(i = 1, \cdots, m)$.

Next, consider the minimax problem

$$\text{Minimize } \underset{i=1,\cdots,m}{\text{maximum}} \ f_i(\mathbf{x}). \tag{18.1.3}$$

Lemaréchal notes that this problem can be reformulated as

$$\text{Minimize } x_{n+1} \tag{18.1.4}$$

$$\text{subject to } f_i(\mathbf{x}) \le x_{n+1} \ (i = 1, \cdots, m).$$

Note that in each of Lemaréchal's reformulations, it is necessary that at least one of the introduced constraints be active at the solution. This is ensured by the fact that the objective function is minimized for each problem.

With Lemaréchal's reformulations, we are able to replace the absolute value function and the max function by differentiable functions provided they occur in the objective function. We now give different reformulations that allow these functions to occur in the constraints of the original problem as well as in more general forms in the objective function.

We first consider the absolute value function. Suppose the function $|t(\mathbf{x})|$ occurs somewhere in the statement of a given optimization problem. We replace $|t(\mathbf{x})|$ by a new variable x_{n+1} and add the constraints $x_{n+1} \ge 0$ and

$$[t(\mathbf{x})]^2 - x_{n+1}^2 = 0. \tag{18.1.5}$$

The presence of the constraint (18.1.5) causes x_{n+1} to take on the required value of $|t(\mathbf{x})|$ at the solution point. Lemaréchal's use of inequality constraints allows slack at the solution, in general.

Next, we consider the max function. We begin by noting that

$$\max\{t_1, t_2\} = 0.5(t_1 + t_2 + |t_1 - t_2|).$$

Using this relation and our procedure for replacing the absolute value function, we can replace $\max\{t_1, t_2\}$ by $0.5[t_1(\mathbf{x}) + t_2(\mathbf{x}) + x_{n+1}]$ if we add the constraints $x_{n+1} \geq 0$ and

$$[t_1(\mathbf{x}) - t_2(\mathbf{x})]^2 - x_{n+1}^2 = 0.$$

This enables us to treat the max of two functions. If there are more than two functions, we can use the relation

$$\max\{t_1, t_2, t_3\} = \max\{t_1, \max(t_2, t_3)\}$$

recursively.

Note that the minimum of two or more functions can be treated by using the relation

$$\min\{f_1, f_2\} = -\max\{-f_1, -f_2\}.$$

These procedures produce differentiability at the expense of added variables. Therefore, we have two options. First, we can solve the original problem using a simple (and hence slow) interval algorithm that does not require differentiability. Second, we can solve a reformulated problem involving more variables using a more efficient algorithm.

It is not clear which option is best for a given problem. However, it seems probable that the second approach requires less computing, in general.

The minimax problem with or without inequality constraints can be solved directly by interval methods. That is, a reformulation such as that described above is not needed. See Wolfe (1999).

Optimization problems with certain max-min constraints can also be converted to standard optimization problems by a method given by Kirjner and Polak (1998). Their method is applicable when the constraints have the form

$$\max_{k \in R} \min_{j \in Q_k} f_k^j(\mathbf{x})$$

where R is the set of integers $\{1, \cdots, r\}$ for some integer r and for any $k \in R$, Q_k is a set of integers $\{1, \cdots, q_k\}$.

18.2 INTEGER AND MIXED INTEGER PROBLEMS

The approach to the global optimization problem that we have used is applicable to problems in which some or all of the variables are required to take integer values only. In this section, we briefly discuss such problems.

As we have pointed out before, our approach to solving a global optimization problem is as follows: Begin with a box $\mathbf{x}^{\mathbf{I}(0)}$ in which the solution is sought. Delete subboxes that can be proved (using interval methods) not to contain the global solution. Continue until the remaining set of points is small.

For simplicity, assume that all variables must take integer values. First consider imposing this condition on the unconstrained problem. We can delete or reduce a subbox $\mathbf{x}^{\mathbf{I}}$ of $\mathbf{x}^{\mathbf{I}(0)}$ using the following procedures.

1. For any box $\mathbf{x}^{\mathbf{I}}$ generated during the application of the algorithm, reduce each interval component until its endpoints are integers.

2. If the i-th component of the gradient of f is positive (negative), replace the i-th component X_i of $\mathbf{x}^{\mathbf{I}}$ by the degenerate interval equal to the smallest (largest) integer in X_i. Compare the procedure in Section 12.4.

3. Generate sample values of the objective function f as in Section 12.5 to obtain an upper bound $\bar{f}$ on the global minimum f^*. Now, however, the sample point must have integer components. As in Section 12.5, delete subboxes of the current box $\mathbf{x}^{\mathbf{I}}$ where $f > \bar{f}$.

4. Use hull consistency and box consistency as discussed in Chapter 10.

We leave it to the reader to put together algorithms similar to those in earlier chapters for solving an integer optimization problem using the procedures listed above. Such an algorithm is not very efficient.

Another approach that works when the variables must be integers is the following: Add the constraints

$$\sin(\pi x_i) = 0 \ (i = 1, \cdots, n). \tag{18.2.1}$$

These constraints force the variables to be integers. The problem can now be treated as if there are no conditions that variables take integer values.

Solving an integer or mixed integer problem in this form can be a slow process. The constraints (18.2.1) are of little use unless the intervals bounding the variables have width less than (say) 1. However, the method produces the global solution.

REFERENCES

1. Alefeld, G. (1968). Intervallrechnung über den Komplexen Zahlen und einige Anwendungen, doctoral dissertation, University of Karlsruhe.

2. Alefeld, G., (1977), Das symmetrische Einzelshrittverfahren bei linearen Gleichungen mit Intervallen als Koeffizienten, Computing, 18, 329–340.

3. Alefeld, G. (1979). Intervallanalytische Methoden bei nichtlinearen Gleichungen, in *Jahrbuch Überlicke Mathematik 1979*, S.D. Chatterji et al. (eds.) Bibliographisches Institut, Mannheim, pp. 63–78.

4. Alefeld, G. (1981). Bounding the slope of polynomial operators and some applications, Computing 26, 227–237.

5. Alefeld, G. (1984). On the convergence of some interval arithmetic modifications of Newton's method, SIAM J. Numer. Anal. 21, 363–372.

6. Alefeld, G. (1999). Interval arithmetic tools for range approximation and inclusion of zeros, in Bulgar and Zenger (1999), pp. 1–21.

7. Alefeld, G. and Herzberger, J. (1970). ALGOL-60 Algorithmen zur Auflösung linearer Gleichungs-systeme mit Fehlerfassung, Computing 6,28–34.

8. Alefeld, G. and Herzberger, J. (1974). *Einführung in die Intervalrechnung,* Bibliographisches Institut, Mannheim.

9. Alefeld, G. and Herzberger, J. (1983). *Introduction to Interval Computations,* Academic Press, Orlando, Florida

10. Alefeld, G. and Rokne, J. (1984). On improving approximate triangular factorizations iteratively, Beitr. Numer. Math. 12, 7–20.

11. Bao, P. G. and Rokne, J. G. (1988). Low complexity k-dimensional Taylor forms, Appl. Math. Comput. 27, 265–280.

12. Bauer, W. and Strassen, V. (1983). The complexity of partial derivatives, Theoret. Comput. Sci. 22, 317–330.

13. Bazaraa, M. S. and Shetty, C. M. (1979). *Nonlinear Programming. Theory and Practice,* Wiley, New York.

14. Bliek, C. (1992). Computer methods for design automation, Ph. D. thesis, Dept. of Ocean Engineering, Massachusetts Inst. of Tech.

15. Bliek, C., Jermann, C., and Neumaier, A. (eds.) (2003). *Global optimization and Constraint Satisfaction: First International Workshop Global Constraint Optimization and Constraint Satisfaction, COCOS 2002 Valbonne-Sophia Antipolis, France, October 2–4, 2002.* Volume number: LNCS 2861.

16. Boche, R. (1966). Complex interval arithmetic with some applications, Lockheed Missiles and Space Co. Report 4-22-66-1.

17. Boggs, P. T., Byrd, R. H., and Schnabel, R. B. (eds.) (1985). *Numerical Optimization* 1984, SIAM Publications.

18. Broyden, C. G. (1971), The convergence of an algorithm for solving sparse nonlinear systems, Math. Comp., 25, 285–294.

19. Bulgar, H. and Zenger, C. (eds.). *Error Control and Adaptivity in Scientific Computing,* Kluwer, Netherlands.

20. Burke, J. (1990). On the identification of active constraints II: The nonconvex case, SIAM J. Numer Anal. 27, 1081–1102.

21. Caprani, O. and Madsen, K. (1980). Mean value forms in interval analysis, Computing 25, 147–154.

22. Casado, L, Garcia, I., and Sergeyev, Y. (2000). Interval branch and bound algorithm for finding the first-zero-crossing-point in one-dimensional functions, Reliable Computing 6, 179–191.

23. Collavizza, H., Delobel, F., and Rueher, M. (1999). Comparing partial consistencies, Reliable Computing, 5, 213–228.

24. Corliss, G. F. (1995). Personal Communication.

25. Corliss, G. F. and Rall, L. B., Bounding Derivative Ranges, in *Encyclopedia of Optimization*, Panos, M. P. and Floudas, C. A. (eds.), Kluwer, Dordrecht, (to appear).

26. Dantzig, G. and Eaves, B. C. (1975). Fourier-Motzkin elimination and its dual with applications to integer programming, in *Combinatorial Programming: Methods and Application,* Proceedings of the NATO Advanced Study Institute, B. Roy (ed.), Reidel, Dordrecht, Netherlands.

27. Deif, A. (1986). *Sensitivity Analysis in Linear Systems,* Springer-Verlag, Berlin.

28. Dennis, J. E., Jr. and Schnabel, R. B. (1983). *Numerical Methods for Unconstrained Optimization and Nonlinear Equations,* Prentice-Hall, Englewood Cliffs, New Jersey.

29. Dinkel, J. J. and Tretter, M. J. (1987). An interval arithmetic approach to sensitivity analysis in geometric programming, Oper. Res. 35, 859–866.

30. Dinkel, J. J. Tretter, M. J., and Wong, D. (1988). Interval Newton methods and perturbed problems, Appl. Math. Comput. 28, 211–222.

31. Dixon, L. C. W., and Szegö, G. P. (1975). *Towards Global* Optimization, North Holland/American Elsevier, New York.

32. Daumas, M. (2002). Past and future formalizations of the IEEE 754, 854 and 754R standards, Talk presented to the IEEE 754 committee. July 18, 2002 at Cupertino, CA. (See: `grouper.ieee.org/groups/754/meeting-materials/2002-07-18-daumas.pdf`)

33. Dwyer, P. S. (1951). Computation with approximate numbers, in *Linear Computations,* P. S. Dwyer (ed.), Wiley, New York, pp. 11–34.

34. Floudas C. A., and Pardalos, P. M. (eds.) (1992). *Recent Advances in Global Optimization*, Princeton University Press, Princeton, New Jersey.

35. Forte^{TM} Developer 6 Fortran 95. @ `www.sun.com/forte/` by Sun Microsystems Inc., July, 2000.

36. Forte^{TM} Developer 6 Update 1 C++. @ `www.sun.com/forte/` by Sun Microsystems Inc., October, 2000.

37. Forte^{TM} Developer 6 Update 1 Fortran 95. @ `www.sun.com/forte/` by Sun Microsystems Inc., October, 2000.

38. Frommer, A. and Mayer, G. (1990). On the R-order of Newton-like methods for enclosing solutions of nonlinear equations, SIAM J. Numer. Anal. 27,105–116.

39. Gargantini, I. (1975a). Parallel square root iterations, in Nickel (1975), pp. 195–204.

40. Gargantini, I. (1975b). Parallel Laguerre iterations, Numer. Math. 26, 317–323.

41. Gargantini, I. (1978). Further applications of circular arithmetic: Schroeder-like algorithms with error bounds for finding zeros of polynomials, SIAM J. Numer. Anal. 15, 497–510.

42. Gargantini, I. and Henrici, P. (1972). Circular arithmetic and the determination of polynomial zeros, Numer. Math. 18, 305–320.

43. Garloff, J. and Smith, A. P. (2000), Investigation of a subdivision based algorithm for solving systems of polynomial equations, Proc. of the 3rd World Congress of Nonlinear Analysis (WCNA 2000), Catania, Italy.

44. Glatz, G. (1975). Newton-Algorithmen zur Bestimmung von Polynomwurzeln unter Verwendung komplexer Kreisarithmetic, in Nickel (1975), pp. 205–214.

45. Griewank, A. (1989). On automatic differentiation, in *Mathematical Programming 88*, Kluwer Academic Publishers, Boston.

46. Griewank, A. (1991). The chain rule revisited in scientific computing, SIAM News, May.

47. Griewank, A. (2000). *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation,* SIAM Publ., Philadelphia.

48. Gustafson, J. L. (1994a). A Paradigm For Grand Challenge Performance Evaluation, (HTML version: `www.scl.ameslab.gov/Publications/Grand Challenge/Paradigm.html`), (PDF version: `www.scl.ameslab.gov/Publications/Paradigm.pdf`) *Proceedings of the Toward Teraflop Computing and New Grand Challenge Applications Mardi Gras '94 Conference*, Baton Rouge, Louisiana, February 1994.

49. Gustafson, J. L. (1994b). Teraflops and other false goals, *Parallel and Distributed Technology*, Summer 1994.

50. Gustafson, J. L. (1995). HINT—A New Way To Measure Computer Performance, (HTML version: `www.scl.ameslab.gov/Publications/HINT/ComputerPerformance.html`) (PDF version: `www.scl.ameslab.gov/Publications/HINT/HINT Acrobat.pdf`), in Gustafson and Snell (1995).

51. Gustafson, J. L. and Snell, Q. O. (1995). Proceedings of the HICSS-28 Conference, Wailela, Maui, Hawaii, January 3–6, 1995.

52. Gustafson, J. L. (1998). Computational verifiability of the ASCI program, *IEEE Computational Science and Engineering*, March-June 1998.

53. Hansen, E. R. (1965). Interval arithmetic in matrix computations, part I, SIAM J. Numer. Anal. 2, 308–320.

54. Hansen, E. R. (1968). On solving systems of equations using interval arithmetic, Math. Comput. 22, 374–384.

55. Hansen, E. R. (1969a). *Topics in Interval Analysis*, Oxford University Press, London.

56. Hansen, E. R. (1969b). On linear algebraic equations with interval coefficients, in Hansen (1969a), pp. 35–46.

57. Hansen, E. R. (1969c). On the centered form, in Hansen (1969a), pp. 102–106.

58. Hansen, E. R. (1975). A generalized interval arithmetic, in Nickel (1975), pp.7–18.

59. Hansen, E. R. (1978a). Interval forms of Newton's method, Computing 20, 153–163.

60. Hansen, E. R. (1978b). A globally convergent interval method for computing and bounding real roots, BIT 18, 415–424.

61. Hansen, E. R. (1979). Global optimization using interval analysis-the one dimensional case, J. Optim. Theory Appl. 29, 331–344.

62. Hansen, E. R. (1980). Global optimization using interval analysis-the multi-dimensional case, Numer. Math. 34, 247–270.

63. Hansen, E. R. (1984). Global optimization with data perturbations, Comput. Ops. Res. 11, 97–104.

64. Hansen, E. R. (1988). An overview of global optimization using interval analysis, in Moore (1988), pp. 289–307.

65. Hansen, E. R. (1991). Bounding the set of solutions of a perturbed global optimization problem, in *Proceedings of the Second Workshop on Global Optimization*, International Institute for Applied Systems Analysis.

66. Hansen, E. R. (1992). Bounding the solution of interval linear equations, SIAM J. Numer. Anal., 29, 1493–1503.

67. Hansen, E. R. (1993). Computing zeros of functions using generalized interval arithmetic, Interval Computations, 3, 3–28.

68. Hansen, E. R. (1997a). Preconditioning linearized equations, Computing, 58, 187–196.

69. Hansen, E. R. (1997b). Sharpness in interval computations, Reliable Computing, 3,17–29.

70. Hansen, E. R. (2000). The hull of preconditioned interval linear equations, Reliable Computing, 6, 95–103.

71. Hansen, E. R. (2002). Sharp solutions for interval linear equations (submitted).

72. Hansen, E. R. and Greenberg, R. I. (1983). An interval Newton method, appl. Math. Comput. 12, 89–98.

73. Hansen, E. R. and Sengupta, S. (1980). Global constrained optimization using interval analysis, in Nickel (1980), pp. 25–47.

74. Hansen, E. R. and Sengupta S. (1981). Bounding solutions of systems of equations using interval analysis, BIT 21, 203–211.

75. Hansen, E. R. and Sengupta, S. (1983). Summary and steps of a global nonlinear constrained optimization algorithm, Lockheed Missiles and Space Co. report no. D889778.

76. Hansen, E. and Smith, R. (1967). Interval arithmetic in matrix computations, Part 2, SIAM J. Numer. Anal., 4, 1–9.

77. Hansen, E. R. and Walster, G. W. (1982). Global optimization in nonlinear mixed integer problems, in *Proc. 10th IMACS World Conference on System Simulation and Scientific Computation*, vol I, pp. 397–381.

78. Hansen, E. R. and Walster, G. W. (1992a). Bounds for Lagrange multipliers and optimal points, in the Second Special Issue on Global Optimization, Control, and Games, Comput. Math. Appl., pp. 59–69.

79. Hansen, E. R. and Walster, G. W. (1992b). Nonlinear equations and optimization, in the Second Special Issue on Global Optimization, Control, and Games, Comput. Math. Appl.

80. Hansen, E. R. and Walster, G. W. (1992c). Equality constrained global optimization, submitted to SIAM Journal On Control and Optimization. Note: The authors finished this paper and submitted it for publication in 1987. It was to be published, but the authors have no record of this fact. The paper will be resubmitted. In the meantime, a preprint can be found at: www.cs.utep.edu/interval-comp/EqualityConstraints.pdf

81. Hansen, E. R. and Walster, G. W. (2002). Sharp bounds on interval polynomial roots. Reliable Computing, 8–2, 115–112.

82. Hansen, E. R. and Walster, G. W. (2003). Solving overdetermined systems of linear equations. Submitted to Reliable Computing.

83. Hanson, R. J. (1968). Interval arithmetic as a closed arithmetic system on a computer, Jet Propulsion Laboratory report 197.

84. Hartfiel, D. J. (1980). Concerning the solution of $Ax = b$ when $P \leq A \leq Q$ and $p \leq b \leq q$, Numer. Math. 35, 355–359.

85. Hebgen, M. (1974). Eine scaling Pivotsuche für Intervallmatrizen, Computing 12, 99–106.

86. Heindl, G., Kreinovich, V., and Lakeyev, A. (1998), Solving linear interval systems is NP-hard even if we exclude overflow and underflow, Reliable Computing, 4, 383–388.

87. Hennessy, J. L. and Patterson, D. A. (1994). *Computer Organization and Design*, Morgan Kaufmann, San Mateo, California.

88. Henrici, P. (1975). Einige Anwendungen der Kreisscheibenarithmetic in der Kettenbruchtheorie, in Nickel (1975), pp. 19–30.

89. Hickey, Q. J. and Van Emden M. H. (1999). Interval Arithmetic: From Principles to Implementation, Technical Report DCS-260-IR, Department of Computer Science, Victoria, B.C. Canada.

90. Hock, W. and Schittkowski, K. (1981). *Test Examples for Nonlinear Programming Codes*, Lecture Notes in Economics and Mathematical Systems, Springer-Verlag, New York.

91. IBM (1986a). IBM high accuracy arithmetic subroutine library (ACRITH). General information manual GC33-6163-02, third edition.

92. Ichida, K., and Fujii, Y. (1990). Multicriterion optimization using interval analysis, Computing 44, 47–57.

93. IEEE (1985). IEEE standard for binary floating-point arithmetic, ANSI/IEEE STD 754–1985 Technical Report, New York.

94. Iri, M. (1984). Simultaneous computation of functions, partial derivatives, and estimates of rounding errors-complexity and practicality, Jpn. J. Appl. Math. 1, 223–252.

95. Jacobs, D. (ed.) (1976). The state of the art in numerical analysis, in *Proc. Conference on the State of the Art in Numerical Analysis*, University of York.

96. Jaulin, L., Kieffer, M., Didrit, O., and Walter, É. (2001) *Applied Interval Analysis*, Springer-Verlag, London.

97. Kahan, W. M. (1968). A more complete interval arithmetic, Lecture notes for a summer course at the University of Michigan.

98. Kearfott, R. (1992). An interval branch and bound algorithm for bound constrained optimization problems, J. Global Optimiz., 2, 259–280.

99. Kearfott, R. (1996). *Rigorous Global Search: Continuous Problems,* Kluwer Academic Publ., Dordrecht.

100. Kearfott, R. B. and Dian, J. (2000). Existence verification for higher-degree singular zeros of complex nonlinear systems, preprint, `interval.louisiana.edu.cplx.0302.pdf`.

101. Kirjner, C. and Polak, E. (1998). On the conversion of optimization problems with max-min constraints to standard optimization problems, SIAM J. Optim., 8, 887–915.

102. Krawczyk, R. (1969). Newton-Algorithmen zur Bestimmung von Nullstellen mit Fehlershranken, Computing 4, 187–201.

103. Krawczyk, R. (1983). A remark about the convergence of interval sequences, Computing 31, 255–259.

104. Krawczyk, R. (1986). A class of interval Newton operators, Computing 37, 179–183.

105. Krawczyk, R., and Neumaier, A. (1985). Interval slopes for rational functions and associated centered forms, SIAM J. Numer. Anal. 22, 604–616.

106. Kreier, N. (1974). Komplexe Kreisarithmetic, Z. Angew. Math. Mech. 54, 225–226.

107. Kreier, N., and Spellucci, P. (1975). Einschliessungsmengen von Polynom-Nullstellen, in Nickel (1975), pp.223–229.

108. Kreinovich, V., Lakeyev, A., Rohn, J., and Kahl, P., *Computational complexity and feasibility of data processing and interval computations*, Kluwer, Dordrecht, 1998.

109. Kulisch, U. (ed.) (1987). *Computer Arithmetic, Scientific Computation and Programming Languages*, Teubner, Stuttgart.

110. Kulisch, U., and Miranker, W. L. (eds.) (1983). *A New Approach to Scientific Computation*, Academic Press, Orlando, Florida.

111. Lemaréchal, C. (1982). Nondifferentiable optimization, in Powell (1982), pp. 85–89.

112. Levy, A. V., and Gomez, S. (1985). The tunneling method applied to global optimization, in Boggs, Byrd, and Schnabel (1985), pp. 213–244.

113. Levy, A. V., Montalvo, A., Gomez, S., and Calderon, A. (1981). *Topics in Global Optimization*, Lecture Notes in Mathematics No.909, Springer-Verlag, New York.

114. Loh, E. and Walster, G. W. (2002). Rump's example revisited. Reliable Computing, 8 (2) 245–248.

115. Mancini, L. J., (1975). Applications of interval arithmetic in signomial programming, Technical report SOL 75–23, Systems Optimization Laboratory, Department of Operations Research, Stanford University.

116. Mancini, L. J., and McCormick, G. P. (1976). Bounding global minima, Math. Oper. Res. 1, 50–53.

117. McAllester, D., Van Hentenryck, P., and Kapur, D. (1995), Three cuts for accelerated interval propagation, Massachusetts Inst. of Tech. Artificial Intelligence Lab. memo no. 1542.

118. Metropolis, N., et al. (eds.) (1980). *A History of Computing in the 20th Century*, Academic Press, New York.

119. Mohd, I. B., (1990). Unconstrained global optimization using strict complementary slackness, Appl. Math. Comput. 36, 75–87.

120. Moore, R. E. (1965). The automatic analysis and control of error in digital computation based on the use of interval numbers, in Rall (1965), pp. 61–130.

121. Moore, R. E. (1966).*Interval Analysis*, Prentice-Hall, Englewood Cliffs, New Jersey.

122. Moore, R. E. (1977). A test for existence of solutions of nonlinear systems, SIAM J. Numer. Anal. 14, 611–615.

123. Moore, R. E. (1979).*Methods and Applications of Interval Analysis*, SIAM Publ., Philadelphia, Pennsylvania.

124. Moore, R. E. (1980). Microprogrammed interval arithmetic, ACM SIGNUM Newsletter. 15(2), p. 30.

125. Moore, R. E. (ed.) (1988).*Reliability in Computing*, Academic Press, San Diego.

126. Moore, R. E. (1991). Global optimization to prescribed accuracy, Comput Math. Appl. 21, 25–39.

127. Moore, R. E., Hansen, E. R,, and Leclerc, A. (1992). Rigorous methods for parallel global optimization, in Floudas and Pardalos (1992), pp.321–342.

128. Moré, J. J., Garbow, B. S., and Hillstrom, K. E. (1981). Testing unconstrained optimization software, ACM Trans. Math. Software 7, 17–41.

129. Nataraj, P. S. V. and Sardar, G. (2000). Computation of QFT bounds for robust sensitivity and gain-phase margin specifications, Trans. ASME Journal of Dynamical Systems, Measurement, and Control, Vol. 122, pp. 528–534.

130. Nataraj, P. S. V. (2002a). Computation of QFT bounds for robust tracking specifications, Automatica, Vol. 38, pp. 327–334.

131. Nataraj, P. S. V. (2002b). Interval QFT: A mathematical and computational enhancement of QFT, Int. Jl. Robust and Nonlinear Control, Vol. 12, pp. 385–402.

132. Neumaier, A. (1981). Interval norms, Inst. angew. Math. report 81/5, University of Freiburg.

133. Neumaier, A. (1985). Interval iteration for zeros of systems of equations, BIT 25, 256–273.

134. Neumaier, A. (1986). Linear interval equations, in Nickel (1986), pp. 114–120.

135. Neumaier, A. (1988). The enclosure of solutions of parameter-dependent systems of equations, in Moore (1988), pp. 269–286.

136. Neumaier, A. (1990). *Interval Methods for System of Equations*, Cambridge University Press, London.

137. Neumaier, A. (1999). A simple derivation of the Hansen-Bliek-Rohn-Ning-Kearfott enclosure for interval linear equations, Reliable Computing, 6, 131–136.

138. Neumaier, A. (2000). Erratum to: A simple derivation of the Hansen-Bliek-Rohn-Ning-Kearfott enclosure for interval linear equations, Reliable Computing, 6, 227.

139. Nickel, K. L. (1969). Zeros of polynomials and other topics, in Hansen (1969a), pp. 25–34.

140. Nickel, K. L. (1971). On the Newton method in interval analysis, University of Wisconsin, Mathematics Research Center report 1136.

141. Nickel, K. L. (ed.) (1975). *Interval Mathematics*, Springer-Verlag, New York.

142. Nickel, K. L. (1976). Interval-Analysis, in Jacobs (1976), pp. 193–225.

143. Nickel, K. (1977). Die Überschätzung des Wertebereichs einer Function in der Intervallrechnumg mit Anwendungen auf lineare Gleichungssysteme, Computing, 18, 15–36.

144. Nickel, K. L. (ed.) (1980). *Interval Mathematics 1980*, Academic Press, New York.

145. Nickel, K. L. (ed.) (1986). *Interval Mathematics 1985*, Springer Lecture Notes in Computer Science No. 212, Springer-Verlag, New York.

146. Nickel, K. L., and Ritter, K. (1972). Termination criteria and numerical convergence, SIAM J, Numer. Anal. 9, 277–283.

147. Ning, S. and Kearfott, R. (1997). A comparison of some methods for solving linear interval equations, SIAM J. Numer. Anal., 34, 1289–1305.

148. Oettli, H. (1965). On the solution set of a linear system with inaccurate coefficients, SIAM J. Numer. Anal. 2, 115–118.

149. Oettli, H., Prager, H., and Wilkinson, J. H. (1965). Admissible solutions of linear systems with not sharply defined coefficients, SIAM J. Numer. Anal. 2, 291–299.

150. Palmer, J. F. and Morse, S. P. (1984). *The 8087 Primer*, Wiley Press, New York.

151. Pintér, J. (1990). Solving nonlinear equation systems via global partition and search: some experimental results, Computing 43, 309–323.

152. Powell, M. J. D. (ed.) (1982).*Nonlinear Optimization, 1981*, Academic Press, New York.

153. Qi. L. (1982). A note on the Moore test for nonlinear systems, SIAM J. Numer. Anal. 19, 851–857.

154. Rall, L. B. (1965). *Error in Digital Computation*, Vol. I, Wiley, New York.

155. Rall, L. B. (1969). *Computational Solution of Nonlinear Operator Equations*, Wiley, New York.

156. Rall, L. B. (1981).*Automatic Differentiation-Techniques and Applications*, Springer Lecture Notes in Computer Science, Vol. 120, Springer-Verlag, New York.

157. Rall, L. B. (1983). Differentiation and generation of Taylor coefficients in Pascal-SC, in Kulisch and Miranker (1983), pp. 291–309.

158. Rall, L. B. (1984). Differentiation in PASCAL-SC: type gradient, ACM Trans. Math. Software 10, 161–184.

159. Rall, L. B. (1987). Optimal implementation of differentiation arithmetic, in Kulisch (1987).

160. Ratschek, H., and Rokne, J. (1984). *Computer Methods for the Range of Functions*, Halstead Press, New York.

161. Ratschek, H., and Rokne, J. (1988). *New Computer Methods for Global Optimization*, Wiley, New York.

162. Ratz, D. (1994). Box splitting strategies for the interval Gauss-Seidel step in a global optimization method, Computing, 53, 337–353.

163. Ratz, D. (1998). *Automatic Slope Computation and Its Application In Nonsmooth Global Optimization*, Shaker Verlag, Aachen.

164. Ratz, D. (1999). A nonsmooth global optimization technique using slopes — the one-dimensional case. Jour. Global Optim. 14, 365–393

165. Reichmann, K. (1979). Abbruch beim Intervall-Gauss-Algorithmus, Computing 22, 355–362.

166. Rigal, J. L., and Gaches, J. (1967). On the compatibility of a given solution with the data of a linear system, J. Assoc. Comput. Mach. 14, 543–548.

167. Ris, F. N. (1972). Interval analysis and applications to linear algebra, Ph.D. dissertation, Oxford University.

168. Ris, F. N. (1975). Tools for the analysis of interval arithmetic, in Nickel (1975), pp. 75–98.

169. Robinson, S. M. (1973). Computable error bounds for nonlinear programming, Math. Programming 5, 235–242.

170. Rohn, J. (1993). Cheap and tight bounds: The recent result of E. Hansen can be made more efficient, Interval Computations, 4, 13–21.

171. Rokne, J. G. (1986). Low complexity k-dimensional centered forms, computing 37, 247–253.

172. Rokne, J. G., and Bao, P. (1987). Interval Taylor forms, Computing 39, 247–259.

173. Rump, S. M. (1988). Algorithm for verified inclusions-theory and practice, in Moore (1988), pp. 109–126.

174. Rump, S. M. (1990). Rigorous sensitivity analysis for systems of linear and nonlinear equations, Math. Comput. 54, 721–736.

175. Rump, S. M. (1996). Expansion and estimation of the range of nonlinear functions, Math. Comp., 65, 1503–1512.

176. Schwefel, H. (1981).*Numerical Optimization of Computer Models,* Wiley, new York.

177. Sengupta, S. (1981). Global nonlinear constrained optimization, Ph.D. dissertation, Washington State University.

178. Shary, S. (1995), On optimal solution of interval linear equations, SIAM J. Numer. Anal., 32, 610–630.

179. Shary, S. (1999). Outer estimation of generalized solution sets to interval linear systems, Reliable Computing, 5, 323–335.

180. Shearer, J. M., and Wolfe, M. A. (1985a). some computable existence, uniqueness, and convergence tests for nonlinear systems, SIAM J. Numer. Anal. 22, 1200–1207.

181. Shearer, J. M., and Wolfe, M. A. (1985b). An improved form of the Krawczyk-Moore algorithm, Appl. Math. Comput. 17, 229–239.

182. Speelpenning, B. (1980). Compiling fast partial derivatives of functions given by algorithms, Ph.D. dissertation, Department of Computer Science, University of Illinois at Urbana-Champaign.

183. Stewart, G. W. (1973). *Introduction to Matrix Computations*, Academic Press, New York.

184. Sunaga, T. (1958). Theory of interval algebra and its application to numerical analysis, RAAG Memoirs 3, 29–46.

185. Van Hentenryck, P., McAllester, D., and Kapur, D. (1997), Solving Polynomial systems using branch and prune approach, SIAM J. Numer. Anal., 34, 797–827.

186. Van Hentenryck, P., Michel, L., and Deville, Y. (1997), *Numerica. A Modeling Language for Global Optimization,* The MIT Press, Cambridge.

187. Walster, G. W. (1988). Philosophy and practicalities of interval arithmetic, pages 309–323 of Moore (1988).

188. Walster, G. W. (1996). Interval Arithmetic: The New Floating-Point Arithmetic Paradigm, Sun Solaris Technical Conference, June 1996.

189. Walster, G. W. (2000a). The “Simple” closed interval system, Technical report, Sun Microsystems Inc., February, 2000.

190. Walster, G. W. (2000b). Interval arithmetic programming reference: Forte™ workshop 6 Fortran 95. Sun Microsystems Inc., May 2000.

191. Walster, G. W. (2000c). Interval arithmetic programming reference: Forte™ workshop 6 update 1 Fortran 95. Sun Microsystems Inc., Nov., 2000.

192. Walster, G. W. (2002). Implementing the “Simple” Closed Interval System. See:
www.sun.com/software/sundev/whitepapers/simple-implementation.pdf

193. Walster, G. W. (2003a). Exterior interval continuity, to be submitted for publication.

194. Walster, G. W. (2003b). The containment set based interval arithmetic standard, to be submitted for publication.

195. Walster, G. W. and Chiriaev, D. (2000). Interval arithmetic programming reference: Forte™ workshop 6 update 1 C++. Sun Microsystems Inc., Nov, 2000.

196. Walster, G. W. and Hansen, E. R. (1997). Interval algebra, composite functions, and dependence in compilers, Technical Report, Sun Microsystems Inc.

197. Walster, G. W. and Hansen, E. R. (1998). Composite functions in interval mathematics. Preprint available at:
www.mscs.mu.edu/~globsol/readings.html#Walster
March, 1998.

198. Walster, G. W. and Hansen, E. R. (2003). Computing interval parameter bounds from fallible measurements using overdetermined (tall) systems of nonlinear equations, in Bliek, C., *et. al.* (2003)

199. Walster, G. W. and Hansen, E. R. (2004). Using pillow functions to efficiently compute crude range tests, in the SCAN 2000 Proceedings to appear in Num. Alg.

200. Walster, G. W., Hansen, E. R., and Pryce, J.D. (2000). Extended real intervals and the topological closure of extended real relations, Technical Report, Sun Microsystems Inc., February, 2000.

201. Walster, G. W., Hansen, E. R., and Sengupta, S. (1985). Test results for a global optimization algorithm, in Boggs, Byrd, and Schnabel (1985), pp. 272–287.

202. Walster, G. W. and Kreinovich, V. (2003). Computational complexity of optimization and crude range testing: a new approach motivated by fuzzy optimization, Fuzzy Sets and Systems, 9, 179–208.

203. Walster, G. W., Liddiard, L. and Tretter, M. J. (1980). INTLIB User Manual (unpublished).

204. Walster, G., Pryce, J.D., and Hansen, E. R. (2002). Practical, exception-free interval arithmetic on the extended reals. SIAM Journal on Scientific Computing. The paper was provisionally accepted for publication; has since been significantly revised; and will be resubmitted for publication.

205. Warmus, M. (1956). Calculus of approximations, Bull. de l'Academie Polonaise des Sciences, 4, 253–259.

206. Warmus, M. (1960). Approximations and inequalities in the calculus of approximations. Classification of approximate numbers, Bull. de l'Academie Polonaise des Sciences, 9, 241–245.

207. Watson, L. T. (1986). Numerical linear algebra aspects of globally convergent homotopy methods, SIAM Rev. 28, 529–545.

208. Wilkinson, J. H. (1965). *The Algebraic Eigenvalue Problem*, Oxford University Press, London.

209. Wilkinson, J. H. (1980). Turing's work at the NPL and the Pilot ACE, DEUCE, and ACE, in Metropolis, et al. (1980).

210. Wolfe, M. A.(1999). On discrete minimax problems in R using interval arithmetic, Reliable Computing 5, 371–383.

211. Wolfe, P. (1982). Checking the calculation of gradients, ACM Trans. Math. Software 6, 337–343.

212. Wongwises, P. (1975a). Experimentelle Untersuchungen zur numerische Auflösungen von linearen Gleichungssystemen mit Fehlershranken, dissertation, Instituts für Praktische Mathematik, report 75/1, University of Karlsruhe.

213. Wongwises, P. (1975b). Experimentelle Untersuchungen zur numerische Auflösungen von linearen Gleichungssystemen mit Fehlershranken, in Nickel (1975), pp. 316–325.

214. Yohe, J. M. (1979). Implementing nonstandard arithmetic, SIAM Rev. 21, 34–56.

215. Zuhe, S., and Wolfe, M. A. (1990). On interval enclosures using slope arithmetic, Appl. Math. Comput. 39, 89–105.