Lazy exact real computation

Langley, Simon

Award date:
2005

Awarding institution:
University of Bath

LAZY EXACT REAL COMPUTATION

Submitted by Simon Langley
for the degree of
Doctor of Philosophy
of the University of Bath
January 2005

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

Summary

Given a box in $\mathbb{R}^n$ containing an isolated root ξ of a set, S, of polynomials in $\mathbb{Z}[x_1, \ldots, x_n]$ how can the sign of a $q(\xi)$ be determined where $q \in \mathbb{Z}[x_1, \ldots, x_n]$? An effective method for solving this problem would have applications in graphics programming and in several areas of computer algebra.

A number of algorithms are presented which solve the problem in both general and special cases (e.g. univariate polynomials, triangular systems, etc). The emphasis is on finding 'gap functions', i.e. functions $K(S, q, \xi)$ such that $|q(\xi)| < K(S, q, \xi)$ implies $q(\xi) \neq 0$. If $q(\xi) \neq 0$ its sign can be found by numerical means.

The numerical techniques make considerable use of interval arithmetic and include an algorithm for determining whether a given box does contain exactly one isolated solution of S.

An extension of the problem to 'elementary expressions' which include log and exp terms is also discussed and some conjectures and experimental findings presented.

Contents

Acknowledgements

Prof Ken Jukes at the University of the West of England agreed to my doing this work despite it greatly diverging from the general research direction of UWE. He, and his successors in the Faculty of Computing and Mathematical Sciences, were generous in providing time and resources throughout.

Dr Dan Richardson was the ideal supervisor and is a good friend. Hopefully, as we both approach retirement we will have more time to sit in the BTP, drink coffee and just talk - without thinking too much about mathematics.

Finally, this thesis would never have been written without the unwavering and unconditional support of my wife, Eleanor. She was understanding and encouraging throughout the eight years it took to do and in the last few months especially provided the environment that ensured it was finally completed. I owe her more that I can say for this and for everything.

Chapter 1

Lazy Exact Real Computation

1.1 Exact Computation

The aim of this thesis is to describe practical algorithms for the implementation of exact computation with subsets of the real numbers. Given some subset $\mathbb{K}$ of the real numbers[1], we interpret the term *exact computation* to refer to the possibility of defining algorithms for numbers in $\mathbb{K}$ including at least the following:

1. (Approximation) Provide approximate numerical representations of numbers to arbitrary precision, i.e. given $\epsilon > 0$ and a representation of $\alpha \in \mathbb{K}$ find $f \in \mathbb{Q}$ such that $|f - \alpha| < \epsilon$.

2. (Field Operations) Given representations for $\alpha, \beta \in \mathbb{K}$, find a representation for $\alpha + \beta$, $\alpha - \beta$, $\alpha \times \beta$, α^r where $r \in \mathbb{Q}$ (assuming the result to be real and taken to be positive where possible) and α/β (if $\beta \neq 0$).

3. (Sign Determination) Decide if $\alpha > \beta$, $\alpha = \beta$ or $\alpha < \beta$. Equivalently, determine the sign of $\alpha - \beta$.

[1] References to 'reals' should always be taken as shorthand for 'computable reals'

The term *exact arithmetic* is used here for algorithms and/or systems meeting the first two requirements; the term *exact computation* is reserved for when the last requirement is also met.

Exact computation is clearly possible in $\mathbb{Z}$ and $\mathbb{Q}$. On the other hand, from the time of Turing's earliest papers [Tur36, Tur37], it has been known that the third requirement, sign determination, is generally undecidable. Thus any theorem containing such statements as 'if $a = 0 \ldots$', 'if $a > b \ldots$' or 'if $a \neq b \ldots$' fails to be constructive if a and b are arbitrary reals. Equivalently, and of greater practical importance, algorithms cannot depend upon decisions about the equality or inequality of real numbers.

One response to the general undecidability of sign determination has been the development of algorithms for exact arithmetic with sign determination replaced by an approximate equality test $=_\epsilon$ which compares values to within some predefined tolerance ϵ. Clearly any system supporting the first two criteria of exact computation can support $=_\epsilon$ testing. Examples include iRRam [Mül00] and XR (an implementation can be found at [Bri02]), while programming libraries such LEDA [Alg] and LiDIA [LiD] provide data types with similar properties.

A less common response, the one adopted here, is to seek algorithms for classes of number beyond $\mathbb{Q}$ for which exact computation is possible. That it is possible in $\mathbb{A}$ (the algebraic numbers) has been known for many years, though few good algorithms exist. An important subset of the algebraic numbers are those defined by *nested radical expressions* (e.g. $\sqrt[3]{1+\sqrt{3/7}}$) which are of particular relevance to geometric computations. Exact [Fv93] provides an early implementation of sign determination in geometric computation. The more recent CORE library for computational geometry [Yap] provides a more sophisticated zero testing en-

vironment.

Papers by [Cav70] and later by Richardson [Ric97] and MacIntyre & Wilkie [MW96] show that (assuming the correctness of Schanuel's conjecture) sign determination can be extended to the elementary numbers (the closure of the algebraic numbers under $\exp(\cdot)$). Richardson's proof is constructive and provides an algorithm (which may fail to terminate in the case that Schanuel's conjecture is false) for deciding if an elementary constant is zero.

Schanuel's conjecture states

Conjecture 1. *If* $\alpha_1, \ldots, \alpha_n \in \mathbb{C}$ *are linearly independent over* $\mathbb{Q}$ *then the transcendence degree of* $\mathbb{Q}[\alpha_1, \ldots, \alpha_n, e^{\alpha_1}, \ldots, e^{\alpha_n}] : \mathbb{Q}$ *is at least* n.

The conjecture, if true, implies several important results (including the algebraic independence of π and e). Unfortunately it seems particularly resistant to proof or dis-proof [Ax71]. It is uncertain for what further sets of reals sign determination is possible.

Even if we assume Schanuel's conjecture, we do not have a practical way to solve the fundamental problem. That is, we do not have any solution method whose complexity is polynomial in the (bit) size of the input, and many people believe that no such solution method exists. On the other hand, if the problem is truly intractable, there is a remarkable scarcity of difficult specific examples.

Nevertheless, the problem of deciding whether a constant expression is zero is important to many areas of mathematics, especially computer algebra. Van der Hoeven [vdH00] gives some examples including: simplifying expressions e.g. $\sqrt{9+4\sqrt{2}}-2\sqrt{2}=1$ and deciding if expressions are defined e.g. $\log(\sqrt{9+4\sqrt{2}}-$

$1 - 2\sqrt{2}$). To these we could add the general comments above and, as additional examples: finding limits automatically, e.g. the behaviour of $\lim_{x\to\infty} A + Bx$ depends on whether $B = 0$ and deciding if over constrained equations have solutions, e.g. is there a solution to $x_1^2 - 2 = 0, x_2^2 - 9 - 4x_1 = 0, x_2 - 2x_1 - 1 = 0$? (decide if any two equations have a common root then decide if the third is zero at that point).

Another application occurs as part of quantifier elimination in semi-algebraic sets, most familiarly within Collins' CAD algorithm [ACM84]. See also [Ped91b] and [Can91] for sign determination methods used as part of CAD.

In this thesis therefore we describe algorithms which could be used for those classes for which the problem is known to be solvable including the discussion of some practically useful subsets (such as nested radical and exp-log expressions).

1.2 A Model of Exact Computation

Following the ideas of Richardson [Ric97, Ric96a, Ric96b, Ric99a] we define exact computation more precisely in the following way.

Definition 1. *An* n-box $\mathbf{B} \subset \mathbb{R}^n$ *is the Cartesian product of* n *closed intervals with rational endpoints. The* width *of a box,* $w(\mathbf{B})$ *is the maximum width of its intervals and* $m(\mathbf{B})$ *denotes the (rational) vector composed of the midpoints of the intervals*[2].

Definition 2. *Let* $\mathbf{B}$ *be an* n*-box and let* $F :\subset \mathbb{R}^n \to \mathbb{R}^n$ *be continuous over* $\mathbf{B}$ *with continuous derivatives then if there exists exactly one point* $\xi \in \mathbf{B}$ *at which*

[2]A fuller table of notation used for boxes and intervals is given in §2.2.1 where other constructs are needed.

the Jacobian $J(F,\xi)$ is non-singular and $F(\xi) = 0$ then $F_{\mathbf{B}}$ (the restriction of F to $\mathbf{B}$) defines ξ.

The focus of this thesis will be exact computation over $\mathbb{Z}(\xi)$ the set of rational expressions in components of ξ. Strictly, since $\xi = (\xi_1, \ldots, \xi_n)$ is only given implicitly, we should, in a constructive context, view the ξ_i as indeterminates rather than 'values'. This can be confusing and instead we take the elements of $\mathbb{Z}(\xi)$ to be represented by corresponding elements of $\mathbb{Z}(x_1, \ldots, x_n)$. In general it is useful to have a notation which emphasises the structure of the representation rather than the value and so $q \in \mathbb{Z}(x_1, \ldots, x_n)$ will represent the value expressed as $q(\xi)$.

Definition 3. *An* exact real computation system *is a 4-tuple $(\mathcal{F}, \mathcal{V}, \mathcal{A}, \mathcal{Z})$ where $\mathcal{F}$ be a countable set of functions from $\mathbb{R}^n$ to $\mathbb{R}$ and $\mathcal{V}, \mathcal{A}$ and $\mathcal{Z}$ are algorithms such that:*

1. *(Validation) Given a function $F \in \mathcal{F}^n$ and an n-box $\mathbf{B}$, $\mathcal{V}(F, \mathbf{B})$ determines whether $F_{\mathbf{B}}$ defines a point.*

2. *(Approximation) If $F_{\mathbf{B}}$ defines a point ξ say, then given any $\epsilon > 0$, $\mathcal{A}(F, \mathbf{B}, \epsilon)$ returns $\mathbf{C}$ such that $\xi \in \mathbf{C} \subset \mathbf{B}$ and $w(\mathbf{C}) < \epsilon$.*

3. *(Zero Testing) If $F_{\mathbf{B}}$ defines ξ then given $q \in \mathbb{Z}(x_1, \ldots, x_n)$, $\mathcal{Z}(F, \mathbf{B}, q)$ decides if $q(\xi) = 0$.*

The first algorithm is required because of the general method used to define ξ. In general, given F and $\mathbf{B}$ it will not be obvious whether $F_{\mathbf{B}}$ is a definition of a point. An alternative approach is to assume the function has the required property and continue from there. But at some point such verification is needed and it seems

appropriate to include it here. This requirement might be very simple in some cases: for example if $F = (a_1x_1 - b_1, \ldots a_nx_n - b_n)$ with $a_i, b_i \in \mathbb{Z}$. It is of course possible to define functions whose validation would be undecidable, see for example [Ric68].

The need for the second two algorithms is directly related to the requirements at the head of the chapter.

Proposition 1. *An exact real computation system as defined above is capable of meeting the requirements for exact computation given at the start of this chapter for* $\mathbb{K} = \mathbb{Z}(\xi)$ *with each* $\alpha \in \mathbb{K}$ *represented by the the corresponding expression in* $\mathbb{Z}[x_1, \ldots, x_n]$ *if* $\mathbb{Z}[x_1, \ldots, x_n] \subset \mathcal{F}$.

Proof. 1. (Approximation) Using $\mathcal{A}$, for any $\delta > 0$ a box $\mathbf{C}$ can be found containing ξ with $w(\mathbf{C}) < 2\delta$. ξ can be approximated by $m(\mathbf{C})$ (the vector of mid-points of the intervals comprising $\mathbf{C}$) with accuracy $\leq \pm\delta$. For any $\alpha \in \mathbb{Z}(x_1, \ldots, x_n)$, $v = \alpha(m(\mathbf{C}))$ gives an approximation to $\alpha(\xi)$. Clearly $|\alpha(\xi) - v| < \epsilon$ for some δ.

2. (Field Operations) For $\alpha, \beta \in \mathbb{Z}(x_1, \ldots, x_n)$ so are $\alpha + \beta$, $\alpha - \beta$, $\alpha \times \beta$ and α/β. For the last case algorithm $\mathcal{Z}$ can decide if $\beta(\xi) = 0$ and thus whether α/β is a valid representation.

   For α^r it is sufficient to consider the case where $r = 1/m$ and $\alpha \in \mathbb{Z}[x_1, \ldots, x_n]$. Replace $F \equiv F(x_1, \ldots, x_n) \equiv (f_1, \ldots, f_n)$ by $F' = (f_1, \ldots, f_n, x_{n+1}^m - \alpha)$ and augment $\mathbf{B}$ with an interval containing $\alpha(\xi)^{1/m}$ (which can be done by rational arithmetic).

3. (Sign Determination) $\mathcal{Z}$ can decide if $\delta = 0$. If not, let v_i be a sequence of approximations to $\delta(\xi)$ such that $|\delta(\xi) - v_i| < 2^{-i}$. Such a sequence exists

from part 1 of the proof. Since $\delta(\xi) \neq 0$, for some i it must be the case that $0 \notin [v_i - 2^{-i}, v_i + 2^{-i}]$. The sign of $\delta(\xi)$ is the same as that of v_i.

□

The model described above may be called 'lazy' computation in the sense that no attempt is made to produce a simpler representation of, say, $\alpha(\xi) + \beta(\xi)$ than the expression $\alpha + \beta$. Ideally we should like algebraic computation to be lazy in another sense which is described in the next section.

1.3 Lazy Algebraic Computation

The 'standard' representation of a real algebraic number is its minimal polynomial together with an interval identifying which root is being specified. In this case, $F = (p_1(x_1), \ldots, p_n(x_n))$ where the p_i are the minimal polynomials.

Of course univariate defining polynomials are often not 'given' as part of a problem. For example, the first quadrant intersection of $x^2 + y^2 = 4$ and $x + y = 1$ defines a pair of real algebraic numbers. Such implicit definitions arise quite naturally, and though it is possible to extract univariate polynomials which define the same numbers, it adds a further overhead even before any arithmetic is done.

The standard model is also difficult to extend to the elementary number case. It is not known whether a set of n polynomials in $\mathbb{Z}[x_1, \ldots, x_n, \exp(x_1), \ldots, \exp(x_n)]$ can be reduced to n univariate forms (each a polynomial in some x_i and $\exp(x_i)$), for example[3].

[3]Discussion with Dr Richardson

The ideal scenario would require only that the function defining ξ is a general polynomial mapping. Current experience suggests that this is an unrealistic hope, especially if, as implied here, the aim is to do repeated calculations with the ξ_i. In such a case, for all but the smallest systems, some 'pre-processing' may convert F to a different form which enables subsequent computations to be carried out far more efficiently. For example, a Gröbner basis for the polynomials could be computed - this has the advantage that it can be adapted to deal with the elementary case also. Alternatively, in the purely algebraic case, some linear combination ϕ of the ξ_i will be a primitive element of the field $\mathbb{Q}[\xi_1, \ldots, \xi_n]$ allowing each ξ_i to be represented as a rational function of ϕ.

1.4 Methodology

Symbolic manipulation tends to be slow on computers whereas arithmetic is fast. Again following Richardson [Ric96b], we use of a combination of algebraic and numerical methods. Our model of exact computation assumes three algorithms. The next chapter addresses the first two problems: deciding whether a function F and a box $\mathbf{B}$ defines a point and providing an efficient way of approximating a value. Both of these are solved by primarily numerical methods.

How to decide, for $q \in \mathbb{Z}(x_1, \ldots, x_n)$, whether $q(\xi) = 0$, the third algorithm of our model is the major burden of this report.

In the purely algebraic case there are a spectrum of approaches which might be used. At one extreme it is possible to determine a polynomial $\chi(z)$ such that $\chi(q(\xi)) = 0$. A necessary condition for $q(\xi) = 0$ is that $\chi(0) = 0$, sufficiency is achieved by showing that no other root of χ could correspond the value of $q(\xi)$.

One step toward a numerically based solution is to find not χ, but estimates of its coefficients. A knowledge of the upper bounds of these is sufficient to provide a lower bound for any non-zero root and from this numerical methods alone are sufficient. The practical difficulty encountered is that such estimates can give very poor bounds and lead to very large floating point mantissæ being required.

An ideal scenario would be to bypass χ and use a purely numerical approach. Since $J(F, \xi)$ is non-singular, ξ is an isolated root of F. Thus either $q(\xi) = 0$ or there exists $L > 0$ such that $|q(\xi)| \geq L$.

Functions yielding bounds such L above are called *gap functions* or *witness conjectures* [vdH00], i.e. a gap function is a mapping G such that $0 < G(F, B, q) \leq |q(\xi)|$. Equivalently, $|\log_2(G(F, B, q))|$ is a measure of the number of bits necessary to distinguish $q(\xi)$ from zero. Unfortunately few gap functions are known and those that are almost always badly underestimate the actual bound.

In the context of this thesis, a modified definition of exact computation is appropriate. The version used here will be:

Definition 4. *An* exact real algebraic computation system *is a 4-tuple* $(\mathcal{F}, \mathcal{V}, \mathcal{A}, \mathcal{Z})$ *where* $\mathcal{F} = \mathbb{Z}[x_1, \ldots, x_n]$ *and* $\mathcal{V}, \mathcal{A}$ *and* $\mathcal{Z}$ *are algorithms such that*

1. *(Validation) Given a function* $F \in \mathcal{F}^n$ *and an* n*-box* $\mathbf{B}$, $\mathcal{V}(F, \mathbf{B})$ *determines whether* $F_{\mathbf{B}}$ *defines a point.*

2. *(Approximation) If* $F_{\mathbf{B}}$ *defines a point* ξ *say, then given any* $\epsilon > 0$, $\mathcal{A}(F, \mathbf{B}, \epsilon)$ *returns* $\mathbf{C}$ *such that* $\xi \in \mathbf{C} \subset \mathbf{B}$ *and* $w(\mathbf{C}) < \epsilon$.

3. *(Bounding) If* $F_{\mathbf{B}}$ *defines* ξ *then given* $q \in \mathbb{Z}(x_1, \ldots, x_n)$, $\mathcal{Z}(F, \mathbf{B}, q)$ *returns a value* $L \in \mathbb{Q}$ *such that* $|q(\xi)| < L$ *implies* $q(\xi) = 0$.

The restriction of $\mathcal{F}$ to polynomials is shown in Chapter 2 to provide a possible validation algorithm. In fact the algorithm is more general Chapter 2 shows how it may be extended.

It is not necessary that a polynomial map have only isolated zeros, since an additional polynomial $1 - x_{n+1} \det J(F)$ can be appended to the map to restrict the solutions to isolated points (note that $F_{\mathbf{B}}$ defining a point does not not necessarily imply that F has only one zero in $\mathbf{B}$, just that there is only one at which $J(F)$ is non-singular). It might seem that this could be used more widely but there is one problem: over what interval is x_{n+1} to be defined? In the algebraic case there is a gap function which can be used to define a lower bound for $\det J(F)$ and the upper bound over a box $\mathbf{B}$ can easily be estimated.

Combining a gap function L with part 3 of Prop. 1 provides a sign determination algorithm. As in that proof consider a sequence v_i such that $|q(\xi) - v_i| < 2^{-i}$. It may be that for some v_i, $q(\xi)$ is shown to be non-zero. If not, $\lim_{i\to\infty} v_i = 0$ and thus for some i, $\max(|v_i - 2^{-i}|, |v_i + 2^{-i}|) < L$ at which point $q(\xi)$ is deduced to be zero.

Gap functions are only one way of deciding the sign of q. There are sometimes algebraic or other ways of making the decision for particular q or F (consider the case where F is a linear map). In some systems there is a natural canonical form. For example in Chapter 7 a gap function is given for expressions formed by field operations on rational numbers, nevertheless it is computationally simpler to reduce the expression to its canonical form to decide if it is zero.

The alternative definition above is not forced to rely on gaps. For example, if in a particular case q is easily shown to be zero then the value returned as L could

be $1 + |q(\xi)|$ with q computed with minimal accuracy.

1.5 The Rest of the Report

Chapter 2 focuses on interval arithmetic as a way of implementing exact arithmetic in a situation where, as in this case, values are given implicitly as roots. Interval based variants of Newton's and other root finding algorithm provide the mechanism for approximating value. Convergence criteria for such methods are shown to provide a basis for an algorithm to decide if a set of equations has a unique solution in a box.

The three chapters on algebraic systems use a common notation and depend on some well known results. It was convenient to group these together and they appear in Chapter 3. Before considering general algebraic systems, a number of special cases (for example sets of univariate polynomials) are described in Chapter 4 and some results given. Given the absence of good algorithms for the general case, it is worth considering whether systems of polynomials should be converted to a more convenient equivalent form, e.g. finding a set of univariate polynomials sharing a root with a given system. Doing this algebraically is possible by standard techniques but an alternative possibility is suggested, using lattice reduction algorithms.

Chapter 5 considers the application of Gröbner basis methods to the general case. Though Gröbner bases have many advantages and provide an essentially canonical way of representing polynomials, the cost of computing the basis is high. The main alternative is the use of multivariate resultant methods two examples of which are given in Chapter 6 and which both lead to a new gap functions for

the general case.

The extension to a subset of the elementary numbers is discussed in Chapter 7. The chapter is largely concerned with experiments and conjectures about sign determination of exp – log expressions and the Uniformity Conjecture.

1.6 An Example

It is convenient to have an example which can be used to illustrate a variety of methods. The one chosen is based on the identity $\sqrt{9+4\sqrt{2}} = 1+2\sqrt{2}$

Set $x_1 \equiv \sqrt{2}$ and $x_2 \equiv \sqrt{9+4\sqrt{2}}$. Proving the identity is equivalent to showing

$$\begin{aligned} x_1^2 - 2 = 0 \quad & \text{where } x_1 \in [1.41, 1.42] \\ x_2^2 - 9 - 4x_1 = 0 \quad & \text{where } x_2 \in [3.82, 3.84] \\ \text{implies} \quad x_2 - 2x_1 - 1 = 0 & \end{aligned} \tag{1.1}$$

As a triangular system this is not the most general form (the equations also form a Gröbner basis) so where appropriate one of the following two equivalent systems will be used.

$$\begin{aligned} x_1 x_2 - x_1 - 4 = 0 \\ x_2^2 - 9 - 4x_1 = 0 \end{aligned} \tag{1.2}$$

or

$$\begin{aligned} x_1^2 - 2 = 0 \\ x_2^4 - 18x_2^2 + 49 = 0 \end{aligned} \tag{1.3}$$

The application of particular results to this example use the format below:

In Example 1.1 $\quad p_1 = x_1^2 - 2, \;\; p_2 = x_2^2 - 9 - 4x_1, \;\; q = x_2 - 2x_1 - 1$

Using resultants,

$$q^4 + 4q^3 - 28q^2 = 0$$

and the non-zero roots are ≈ 3.65 and -7.65.

Chapter 2

Exact Arithmetic

2.1 Introduction

The first chapter of this report made a distinction between 'exact arithmetic' and 'exact computation'. This chapter is concerned with the former topic in so far as it can be used to implement the latter. The ideas described here are adequate to resolve two problems required to implement exact computation.

Firstly, some systems supporting exact arithmetic have already been mentioned but all are designed to do so in terms of field operations and function evaluations on given (floating point) values. The system described here provides a higher level structure appropriate dealing with values defined as an 'unknown' zero of a function.

Secondly, the formalism adopted for exact computation defines a real vector $\xi \in \mathbb{R}^n$ as the unique isolated solution of a function within a given n-box. For such a model to be viable it is necessary to be able to validate the claim that a function and box do indeed define ξ. Part of this chapter describe an algorithm for this.

The solution to both problem will be based on the use of interval arithmetic.

2.2 Interval Arithmetic

Numeric calculation traditionally uses floating point arithmetic for speed and accepts the penalty of inexact results. These days floating point packages, whether hardware or software based, provide values which are accurate to one bit in the mantissa per operation but evaluation of an expression involving a number of operations may still require considerable *post hoc* error analysis to determine the reliability of the result.

To overcome this various system have been developed which allow a programmer to specify the required accuracy of a final result rather than individual operations. Such systems are usually support the notion of $=_\epsilon$ (i.e. $a =_\epsilon b$ iff $|a - b| < \epsilon$).

The starting point of such approaches could be traced to Turing's first papers on undecidability [Tur36, Tur37] which model real numbers as sequences of intervals of decreasing width. Rice [Ric54] used recursive functions to define convergent sequences. The term 'exact' is used by Vuillemin in [Vui90] where real values are defined as sequences of nested intervals (using continued fractions) which are guaranteed to converge to a specified value. This paper is the basis of a considerable amount of later work, both theoretical and practical, two illustrative examples being [Pot96] and [EP97] which continue Vuillemin's work with continued fraction definitions of expressions.

All of these methods are based on some form of lazy evaluation. A numerical 'object' is represented by a mathematical expression which is evaluated to only

as much precision as is required (for example by using a truncated power series representation or by continued fractions). It is less easy to deal with the situation where, as in much of this report, values are given implicitly as a solution of a set of equations. The implicit approach is on, the other hand, quite easily adapted to dealing with explicitly given expressions: as a simple example, π is easily and naturally defined as the solution of $sin(x) = 0$ in the interval $[3.1415, 3.1416]$.

Interval arithmetic, introduced originally by Ramon Moore [Moo79] and now relatively widely used and mature (see e.g. [Abe94, AH83, Han92, Neu90]), developed more from a desire to replace *post hoc* error analysis of numerical methods with automatically computed error bounds than from theoretical concerns with precise accuracy.

In practical applications, input values are often known only approximately. Even when they are known, it may not be possible to represent them as floating point numbers and finally, calculation with floating point values introduces rounding errors. Interval arithmetic therefore replaces each real value x with a closed interval $\mathbf{x} = [\underline{x}, \overline{x}]$ such that $x \in \mathbf{x}$. We write $\mathbb{IR} \equiv \{[\underline{x}, \overline{x}] | \underline{x}, \overline{x} \in \mathbb{Q}\}$ for the set of real intervals (it is assumed throughout that intervals have rational endpoints).

Field operations on numbers are replaced by corresponding operations on intervals in such a way that the resulting interval is guaranteed to contain the true result. Detailed descriptions of interval arithmetic can be found in the papers already

quoted above but, as an example, field operations on intervals can be defined as

$$
\begin{aligned}
[a,b]+[c,d] &= [a+b, c+d] \\
[a,b]-[c,d] &= [a-d, b-c] \\
[a,b]\times[c,d] &= [\ \min\{a\times c, a\times d, b\times c, b\times d\}, \\
&\qquad\qquad \max\{a\times c, a\times d, b\times c, b\times d\}] \\
1/[a,b] &= [1/b, 1/a] \ \text{ if } \ 0\notin[a,b]
\end{aligned}
$$

Interval arithmetic is 'sub-distributive', for $\mathbf{a},\mathbf{b},\mathbf{c}\in\mathbb{IR}, \mathbf{a}(\mathbf{b}+\mathbf{c})\subset\mathbf{ab}+\mathbf{bc}$ but, in general, not distributive. Also familiar identities may fail when evaluated using interval arithmetic. For example, $\mathbf{x}-\mathbf{x}=[\underline{x}-\overline{x}, \overline{x}-\underline{x}]\neq[0,0]$.

In practice, floating point arithmetic is used to implement arithmetic and, for example, $[a,b]+[c,d]=[a+b,c+d]$ is replaced by $[a,b]+[c,d]=[(a+b)\downarrow,(c+d)\uparrow]$ where $x\downarrow$ is the largest floating point number such that $x\downarrow\leq x$, and $x\uparrow$ is the smallest floating point number such that $x\uparrow\geq x$. It is assumed throughout that arbitrary precision floating point arithmetic is available and that the error in replacing x with $x\uparrow$ or $x\downarrow$ can be reduced to any pre-chosen value.

2.2.1 Notation

Various notations have been used for interval arithmetic with no two of the authors cited above using the same. The notation used here is based on that suggested by Kearfott in [Kea] and is perhaps nearest to standard.

Notation	Definition	Interpretation
$\mathbf{x}$	$[\underline{x}, \overline{x}]$	an interval
$\breve{x}$	$\breve{x} \in \mathbf{x}$	an arbitrary value from a interval
$w(\mathbf{x})$	$\overline{x} - \underline{x}$	the width
$m(\mathbf{x})$	$(\underline{x} + \overline{x})/2$	the midpoint
$\lvert\mathbf{x}\rvert$	$\max(\lvert\underline{x}\rvert, \lvert\overline{x}\rvert)$	absolute value
$r(\mathbf{x})$	$(\overline{x} - \underline{x})/2$	the radius
$\text{int}(\mathbf{x})$	$(\underline{x}, \overline{x})$	the interior of an interval
$d(\mathbf{x}, \mathbf{y})$	$\max(\lvert\underline{x} - \underline{y}\rvert, \lvert\overline{x} - \overline{y}\rvert)$	distance metric
$\mathbf{X}$	$(\mathbf{x_1}, \ldots, \mathbf{x_n})^T$	a vector of intervals - an n-box
$\breve{X}$	$(\breve{x}_1, \ldots, \breve{x}_n)^T$	an arbitrary point in the box $\mathbf{X}$
X	$(x_1, \ldots, x_n)^T$	a vector in $\mathbb{R}^n$
$\lvert\mathbf{X}\rvert$	$\max(\lvert\mathbf{x_i}\rvert)$	
$m(\mathbf{X})$	$(m(\mathbf{x_1}), \ldots, m(\mathbf{x_n}))^T$	
$w(\mathbf{X})$	$\max(w(\mathbf{x_i}))$	
$r(\mathbf{X})$	$\max(r(\mathbf{x_i}))$	
$d(\mathbf{X}, \mathbf{Y})$	$\max(d(\mathbf{x_i}, \mathbf{y_i}))$	
$Y \in \mathbf{X}$	$\forall i,\ y_i \in \text{int}(\mathbf{x_i})$ or $\mathbf{x_i} = [y_i, y_i]$	
$\mathbf{X} \cap \mathbf{Y}$	$(\mathbf{x_1} \cap \mathbf{y_1}, \ldots, \mathbf{x_n} \cap \mathbf{y_n})^T$	$\mathbf{X} \cap \mathbf{Y} = \emptyset$ if any $\mathbf{x_i} \cap \mathbf{y_i} = \emptyset$
$\mathbf{X} \cup \mathbf{Y}$	$(\mathbf{x_1} \cup \mathbf{y_1}, \ldots, \mathbf{x_n} \cup \mathbf{y_n})^T$	undefined if $\mathbf{x_i} \cap \mathbf{y_i} = \emptyset$, for any i

The notation used for vectors is also used for matrices. A single value x can be treated as a *thin interval* $\mathbf{x} \equiv [x, x]$, thus it is not unreasonable to write both $x \in \mathbf{y}$ and $x \subset \mathbf{y}$. Vectors and matrices, both interval and point are used similarly.

2.2.2 Interval Extensions

Definition 5. *[Moo79, p20]. Given a function $F : \mathbf{X} \subset \mathbb{R}^n \to \mathbb{R}^m$, an* inclusion monotonic interval extension *of F is a function $\widehat{F} : \mathbb{IR}^n \to \mathbb{IR}^m$ such that*

1. $\mathbf{Y} \subset \mathbf{X}$ *implies* $\widehat{F}(\mathbf{Y}) \subset \widehat{F}(\mathbf{X})$

2. $w(\mathbf{Y}) = 0$ *(i.e. $\mathbf{Y}$ is a point vector, Y) implies* $\widehat{F}(\mathbf{Y}) = F(Y)$

Usually, F will be used both for the function and its interval extensions - the use of bold face for the intervals serving to distinguish the two. Where this is not clear from context the form $\widehat{F}$ is used.

2.2.2.1 The Natural Extension

For rational expressions the 'natural' extension is that obtained by replacing each field operation with its equivalent interval equivalent. For monotonic functions, a natural extension can be obtained by evaluating the function at the end points of the interval.

Definition 6. *(Based on [Moo79, p19]) For* $F : \mathbf{X} \subset \mathbb{IR}^n \to \mathbb{R}^m$, *write* $F^u(\mathbf{X}) \subset \mathbb{IR}^m$ *for the smallest interval vector containing the image of* F *over* $\mathbf{X}$ *(for this case alone the interval endpoints need not be rational)*[1].

Clearly $F^u(\mathbf{X}) \subset \widehat{F}(\mathbf{X})$ and ideally an interval extension will be as 'near' to the image as possible. A suitable restriction of interval extensions, which covers the elementary and most useful functions, is to those which are *Lipshitz continuous*

Definition 7. *[Neu90, p33] A function* $F : \mathbf{X} \subset \mathbb{IR}^n \to \mathbb{IR}^m$ *is* Lipshitz continuous *on* $\mathbf{X}$ *if* $F(\mathbf{X})$ *is defined and for any* $\mathbf{X_1}, \mathbf{X_2} \subset \mathbf{X}$, *there exists* $k \in \mathbb{R}^+$ *such that*

$$d(F(\mathbf{X_1}), F(\mathbf{X_2})) \leq k\, d(\mathbf{X_1}, \mathbf{X_2})$$

It is shown in the standard texts, e.g. [Neu90, AH83], that the natural interval extensions for elementary functions and field operations are Lipshitz continuous.

[1] Moore's definition allows for maps of families of arbitrary subsets of $\mathbb{R}^n$ to an interval vector.

Interval extensions are not unique and ways of representing expressions give better results (in the sense of evaluating to narrower intervals) than others[2].

2.2.2.2 The Centred Extension

For algebraic expressions better extensions can often be found by using Horner's scheme rather than direct interval evaluation (in general the smaller the number of operations used in computing the expression the smaller the enclosure) but a better technique is to use 'centred forms' or higher order Taylor series forms.

Centred forms start from the Mean Value theorem: for $A, B \in \mathbf{X}$

$$F_i(A) = F_i(B) + \sum_j \frac{\partial F_i}{\partial x_j}(\alpha_{ij})(A_j - B_j)$$

In particular if $B = m(\mathbf{X})$ then

$$F_i^u(\mathbf{X}) \subset F_i(m(\mathbf{X})) + \sum_j \frac{\partial F_i}{\partial x_j}(\alpha_{ij})(\mathbf{X}_j - m(\mathbf{X}_j)$$

However each of the derivative values are in $\widehat{J}$ (the interval evaluation of the Jacobian of F) and so

$$F^u(\mathbf{X}) \subset F(m(\mathbf{X})) + \widehat{J}(\mathbf{X} - m(\mathbf{X}))$$

The right hand side provides an alternative interval evaluation for F to the natural form which can give a tighter estimate for $F^u(\mathbf{X})$.

[2]This is, of course, not a problem unique to interval arithmetic: a similar situation arises in floating point calculations. Unfortunately, in interval arithmetic, you have both problems.

For example, if $F(x) = x^4 - 11x^3 + 9x^2 - 25x + 4$ evaluated over $[0.2, 0.4]$,

$$\begin{aligned} F^u([0.2, 0.4]) &\subset [-5.24, -0.727] \\ F([0.2, 0.4]) &= [-6.35,\ 0.378] \text{ evaluated as presented} \\ &= [-5.63, -0.449] \text{ rewritten in Horner form} \\ &= [-5.30, -0.664] \text{ in centred form} \end{aligned}$$

Note that the simplest interval extension falsely includes zero in the computed range.

Several authors have some claim to devising the Taylor series method which extends centred forms by using higher order derivatives. The approach of Berz [MB01, MB03] seems most developed. There has been some controversy in the Interval Arithmetic community over what level of accuracy can be claimed for Taylor models. An analysis is given in [Neu02] but the following result for natural and centred extensions is sufficient here.

Lemma 1. *If F is Lipshitz continuous over $\mathbf{X}$ and F_1 and F_2 are the natural and centered interval extensions of F then there exist $k_1, k_2 \in \mathbb{R}^+$ such that*

$$d(F^u(\mathbf{X}), F_i(\mathbf{X})) < k_i w^i(\mathbf{X}) \tag{2.1}$$

Proof. See [AH83, pp24-27]. □

Thus if a sequence of intervals tends to a point, the centred extension of F tends quadratically to the image.

2.3 Interval Iteration Methods

The numerical solution of a system, F, of equations is often based on an iterative process of the form:

$$X' = \Phi(F, X) \tag{2.2}$$

where Φ has a fixed point at a zero of F. Simply replacing values by intervals and functions by interval extensions gives an interval scheme:

$$\begin{aligned} \mathbf{K} &= \widehat{\Phi}(\widehat{F}, \mathbf{X}) \\ \mathbf{X}' &= \mathbf{K} \cap \mathbf{X} \end{aligned} \tag{2.3}$$

Such a scheme is called convergent if the sequence $\mathbf{X}^{(i)}$ converges to a point vector where $\mathbf{X}^{(0)} = \mathbf{X}$ and for $k > 0$, $\mathbf{X}^{(k+1)} = \widehat{\Phi}(\widehat{F}, \mathbf{X}^{(k)}) \cap \mathbf{X}^{(k)}$.

The advantage of the interval version is that, if the iteration converges, the true solution is guaranteed to be within each interval vector of the iteration. More generally if $F(\xi) = 0$ and $\xi \in \mathbf{X}$ then $\xi \in \mathbf{X}'$, in particular, if $\mathbf{X}' = \emptyset$ then F has no zeros in $\mathbf{X}$.

Though direct analogues of Newton's method exist, in practice the most commonly used algorithms are the Krawczyk [Kra86] and Hansen-Sengupta [HS81] operators. The latter being a Gauss-Seidel variant of the former. The Krawczyk version has been used to illustrate this section only because it is slightly simpler. With small modifications the Hansen-Sengupta operator could be substituted (and would never give worse results).

2.3.1 The Krawczyk Operator

Definition 8. *The* Krawczyk operator *for a function F and interval vector $\mathbf{X}$ is*

$$K(F, \mathbf{X}, \breve{X}, C) \equiv \breve{X} - CF(\breve{X}) - (CJ(F, \mathbf{X}) - I)(\mathbf{X} - \breve{X})$$

where C is a (real) conditioning matrix.

Theorem 1 (Krawczyk, Kahan, Hansen & Sengupta). *Let $F : \mathbf{X} \to \mathbb{R}^n$ be Lipshitz continuous on $\mathbf{X}$ and let $\mathbf{X}' = K(F, \mathbf{X}, \breve{X}, C)$ then ([Neu90, p177])*

1. *If $\breve{Y} \in \mathbf{X}$ and $F(\breve{Y}) = 0$ then $\breve{Y} \in \mathbf{X}'$.*
2. *If $\mathbf{X}' \cap \mathbf{X} = \emptyset$ then F has no zero in $\mathbf{X}$.*
3. *If $\breve{X} \in \text{int}(\mathbf{X})$ and $\emptyset \neq \mathbf{X}' \subset \text{int}(\mathbf{X})$ then F has a unique zero in $\mathbf{X}$ (and therefore in $\mathbf{X}'$).*

Neumaier also shows $K(F, \mathbf{X}, m(\mathbf{X}), J^{-1}(F, m(\mathbf{X}))) \subset K(F, \mathbf{X}, \breve{X}, C)$ and so $m(X)$ is always an optimal choice for $\breve{X}$ (and the first condition of point 3 above is automatically satisfied), while a floating point approximation to $J^{-1}(F, m(\mathbf{X}))$ is used for C. In future $K(F, \mathbf{X})$ will be used as shorthand for the optimal form.

Moore [Moo79, p63] presents a range of iteration schemes which are guaranteed to converge if $K(F, \mathbf{X}, \breve{X}, C) \subset \mathbf{X}$. In practice two variants of the Krawczyk operator can be used directly. The first is to use $\mathbf{X}' = \mathbf{X} \cap K(F, \mathbf{X}, m(\mathbf{X}), C)$ where C approximates $J^{-1}(F, m(\mathbf{X}))$. As mentioned above this gives the 'best' enclosure. Alternatively, C can be computed for the initial matrix and reused at each iteration.

Note that the Theorem does not provide a guarantee of convergence or proof of non-existence of roots. Consider the case where F has a zero on each face of $\mathbf{X}$. Point 1 then guarantees that every zero is also in $\mathbf{X}'$ and so $\mathbf{X}' = \mathbf{X}$. Even if F has only one zero, there is no guarantee that the condition $\mathbf{X}' \subset \operatorname{int}\mathbf{X}$ is eventually met. This is addressed, and the problem resolved, in the context of the algorithm described below at §2.4.1.

2.3.2 Exclusion Tests

Interval iteration methods have an advantage over traditional iteration schemes which Th. 1 and Prop. 2 exemplify: some boxes can be shown not to contain roots. Tests which reveal the presence of roots are sometimes called *inclusion tests*, those which show their absence are *exclusion tests.*

It is tempting to make use of exclusion tests since any division of a box into sufficiently small sub-boxes is likely to yield more boxes without than with roots. A number of different exclusion tests have been invented. The simplest being to evaluate $F(\mathbf{X})$ since $0 \notin F(\mathbf{X}) \subset F^u(\mathbf{X})$ implies F has no zero in $\mathbf{X}$.

Given a box not containing any zero, a sequence of bisections will eventually yield sub-boxes small enough for the exclusion test to work on them. But an individual exclusion test may fail (in the sense of not being able to show that no zero is present) if a root is just outside the box being tested. The box may then be split repeatedly with tests on sub-boxes close to the root still failing. [SNar] show that the effects of this 'clustering' of failed tests due to a nearby zero is reduced dramatically if the centred form is used (as would be expected from Eq. 2.1).

Other methods of exclusion testing have been suggested. Xu *et al* [XZW96] attempt to use only exclusion testing to find zeros of polynomial systems (eliminating regions definitely not containing zeros). They explicitly reject interval methods and rely on rational arithmetic. Unfortunately their method seems no different from using intervals: they are just computing upper and lower bounds separately.

For specifically polynomial systems a range of methods is possible. In the univariate case, Sturm sequences could be used. [Ped91a, Ped91b] extends these to the multivariate case.

Dedieu *et al* [DGY96] give a lower bound for the distance from a point to an algebraic surface. From this a box can be defined which contains no zeros of a given polynomial system. Their method has the advantage that it could be used more generally in algebraic systems containing zero components of dimension greater than zero.

Representing a polynomial system as Bernstein polynomials provides an algorithm which uses the convex hull properties of Bernstein polynomials to exclude large regions rapidly [MO91]. Strictly the algorithm they describe also provides an inclusion test, from which exclusion tests are produced. A serious practical disadvantage of the method is the need to recompute the Bernstein polynomial coefficients at each subdivision of the box.

It seems unlikely that any of these methods, other than the simplest, warrant being used.

2.4 Exact Arithmetic using Intervals

The definition of exact computation in Def. 3 requires arithmetic with two properties. It must be possible to approximate a value with arbitrary accuracy and it must be possible to decide if the restriction of a function F to an interval box $F_{\mathbf{B}}$ defines a point. In this section an algorithm is presented for the more restricted version of Def. 4 where F is a polynomial map. Since the algorithm is based on isolating roots using the Krawczyk operator it conveniently solves both problems: called to verify $F_{\mathbf{B}}$ defines a point it will return either that there are no roots, more than one, or a smaller box containing the root. If $F_{\mathbf{B}}$ does define a point, successive calls return a sequence of nested, strictly smaller root enclosures which converge to a point.

Definition 9. *For any box* $\mathbf{X}$ *define* $\dim \mathbf{X}$ *to be the number of components of* $\mathbf{X}$ *with non-zero width. If* $w(\mathbf{x_i}) = 0$, *i will be called a* point dimension.

The term 'dimension' as used above could be slightly confusing. All the boxes described in this chapter are embedded in $\mathbb{R}^n$ for some fixed given n. An individual box, $\mathbf{B}$ then has $\dim \mathbf{B} \leq n$.

Definition 10. *The* volume *of a box* $\operatorname{vol} \mathbf{X}$ *corresponds to the usual notion except that point dimensions are ignored unless the box has dimension zero. I.e.* $\operatorname{vol} \mathbf{X} = 0$ *if* $w(\mathbf{X}) = 0$ *and otherwise is* $\prod_{w(\mathbf{x_i}) \neq 0} w(\mathbf{x_i})$

Boxes of different dimension are incommensurable. The notation $\mathbf{X} \preceq k\mathbf{Y}$ will be interpreted as meaning either $\dim \mathbf{X} < \dim \mathbf{Y}$ or $\dim \mathbf{X} = \dim \mathbf{Y}$ and $\operatorname{vol} \mathbf{X} \leq k \operatorname{vol} \mathbf{Y}$. In a set of boxes $\mathcal{R}$ this is extended to the (not necessarily unique) box $\max \mathcal{R}$.

The Krawczyk operator can be applied to boxes containing point dimensions with only one minor modification, the statement $\mathbf{X}' \in \text{int}(\mathbf{X})$ is interpreted as: for each $\mathbf{x_i}$ either i is a point dimension or $\mathbf{x}'_\mathbf{i} \in \text{int}(\mathbf{x_i})$. Point dimensions are always rational numbers (though represented in floating point). An equivalent to interpreting $\text{int}(\cdot)$ in this way for, say, $\mathbf{x_i} = r$ would be to replace all reference to the variable in F by its value and add an equation $x_i - r = 0$ with $x_i \in [\max(\underline{\mathbf{b_i}}, r - \epsilon), \min(\overline{\mathbf{b_i}}, r + \epsilon)]$ for some small value of ϵ such that the new interval has non-zero width.

If $\dim \mathbf{X} = 0$ then $\mathbf{X}$ is just a (rational) point. Since F is a polynomial map deciding if $F(\mathbf{X}) = 0$ is trivial.

The algorithm has been split into four parts for ease of comprehension. In an implementation they could be combined into a single algorithm without difficulty.

- *Validate* - applies the Krawczyk operator to produce a new box. If this is not sufficient to either prove that the input box has no roots, or that it has exactly one root, one of the other algorithms below is used to test for multiple roots or roots on the box boundary. If there is exactly one root, *Validate* returns a new box guaranteed to contain it.

- *Bisect* - applied when a box has to be subdivided because the iteration appears not to be converging. *Bisect* has to ensure that when a box is subdivided, double counted roots, lying on the boundary of two boxes are not introduced. If the maximum number of roots is known to be one, *Bisect* can return a box of lower dimension.

- *Boundary* - when there can be at most one root in a box, this determines whether it is in the interior (in which case the iteration will continue) or on

a boundary (in which case the dimension of the box being checked drops by at least one).

- *Approximate* - calls *Validate* repeatedly until a given accuracy is achieved.

In the implementation changes to floating point precision have to be made because of the difficulty of predicting the final precision. This isn't shown in the algorithms but involves a step which checks that the minimum width of given intervals (or a predicted width) is not so small that it approaches the software-ϵ of the floating point package. Typically, suppose an interval $\mathbf{x}$ has width $w(\mathbf{x}) = W$. The most significant operations are concerned with increasing the number of bits of accuracy of the interval. Quadratic convergence will double the number of significant bits, this suggests an heuristic to use in conjunction with the Krawczyk operator: if $|\log_2 w(\mathbf{x})| \geq M/2$ where M is the size of the mantissa, increase floating point precision to $2M$.

The notation used for the algorithms is straightforward and is summarised in App. A. Probably the only clarification needed is that:

$$\mathbf{Y}(i : \mathbf{a}) \equiv (\mathbf{y_1}, \ldots, \mathbf{y_{i-1}}, \mathbf{a}, \mathbf{y_{i+1}}, \ldots, \mathbf{y_n})$$

and if X is a variable in an algorithm, $\underset{\text{init}}{X}$ is its initial value (X is used if the meaning is clear).

2.4.1 Algorithm *Validate*

As stated above the basis of this algorithm is to use the Krawczyk iteration to converge to roots of F in some box $\mathbf{B}$. Even if a box contains just one root there may be a problem detecting it if it lies on the box boundary (a possibility not

catered for by Th. 1). Also, an iteration may fail to converge because the box contains more than one root or because the box is 'too large'.

For these reasons the algorithm, after applying the operators must make decision based on the new box it has generated. An empty box implies no root, a box interior to the previous one implies a unique root. If neither of these obtain other methods have to be adopted, for example bisecting the box and applying the algorithm to the two parts separately. Bisection introduces its own problems. To avoid double counting of roots (one root on the boundary of two boxes) the *Validate* algorithm has to be called recursively to ensure that the bisection boundary does not itself contain a root.

Some preliminary results are needed to guarantee convergence and to identify the case where a box contains at most one root. These are based on results in [Neu90] modified for the needs of this section (in general [Neu90] describes a Kantorowich style iteration which may converge only linearly and does not take account of roots on a boundary).

Lemma 2. *Let* $\breve{X} \in \mathbf{X}$, *if* $C = J^{-1}(F, \breve{X})$ *exists then* $0 \in \mathbf{U}_{ij}$ *where* $\mathbf{U} = CJ(F, \mathbf{X}) - I$. *Further there exists* $\kappa \in \mathbb{R}^+$ *such that* $w(\mathbf{U}) \leq \kappa w(\mathbf{X})$

Proof. for the first part $CJ(F, \breve{X}) - I = 0$ and $J(F, \breve{X}) \in J(F, \mathbf{X})$ Thus $0 = (CJ(F, \breve{X}) - I)_{ij} \in (CJ(F, \mathbf{X}) - I)_{ij}$

For the second part, each term in $J(F)$ is Lipshitz continuous and so for some $\delta \in \mathbb{R}^+$, all terms satisfy $w(J_{ij}(F, \mathbf{X})) \leq \delta w(\mathbf{X})$ which makes $w(\mathbf{U}_{ij}) \leq n\delta w(\mathbf{X})$.

□

Lemma 3. *For any box,* $\mathbf{X}$, $w(K(F,\mathbf{X})) \leq \|CJ(F,\mathbf{X}) - I)\|_1 w(\mathbf{X})$.

Proof.

$$\begin{aligned} w(K(F,\mathbf{X})) &= w(m(\mathbf{X}) - CF(m(\mathbf{X})) - (CJ(F,\mathbf{X}) - I)(\mathbf{X} - m(\mathbf{X}))) \\ &= w((CJ(F,\mathbf{X}) - I)(\mathbf{X} - m(\mathbf{X}))) \\ &\leq \|CJ(F,\mathbf{X}) - I\|_1 w(\mathbf{X}) \end{aligned}$$

□

Proposition 2. *For any box,* $\mathbf{X}$, *write* $\mathbf{X}^{(0)} = \mathbf{X}$, $\mathbf{X}^{(n)} = K(F,\mathbf{X}^{(n-1)}) \cap \mathbf{X}^{(n-1)}$. *If, for some* n, $C^{(n)} = J^{-1}(F, m(\mathbf{X}^{(n)}))$ *exists and* $\|C^{(n)}J(F,\mathbf{X}^{(n)}) - I\|_1 < 1$ *then* F *has at most one root in* $\mathbf{X}$

Proof. From Lemma 3,

$$w(K(F,\mathbf{X}^{(n)})) \leq \|CJ(F,\mathbf{X}^{(n)}) - I\|_1 w(\mathbf{X}^{(n)})$$

and, so if the condition holds, $w(\mathbf{X}^{(n+1)}) = w(K(F,\mathbf{X}^{(n)})) < w(\mathbf{X}^{(n)})$. Continue the iteration using the value $C^{(n)}$ rather than computing $C^{(n+1)}$. Now $J(F,\mathbf{X}^{(n+1)}) \subset J(F,\mathbf{X}^{(n)})$ since $\mathbf{X}^{(n+1)} \subset \mathbf{X}^{(n)}$ and so $\|C^{(n)}J(F,\mathbf{X}^{(n+1)}) - I\| < 2$. Thus $w(\mathbf{X}^{(n+2)})$ and all subsequent iterates have strictly decreasing widths. If at some point $\mathbf{X}^{(N)} = \emptyset$ then there is no zero, otherwise $\lim_{n\to\infty} w(\mathbf{X}^{(N)}) = 0$, i.e. F has exactly one root in $\mathbf{X}^{(n)}$. □

Corollary 1. *If* $\mathbf{X}$ *contains a unique root of* F, *the Krawczyk iteration, with a re-computation of* C *and* $J(F,\mathbf{X})$ *at each step, converges quadratically.*

Proof. From Lemma 3,

$$\begin{aligned} w(K(F,\mathbf{X})) &\leq \|CJ(F,\mathbf{X}) - I)\|_1 w(\mathbf{X})/2 \\ &= O(w^2(\mathbf{X})) \text{ from Lemma 2} \end{aligned}$$

□

An alternative test for a box to contain at most one root is

Lemma 4. *If* $J(F, \mathbf{X})$ *is* regular *(i.e for all* $\breve{X} \in \mathbf{X}$, $J(F, \breve{X})$ *is non-singular [Neu90]) then* F *has at most one zero in* $\mathbf{X}$.

Proof. If for $X, X' \in \mathbf{X}, F(X) = F(X') = 0$ then by the Mean Value Theorem there exists a matrix M such that $F(X) = F(X') + M(X - X')$ or $M(X - X') = 0$. Since $J(F, \mathbf{X})$ is regular and $M \in J(F, \mathbf{X})$, M is non-singular and so $X = X'$ □

Thus if an interval evaluation of det $J(F, \mathbf{X})$ yields a interval not containing zero then F has at most one zero in $\mathbf{X}$. Unfortunately the Jacobian tends to be a complex expression and the width of the computed interval can be large. A better way of using the result is to evaluate the determinant symbolically just once and evaluate the result as efficiently as possible.

The algorithm below uses only the test from Lemma 3 since the matrix on which it is based is computed anyway.

It is convenient to sum up what results are now available in applying the Krawczyk operator

1. Exclusion tests: F has no root in $\mathbf{X}$ if
   - $\dim \mathbf{X} = 0$ and $F(\mathbf{X}) \neq 0$, or
   - $0 \notin F(\mathbf{X})$, or
   - $K(F, \mathbf{X}) \cap \mathbf{X} = \emptyset$

2. Uniqueness Test:

   - $\dim \mathbf{X} = 0$ and $F(\mathbf{X}) = 0$, or
   - $K(F, \mathbf{X}) \subset \text{int}(\mathbf{X})$

3. At most one root:

   - $0 \notin \det J(F, \mathbf{X})$, or
   - $\|CJ(F, \mathbf{X}) - I\|_1 < 1$

The algorithm assumes an arbitrary but small value $\delta \in \mathbb{R}^+$ (small with respect to the width of the initial box). δ is used to prevent endless iteration in the case that a zero lies on a box boundary (when the Krawczyk uniqueness will never be attained).

```
1   Algorithm Validate
2   Input: Y                                  ▷ box to be validated/approximated
3   W ⟵ vol Y
4   R ⟵ {Y}                                   ▷ boxes to process
5   T ⟵ ∅                                     ▷ boxes containing a root
6   while R ≠ ∅
7      S ⟵ ∅                                  ▷ new boxes which may contain roots
8      for Y ∈ R
9         if dim Y = 0
10           if F(Y) = 0   T ⟵ T ∪ {Y}
11           next loop
12        end if
13        V ⟵ F(Y)
14        if 0 ∉ V   next loop                ▷ Cannot contain a zero
15        X ⟵ Y
16        C ⟵ an approximation to J^{-1}(F, m(X))
17        U ⟵ CJ(F, X) − I
18        Y ⟵ (m(X) − Cm(V) − U(X − m(X))) ∩ X
19        if Y = ∅   next loop                ▷ can't contain a zero
20        if Y ⊂ int(X)                       ▷ exactly one root in Y
21           T ⟵ T ∪ {Y}
22        else if ‖U‖ < 1                     ▷ at most one root
```

```
23              if vol Y < δ                    ▷ very small box
24                 ▷ does it lie on the boundary?
25                 Z ⟵ Boundary(Y, X)
26                 if Z = ∅                     ▷ no root on boundary
27                    S ⟵ S ∪ Bisect(Y, true)
28                 else
29                    T ⟵ T ∪ {Z}              ▷ one root on boundary
30                 end if
31              else
32                 S ⟵ S ∪ Bisect(Y, true)
33              end if
34           else
35              S ⟵ S ∪ Bisect(Y, false)
36           end if
37        end for
38        if #T > 1  fail                       ▷ at least two roots
39        R ⟵ S
40     end while
41     if R = ∅
42        return ∅
43     else if R = {X}
44        if vol X > (2/3)W   X ⟵ Bisect(X, true)
45        return X
```

Lemma 5. *Assuming the correctness of algorithms Boundary and Bisect*

1. *After each iteration of the outer loop, if* $\alpha \in \underset{init}{\mathbf{Y}}$ *and* $F(\alpha) = 0$ *then* α *is in exactly one of the boxes comprising* $\mathcal{R}$ *or* $\mathcal{T}$ *unless the algorithm fails as a result of detecting more than one root.*

2. *if at line 7,* $\mathbf{Z} = \max \mathcal{R}$ *then at the next iteration the maximum volume is* $\leq (1+\epsilon)\operatorname{vol}\mathbf{Z}/2$.

3. *For any* $F_{\mathbf{B}}$ *there exists* $\mathbf{V}^*$ *with* $\dim \mathbf{V}^* = n$ *such that for all* $\mathbf{Y} \subset \mathbf{B}$ *with* $\mathbf{Y} \preceq \mathbf{V}^*$ *the main loop is executed at most* $\dim \mathbf{Y}$ *times.*

4. *If* $N \equiv \operatorname{vol}(\underset{init}{\mathbf{Y}})/\operatorname{vol}\mathbf{V}^*$, *the code in the inner loop is executed at most* $O(N \log N)$ *times.*

5. *If the Krawczyk iteration convergences quadratically, the code in the inner loop may be executed* $O(N \log M)$ *times where* $M \equiv \log(\operatorname{vol} \underset{init}{\mathbf{Y}}) / \log \operatorname{vol} \mathbf{V}^*$.

6. *The algorithm always terminates. If* $\underset{init}{\mathbf{Y}}$ *contains one root a box enclosing it is returned with volume* $\prec \frac{1+\epsilon}{2} \operatorname{vol} \underset{init}{\mathbf{Y}}$, *if no roots the empty set is returned, otherwise the algorithm raises a 'fail' condition.*

Proof. 1. The condition is trivially true on first entry to the loop. Each box is processed by the inner loop (lines 8–37). If the condition is true on entry to the outer loop then, for any box $\mathbf{X}$ in $\mathcal{R}$ containing a root, one of the following actions must occur:

   (a) at line 20 a sub-box of $\mathbf{X}$ containing the root is added to $\mathcal{T}$,

   (b) at line 25 there can be at most one root in the box, either on a boundary sub-box of dimension $\leq \dim \mathbf{X} - 1$ (line 29) or in at most one of two boxes generated by *Bisect* (line 27).

   (c) at line 35 any other box is split into at least two sub-boxes with no common zero.

2. Boxes added to $\mathcal{S}$ (assigned to $\mathcal{R}$ at the end of the loop), are created by the *Bisect* algorithm. In the worst case, *Bisect* returns on box with volume $\frac{1+\epsilon}{2} \operatorname{vol} \mathbf{Y}$.

3. It can be assumed that $\underset{\text{init}}{\mathbf{Y}}$ contains at most one root since there are finitely many roots and $\mathbf{V}^*$ can be small enough to exclude boxes with more than one root. If $\mathbf{Y}$ contains no roots, and $\operatorname{vol} \mathbf{Y}$ is small enough, one of the two exclusion tests (lines 14 and 19) will detect the fact.

   If $\mathbf{Y}$ contains a root in its interior, Prop. 2 guarantees it will be detected if $\operatorname{vol} \mathbf{Y}$ is sufficiently small. So the cases of no roots and an interior root are

detected. This leaves only the possibility that there is a root on a boundary box (line 27) but, since its dimension must be at least one less, at most a further $\dim \mathbf{Y}$ iterations passing through the same path will reduce it to a rational point if no other test terminates the loop before that.

4. From item 2, box size drops by at least $(1+\epsilon)/2$ on each iteration, so if volume $\mathbf{V}^*$ is reached after k iterations $\mathbf{V}^* \preceq ((1+\epsilon)/2)^k \underset{\text{init}}{\mathbf{Y}}$, or there are $O(\log(N))$ iterations of the outer loop. Since the boxes in $\mathcal{R}$ are obtained by sub-dividing the original box, and have dimension $< \dim \underset{\text{init}}{\mathbf{Y}}$ the sum of the volumes of the boxes is $\leq N$ and the result follows.

5. If convergence is quadratic, there will still be at most N boxes when $\mathbf{V}^*$ is reached after $O(\log(\log(\text{vol} \underset{\text{init}}{\mathbf{Y}})/\log(\text{vol}\,\mathbf{V}^*))) = O(\log M)$ steps.

6. Combining the previous steps. Note that a box which is passed immediately to the set $\mathcal{T}$ of boxes with one root might not have volume $< \frac{1+\epsilon}{2} \text{vol} \underset{\text{init}}{\mathbf{Y}}$, hence the (strictly unnecessary) call on line 44.

□

2.4.1.1 Algorithm *Boundary*

Boundary checks for a root on one or more of the boundaries of a box. It is only called if the input box contains at most one root and uses *Validate* to check each (lower dimension) boundary box.

Boundary$(\mathbf{Y}, \mathbf{X})$ is called with $\mathbf{Y}$ being the result of applying the Krawczyk operator to $\mathbf{X}$ and $\mathbf{Y} \not\subset \text{int}(\mathbf{X})$. So at least one of $\mathbf{Y}$'s boundaries must coincide with one of $\mathbf{X}$'s. Other boundary boxes need not be checked.

```
1   Algorithm Boundary                          ▷ at most one root on boundary
2   Input: Y                                    ▷ Krawczyk operator applied to X
3   Input: X                                    ▷ Y ⊂ X
4   ▷ At most one root, check boundary
5   for i from 1 to n                           ▷ all possible dimensions
6      if w(x_i) ≠ 0                            ▷ ignore point dimensions
7         if x̲_i = y̲_i                          ▷ coincident boundary
8            U ⟵ Validate(Y(i : [y̲_i, y̲_i]))
9            if U ≠ ∅  return U
10        else if x̄_i = ȳ_i
11           U ⟵ Validate(Y(i : [ȳ_i, ȳ_i]))
12           if U ≠ ∅  return U
13        end if
14     end if
15  end for
16  return ∅
```

The correctness of the algorithm, assuming that of *Validate* is correct for boxes of dimension $< \dim \mathbf{Y}$, is clear. A worst case scenario for *Boundary* is that calling *Validate* leads to further calls of *Boundary* recursing down to the point case.

Lemma 6. *A box of dimension n has $2^{n-k}\binom{n}{n-k}$ boundaries of dimension k and the total number of boundaries of dimension $\leq n$ is 3^n.*

Proof. A boundary of dimension k is obtained by choosing $n-k$ point dimensions (which can be done in $2^{n-k}\binom{n}{n-k}$ ways). For each dimension chosen to become a point, either the upper or lower bound of the interval can be taken giving $2^{n-k}\binom{n}{n-k}$ boundaries. The total number of boundaries is therefore $\sum_{0\leq k\leq n} 2^{n-k}\binom{n}{n-k} = 3^n$ □

and thus up to 3^n points to be checked, unfortunately in the worst case this is done $2n$ times (when $\mathbf{X} = \mathbf{Y}$). However, as in the analysis of *Validate*, there is some minimum box below which iteration stops.

2.4.1.2 Algorithm *Bisect*

Bisect is used both when there is no knowledge of how many roots a box may contain and when it is known to contain at most one. The latter case is simpler, the box is divided in half along a dividing plane (of dimension $n-1$) perpendicular to the longest axis and the algorithm *Validate* used to check if the dividing plane contains a root. If it does the plane itself is returned. If not the two boxes produced by the division are returned

If there is no knowledge of the number of roots, again a check is made of the dividing plane. If this contains no zero the situation is as before. If it contains a root then another division is made a short distance $\epsilon/2$ from, and parallel to, the plane. If the new plane contains a no root the box is divided along it, otherwise the box contains at least two roots and the algorithm raises the fail condition to terminate execution.

```
1   Algorithm Bisect
2   Input: Y                                  ▷ Box
3   Input: singleRoot                         ▷ boolean - true = max of one root
4   m ⟵ w(Y)/2
5   i ⟵ index of widest dim of Y              ▷ bisect box
6   a_1 ⟵ y_i + m                             ▷ first try at about 1/2 length
7   Z_1 ⟵ Validate(Y(i : [a_1, a_1]))
8   if Z_1 = ∅
9      return {Y(i : [y_i, a_1]), Y(i : [a_1, ȳ_i])}
10  else if singleRoot
11     return Z_1
12  end if
13  ▷ Try again near first division
14  a_2 ⟵ a_1 ± +ε/2
15  Z_2 ⟵ Validate(Y(i : [a_2, a_2]))
16  if Z_2 ≠ ∅  fail                          ▷ More than one root
17  return {Y(i : [y_i, a_2]), Y(i : [a_2, ȳ_i])}
```

2.4.2 Algorithm *Approximate*

```
1   Algorithm Approximate
2   Input: X                          ▷ validated box
3   Input: W                          ▷ find approx. with width < W
4   while w(X) >= W
5       X ⟵ Validate(X)
6   end while
7   return X
```

2.4.3 Correctness and Complexity of *Validate*

Proposition 3. *If the zeros of* $F_{\mathbf{B}}$ *are finite in number and non-singular then the algorithm Validate will raise a fail condition if there are* > 1*, return* $\emptyset$ *if there are none and otherwise return a box* $\mathbf{B}'$ *containing the only root with* $\mathbf{B}' \preceq (1+\epsilon)\mathbf{B}/2$.

Proof. The proposition is correct if $\dim \mathbf{B} = 0$ since the algorithm terminates after at most one iteration. Suppose it is correct for $\dim \mathbf{B} < n$, and consider a call with input of dimension n. Both *Boundary* and *Bisect* call *Validate* but always for a box of dimension at most $n - 1$ which will always yield a correct result. □

The performance of the Krawczyk [Kra86] and Hansen-Sengupta [HS81] operators are impressive. Interval arithmetic is generally assumed to increase the time cost of arithmetic operations by a factor of two to four. Since both operators provide quadratic convergence[3] this is a small price to pay for the existence and uniqueness

[3]The speed of convergence depends on a number of factors. As presented initially in [Neu90] convergence can be linear when a Kantorwich style iteration is used with the same approximation to the Jacobian being used for the complete algorithm. In the method presented in this report the Jacobian is recomputed giving at least quadratic convergence in a sufficiently small

tests they provide.

The complexity of deciding if $F_{\mathbf{B}}$ represents a value is hard to ascertain. The method described above could be improved in many ways but it is, at its heart, intractable. Even if some vastly improved root finding technique were known we would not expect it to be polynomial in the input size.

Proposition 4. *Let $F : \mathbb{R}^n \to \mathbb{R}^n$ be a polynomial map where no polynomial has degree > 2. Deciding whether F has at least one zero in a box $\mathbf{B}$ is at least NP-Complete.*

Proof. The well known NP-Complete boolean satisfiability problem [GJ79, p39] is easily reduced to finding a zero in a box. Consider a boolean expression of the form $D = \bigwedge_{1 \le i \le s} D_i$ where the D_i are disjunctions of boolean variables $b_1, \ldots b_n$ and their inverses. This can be converted to a polynomial map F where

$$F_i = \begin{cases} x_i(x_i - 1) & \text{for } 1 \le i \le n \\ z_i + \sum_j a_{ij} x_j & \text{for } n < i \le n + s \text{ and } a_{ij} \in \{-1, 0, 1\} \end{cases}$$

where the first group of equations define the requirement that each x_i must be 0 or 1 as b_i is false or true. The second set are formed from the disjunctions in the following way. Firstly, replace $\vee$ by $+$, each b_i by x_i and $\neg b_i$ by $1 - x_i$. The j^{th} disjunction now has the form $k_j + \sum a_i x_i$ where k_j is a integer. The a_i may be 0 (b_i doesn't appear the disjunction), 1 (b_i is present) or -1 ($\neg b_i$ is present). k_j is equal to the number of b_i which appear negated. For this expression to correspond to 'true' its value at a root must be in $[1, n]$. Define a variable z_j whose range $[\underline{z_j}, \overline{z_j}]$ guarantees that $z_i + \sum a_{ij} x_j = 0$ has a solution with $z_i \in [\underline{z_i}, \overline{z_i}]$ when the x_i have values one or zero.

This transformation can be done in polynomial time.

region.

Suppose the Jacobian matrix is blocked as $\begin{pmatrix} A & B \\ C & D \end{pmatrix}$ where A is $n \times n$ and $A_{ij} = \partial F_i/\partial x_j$ for $1 \leq i, j \leq n$. This is zero apart from the diagonal where $A_{ii} = 2x_i - 1$. $B_{ij} = \partial F_i/\partial z_j = 0$ for $1 \leq i \leq n$, $n < j \leq n + s$. Finally D is the unit matrix $D_{ij} = \partial F_i/\partial z_j = 0$ for $n < i, j \leq n + s$. It follows that at any root $\det J = \prod_{1 \leq i \leq n}(2x_i - 1)$ is so non-singular.

F is a polynomial map with degree ≤ 2 as required then deciding whether it has a zero in $([0,1], \ldots, [0,1], [\underline{z_{n+1}}, \overline{z_{n+1}}], \ldots, [\underline{z_{n+s}}, \overline{z_{n+s}}])$ is equivalent to solving the satisfiability problem. □

Some estimates of how complex it is to validate a box in the polynomial case can be made.

Proposition 5. *If evaluating a function has unit cost then, in the worst case, for $F : \mathbb{R}^n \to \mathbb{R}^n$, the complexity of $Krawczyk(\mathbf{Y})$ is $O(n^3(AnN \log N)^n)$ where $N = \mathrm{vol}(\mathbf{Y})/\mathrm{vol}(\mathbf{V}^*)$ with $\mathbf{V}^*$, as before and A is some constant.*

If the Krawczyk iteration converges quadratically this can be reduced to $O(n^3(AN \log M)^n)$ where $M = \log(\mathrm{vol}\,\mathbf{Y})/\log(\mathrm{vol}\,\mathbf{V}^*)$.

Proof. One computation of the Krawczyk operator has cost $O(n^3)$. From Lemma 5, this is done directly $O(N \log N)$ times. However the calls to *Boundary* and *Bisect* also call *Validate* and at least one may be done on each iteration of the original algorithm. *Boundary* dominates the calls since each can require up to $2n$ calls to *Validate*. Each call applies only to boxes of dimension one less therefore calls to *Boundary* are never more that n deep. This gives a number of steps of order $O(n^3(nN \log N)^n)$.

The time for quadratic convergence follows similarly from Lemma 5 which gives $O(N \log M)$ for the number of iterations. □

Assuming a polynomial map, at each iteration a new Jacobian matrix and inverse is computed involving $O(n^2)$ polynomial evaluations (assuming arithmetic has unit cost). Evaluating a function using the centred form already involves computing each of the partial derivatives anyway so this can be done as a byproduct.

Lemma 7. *If $F \in \mathbb{Q}[x_1, \ldots, x_n]$, the cost of evaluating $F(\mathbf{B})$ and $J(F, \mathbf{B})$ is $O(n^3 D)$ addition/multiplication operations where $D = \prod \deg_i F$.*

Proof. For F with $n = 1$ this can be achieved by using Horner's scheme. By induction the result for $n > 1$ follows by considering polynomials in recursive form. □

This takes the cost to $O(n^6 D(nN \log N)^n)$.

We have no estimate of $\mathbf{V}^*$ but a reasonable guess would be to base it on the minimum size of a component of a root of F. Chapter 6 establishes a lower bound (using Eq. 6.3) of

$$|\xi_k| \geq L^{-(eD)^n}$$

where L is the maximum of the lengths of the polynomials comprising F, giving

$$\begin{aligned} N &= \operatorname{vol} \mathbf{Y} / \operatorname{vol} \mathbf{V}^* \\ &= O(L^{(eD)^n} \operatorname{vol} \mathbf{Y}) \end{aligned}$$

2.5 Non-Polynomial Functions

Validate was written to deal with validation of polynomial functions but it could easily be extended. For example it can be modified to count zeros. Apart from a trivial change of not failing when two roots are found in *Validate* a small change is needed to *Bisect*. As currently used an error is raised if two zeros are found in a box. If instead the algorithm is applied recursively to the box eventually it must be possible to separate roots into different sub-boxes.

More significantly, it be applied to a wider range of functions.

Proposition 6. *Let* $\mathbb{K} \equiv \mathbb{Q}[x_1, \ldots, x_n, e^{x_1}, \ldots, e^{x_n}]$ *and suppose* $F \in \mathbb{K}^n$. *If* $F_{\mathbf{B}}$ *has only finitely many isolated simple zeros then they can be counted by the Validate routine.*

Proof. The only issue is to decide if $F(\alpha) = 0$ for a rational point α. Each component of F can be written as $\sum_i r_i e^{s_i}$ with the r_i and s_i rational. This either vanishes trivially or is a polynomial in $e^{1/m}$ where m is the lcm of the denominators of the s_i. This is impossible since $e^{1/m}$ is never algebraic for $m \neq 0$. □

Proposition 7. *Let* $\mathbb{K} \equiv \mathbb{Q}[x_1, \ldots, x_n, \sin(x_1), \cos(x_1), \ldots, \sin(x_n), \sin(x_n)]$ *and suppose* $F \in \mathbb{K}^n$. *If* $F_{\mathbf{B}}$ *has only finitely many isolated simple zeros then they can be counted by the Validate routine.*

Proof. Validate could be easily extended to functions $F : \mathbb{C}^n \to \mathbb{C}^n$ but remaining with the reals, a similar argument to the above applies. Each component of $F(\alpha)$ for rational α can be written as a polynomial in $\mathbb{Q}[\sin(1/m), \cos(1/m)]$ for some $m \in \mathbb{Z}$. Either sin or cos can be eliminated using resultants. $F(\alpha) = 0$ only

if the resulting polynomial is trivially zero (since $\sin(1/m)$ is never algebraic for rational arguments). □

In [Mai98] and [Mai00], Maignan shows that how to count zeros of systems such as these in the one and two dimensional case using Sturm sequences. Her papers also deal with the case of an infinite box.

Proposition 8. *Let $F : \mathbb{R}^n \to \mathbb{R}^n$ be Lipshitz continuous on a box $\mathbf{B}$ and further suppose that*

1. *F has only simple isolated zeros (i.e. $F(X) = 0$ implies $\det J(F, X) \neq 0$);*
2. *for some $d < \dim \mathbf{B}$ there is a means of deciding if a box $\mathbf{X} \subset \mathbf{B}$ with $\dim \mathbf{X} \leq d$ contains exactly one root of F if it contains at most one.*

then algorithm Validate can be modified by replacing the test

```
1    if dim Y = 0
2        if F(Y) = 0   T ⟵ T ∪ {Y}
3        next loop
4    end if
```

at lines 9–12 by

```
1    if dim Y ≤ d
2        if F(Y) has exactly one zero in Y   T ⟵ T ∪ {Y}
3        next loop
4    end if
```

Proof. The original version of *Validate* is able to test $F(\mathbf{Y}) = 0$ since F is a polynomial and if $\dim \mathbf{Y} = 0$, $\mathbf{Y}$ is a rational point. No other properties of polynomials are used. □

Even where $\mathbf{B}$ may contain singular zeros, *Validate* may be usable:

Corollary 2. *Suppose* $F : \mathbf{B} \subset \mathbb{R}^n \to \mathbb{R}^n$ *is Lipshitz continuous on* $\mathbf{B}$ *and the second condition of Prop. 8 above holds. If in addition there is a computable lower bound* $b > 0$ *such that for all* $\breve{B} \in \mathbf{B}$, $\det J(F, \breve{B}) \neq 0$ *implies* $|\det J(F, \breve{B})| \geq b$ *then it is possible to determine whether* $F_{\mathbf{B}}$ *defines a point.*

Proof. Let $[u, v] \equiv \det J(F, \mathbf{B})$. If $0 \notin [u, v]$ then Lemma 4 applies, if not consider the function $F' = (f_1, \ldots, f_n, 1 - x_{n+1} \det J(F))$. Clearly $J(F') = -J(F)^2$ and, by construction, $F'(X) = 0$ implies $J(F, X) \neq 0$. Thus at any solution of F', either $x_{n+1} \in [\min(1/v, 1/b), \max(1/v, 1/b)]$ or $x_{n+1} \in [\min(-1/b, 1/u), \max(-1/b, 1/u)]$. The proposition can be applied to the two cases separately and the results combined. □

This approach can be used for polynomial systems with one of the lower bounds from Chapter 6.

2.6 Conclusion

This chapter has demonstrated the possibility of using interval arithmetic combined with interval root finding techniques as a form of exact arithmetic. From a theoretical point of view all calculations can be done in $\mathbb{Q}$ even those involving transcendental functions.

In practice it is convenient to use arbitrary precision floating point arithmetic. Extensions based on floating point values are easy to compute (assuming facilities for rounding calculations up or down are available). The only problem is where intervals are created which have widths indistinguishable from zero. When this

happens it can be resolved by increasing the precision and then repeating the calculation.

Our implementations of interval arithmetic (in MuPAD, Maple and using the C++ GiNaC library) all support both rational and floating point arithmetic, swapping to floating point if operations not yielding exact rational values are used. Though no check has been made, the speed of even software floating point seems to outweigh the greater precision gained from using rational values.

The main alternative to the method for validating boxes used here seems to be based on computing the topological degree (also done using interval arithmetic) which is described in [Abe94, Abe98]. This has been extended by [KDN00] to the more difficult case of singular zeros of complex systems. Its disadvantage is a practical one: it proceeds by bisection where root finding methods can converge quadratically (though of course in the worst case they don't).

No attempt has yet been made to compare the methods. It is expected that the new method will do considerably better in the general case. Aberth's method computes the topological degree using bisection of boxes. The method here has a single step of comparable complexity: computing the Krawczyk operator rather than the determinant. However the 'unreasonable efficiency' of the iteration means that at a single step the volume of the box being tested is reduced by a factor of $\approx w(\mathbf{Y})^2$ rather than halved as in Aberth's algorithm.

Other methods of root counting exist but most are primarily symbolic in operation. For specifically polynomial systems a range of methods is possible. In the univariate case, Sturm sequences could be used. [Ped91a, Ped91b] extends these to the multivariate case. An alternative is described by [Can91]. A more modern

approach is through homotopic methods [GLW99].

Chapter 3

Algebraic Notation and Basic Results

This chapter summarises the notation used in the rest of the report and basic results which are used throughout the text and which seem more sensibly grouped together rather than be introduced at their first use.

3.1 Notation

The notation used for interval arithmetic is used here as far as possible. $\mathbf{B} \subset \mathbb{IR}^n$ is a box in $\mathbb{R}^n$ with rational endpoints and $\xi \in \mathbf{B}$ is a point defined by a polynomial mapping $P : \mathbb{R}^n \to \mathbb{R}^n$. It is convenient to ignore situations where any of the ξ may be zero and so it will be assumed that $0 \notin B_i$ for any i.

Capital letters (avoiding those with well established usages in algebraic number theory) are used for polynomial maps and lower case letters for individual polynomials.

Unless otherwise stated $P \equiv (p_1, \ldots, p_n)$ and $p_i \in \mathbb{Z}[x_1, \ldots, x_n]$ will be the system considered. The aim is to decide if $q(\xi) = 0$ ($q \in \mathbb{Z}[x_1, \ldots, x_n]$) given that $\det J(P, \xi) \neq 0$. The degree of polynomial p in variable x (the highest exponent of x) will be written as $\deg_x p$. Since $x_1, \ldots, x_n$ are commonly used as variables $\deg_i(p)$ is used for $\deg_{x_i} p$. Used without subscript $\deg p$ is the total degree of p. For a polynomial map, P the same notation refers to the maximum of the degrees over the components of P.

Symbols such as $\mathcal{X}$ and $\mathcal{Y}$ will are sometimes used to represent sets of monomials in $x_1, \ldots, x_n$ or $y_1, \ldots, y_n$ respectively. For $p \in \mathbb{Z}[x]$ we will write $p \equiv \sum p_i x^i$ with summation assumed to be over $0 \leq i \leq \deg(p)$. For $p \in \mathbb{Z}[x_1, \ldots, x_n]$ the notations $\sum_{k \in \mathbb{Z}^n} p_k \mathcal{X}^k$ or $\sum_{k \in \mathbb{Z}^n} p_k x_1^{k_1} \ldots x_n^{k_n}$ or simply $\sum p_k \mathcal{X}^k$ are used depending on which is felt to be clearest.

The notations $<P>$ and $<p_1, \ldots, p_n>$ are both used to refer to the ideal generated by the $p_1, \ldots, p_n$.

3.2 Measures of Size

For a polynomial $P = \sum_{0 \leq i \leq n} p_i x^i \in \mathbb{Z}[x]$ with roots $z_1, \ldots, z_n$, common measures of 'size' are:

$$\|p\|_k \equiv \sqrt[k]{\sum |p_i|^k}$$

$$L(p) \equiv \|p\|_1 \quad \text{'length'}$$

$$H(p) \equiv \|p\|_\infty \quad \text{'the usual height'}$$

$$M(p) \equiv p_n \prod \max(1, |z_i|) \quad \text{'Mahler measure'}$$

$$h(p) \equiv \log(M(p))/n \quad \text{'absolute logarithmic height'}$$

As with $\deg(\cdot)$, the formulæ have the obvious interpretations when p is replaced by an algebraic number α, e.g. $L(\alpha) \equiv L(Irr(\alpha))$ where $Irr(\alpha)$ is the minimal polynomial in $\mathbb{Z}[x]$ having α as a root and similarly for other measures. The table below (abbreviated from [Wal00, p114]) provides a convenient summary of the relationship between these measures for a polynomial p with degree d.

	$M(p)$	$H(p)$	$\|p\|_2$	$L(p)$
$M(p) \leq$	1	$\sqrt{d+1}$	1	1
$H(p) \leq$	$\binom{d}{\lceil d/2 \rceil}$	1	1	1
$\|p\|_2 \leq$	$\binom{2d}{d}^{1/2}$	$\sqrt{d+1}$	1	1
$L(p) \leq$	2^d	$d+1$	$\sqrt{d+1}$	1

Commonly these measures are given for monic polynomials, the effect of allowing $p_n \neq 1$ is to make some upper (lower) bounds larger (small) than they strictly need to be. Unfortunately in most cases there is insufficient information to do better.

3.3 Roots of Polynomials

Lower case Greek letters are used to represent algebraic numbers and vectors of numbers, e.g. $\xi = (\xi_1, \ldots, \xi_n)$. For any algebraic number α, $\mathrm{Irr}(\alpha)$ is the minimal polynomial of α, $\deg(\alpha)$ is its degree and $\lceil \alpha \rceil = \max(|\alpha^{(1)}|, \ldots, |\alpha^{(k)}|)$ where $\alpha = \alpha^{(1)}, \ldots, \alpha^{(k)}$ are the algebraic conjugates of α. $\mathrm{den}(\alpha)$ is the smallest integer d such that $d\alpha$ is an algebraic integer.

A significant part of what follows is based on estimating the sizes of roots of univariate polynomials with integer coefficients by estimating their heights (because

$M(\cdot)$ and so $h(\cdot)$ are generally harder to estimate). Mignotte [Mig92b, p146] provides a summary of estimates of which the most useful here is, if $p(z) = 0$ then

$$\frac{1}{1+H(p)} \leq |z| \leq H(p)+1 \tag{3.1}$$

For a few cases, where the coefficients are known, better estimates of $|z|$ could be used (indeed numerical methods could be used to obtain arbitrarily precise results) but for most case only a very crude estimate is known. Where $p \in \mathbb{Z}[x]$ it is often neater to use

$$\frac{1}{2H(p)} \leq |z| \leq 2H(p) \tag{3.2}$$

If even the height isn't known, a very crude bound for the smallest root ('smallest' and 'largest' always refer to absolute magnitude) is that if α the is any root then

$$|\alpha| \geq \frac{1}{\lceil \alpha \rceil^{\deg(\alpha)-1}}$$

3.4 Polynomials as Determinants

It frequently occurs in this report that a polynomial is given in the the form of a determinant. Usually there are matrices $A, B \in \mathbb{Z}^{n\times n}$ and the polynomial $\chi(z) = \det(Az+B)$ is of interest. It may be required to evaluate the determinant and the polynomial, or more usually, to bound its roots. In general the matrices may not be square in which case a maximal sub-matrix is sought (one which doesn't vanish identically for all z, and such that any larger sub-matrix does vanish identically).

When $B = I$ this is standard eigen-problem. Danilevsky's algorithm (described in

[Ham70] for example) is perhaps the earliest algorithm for this. More commonly conversion to Smith normal form is used which has the advantage of working in the general case ($B \neq I$ and not necessarily square). Modern Smith normal form algorithms are given in [Gie95a, Gie95b, GS02] and [Gie01] for the sparse case.

Solving the general eigenvalue problem numerically is described in [GL96, pp375-390]. Unfortunately the methods discussed there are not easily converted to an interval setting because of problems of dividing by intervals containing zero. LU decomposition (which requires only rational arithmetic) is an alternative, especially as $A^{-1}B$ can be efficiently computed from an LU decomposition of A [PTVF92]. [Tra98] computes one of the one-sided inverses of A (i.e. A need not be square nor non-singular) to solve this problem but it is not clear how this improves on LU decomposition.

Depending on what is known about the entries in A and B bounds on the roots of χ can be found. The bounds are mostly based on the following result

Lemma 8. *Let* $M(z) = (a_{ij}z + b_{ij})^{n\times n}$ *with* $a_{ij}, b_{ij} \in \mathbb{Z}$. *and write* $L_k = \sum_{1\leq i\leq n}|a_{ki}| + |b_{ki}|$. *If* $\chi(z) = \det M$, *then* $H(\chi) \leq \prod_i L_i$.

Proof. The result is trivially true for $n = 1$. Assume it is true for $k < n$ Let the elements in the first row of M be $a_1z + b_1, \ldots, a_nz + b_n$ and their cofactors be $A_1, \ldots, A_n$, if $A_{i,j}$ is the coefficient of z^j in A_i then the coefficient of z^k in $\det M$ is

$$M_k = \sum_{1\leq i\leq n} (-1)^{i+1}(a_iA_{i,k-1} + b_iA_{i,k})$$

or

$$\begin{aligned} |M_k| &\leq \sum_{1\leq i\leq n} (|a_i| + |b_i|) \max(|A_{i,k-1}|, |A_{i,k}|) \\ &\leq L_k \prod_{1\leq i<k} L_i = \prod_{1\leq i\leq k} L_i \end{aligned}$$

□

Note that this is just a variant of the Hadamard bound [Mig92b, p303] for the size of a matrix using the $\|\cdot\|_1$ row norm and defining $|az+b| = |a| + |b|$.

Corollary 3. *If* $|a_{ij}| \leq H_a$ *and* $|b_{ij}| \leq H_b$ *for all* i, j *then*

$$\begin{aligned} |M_k| &\leq n!(H_a + H_b)^n \\ &\leq n!\, 2^n \max(H_a, H_b)^n \end{aligned}$$

Proof. Referring to the previous proof, $L_i \leq i(H_a + H_b)$ since the minors in the expansion are successively narrower by one column. □

Corollary 4. *If* $\det M(\xi) = 0$ *and* $\xi \neq 0$,

$$\frac{1}{2\prod_i L_i} \leq |\xi| \leq 2\prod_i L_i \tag{3.3}$$

and

$$\frac{1}{2}\frac{(H_a + H_b)^{-n}}{n!} \leq |\xi| \leq 2n!(H_a + H_b)^n \tag{3.4}$$

In the simple eigenvalue case, $H_a = 1$, thus if $Iz + B = 0$ has a solution $z = \xi \neq 0$ then

$$|\xi| \geq \frac{1}{2}\frac{(1 + H_b)^{-n}}{n!}$$

3.5 The Bezout Number

The *Bezout number* is the number of complex zeros of a (zero dimensional) polynomial map. More generally it is the number of connected components of the variety generated by the map. If $P = (p_1, \ldots, p_n)$ with all $p_i \in \mathbb{Z}[x_1, \ldots, x_n]$ it is

$$N(P) \leq \prod \deg(p_i)$$

The estimate may be much greater than the Bezout number. An example which is important in the context of this report occurs when creating a system $P \cup \{1 - x_{n+1} \det J(P)\}$ to guarantee a finite number of roots. In general the degree of $\det J(P)$ is greater that that of P but the number of roots in the new system is no more than the estimate based on P alone would give.

The true value of $N(P)$ can be computed from the Newton polytope of the system [CLO98] or, for a zero dimensional ideal, by computing a Gröbner basis. More efficiently, a better bound can be obtained in many cases by *multi-homogenisation*, originally introduced in [MS87a, MS87b] in the context of solving equations by homotopy methods. The basic idea is to partition power products into sets and determine the degree in each element of each polynomial in the system. The permanent of this degree matrix formed in this way generally gives a lower bound than the product of degrees.

The simplest bound is used here since finding the true Bezout number will not always be practicable and, for gap functions, a bound is required based on input characteristics only.

The Bezout number is often bigger than strictly required by the problem. If the

ideal $<P>$ is decomposable there may exist points at which $P(\xi) = 0$ implies $q(\xi) = 0$ and others where it does not (i.e. $Q[\xi_1, \ldots, \xi_n]/<P(\xi)>$ is not isomorphic to $Q[x_1, \ldots, x_n]/<P>$). The perfect solution would be to know the Bezout number for each irreducible component of P.

Chapter 4

Special Cases

4.1 Introduction

Historically, finding lower bounds for algebraic numbers is more associated with the traditions of Algebraic Number Theory and Diophantine Approximation (both of which normally work in terms of univariate polynomials) than Algebraic Geometry (where more general representations are used). Though a goal of this research is to place as few restrictions as possible on how numbers are represented, the evidence to date suggests that this is incompatible with efficient zero detection. A practical approach may be to combine a more restrictive approach to representing the numbers but still to use lazy arithmetic for intermediate results. Even if the attempt is made to work with more general systems of equations, it is often worth checking for those special cases where better algorithms exist. Finally special cases often provide an insight into the more general scenario.

This chapter considers a number of special cases including results involving single values and points defined by univariate and triangular systems of equations. In doing so it is convenient also to discuss how a general set of polynomials might

be reduced to one of these special forms.

4.2 Univariate Representation

4.2.1 The Single Variable Case

It is convenient to consider initially the one dimensional version of the problem; that is, given polynomials $p, q \in \mathbb{Z}[x]$ and an interval $\mathbf{x}$ containing exactly one root ξ of p, decide if $q(\xi) = 0$. It is not necessary to assume that either p or q is irreducible but it is assumed that $p'(\xi) \neq 0$ and so ξ is not a zero of any repeated factor of p.

There is an easy algebraic method of solution in this case. Let $g = \gcd(p, q)$ and $h = p/g$, then either $g(\xi) \neq 0$ (implying $q(\xi) \neq 0$) or $h(\xi) \neq 0$ (implying the converse). Thus an interval $\mathbf{x}^* \subset \mathbf{x}$ can be found for which either $0 \notin g(\mathbf{x}^*)$ or $0 \notin h(\mathbf{x}^*)$.

The simplest numerical method is probably the following

Proposition 9. *For $p, q \in \mathbb{Z}[x]$ with $\deg(p) = d$, $\deg(q) = m$, with $p(\xi) = 0$ and $\lceil \xi \rceil \leq k$ then if $q(\xi) \neq 0$*

$$|q(\xi)| > A^{1-d} \tag{4.1}$$

where $A = \mathrm{den}(p)^m \sum_{0 \leq i \leq m} |q_i| k^i$

Proof. $\mathrm{den}(p)^m q(\xi)$ is an algebraic integer and so $\lceil \mathrm{den}(p)^m q(\xi) \rceil \geq 1$ and the smallest root is $\geq \lceil \mathrm{den}(p)^m q(\xi) \rceil^{1-d} = \mathrm{den}(p)^{m(1-d)} \lceil q(\xi) \rceil^{1-d}$. Since $\lceil \xi \rceil \leq k$, $\lceil q(\xi) \rceil \leq \sum_{0 \leq i \leq m} |q_i| k^i$ □

An alternative algorithm giving a better lower bound is described and analysed by Johnson [Joh92] based on the following result

Theorem 2 (Collins and Loos). *Let* $p, q \in \mathbb{Z}[x]$ *with* $\deg(p) = d$, $\deg(q) = m$. *If* $p(\xi) = 0$ *and* $q(\xi) \neq 0$ *then*

$$|q(\xi)| > \frac{1}{2} L(p)^{-m} (L(q) + 1)^{-d} \tag{4.2}$$

Proof. Johnson cites [CL76] for a proof of Eq. 4.2 by consideration of the lower bound of the resultant matrix of $res_x(p(x), z - q(x))$. However the result is also a direct application of Eq. 3.3. The resultant matrix has the first m rows containing the (shifted) coefficients of p and the remaining rows contain $z - q_0, q_1, \ldots, q_m$. □

Comparing this with Eq. 4.1,

$$\begin{aligned} A &= \max(1, \mathrm{den}(p)^m \sum_{0 \leq i \leq m} |q_i| k^i) \\ &\leq \mathrm{den}(p)^m L(q) L(p)^m \end{aligned}$$

implying $|q(\xi)| \geq L(q)^{1-d} (\mathrm{den}(p) L(p))^{m(1-d)}$ which is considerably weaker in general (though see the example below).

Take

$$\begin{aligned} p_1 &= x^3 + 11x^2 + 17x - 44 \\ p_2 &= 3x^4 + 21x^3 - 35x^2 - 14x + 22 \end{aligned} \tag{4.3}$$

with $\xi \approx 1.32$ and $\lceil \xi \rceil \leq 8.4$.

$$|p_2(\xi)| \geq 1.2 \times 10^{-9} \text{ Using Eq. 4.1}$$

$$|p_2(\xi)| \geq 1/2 L(p_1)^{-\deg p_2} (1 + L(p_2))^{-\deg p1}$$

$$\approx 1.9 \times 10^{-14} \text{ Using Eq. 4.2}$$

(In fact p_1 and p_2 have a common factor: $x^2 + 7x - 11$ of which ξ is a root.)

4.2.2 Equality of Algebraic Numbers

Finding lower bounds for $|\alpha - \beta|$ with algebraic α and β provides a natural extension to the Thue-Siegel-Roth theorem [Wal00, p8] and has been studied extensively. A summary of current knowledge can be found in [Eve00a] and [Eve00b]. Unfortunately many of bounds are non-constructive. Schmidt [Sch83] (quoted in [Eve00b]) conjectured that, for any $\delta > 0$ in any fixed algebraic field, there are only finitely many α and β satisfying

$$|\alpha - \beta| < \max(M(\alpha), M(\beta))^{-2-\delta} \tag{4.4}$$

which makes the link with the Thue-Siegel-Roth theorem explicit[1]. If correct this gives a probability 1 test for $\alpha = \beta$ by choosing any value for δ.

Using Liouville's theorem it is easy to show that

$$|\alpha - \beta| \geq (2M(\alpha))^{-\deg(\beta)}(2M(\beta))^{-\deg(\alpha)} \tag{4.5}$$

where $M(\cdot)$ is the Mahler measure. This can be expressed in terms of the more usual height/length by using $M(\alpha) \leq H(\alpha)\sqrt{\deg(\alpha)+1}$ or $M(\alpha) \leq L(\alpha)$.

By a different method Stolarsky [Sto74, p40] gives a result which in the two variable case is very close to Eq. 4.5 as does Mignotte [Mig92a].

It is also possible to reduce the problem to two single variable problems.

[1] We assume the claim is for α and β in a given algebraic number field.

Proposition 10. *Let* $d_i = \deg(p_i)$ *then* $\xi_1 = \xi_2$ *if* $|\xi_1 - \xi_2| < d_1^{-(d_1+2)/2}\|p_1\|_2^{1-d_1}$ *and* $|p_1(\xi_2)| < \frac{1}{2}L(p_2)^{-d_1}(L(p_1)+1)^{-d_2}$.

Proof. Trivially $p_1(\xi_1) = p_1(\xi_2) = 0$ is necessary if $\xi_1 = \xi_2$. Thus ξ_2 is also a root of p_1 which is tested by the first condition. For $\xi_1 \neq \xi_2$ the second equation above defines a minimum for the root separation [Mig92b, p168] □

A result which pre-figures the methods of Chapter 5 is

Proposition 11. *For* $i = 1, 2$ *suppose* $p_i(\xi_i) = 0$ *where* $p_i(x_i) = \sum_{0 \le j \le d_i} p_{ij} x_i^j$ *and write* $k_i = p_{id_i}$ *(assumed to be positive), then* $\xi_1 \neq \xi_2$ *implies*

$$|\xi_1 - \xi_2| \geq \frac{3^{-(d_1-1)(d_2-1)+3}}{4K}(L(p_1)+k_1)^{-d_2}(L(p_2)+k_2)^{-d_1} \tag{4.6}$$

where $K = k_1k_2 + k_1H'(p2) + k_2H'(p_1)$ *and* $H'(p_i) = \max_{0 \le k < d_i}\{|p_{ik}|\}$.

Proof. Consider the set of equations $(z - x_1 + x_2)x_1^i x_2^j = 0$ for $0 \le i < d_i$, $0 \le j < d_2$. The equations can be partitioned into four sets:

1. $i < d_1 - 1$ and $j < d_2 - 1$: $(d_1 - 1)(d_2 - 1) - 2$ equations. These have the form $zx_1^i x_2^j - x_1^{i+1}x_2^j + x_1^i x_2^{j+1}$

2. $i = d_1 - 1$ and $j < d_2 - 1$: d_2 equations. For these

$$zx_1^{d_1-1}x_2^j - x_1^{d_1}x_2^j + x_1^{d_1-1}x_2 \equiv zx_1^{d_1-1}x_2^j + x_1^{d_1-1}x_2^{j+1} + \frac{x_2^j}{k_1}\sum_{0 \le j < d_1} p_{1j}x_1^j$$

or

$$k_1 zx_1^{d_1-1}x_2^j + k_1 x_1^{d_1-1}x_2^{j+1} + x_2^j \sum_{0 \le k < d_1} p_{1k}x_1^k = 0$$

3. $i < d_1 - 1$ and $j = d_2 - 1$: d_1 equations with the form

$$k_2 z x_1^i x_2^{d_2-1} - k_2 x_2^{d_2-1} x_1^{i+1} - x_1^j \sum_{0 \le k < d_2} p_{2k} x_2^k = 0$$

4. $i = d_1 - 1$ and $j = d_2 - 1$: one equation:

$$k_1 k_2 z x_1^{d_1-1} x_2^{d_2-1} - k_2 x_2^{d_2-1} \sum_{0 \le k < d_1} p_{1k} x_1^k + k_1 x_1^{d_1-1} \sum_{0 \le k < d_2} p_{2k} x_2^k = 0$$

Treating these as a set of linear equations in the $x_1^i x_2^j$ leads to a matrix with $(d_1 - 1)(d_2 - 1) - 2$ rows of 'length' 3, d_2 of $k_1 + L(p_1)$, d_1 of $k_2 + L(p_2)$ and one of $k_1 L(p_2) + k_2 L(p_1) - k_1 k_2$.

Applying Eq. 3.3 almost gives the result with one small modification. In the proof of Eq. 3.3 no account is taken of the fact that the number of entries in the rows of the matrix minors decrease, thus rather than using the full length of the rows the sizes decrease. In particular, one row 'length' can be replaced by its height.

Taking the last equation gives a height of $\le k_1 k_2 + k_1 H'(p2) + k_2 H'(p_1)$. Also, a trivial gain, one row can be chosen to have only two elements, for this one of the rows from the first set is chosen, replacing a 3^{-1} by 2^{-1} and the result follows. □

Take $p_1 = x_1^3 + 11x_1^2 + 17x_1 - 44$, $p_2 = 3x_2^4 + 21x_2^3 - 35x_2^2 - 14x_2 + 22$ with ξ_1 and $\xi_2 \approx 1.32$ (in fact p_1 and p_2 have a common factor: $x^2 + 7x - 11$). $L(p_1) = 73$, $L(p_2) = 95$

Assuming $\xi_1 \neq \xi_2$, using Eq. 4.5

$$|\xi_1 - \xi_2| > \approx 3.2 \times 10^{-16}$$

Using Prop. 10, $\xi_1 = \xi_2$ if

$$|p_1(\xi_2)| < \frac{1}{2}95^{-3}74^{-4} \approx 1.9 \times 10^{-14} \text{ and}$$
$$|\xi_1 - \xi_2| < 3^{-5/2}48.5^{-2} \approx 2.7 \times 10^{-5}$$

Using Eq. 4.6

$$|\xi_1 - \xi_2| > \approx 1.9 \times 10^{-18}$$

4.2.3 The General Univariate Case

Hur [HD99] and Strzeboński [Str97] describe resultant based arithmetic for the case of $q \in \mathbb{Z}[x_1, \ldots, x_n]$ and $P_i \in \mathbb{Z}[x_i]$ for $1 \leq i \leq n$. Their concern is more with arithmetic and with identifying the polynomial representing the result by p-adic (Hur) or interval (Strzeboński) methods. Since they produce a characteristic polynomial for $q(\xi)$, detecting zero is a trivial issue.

Both authors use a term by term approach to finding the the final polynomial, a multivariate resultant method could be used to eliminate $n-1$ variables from q reducing it to the single variable case but this is hardly less expensive.

The field operations involve the greatest amount of computation. Strzeboński [Str97] (see also [HD99]) suggests using resultants. For example, to add α, β, defined by polynomials p and q respectively, he computes $\text{Res}_y(p(x-y), q(y))$ which has amongst its roots $\alpha + \beta$. By factorising the resultant and isolating numerically the factor having $\alpha + \beta$ as a root a representation for the sum is found. The other field operations are defined similarly. This is a costly way of doing arithmetic. The cost of resultant calculation is $O(d^3)$ where d is the sum

of the degrees of the polynomials involved. Typically the degree of the resultant is the product of the degrees of its constituent polynomials. Thus if $\alpha, \beta, \gamma \in \mathbb{A}$ are all of degree n, a simplistic evaluation of $\alpha\beta - (\beta + \gamma)\alpha + \alpha\gamma$ requires six resultant computations with intermediate polynomials having degrees as high as n^6 being factorised before the final collapse to zero.

A simple numerical method would be to extend Eq. 4.1.

Proposition 12. *Let* $q = \sum_{t\in\mathbb{Z}^n} q_t x_1^{t_1} \dots x_n^{t_n}$ *and let* K_i *be an upper bound for the roots of* p_i *then*

$$|q(\xi)| > 1/A^{D-1} \tag{4.7}$$

where $A = \max(1, \sum_{t\in\mathbb{Z}^n} |q_t| K_1^{t_1} \dots K_n^{t_n})$ *and* $D = \prod_i \deg(\xi_i)$

The chapter on resultant methods below could extend the result at Eq. 4.2 to this case, but the bound is generally weaker.

Another gap function is provided by Liouville's Theorem [Wal00, p83]

Theorem 3 (Liouville). *Let* $d_i = \deg(\xi_i), N_i = \deg_i(q)$ *and* $D = \prod d_i$ *then*

$$|q(\xi)| \geq L(q)^{1-D} \prod_i M(\xi_i)^{-DN_i/d_i} \tag{4.8}$$

As usual, $M(\xi_i)$ can be replaced with $L(\xi_i)$, alternatively the Dandelin-Graeffe iteration [Mig92b, p155] can be used to compute a rational upper bound for $M(\xi_i)$

In Example 1.3 $p_1 = x_1^2 - 2$, $p_2 = x_2^4 - 18x_2^2 + 49$, $q = x_2 - 2x_1 - 1$

Using Eq. 4.7: $|q(\xi)| > \approx 8.1 \times 10^{-15}$

Using Eq. 4.8: $|q(\xi)| > \approx 1.6 \times 10^{-10}$

These are comparable because the system is so small. For larger system Liouville's result is usually better.

4.2.4 Obtaining a Univariate Representation

Given a general systems S how may an equivalent univariate system P be produced? By 'equivalent' we mean a system which represents ξ, i.e. $\xi \in V(P) \subset V(S)$ (that is P need not include all zeros of S as long as it includes ξ). Two possibilities would be to use algebraic techniques such as multivariate resultants or Gröbner bases, these are briefly mentioned in later sections. An alternative, more in keeping with the ethos of this thesis, is numerical: reconstruct the polynomial representation of each ξ_i from its approximate value $\widehat{\xi_i}$, i.e. if $\deg \xi_i = d$ then find integers $a_0, \ldots, a_d$, not all zero, such that $|\sum_{0 \le i \le d} a_i \widehat{\xi_i}^i|$ is small. This gives a putative univariate representation of ξ_i. Of course such a polynomial may not be correct but the lemma below shows how the representation can be verified or a 'better' one found.

$a_0, \ldots, a_d$ can be found using 'lattice reduction algorithm' such as PSLQ [FBA99, BP98] or LLL [Poh93, pp11–26]. (We have found PSLQ simpler to implement and use.)

Using PSLQ a putative univariate representation P for ξ can be found. To confirm that P is a correct representation we need

Lemma 9. *Let $S_{\mathbf{B}}$ represent ξ and for $P = \{p_i | 1 \leq i \leq n,\ p_i \in \mathbb{Z}[x_1, \ldots, x_n]\}$ let $P_{\mathbf{B}}$ represent a point α. If $S_i(\alpha) = 0$ for all S_i then $\alpha = \xi$*

Proof. Since $S_i(\alpha) = 0$ for all S_i, $S(\alpha) = 0$ but ξ is the only zero of S in $\mathbf{B}$ so $\alpha = \xi$. □

In the case where the p_i are to be univariate this gives an effective technique to find a P which represents ξ.

1. Use PSLQ to find a candidate polynomials p_i for each ξ_i.

2. If the root represented by $P_{\mathbf{B}}$ is α check that $S_i(\alpha) = 0$ for each S_i using any of the above methods.

3. If $S(\alpha) = 0$ the algorithm terminates

4. If $S(\alpha) \neq 0$ then for at least one p_i, $p_i(\xi) \neq 0$. By numerical methods refine $\mathbf{B}$ until the non-zero polynomial is found.

5. Reapply the PLSQ algorithm to get a new approximation and repeat from step 2.

The Bezout number $N(S)$ gives a bound on the degree of the first polynomial to be extracted. The PSLQ algorithm's time complexity is polynomial in N (and hence exponential in n).

4.3 Rational Representation

An algebraic number field $\mathbb{Q}[\xi_1, \ldots, \xi_n]$ contains a *primitive element [Lan65, p183]* $\eta = \sum c_i \xi_i$, $c_i \in \mathbb{Q}$ such that $\xi_i = v_i(\eta)$ where $v_i \in \mathbb{Z}(\eta)$. If the minimal polynomial of η is $f(t)$, the polynomial system $(p_1, \ldots, p_n)$ can be replaced by a *rational representation*[2] $(f(t), x_1 - v_1(t), \ldots, x_n - v_n(t))$. Such a representation reduces zero testing to the univariate case: replace each variable in q by the appropriate rational expression to obtain a rational expression in t and use, for example, the bound from Eq. 4.2.

Obtaining a rational representation can be done in a number of ways, most obviously by the use of Gröbner bases (see Chapter 5 below) once a primitive element has been found. Fortunately, with probability one, any integer linear combination of the x_i will be a primitive element. Unfortunately, as illustrated in [ABRW96], representations found by such a method often have very large coefficients.

Several improvements have been suggested, Rouillier [Rou99] describes a method without using Gröbner bases to obtain a representation

$$(f(t), x_1 - g_1(t)/g(t), \ldots, x_n - g_n(t)/g(t))$$

with $f, g, g_1, \ldots, g_n \in \mathbb{Z}[t]$. This reduces the amount of work by requiring only one denominator to be found. Once f and g are known, the algebraic method used by Rouillier could replaced by numeric methods similar to the conversion to univariate form.

Rouillier's approach can still lead to polynomials with very large coefficients. [ABRW96] shows that the polynomial f typically has small coefficients and a

[2]Something of the rather combative history of the development of this idea can be found in [GH]. An early description of the method by Macaulay is available as a reprint: [Mac16].

representation

$$(f(t), x_1 - g_1(t)/f'(t), \ldots, x_n - g_n(t)/f'(t))$$

exists and can be constructed.

A modern probabilistic algorithm derived from Kronecker's idea, which is claimed to work well in practice, is given in [GLS]. It finds a representation incrementally and never involves working with more that bivariate polynomials.

Rational representation has a major advantage: it very simply reduces problems in n variables to a single variable case. However, there is no evidence to suggest that, for this problem at least, a conversion to rational form is simpler than conversion to univariate form.

4.4 Triangular Systems

A *triangular system* $P = \{p_1, \ldots, p_n\}$ is understood to be one in which for each i there is at most one $p_i \in \mathbb{Z}[x_1, \ldots, x_i] \setminus \mathbb{Z}[x_1, \ldots, x_{i-1}]$. The form of triangular systems makes them convenient for computing numerically the solutions of sets of equations. An important sub-class are systems derived from nested radical expressions such as $\sqrt{9 + 4\sqrt{2}} - 1 + 2\sqrt{2}$.

Nested radical expressions built from $\mathbb{Z}$ using $+, -, \times, /$ and$\sqrt{n}\cdot$ are an important class of expressions because of their frequent appearance in geometric problems.

They are also of interest because it is easy to find putative identities such as

$$\begin{aligned}
\sqrt{9+4\sqrt{2}} &= 1+2\sqrt{2} \\
\sqrt{5+2\sqrt{6}}+\sqrt{5-2\sqrt{6}} &= 2\sqrt{3} \\
\sqrt{16-2\sqrt{29}+2\sqrt{55-10\sqrt{29}}} &= \sqrt{22+2\sqrt{5}} \\
&\quad -\sqrt{11+2\sqrt{29}}+\sqrt{5} \\
\sqrt[3]{\sqrt[5]{32/5}-\sqrt[5]{27/5}} &= (1+\sqrt[5]{3}-\sqrt[5]{3}^2)/\sqrt[5]{25} \\
\sqrt{112+70\sqrt{2}+(46+34\sqrt{2})\sqrt{5}} &= 5+4\sqrt{2}+(3+\sqrt{2})\sqrt{5}
\end{aligned}$$

(from [DST93, p93]) which are easy to prove using pen and paper but which are non-trivial to resolve by a symbolic algorithm. The link with triangular systems is clear when, say, deciding if $\sqrt{9+4\sqrt{2}} = 1+2\sqrt{2}$ is rephrased as asking whether

$$x_1^2 - 2 = 0 \quad \text{where } x_1 \in [1.41, 1.42]$$

$$x_2^2 - 9 - 4x_1 = 0 \quad \text{where } x_2 \in [3.82, 3.84]$$

implies $x_2 - 2x_1 - 1 = 0$ (this is the example from the introduction).

One method of deciding if a division free nested radical expression is zero has been re-invented a number of times, including by the authors of [Li01] or [BFMS00] (and of this report). If e is a nested radical expression, a bound on its algebraic degree is easily computed

$$\deg e \leq \begin{cases} 1 & \text{if } e \in \mathbb{Z} \\ \deg a \ \deg b & \text{if } e = a \pm b \text{ or } a \times b \\ n \deg a & \text{if } e = \sqrt[n]{a} \end{cases}$$

as is an upper bound on the largest absolute root value, $\lceil e \rceil$:

$$\lceil e \rceil \leq \begin{cases} |e| & \text{if } e \in \mathbb{Z} \\ \lceil a \rceil + \lceil b \rceil & \text{if } e = a \pm b \\ \lceil a \rceil \lceil b \rceil & \text{if } e = a \times b \\ \sqrt[n]{\lceil a \rceil} & \text{if } e = \sqrt[n]{a} \end{cases}$$

Without division, a nested radical expression always represents an algebraic integer and so, as in Eq. 4.1 the bound

$$|e| > \lceil e \rceil^{1-\deg e}$$

can be used. If division is included and, the expression is defined, the method also works when

$$\boxed{a/b} \leq \boxed{a}\boxed{b}^{\deg b-1}$$

is added to the definition of $\boxed{e}$.

The same idea can be extended to more general triangular systems.

Lemma 10. *Let P be a triangular system with $P(\xi) = 0$ then* $\deg \xi_i \leq D_i := \prod_{1\leq j\leq i} d_j$ *where* $d_j = \deg_j p_j$.

Proof. The result is trivially true for $i = 1$. Assume it is true for $i < k$ and consider

$$F := \prod_{c_1,\ldots,c_{k-1}} p_k(\xi_1^{(c_1)}, \ldots, \xi_{k-1}^{(c_{k-1})}, x_k)$$

where the c_j run over all conjugates of ξ_j.

By the hypothesis the coefficients of p_k will have degrees $\leq D_{k-1}$ and there will be D_{k-1} terms in the product, i.e. $F \in \mathbb{Q}$ since its coefficients are symmetric polynomials over all conjugates of the $\xi_1, \ldots, \xi_{k-1}$. Since $\deg F \leq \deg_k(p_k)D_{k-1}$ the result follows. □

Corollary 5. *Let P be a triangular system then* $\deg q(\xi) \leq D_n$ *(i.e. the degree of $q(\xi)$ depends on the degree in x_i of the i^{th} polynomial, not on the degrees of x_i appearing elsewhere in the system).*

Proof. By consideration of $S \cup \{x_{n+1} - q\}$ with $\xi_{n+1} = q(\xi)$. □

Using this, bounds on $\boxed{\xi_i}$ can be found sequentially. $\boxed{\xi_1}$ is bounded by any root bound of P_1. Suppose bounds on $\boxed{\xi_1}, \ldots, \boxed{\xi_{k-1}}$ have been found. Since

the coefficients of P_k are in $\mathbb{Z}[x_1, \ldots, x_{k-1}]$ it is possible to find the first non-zero coefficient (alternatively a lower bound for any non-zero coefficient can be found). The upper bounds can be be found easily and from this an upper bound for ξ_k as a root bound from $P_k(\lceil \xi_1 \rceil, \ldots, \lceil \xi_{k-1} \rceil)$. Eq. 4.7 then gives a lower bound for q.

Lemma 11. *If P is a triangular system and for $P_k \in \mathbb{Z}[x_1, \ldots, x_k]$, $P_k = \sum p_{ki} x_k^i$ and $\max_i(\lceil p_{ki} \rceil) \leq H_k$ then $\lceil \xi_k \rceil \leq 2H_k^{D_{k-1}}$ where $D_0 = 1$, $D_i = \deg \xi_i \; D_{i-1}$.*

Proof. The result is trivial for $k = 1$ as it is just a root bound for ξ_1. Assume it is true for $\xi_1, \ldots, \xi_{k-1}$. At least one of the $p_{ki}(\xi_1, \ldots, \xi_{k-1}) \neq 0$ (since $P(\xi)$ is an isolated zero). Assume that the highest degree coefficient which is not zero is the s^{th}. Then $P_k(\xi_1, \ldots, \xi_{k-1}, x_k) = \sum_{0 \leq i \leq s} p_{ki}(\xi_1, \ldots, \xi_{k-1}) x_k^i$ and

$$\lceil \xi_k \rceil \leq 2 |\max_i(p_{ki}(\xi_1, \ldots, \xi_{k-1}))/p_{ks}(\xi_1, \ldots, \xi_{k-1})|$$

but $|p_{ks}(\xi_1, \ldots, \xi_{k-1})| \geq H_k^{1-D_{k-1}}$ and the result follows. □

As with the rational representation, there is no obvious benefit in converting a non-triangular system to triangular form. On the other hand numerous methods of converting to triangular form exists including the use of Gröbner bases and Wu-Ritt characteristic sets. For a comparison of several decomposition see [AM99].

4.5 Gap Functions for General Systems

This chapter has largely used *ad hoc* methods to give gap functions for particular forms of polynomials. In general the gaps predicted are much smaller than the true lower bounds. Not surprisingly when general systems are considered it is harder to obtain a gap function and the few known estimates are even weaker.

Subsequent chapters give several gap functions for more general cases but one example is included here (because it doesn't fit naturally elsewhere).

If $<P>$ is known to generate an irreducible ideal, an effective version of the Hilbert Nullstellensatz provides a gap function.

Theorem 4 (Krick *et al*). *Let $p_1, \ldots, p_s \in \mathbb{Z}[x_1, \ldots, x_n]$ be polynomials without a common zero in $\mathbb{C}^n$. Set $d := \max_i \deg(p_i)$ and $h := \max_i h(p_i)$ where $h(p) \equiv \log H(p)$ then there exist $a \in \mathbb{Z} \setminus \{0\}$ and $g_1, \ldots, g_s \in \mathbb{Z}[x_1, \ldots, x_n]$ such that*

$$a = g_1 p_1 + \cdots + g_s p_s$$

$$\deg(g_i) \leq 4nd^n$$

$$h(g_i) \leq 4n(n+1)d^n(h + \log s + (n+7)\log(n+1)d)$$

Corollary 6. *Writing p_0 for q with $P = \{p_1, \ldots, p_n\}$ set $d := \max_{0 \leq i \leq n} \deg(p_i)$, $h \equiv \max_{0 \leq i \leq n} h(p_i)$ and $K := \max_{0 \leq i \leq n}\{1, |\xi_i|\}$. If $<P>$ is irreducible and $q(\xi) \neq 0$ then*

$$|q(\xi)| \geq \frac{1}{HK^N}$$

where

$$N = 4nd^n \text{ and}$$

$$H = \binom{N+n}{n} 4n(n+1)d^n(h + \log(n+1) + (n+7)\log(n+1)d)$$

Proof. Consider the ideal $<P \cup \{q\}>$, since $<P>$ is, by hypothesis, irreducible then either $q \in \sqrt{<P>}$ or q has no common zero with with P. If $q(\xi) \neq 0$ the latter case must hold and so, as above, there exist $g_0, \ldots, g_n \in \mathbb{Z}[x_1, \ldots, x_n]$ such that

$$a = g_0 p_0 + g_1 p_1 + \cdots + g_n p_n$$

with a a non-zero integer. Each $p_i(\xi) = 0$ so

$$|a| = |g_0(\xi)||q(\xi)| \quad \text{or}$$

$$|q(\xi)| \geq 1/|g_0(\xi)|$$

Since $\deg(g_0) \leq N$ it can have at most $\binom{N+n}{n}$ monomials each of which are $\leq K$ at ξ. The rest of the H is the bound on the height of g_0. □

This doesn't fit the examples well as they don't generate an irreducible ideal. However factorising p_2 in Eq. 1.3 gives a modified example:

In Example 1.3 modified $\quad p_1 = x_1^2 - 2, \;\; p_2 = x_2^2 - 2x_2 - 7, \;\; q = x_2 - 2x_1 - 1$

$$N = 32, \; K \leq 3.9, \; H = \binom{34}{4} 96(7 + \log 3 + 9 \log 6) \approx 10^8$$
$$\text{or } |q(\xi)| \geq 0.1 \times 10^{-26}$$

This is a weak bound, both in magnitude and in requiring <P> to be irreducible. On the other hand it doesn't require P to be zero dimensional and has the interesting feature that it is the only gap function we know of at a specific zero of P.

4.6 Conclusion

The aim of this chapter was to illustrate some rather *ad hoc* techniques for particular cases and to introduce ideas which will be treated more generally in later Chapters. It is interesting to note how few bounds seem to be known - and how weak such bounds are.

In a sense such weaknesses are inherent in the methods adopted. Tests for zeros usually don't depend on the behaviour or a particular algebraic number, $q(\xi)$,

but on a worst case analysis of the equations defining ξ. With the exception of the bound due to Krick *et al*, none of the methods take any notice of what is known about the value of ξ but instead view it in terms of $\lceil \xi \rceil$. And in turn, even $\lceil \xi \rceil$ is derived from properties of its defining polynomial. It is only once a bound is known that numerical methods re-introduce ξ to do the evaluation of $q(\xi)$.

In Chapter 6 a well known bound for a general system (due to Canny) is given and an improved bound is derived (based on a slightly different analysis of the Macaulay resultant). Though the new bound could be improved up in obvious ways it suffers the same flaws as those described above. Indeed because of the generality of the systems it relates to it is considerably cruder.

This chapter has also described ways of producing more tractable representation of numbers. The ideal of lazy computation, espoused in the introduction, had two facets: a general representation of algebraic numbers and arithmetic and zero testing without complex symbolic computation. Though methods of dealing with general systems are discussed in later chapters it seems likely that the first of these is unrealistic.

Chapter 5

Elimination Methods

5.1 Characteristic Equations

If we accept, for the moment at least, that useful gap functions for general systems P are beyond reach, what alternative methods exist? One possibility is to find the characteristic polynomial $\chi(z)$ of q such that $P(\xi) = 0$ *iff* $\chi(q(\xi)) = 0$ (in fact it is sufficient that the minimal polynomial of $q(\xi)$ divide χ). If $q(\xi) = 0$ then $\chi(z) \equiv z^k h(z)$ $(k > 0)$ and $h(q(\xi)) \neq 0$. To prove that $q(\xi) = 0$ is then reduced to showing that $|q(\xi)| < c$ where c is the magnitude of the smallest root of h (for which bounds are well known). But for some sufficiently small box $\mathbf{B}$ enclosing ξ either $0 \notin q(\mathbf{B})$ (and so $q(\xi) \neq 0$) or $|q(\xi)| < c$.

There are three obvious ways by which a suitable χ might be found: using Gröbner bases, multivariate resultants or the characteristic set methods associated originally with Ritt and rediscovered by Wen-tsün Wu [Wu78]. From the perspective of this report the Wu-Ritt method is less useful because of its emphasis on symbolic triangularisation methods (which are also possible with Gröbner bases). Comprehensive details of Wu-Ritt methods are available in [WBC01] with a Maple

software library at [Wan].

5.2 Gröbner Basis Methods

Familiarity with the elementary properties of Gröbner bases is assumed. For details see, e.g. [CLO92, AL94]. $G(P)$ signifies any Gröbner basis of $<P>$. None of the results which follow depend on which monomial ordering is used. Only the following properties of Gröbner bases are required (details can be found in either of the books mentioned).

Lemma 12. *If $<P>$ is 0-dimensional then:*
a) for $1 \leq i \leq n$, one of the generators of $G(P)$ has a leading power product of the form $x_i^{d_i}$ for some d_i,
b) A set of n polynomials in which there is a polynomial with a leading term of the form $x_i^{d_i}$ for $1 \leq i \leq n$ is a Gröbner basis, and
c) $\mathbb{Q}[x_1, \ldots x_n]/<P>$ is a finite dimensional vector space with dimension equal to the number of zeros of P, counted with multiplicity.

If $G(P)$ is known then by computing $q \equiv 0 \pmod{<G(P)>}$ *iff* $q \in <P>$ It will therefore be assumed that $q \notin <P>$.

One way to use Gröbner bases in this context would be to compute $G(P \cup \{z-q\})$ where z is a new variable. By using an appropriate elimination ordering it is possible to obtain a polynomial in $\mathbb{Q}[z]$ whose roots correspond to the values taken by q at zeros of P. (In practice $G(P \cup \{1-zq\})$ is a better choice.) Unfortunately this has several disadvantages: adding an extra variable to a polynomial system can have a drastic effect on the time taken to compute the basis; elimination

orders typically require longer computation times than other orderings; a new basis computation is required each time a sign is to be determined; and finally, in the radical expression case, P will already be a Gröbner basis in $x_1, \ldots, x_n$ - a fact which should be exploited if possible.

There is an alternative. From Lemma 12, polynomials in $\mathbb{Q}[x_1, \ldots x_n]$ form a vector space with basis set of power products, $S \subset \{x_1^{r_1} x_2^{r_2} \ldots x_n^{r_n} | r_i \in \mathbb{N}, 1 \leq i \leq n\}$. [Möl93] and [MS95] showed how this could be used to solve polynomial equations[1].

5.3 Polynomials as Linear Operators

It is simpler to work with polynomials over $\mathbb{Z}$ and equations whose roots are algebraic integers rather than $\mathbb{Q}$ and arbitrary numbers in this section. Such a conversion is easy to make with a Gröbner basis and it is assumed subsequently that it has been done. Thus every generator of the basis is monic.

If $G = \{g_1, \ldots g_m\}$ is a Gröbner basis then any $p \in \mathbb{Z}[x_1, \ldots x_n]$ can be written as

$$p = \sum_{i=1}^{m} c_i g_i + r \text{ where } c_i, r \in \mathbb{Z}[x_1, \ldots x_n]$$

where r is the unique remainder after p is successively divided by each of the g_i. The remainder will be written as $[p]$. If $p = [p]$ we will say that p is reduced.

Lemma 12 a) implies that only a finite number of different monomials can appear in a reduced polynomial. In particular no monomials divisible by $x_i^{d_i}$ are present since they are divisible by one of the generators of $G(P)$. By computing each

[1]Since the first draft of this paper, an excellent description of their methods has appeared in the book "Using Algebraic Geometry" [CLO98]

monomial from 1 to $x_1^{d_1-1}x_2^{d_2-1}\dots x_n^{d_n-1}$ and retaining those which are already reduced, a basis $U = (u_1, u_2, \dots u_s)$ can be constructed for $\mathbb{Z}[x_1, \dots x_n]/<P>$. Every reduced polynomial can be considered as a vector with basis U.

In Example 1.1 $p_1 = x_1^2 - 2$, $p_2 = x_2^2 - 9 - 4x_1$, $q = x_2 - 2x_1 - 1$

---

p_1 and p_2 form a Gröbner basis in lex order with $x_2 > x_1$ and define a vector space spanned by $(1, x_1, x_2, x_1x_2)$.

Now for any $u_i, u_j \in U$

$$[u_iu_j] = \sum_{k=1}^{s} a_k u_k \text{ with } a_k \in \mathbb{Q}$$

and for any $q \in \mathbb{Q}[x_1, \dots x_n]/<P>$

$$q \equiv \sum_{j=1}^{s} q_j u_j \text{ where } q_j \in \mathbb{Q}$$

and for any $u_i \in U$

$$\begin{aligned}[qu_i] &= \sum_{j=1}^{s} q_j [u_j u_i] \\ &= \sum_{j=1}^{s} a_{ij} u_j \text{ for some } a_{ij} \in \mathbb{Q}\end{aligned}$$

This can be rewritten as the matrix equation

$$qU = \mathcal{Q}U \text{ where } \mathcal{Q} = (a_{ij}) \in \mathbb{Q}^{s \times s}$$

Lemma 13. *If the zeros of P are $\{\alpha_1, \dots \alpha_k\}$ then the eigenvalues of the equation $\mathcal{Q}U = \lambda U$ are $\{q(\alpha_1), \dots q(\alpha_k)\}$.*

Proof. by Lemma 12 c), $k = |U|$, thus the equation has k eigenvalues. It is therefore only necessary to show that each $q(\alpha)$ is an eigenvalue for each $\alpha \in$

$\{\alpha_1, \ldots \alpha_k\}$. By the definition of $[\cdot]$

$$qu_i = \sum_{j=1}^{m} b_{ij} g_j + [qu_i]$$

If $P(\alpha) = 0$ then $g_j(\alpha) = 0$ for $1 \leq i \leq m$ giving

$$\begin{aligned} q(\alpha)u_i(\alpha) &= [qu_i](\alpha) \\ &= \sum_{j=1}^{s} c_{ij} u_j(\alpha) \end{aligned}$$

but taking this over all i is equivalent to $\mathcal{Q}U(\alpha) = q(\alpha)U(\alpha)$. Thus $q(\alpha)$ is an eigenvalue. □

In Example 1.1 $p_1 = x_1^2 - 2$, $p_2 = x_2^2 - 9 - 4x_1$, $q = x_2 - 2x_1 - 1$

The matrix representation of $q \equiv x_2 - 2x_1 - 1$ is

$$Q = \begin{pmatrix} -1 & -2 & 1 & 0 \\ -4 & -1 & 0 & 1 \\ 9 & 4 & -1 & -2 \\ 8 & 9 & -4 & -1 \end{pmatrix}$$

The characteristic equation is

$$\begin{aligned} \chi(z) &= z^4 + 4z^3 - 28z^2 \\ &= z^2(z^2 + 4z - 28) \end{aligned}$$

The smallest non-zero root is $\geq 1/(H(\chi) + 1) = 1/29 \approx 0.034$. Using the bounds given in the example, $q(\xi) \in [-0.02, 0.02]$ and so $q(\xi) = 0$.

For a larger example, the characteristic equation could be extracted by converting the matrix to Jordan normal form. A close to optimal algorithm is given in [GS02] which requires $O(s^4(\log s + \log\|Q\|))$ operations where $s = \#U$. However s is the number of zeros of P and is $O(2^{2^n})$ which is doubly exponential in the input size.

5.4 Avoiding the Characteristic Equation

Direct computation of the characteristic polynomial is impractical for all but the smallest cases. However bounds for its coefficients can be obtained using the Hadamard inequality.

Proposition 13. *If q is represented by a matrix Q and $q(\xi) \neq 0$ then*

$$|q(\xi)| > \frac{1}{2\|Q\|_1^s}$$

Proof. From Hadamard's inequality $\det Q \leq \|Q\|_1^s$ and no coefficient of χ is greater than the determinant itself. □

alternatively

Proposition 14. *Let T_i be the maximum absolute sum obtained by taking the $s-1$ largest entries of row i and let $T = \max_{1\leq i\leq s} T_i$. Then $q(\xi) \neq 0$ implies*

$$|q(\xi)| \geq \frac{1}{sT^{s-1}}$$

Proof. In place of the eigenvalues of $Q - qI = 0$ consider those of $Q^{-1} - (1/q)I = 0$. Since Q is an integer matrix its determinant has absolute value ≥ 1 and so the elements of its inverse are $\leq T^{s-1}$ (applying Hadamard's result to any minor of Q). A standard result from linear algebra that if λ is an eigenvalue of an $s \times s$ matrix M then $|\lambda| \leq s\|M\|_\infty$ and so

$$|1/q(\xi)| \leq sT^{s-1}$$

□

For the example, the first bound is $> .001$ and the second > 0.0025. Since $s = O(2^{(2^n)})$ the bounds for larger problems rapidly become impractically small.

Note that it is not necessary to actually create the matrix if the bounds are to be used. As each row is found, only the maximum of the norms is needed. An alternative idea, which seems initially appealing, is to compute, just once, the (sparse) matrix representation for each x_i. (In fact all the matrices can be computed simultaneously by finding the representation for $\sum_i u_i x_i$ where the u_i are unspecified parameters.) The matrix for q can then be produced by evaluating it over the ring of matrix representations. Of course the matrix representation of x_i is just its companion matrix and, if that were known, methods more appropriate to having univariate representation might be used.

5.5 Univariate Systems

Finally, in theory a lower bound could be found by bounding the height of each matrix representation of a variable. This has not been possible in the general case but the situation is easier for univariate systems.

A polynomial system $P = (p_1, \ldots, p_n)$ where $p_i \in \mathbb{Z}[x_i]$ is already a Gröbner basis under any admissible monomial ordering. As before it will be assumed that the p_i are monic. Monomials forming a basis of $\mathbb{Z}[x_1, \ldots, x_n]/<P>$ are $M = \{\prod x_i^{n_i} | \forall i, 0 \leq n_i < d_i = \deg(p_i)\}$. As before $s = \prod d_j$ is the dimension of M.

Let X_i be the matrix of x_i. Then for $m \in M$,

$$[mx_i]_P = \begin{cases} mx_i & \text{if} \deg_i m \neq d_i - 1 \\ \frac{m}{x_i^{d_i-1}}(p_i - x_i^{d_i}) & \text{otherwise} \end{cases}$$

Thus rows in X_i contain a single one or are some permutation of the coefficients of p_i and consequently $\|\prod_i X_i^{k_i}\|_1 \leq \prod_i L(p_i)^{k_i}$ and thus if $\deg_i(q) = n_i$,

$$\|Q\|_1 \leq L(q) \prod_i L(p_i)^{n_i}$$

which could be combined with Prop. 13 to give a bound based solely on the input parameters for the univariate case - but unfortunately one which is weaker than Liouville's Eq. 4.8.

5.6 Computing Upper Bounds for q

Not only are the lower bounds computed by the method above small but the matrix $\mathcal{Q}$ has to be recomputed for each q. Consider instead finding the matrices $\mathcal{X}_i$ for $1 \leq i \leq n$ where $\mathcal{X}_i U = x_i U$, i.e. eigen-equations for the individual roots. From these it would be possible to find T_i such that $|\alpha_i| \leq T_i$ where α is any zero of S. Now for any such α we have

$$q(\alpha) \equiv \sum_{s \in \mathbb{Z}^n} q_s \prod_{1 \leq i \leq n} \alpha_i^{s_i}$$

$$\text{or } |q(\alpha)| \leq \sum_{s \in \mathbb{Z}^n} |q_s| \prod_{1 \leq i \leq n} T_i^{s_i}$$

and so we have a value for T with the advantage that the T_i have only to be computed once.

Computing T_i doesn't require all of $\mathcal{X}_i$ to be found. If G includes the generator $x_i^{d_i} + \ldots$ then only rows of $\mathcal{X}_i$corresponding to monomials with $x_i^{d_i-1}$ as a factor need be calculated.

5.7 Nested Radical Expressions

The cost of computing a Gröbner basis is high which limits the usefulness of the techniques described here. Nested radical expressions are a special case as they give rise to equations of the required form without further processing and the greatest part of the cost can be in computing the matrix representation. Rather than explicitly finding the rows of the matrix to obtain T and D for the lower bound formula, both parameters can be estimated directly from the basis.

A nested radical expression gives rise to a set of n equations in unknowns $x_1, \ldots x_n$ which can be ordered to give

$$x_1^{d_1} - r_1 = 0$$

$$\ldots$$

$$x_n^{d_n} - r_n = 0$$

where $r_i \in \mathbb{Q}[x_1, \ldots x_{i-1}] = \sum_{s \in \mathbb{Z}^{i-1}} r_{i,s} \prod_{1 \leq j < i} x_j^{s_j}$. For this form estimates of the T_i can be computed directly using

$$T_1 \leq |r_1|^{1/d_1} \leq |r_{1,0}|^{1/d_1}$$

$$T_i \leq |r_i|^{1/d_i} \leq \sum_{s \in \mathbb{Z}^{i-1}} |r_{i,s}| \prod_{1 \leq j < i} T_j^{s_j}$$

As an example, consider an example mentioned earlier (due to Ramanujan).

$$\sqrt[3]{\sqrt[5]{32/5} - \sqrt[5]{27/5}} = (1 + \sqrt[5]{3} - \sqrt[5]{3}^2)/\sqrt[5]{25} \tag{5.1}$$

Converted to a system of equations this becomes:

$$5x_1^5 - 32 = 0 \quad \text{or } x_1 = \sqrt[5]{32/5}$$

$$5x_2^5 - 27 = 0 \quad \text{or } x_2 = \sqrt[5]{27/5}$$

$$x_3^5 - 3 = 0 \quad \text{or } x_3 = \sqrt[5]{3}$$

$$25x_4^3 - x_1 + x_2 = 0 \quad \text{or } x_4 = \sqrt[5]{\frac{x_1 - x_2}{25}}$$

$$\text{with } q = x_4 - 1 - x_3 + x_3^2$$

This gives

$$T_1 < 1.45, \quad T_2 < 1.41$$

$$T_3 < 1.38, \quad T_4 < 0.49$$

$$\text{giving } T < 0.49 + 1 + 1.38 + 1.38^2) \approx 5$$

This gives a lower bound for $|q(\xi)| > 0$ of about 5^{-375}, an impracticably small number for calculations, but of interest in illustrating that, in the nested radical case at least, deciding the sign number is theoretically amenable to entirely numerical methods.

5.8 A Digression

If the ξ is known to be the only root of P within an n-sphere, a slight modification to q allows the decision on whether $q(\xi) = 0$ to be 'read off' from the characteristic equation.

Proposition 15. *Let $\xi^* \in \mathbb{Q}^n$ be an approximate rational zero of S with $S(\xi^*) \neq 0$ and ξ the only zero of S in an open ball $B(\xi^*, r)$. Define*

$$d := r^2 - \sum_{1 \leq i \leq n} (x_i - \xi_i^*)^2$$

and let χ be the characteristic equation of $g = dq^2$, then $q(\xi) = 0$ if and only if the variation in sign of the coefficients of χ is even.

Proof. Consider the roots α_i of S at which $g(\alpha_i)$ is real. ξ is one and is the only real root at which d is positive. If at some complex α_i, $g(\alpha_i)$ is real then so is $g(\overline{\alpha_i})$. $\overline{\alpha_i}$ is also a complex root of S. Thus complex roots of S at which g is real lead to repeated roots in χ.

Descartes' Rule of Signs [Mig92b, p197] states that the parity of the number of strictly positive roots of a polynomial is the same as that of the sign variation of the polynomial's coefficients. Now if $g(\xi) \neq 0$ the only positive real roots of of χ are $g(\xi)$ itself plus an even number of roots corresponding to complex roots of S at which g is real. Thus $g(\xi) \neq 0$ implies $q(\xi) \neq 0$ and that the sign variation in χ must be odd. $\square$

Chapter 6

Resultant Methods

6.1 The Macaulay Resultant

6.1.1 Canny's Gap Function

Resultant methods provide the main practical alternative to using Gröbner bases for extracting characteristic equations. Probably the best known multivariant resultant formulation is that of Macaulay [Mac16] which is used by Canny [Can88, p70] to obtain the following result (in the notation of this report):

Theorem 5 (Canny). *Let* $P = (p_1, \ldots, p_n)$ *with* $p_i \in \mathbb{Z}[x_1, \ldots, x_n]$ *have only finitely many solutions. If* $P(\xi) = 0$, $\deg P = D$ *and* $H(p) = H$ *then* $\xi_j \neq 0$ *implies*

$$|\xi_j| > (3DH)^{-nD^n}$$

It is not difficult to find systems with bounds of this form, for example the equations $Hx_1 - 1 = 0, x_i - x_{i-1}^D = 0$ for $i = 2, \ldots, n$ has $x_n = H^{-D^{(n-1)}}$ as a

solution. Note that Canny's prediction is exponentially (in n) less than the true answer. Applying this to the $n+1$ polynomials $P \cup \{z-q\}$ gives

Corollary 7. *Let* $\deg(P \cup \{z-q\}) = D$ *and* $H(P \cup \{z-q\}) = H$ *then* $q(\xi) = 0$ *or*

$$|q(\xi)| > (3DH)^{-(n+1)D^{n+1}} \tag{6.1}$$

In Example 1.2 $p_1 = x_1x_2 - x_1 - 4$, $p_2 = x_2^2 - 9 - 4x_1$, $q = x_2 - 2x_1 - 1$

$$D = 2, \quad H = 9, \quad n = 2$$

$$|q(\xi)| > 54^{-24} \approx 2.6 \times 10^{-42}$$

Comparing this with the Gröbner basis results, a basis has $s \leq D^n$ elements (i.e. the maximum number of solutions of P) but a good estimate for the height of the matrix for q is lacking and one as small as $(3DH)^n$ seems unlikely to be found.

6.1.2 A New Gap Function

The Macaulay resultant generalises the Sylvester resultant to the n variable case. Its construction (based on [CLO98, p97], modified slightly) will only be described for the special case of interest here, polynomials $p_0 = z - q, p_1, \ldots, p_n$, with $p_1, \ldots, p_n, q \in \mathbb{Z}[x_1, \ldots, x_n]$. Let $d_i = \deg p_i$, $d = \sum_{0 \leq i \leq n}(d_i - 1) + 1 = \sum_{0 \leq i \leq n} d_i - n$ and let $\mathcal{X} = \{\prod x_i^{a_i} | \sum a_i \leq d\}$.

$\mathcal{X}$ is partitioned into $n+1$ sets, $S_0 = \{\alpha \in \mathcal{X} \ : \ \deg \alpha \leq d - d_0\}$ and $S_k = \{\alpha \in \mathcal{X} \setminus \bigcup_{1 \leq i < k} S_i \ : \ \alpha / x_i^{d_i} \in \mathcal{X}\}$ for $1 \leq k \leq n$.

The set of equations $\{\alpha(z-q)=0\ :\ \alpha\in S_0\}\cup\{(\alpha/x_i^{d_i})p_i=0\ :\ \alpha\in S_i, 1\le i\le n\}$ has $\binom{d+n}{n}$ elements whose coefficients are the entries in the Macaulay resultant matrix.

Proposition 16. *Let* $S=\sum_{i=0}^{n}d_i$ *and* $L=\max_i\{L(p_i)\}$ *then if* $P(\xi)=0$ *and* $q(\xi)\neq 0$

$$|q(\xi)|\ge\frac{1}{2}(L(q)+1)^{-A}L^{-B} \tag{6.2}$$

where $A=\binom{S-d_0}{n}$ *and* $B=\binom{S}{n}-A$

Proof. This is an application of Eq. 3.3. It is only necessary to show that $\#S_0=\binom{S-d0}{n}$ and $\#\mathcal{X}=\binom{S}{n}$ and the same counting argument applies to both. It is convenient to homogenise $\mathcal{X}$ by introducing a dummy variable x_0 and making every monomial in $\mathcal{X}$ have total degree d. The total number of monomials is the number of ways that d objects can be partitioned into $n+1$ parts, i.e.

$$\binom{d+n}{n}=\binom{\sum_{i=0}^{n}d_i-n+n}{n}=\binom{S}{n}$$

The entries in S_0 are those in which $\deg_{x_0}\ge d_0$ which gives A. □

Note that for $n=1$ the above result reduces to Eq. 4.2.

In Example 1.2 $p_1=x_1x_2-x_1-4,\ p_2=x_2^2-9-4x_1,\ q=x_2-2x_1-1$

$$|q(\xi)|\ge\frac{1}{2}5^{-6}14^{-4}\approx 8.3\times 10^{-10}$$

Clearly it would be possible to compute a bound of the form

$$|q(\xi)|\ge\frac{1}{2}(L(q)+1)^{-\#S_0}\prod_{i=1}^{n}L(p_i)^{-\#S_i}$$

How do this bound and Canny's compare? Canny uses D the maximum total degree where Eq. 6.2 uses S, the sum of the degrees. Since $S < nD$ and $\binom{nD}{n} \leq (eD)^n$ a cruder bound would be

$$|q(\xi)| \geq \frac{1}{2}(L(P)+1)^{-(eD)^n} \tag{6.3}$$

This is close to Eq. 6.1 if $L(P)+1 \approx (3DH)^n$. Each polynomial has $\leq \binom{D+n}{n}$ terms (the number of different monomials with total degree $\leq D$, so

$$L(P) \leq H\binom{D+n}{n} \leq H(e(D+n)/n)^n \leq H(5eD/4)^n \leq H(4D)^n$$

(assuming $n \geq 2$ and $D \geq 2$). Thus

$$|q(\xi)| \geq \frac{1}{2}(2H)^{-(eD)^n}(4D))^{-n(eD)^n} \tag{6.4}$$

against Canny's

$$|q(\xi)| > (3DH)^{-(n+1)D^{n+1}}$$

which is exponentially worse as H increases. Actually the performance of Eq. 6.2 is usually considerably better than Eq. 6.4 since polynomials are generally sparse in the number of terms.

Extracting a characteristic equation from the Macaulay matrix would give much better bounds but for all but the simplest systems the size of the matrix is far larger than that produced by a Gröbner basis computation.

In Example 1.3 $p_1 = x_1^2 - 2$, $p_2 = x_2^4 - 18x_2^2 + 49$, $q = x_2 - 2x_1 - 1$

$d = 5$ and so $\#\mathcal{X} = \binom{5+2+1}{2}$ needing a 28×28 matrix for only 2 equations. Since $< p_1, p_2 >$ is a Göbner basis, an 8×8 matrix is sufficient.

6.1.3 Root Separation

In counting roots in a box, a knowledge of the minimum root separation would provide a criteria for deciding when a box contains at most one root. The gap function above provides a natural though weak formula

Proposition 17. *If* $< P >$ *is a 0-dimensional ideal with* $P(\alpha) = P(\beta) = 0$. *Let* $S = \sum_{i=1}^{n} \deg p_i$ *and* $L = \max_i\{L(p_i)\}$ *then* $\alpha \neq \beta$ *implies*

$$\|\alpha - \beta\|_2 \geq \frac{L^{-B}3^{-A}}{2} \tag{6.5}$$

where $A = \binom{2S}{2n}$ *and* $B = \binom{2S+1}{2n} - A$

Proof. This is a direct application of Eq. 6.2. Replace the system P by $P' = (p_1, \ldots, p_n, s_1, \ldots, s_n)$ where $s_i = p_i(y_1, \ldots, y_n)$, i.e. duplicate the p_i in a new set of polynomials, and set $q = x_k - y_k$ (k arbitrary) and $L(q) = 2$. The degrees of the polynomials in P' sum to $2S$. □

A cruder bound can be found by using $A = \binom{2S}{2n} \leq (eS/n)^{2n}$ and $B = \frac{2n}{2S+1-2n}\binom{2S}{2n} \leq \frac{n}{S-n}(eS/n)^{2n}$

6.2 The Multivariate Dixon Resultant

6.2.1 The Dixon Resultant

Dixon's original idea was to generalise the Bezoutian of univariate polynomials to the general case. The resultant so defined is sometimes known as the Dixon

resultant and sometimes as the multivariate Bezoutian; the former name is used here[1].

The Dixon resultant (see e.g. [KSY94, KS97, KS95]) is less well known than some other formulations. It has advantages, including generally yielding a smaller matrix, and disadvantages: it does not always give a square matrix and even when it does the determinant may vanish identically. Fortunately for the application considered here the disadvantages can be overcome.

The starting point is to lift the polynomials to a larger space by considering them as elements $\mathbb{Z}[z][x_1, \ldots, x_n, y_1, \ldots, y_n]$. We will use x for $(x_1, \ldots, x_n)$, and y for $(y_1, \ldots, y_n)$.

Definition 11. *For* $p(z, x) \in \mathbb{Z}[z][x_1, \ldots, x_n]$ *and* $0 \leq a \leq n$ *define* $p^{(a)} := p(y_1, \ldots, y_a, x_{a+1}, \ldots, x_n)$. *The* Dixon polynomial *of* $p_1, \ldots, p_{n+1}$ *is*

$$\mathcal{D}(z, x, y) \equiv \frac{1}{\prod_{1 \leq i \leq n} (x_i - y_i)} \begin{vmatrix} p_1^{(0)} & p_1^{(1)} & \cdots & \cdots & p_1^{(n)} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ p_{n+1}^{(0)} & p_{n+1}^{(1)} & \cdots & \cdots & p_{n+1}^{(n)} \end{vmatrix} \tag{6.6}$$

It is easily seen that $\mathcal{D}(z, x, y)$ is indeed a polynomial since substituting y_i for x_i within the determinant causes it to vanish (two adjacent columns become identical). Further, if $P(z^*, \alpha) = 0$ then $\mathcal{D}(z^*, \alpha, y) = 0$ independently of the values of the y_i.

[1] Arguments about names are usually sterile, particularly so in this case. Dixon generalised the Bezoutian from the single variable case to three variables. Bezout's supporters argue that Dixon didn't consider the general n variable case - though as far as we can discover, neither did Bezout. To add further obscurity, early text books (such as Burnside and Panton [BP81, pp322-331]) ascribe the Bezoutian concept to Euler and describe a different elimination method as being due to Bezout!

$\mathcal{D}(z,x,y)$ can be written in polynomial or matrix form as

$$\begin{aligned}\mathcal{D}(z,x,y) &= \sum_{\alpha,\beta\in\mathbb{Z}^n} c_{\alpha,\beta}\mathbf{x}^\beta\mathbf{y}^\alpha \\ &= X^T M(z) Y\end{aligned}$$

where $\mathbf{y}^\alpha \equiv \prod_{1\le i\le n} y_i^{\alpha_i}$, $\mathbf{x}^\beta \equiv \prod_{1\le i\le n} x_i^{\beta_i}$ and X (Y) is a vector of the monomials $\mathbf{x}^\beta$ ($\mathbf{y}^\alpha$) appearing in $\mathcal{D}(z,x,y)$.

We will sometimes consider the polynomial $\mathcal{D}(z,x,y)$ as a bilinear form in X and Y. For the bilinear form, we will write it as $\mathcal{D}(z,X,Y)$. It is also sometimes convenient to write $\Delta_k p \equiv (p^{(k)} - p^{(k-1)})/(x_k - y_k)$.

6.2.2 Applying the Dixon Resultant

Consider the Dixon polynomial obtained from P with $p_{n+1} = 1 - zq$.

$$\begin{aligned}&\mathcal{D}_P(z,x,y) = \\ &\frac{1}{\prod_{1\le i\le n}(x_i - y_i)}\begin{vmatrix} p_1^{(0)} & p_1^{(1)} & \dots & p_1^{(n)} \\ \dots & \dots & \dots & \dots \\ p_n^{(0)} & p_n^{(1)} & \dots & p_n^{(n)} \\ 1 - zq^{(0)} & 1 - zq^{(1)} & \dots & 1 - zq^{(n)} \end{vmatrix} \\ &= X^T M_P(z) Y\end{aligned}$$

In the sequel $X(\xi)$ will be the vector X evaluated at ξ and similarly for $Y(\xi)$. The dimensions of $M_P(z)$ will be $N_r \times N_c$. Thus the number of monomials in X is N_r, and the number of monomials in Y is N_c.

The standard resultant argument would be to show that if $q(\xi) \neq 0$ then $\mathcal{D}(1/q(\xi), X(\xi), Y) = 0$ and so, by evaluating $\det(M_P(z))$, find values for $q(\xi)$. Unfortunately $M_P(z)$ is not necessarily square and so may have no determinant. Even when it is square, the determinant is often identically zero and so a more indirect approach is needed.

We will show that there exists a sub-matrix $H(z)$ of $M_P(z)$ so that $H(z)$ is non singular for generic z, but is singular if $q(\xi) \neq 0$ and $z = 1/q(\xi)$. By finding a bound for $|H_{ij}|$ we produce an upper bound for $|1/q(\xi)|$ and so a lower bound for $|q(\xi)|$. Specifically, since H is a sub-matrix of $M_P(z)$ we shall show:

Proposition 18. *If* $N = \min(N_r, N_c)$ *and* $h = \max_{i,j}(|M_P(z)|)$ *(where* $\max(|a + bz|) \equiv \max(|a|, |b|)$*) then* $q(\xi) \neq 0$ *implies*

$$|q(\xi)| \geq \frac{1}{2}(h\sqrt{N})^{-N}$$

or equivalently

$$|\log_2(|q(\xi)|)| \leq 1 + N \log_2(h\sqrt{N}) \tag{6.7}$$

Definition 12. *We call a sub-matrix,* $\mathcal{H}$ *of* $M_P(z)$ maximal *if it is square and its determinant does not vanish identically (i.e. for all* z*) but that of any square sub-matrix of* $M_P(z)$ *containing* $\mathcal{H}$ *does vanish.*

Proposition 19. *Let* $\mathcal{H}(z)$ *be any maximal sub-matrix of* $M_P(z)$*. If* $q(\xi) \neq 0$ *then* $\det(\mathcal{H}(1/q(\xi))) = 0$.

Two preliminary result are required to prove the proposition.

Lemma 14. *If* $P(\xi) = 0$ *then* $\mathcal{D}_P(z, X(\xi), Y) = (1 - zq(\xi))B(Y)$ *and* $\mathcal{D}_P(z, X, Y(\xi)) = (1 - zq(\xi))B'(X)$*, where* B *and* B' *are linear in* Y *and* X *respectively, with coefficients which are polynomials in* ξ *and* $B(Y(\xi)) = B'(X(\xi)) = \det(J_P(\xi))$.

Proof. Since $p_i(\xi) = 0, 1 \le i \le n$

$$\begin{aligned}\mathcal{D}_P(z,\xi,y) &= \frac{1}{\prod\limits_{1\le i\le n}(\xi_i - y_i)} \begin{vmatrix} 0 & p_1^{(1)}(\xi) & \dots & p_1^{(n)}(\xi) \\ 0 & \dots & \dots & \dots \\ 0 & p_n^{(1)}(\xi) & \dots & p_n^{(n)}(\xi) \\ 1-zq(\xi) & \dots & \dots & \dots \end{vmatrix} \\ &= (-1)^n \frac{(1-zq(\xi))}{\prod\limits_{1\le i\le n}(\xi_i - y_i)} \begin{vmatrix} p_1^{(1)}(\xi) & \dots & p_1^{(n)}(\xi) \\ \dots & \dots & \dots \\ p_n^{(1)}(\xi) & \dots & p_n^{(n)}(\xi) \end{vmatrix} \\ &= (1-zq(\xi))B(Y)\end{aligned}$$

$\mathcal{D}_P(z,\xi,y) \equiv \mathcal{D}_P(z,X(\xi),Y)$ and so $B(Y)$ is linear in Y with coefficients which are polynomials in ξ. For the second part, subtracting adjacent columns

$$B(Y) = (-1)^n \begin{vmatrix} \Delta_1 p_1 & \dots & \Delta_n p_1 \\ \dots & \dots & \dots \\ \Delta_1 p_n & \dots & \Delta_n p_n \end{vmatrix}$$

(since $p_i^{(0)}(\xi) = 0$) but

$$\begin{aligned}&\lim_{\substack{y_i\to\xi_i \\ 1\le i\le n}} \Delta_j p = \frac{p^{(j)} - p^{(j-1)}}{\xi_j - y_j} = -\partial p/\partial x_j \mid_\xi \quad \text{so} \\ &\lim_{\substack{y_i\to\xi_i \\ 1\le i\le n}} B(Y) = \det(J_P(\xi))\end{aligned}$$

Exchanging the roles of x and y, the same argument gives $\mathcal{D}_P(z,X,Y(\xi)) = (1-zq(\xi))B'(X)$. □

Since, by hypothesis, $J_P(\xi) \neq 0$ it follows that $\mathcal{D}_P(z^*,\xi;\xi) = 0$ iff $z^* = 1/q(\xi)$.

Lemma 15. *Assume $q(\xi) \neq 0$. The rank of $M_P(z)$ at $z^* = 1/q(\xi)$ is at least 1 less than at generic z.*

Proof. Let $V = (V_1, \dots V_{N_c})$ be an $N_c \times N_c$ matrix with columns $V_1 = Y(\xi)$ and $V_2, \dots, V_{N_c}$ a basis spanning the space $\Gamma = \{a \in \mathbb{C}^{N_c} \mid B(a) = 0\}$. Since B is not identically zero and is linear in a such a basis exists and has dimension $N_c - 1$.

Further V_1 is linearly independent of $V_2, \ldots, V_{N_c}$ since otherwise we should have $V_1 = Y(\xi) = \sum_{2 \leq i \leq N_c} c_i V_i$ for some c_i not all zero implying

$$\begin{aligned} J_P(\xi) &= B(Y(\xi)) \\ &= B(\sum_{2 \leq i \leq N_c} c_i V_i) \\ &= \sum_{2 \leq i \leq N_c} c_i B(V_i) \\ &= 0 \end{aligned}$$

contradicting Lemma 14. Thus V is non-singular. V is also independent of z.

Let $U = (U_1, \ldots U_{N_r})$ be an $N_r \times N_r$ matrix with $U_1 = X(\xi)$ and $U_2, \ldots, U_{N_r}$ constructed in the same way as the V_i but this time spanning $\Gamma' = \{a \in C^{N_r} \mid B'(a) = 0\}$. U is non singular and independent of z.

Consider the matrix $M^* = U^T M_P V$ and let $\{u_i\}$ be the standard basis for $\mathbb{C}^{N_r}$ and $\{v_i\}$ that for $\mathbb{C}^{N_c}$. We have

$$\begin{aligned} u_i^T U^T M_P V v_j &= U_i^T \mathcal{M}_A V_j \\ &= u_i^T M^* v_j = M^*_{i,j}(z) \end{aligned}$$

By construction

$$\begin{aligned} M^*_{1j} &= U_1^T M_P V_j \\ &= X(\xi)^T M_P V_j \\ &= (1 - zq(\xi)) B(V_j) \\ &= (1 - zq(\xi)) \det(J_P(\xi)) \text{ if } j = 1 \text{ and 0 otherwise} \end{aligned}$$

and so the first row of $M^*(z)$ is zero apart from the first element. By the same argument $M^*_{i1} = 0$ if $i \neq 1$, and so the first column is zero except for the first element.

The rank of $M^*(z)$ is the same as that of $M_P(z)$, since they are related by non-singular transformations.

All the entries of $M^*(z)$ depend continuously on z. We suppose $q(\xi) \neq 0$. Consider the point $z = 1/q(\xi)$. Suppose the rank of M at this point is r. This means that there is an r by r non singular sub-matrix, which does not include any entries from the first row or first column, since these are all zero at the point. If z is moved slightly, this sub-matrix will still be non singular, but by appending appropriate entries from the first row and first column, a larger non singular sub-matrix can be constructed.

Thus any maximal sub-matrix of $M^*(z)$ includes M^*_{11} (since if it didn't we could append it and the appropriate entries from row/column 1 to get a larger sub-matrix.)

At $z = 1/q(\xi)$ the first row and column of $M^*(z)$ become zero and thus its rank, and consequently that of $M_P(z)$, drops by at least one. □

This proves Prop. 19 since the last lemma implies that the rank of any maximal sub-matrix will drop at $z^* = 1/q(\xi)$.

Proof of Prop. 18. For any z, M_P has rank no greater than $N = \min(N_r, N_c)$. Thus a maximal sub-matrix $\mathcal{H}$ has rank at most N.

Expanding $\det(\mathcal{H})$ gives a polynomial in z, $\chi(z) \equiv \sum_{0 \leq i \leq N} a_i z^i$ with $|a_i| \leq N^{N/2} h^N$ (using Hadamard's bound). Applying the bound from Eq. 3.2 gives $|z| < 2N^{N/2} h^N$ so $q(\xi) \neq 0$ implies

$$|q(\xi)| \geq \frac{1}{2}(h\sqrt{N})^{-N}$$

□

In Example 1.1 $p_1 = x_1^2 - 2$, $p_2 = x_2^2 - 9 - 4x_1$, $q = x_2 - 2x_1 - 1$

The resultant is

$$(1+z)x_1x_2 + 4zx_2 - 9zx_1 - 8z + (2zx_1x_2 - 4zx_1 + (1+z)x_2 - 9z)y_1$$
$$+ (-zx_1x_2 + (1+z)x_1 + 4z)y_2 + (1 - zx_2 + 2zx_1 + z)y_1y_2$$

giving $h = 9$, $N_r = N_c = 4$ and a lower bound of $\approx 9.0 \times 10^{-6}$ (a bound within reach of hardware floating point arithmetic but not impressive considering the true lower bound is ≈ 3.7).

One appeal of the Dixon resultant is that we wish to find a method to decide the sign of many polynomials from a given number field. The formula

$$\mathcal{D}_P(z,x,y) = \frac{1}{\prod_{1\le i\le n}(x_i - y_i)} \begin{vmatrix} z-u_0 & z-u_1 & \dots & z-u_n \\ p_1^{(0)} & p_1^{(1)} & \dots & p_1^{(n)} \\ \dots & \dots & \dots & \dots \\ p_n^{(0)} & p_n^{(1)} & \dots & p_n^{(n)} \end{vmatrix}$$

can, for small systems, be expanded and applied to different q by substituting $u_i = q^{(i)}$.

6.2.3 A Lower Bound from Input Parameters

One of the advantages of the Dixon resultant is that its size is often considerably less than its theoretical maximum. If, however, worst case behaviour is assumed a lower bound can be found by finding upper bounds for N and h directly from

the heights and degrees of $<p_1, \ldots, p_n, q>$. This gives, of course, a much worse estimate but does obviate the need to compute the resultant.

The following terms are used in what follows:

- $P \equiv \{p_1, \ldots, p_n, p_{n+1} = 1 - zq\}, p_i \in \mathbb{Z}[z][x_1, \ldots, x_n]$
- $d_i \equiv$ the largest exponent of x_i in P, $D \equiv \prod_{1 \leq i \leq n} d_i$
- $t_i \equiv$ the number of distinct monomials in p_i, $T \equiv \prod_{1 \leq i \leq n+1} t_i$
- For $p \equiv \sum_{\alpha,\beta \in \mathbb{Z}^n} c_{\alpha,\beta} \mathbf{x}^\beta \mathbf{y}^\alpha$, $\|p\| \equiv \max(|c_{\alpha,\beta}|)$ where $|a + bz| \equiv \max(|a|, |b|)$
- $H \equiv \prod_{1 \leq i \leq n+1} \|p_i\|$

Proposition 20. *i)* $N \leq n!D$ *and ii)* $h \leq (n+1)!HT$

Proof. i) In Eq. 6.6, x_i appears in the first i columns of the determinant and so its expansion contains no monomial with a power of x_i greater than $\delta_i = id_i$ (a better bound would be to define δ_i as the sum of the i highest powers of x_i appearing in the different polynomials). Dividing the determinant by $(x_i - y_i)$ reduces the degree by one and so the Dixon polynomial can include all monomials $\prod_{1 \leq i \leq n} x_i^{a_i}$ in the x_i with $a_i < \delta_i$. There are $\prod_{1 \leq i \leq n} \delta_i$ such monomials and so $N \leq n! \prod_{1 \leq i \leq n} d_i$.

ii) Write $p_1 = cm + r$ where m is an arbitrarily chosen monomial in p_1, c is its coefficient and r the rest of p_1. The determinant in Eq. 6.6 can be written

$$c \begin{vmatrix} m & m^{(1)} & \ldots & \ldots & m^{(n)} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ p_{n+1}^{(0)} & p_{n+1}^{(1)} & \ldots & \ldots & p_{n+1}^{(n)} \end{vmatrix}$$
$$+ \begin{vmatrix} r & r^{(1)} & \ldots & \ldots & r^{(n)} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ p_{n+1}^{(0)} & p_{n+1}^{(1)} & \ldots & \ldots & p_{n+1}^{(n)} \end{vmatrix}$$

Repeating this process for r and then for subsequent rows, the Dixon Polynomial can be written as

$$D_P = \frac{1}{\prod\limits_{1\le i\le n}(x_i - y_i)} \sum_{1\le i\le T} h_i D_i$$

where each h_i is a product of $n+1$ coefficients taken from the polynomials (i.e. $|h_i| \le H$) and the D_i are determinants following the Dixon pattern but with each entry a single monomial. If D is one of the D_i

$$\frac{D}{\prod\limits_{1\le i\le n}(x_i - y_i)} = \begin{vmatrix} m_1^{(0)} & \Delta_1 m_1 & \dots & \dots & \Delta_n m_1 \\ \dots & \dots & \dots & \dots & \dots \\ m_{n+1}^{(0)} & \Delta_1 m_{n+1} & \dots & \dots & \Delta_n m_{n+1} \end{vmatrix}$$

Now $\Delta_i m = m'(x_i^d - y_i^d)/(x_i - y_i)$ for some $d \le d_i$ where m' is a monomial in $y_1, \dots, y_{i-1}, x_{i+1}, \dots, x_n$. $(x_i^d - y_i^d)/(x_i - y_i) = \sum_{j+k=d-1} x_i^j y_i^k$ and so has at most d_i terms.

To determine the height we have to decide how many of the monomials in the expansion can be the same. We claim that at most $(n+1)!$ can be identical. In case $n = 1$ we have a 2×2 determinant and the result is clearly true: each of the two powers of x_1 in the first column can be multiplied by one term in column 2. Suppose it is true for $k-1$ variables. In the k variable case, y_k appears only in column $k+1$ and could appear to the same power in each of the rows. To obtain identical monomials we take equal powers of y_k and expand with respect to column $k+1$. Each of the minors is a version of the $k-1$ size case (except for a factor which is a power of x_k) and the result is proved by induction.

Combining these results, we conclude the height of the Dixon polynomial is $h \le (n+1)!HT$ □

For the example above this gives $N \le 8$ or $N \le 6$ (using the better bound) and $h \le 3888$ (the true value being 9). With these, the lower bound for the example

becomes $\approx 0.4 \times 10^{-32}$.

An obvious improvement here would be to take account of the fact that many of the monomial determinants will vanish as a result of having identical rows. A better value for T would be the number of ways of choosing one monomial from each of the p_i without duplication. There is no closed formula for this though it is easy to compute. In the example, it would reduce T from 18 to 10 ($h \leq 2160$) which is not a significant improvement.

6.2.4 The Univariate Case

In the case that $p_i \in \mathbb{Z}[x_i]$ with d_i the degree of p_i, a slightly better bound can be obtained. We assume $p_i(0) \neq 0$ but do not require the polynomials to be minimal. Let n_i be the degree of q in x_i, the highest degree in x_i of the Dixon polynomial is now $n_i + d_i - 1$ and the matrix M_z has rank $N \leq \prod_{1 \leq i \leq n}(n_i + d_i)$.

Using the same expansion as before, row i consists of polynomials with at most $d_i + 1$ terms if $1 \leq i \leq n$. If $1 - zq$ contains t monomials there will be $T = t\prod_{1 \leq i \leq n}(d_i + 1)$ determinants of monomials (since $t \leq \prod_{1 \leq i \leq n}(d_i + 1)$ we could also use $T = \prod_{1 \leq i \leq n}(d_i + 1)^2$).

Subtracting adjacent columns as before gives polynomials in which each column after the first contains only two non-zero entries and the same argument as used above gives at most 2^n identical monomials in the expansion of any one determinant. Thus $h \leq 2^n Ht \prod_{1 \leq i \leq n}(d_i + 1)$

6.2.5 The Dixon u-Resultant

Lemma 14 is specific to the particular use of Dixon resultants needed here: a non-singlar Jacobian at a point and an '$(n+1)^{\text{th}}$' polynomial of the form $z-q$ or $1-zq$. Taking for q the u-resultant polynomial $U \equiv \sum_{1\le i\le n} u_i x_i$ gives a Dixon polynomial:

$$\frac{1}{\prod\limits_{1\le i\le n}(x_i-y_i)}\begin{vmatrix} p_1^{(0)} & p_1^{(1)} & \dots & p_1^{(n)} \\ \dots & \dots & \dots & \dots \\ p_n^{(0)} & p_n^{(1)} & \dots & p_n^{(n)} \\ z-U & z-U^{(1)} & \dots & z-U^{(n)} \end{vmatrix}$$

$$= \begin{vmatrix} p_1^{(0)} & \Delta_1 p_1 & \dots & \Delta_n p_1 \\ \dots & \dots & \dots & \dots \\ p_n^{(0)} & \Delta_1 p_n & \dots & \Delta_n p_n \\ z-\sum u_i x_i & u_1 & \dots & u_n \end{vmatrix}$$

Setting $u_k = 1$ and $u_1, \dots, u_{k-1}, u_{k+1}, \dots, u_n$ to zero gives a means of computing a matrix representation for any x_k.

In Example 1.2 $p_1 = x_1x_2 - x_1 - 4$, $p_2 = x_2^2 - 9 - 4x_1$, $q = x_2 - 2x_1 - 1$

This gives a Dixon polynomial for $u_1 = 0$, $u_2 = 1$ of

$$(z-x_2)y_2^2 + ((1+z)x_2 - 4x_1 - z - 9)y_2 - 7 + 4x_1z - zx_2$$

and determinant (removing a zero column):

$$\begin{vmatrix} z & -1 & 0 \\ -z-9 & 1+z & 4 \\ -7 & -z & 4z \end{vmatrix}$$

or $z^3 + z^2 - 9z + 7 = 0$ which has the polynomial for x_2 as a factor

---

Chapter 7

Closed Form Expressions

In scientific and technical programming, as much as in Computer Algebra, there is a need for ways of deciding equality and performing sign determination. The conventional approach in floating point arithmetic is to substitute $|A-B| < \epsilon$ for some small ϵ for $A = B$. In effect, programmers are making use of an heuristic gap function. This chapter investigates what for a true gap function for this situation might be.

Blum, Shub and Smale in [BSS89, BCSS98] and other papers suggested a simple RAM model of scientific based on exact real arithmetic (here called the BSS model). Their aim could be described as modeling floating point computation as a perturbation of exact computation and as such they allowed the numbers in their system to take any real values (even non-computable ones) and assumed that comparison and arithmetic could be done exactly.

The BSS model also interested researchers with a different perspective. It suggested a new way of studying computation over other domains than the rationals, in particular the development of ways of extending complexity theory to other

domains (a survey can be found in [MM97]).

Rather than allowing real inputs, we consider a slight variant of the BSS model. Inputs are restricted to integers but, in addition to field operations, other functions are allowed and all arithmetic is assumed to be done exactly with real values. If the extra functions are $\exp(\cdot)$ and $\log(\cdot)$ the values computed by a BSS style program include most relevant to scientific programming.

7.1 Closed Form Calculation

Definition 13. *Let F be a countable set of function names, and $C \in \mathbb{C}$ a countable set of constants. $\mathbb{E}(F, C)$ is the set of expressions in F with over values in C and is defined as the smallest set such that $C \subset \mathbb{E}(F, C)$[1] and, if $f \in F$ is a function with arity n, then $f(e_1, \ldots, e_n) \in \mathbb{E}(F, C)$ for all $e_1, \ldots, e_n \in \mathbb{E}(F, C)$. $\mathbb{E}(F, C)$ is a set of* closed form expressions.

Definition 14. *For any $e \in \mathbb{E}(F, C)$, $\mathcal{V}(e)$ denotes the value of the expression if it is defined. If $e \in C$ then $\mathcal{V}(c)$ is the constant represented by c and if $e = f(e_1, \ldots, e_n)$ $\mathcal{V}(e)$ is the result of evaluating $f(\mathcal{V}(e_1), \ldots, \mathcal{V}(e_n))$ If $v \in \mathcal{V}(\mathbb{E}(F, C))$ then v will be called a* closed form number *[Cho99].*

For particular cases a more precise definition of $\mathcal{V}$ is needed - for example interpreting which branch to choose for complex valued functions.

With any $e \in \mathbb{E}(F, C)$ a integral *weight* $\mathcal{W}(e)$ will be associated. Crudely, this

[1] $\mathbb{E}(F, C)$ is a set of expressions, 'piles of ink on paper', rather than of numbers and as such it would be more accurate to replace C by 'representations of each constant'. However this seems to expand notation without any extra benefit to clarity. In general $c \in C$ will be used to denote both names and values of constants. Similarly for $f \in F$, f will be both 'the function' and its name.

could be the number of symbols needed to represent e. In some crude sense, very small or very large valued expressions are expected to have a large weight. The weights will always be chosen so that there are a finite number of expressions of a given weight, W, and so there exists

$$B(W) = \min_{e \in \mathbb{E}(F,C)} \{|e| \ : \ \mathcal{V}(e) \text{ is defined}, \ \mathcal{V}(e) \neq 0, \ w(e) \leq W\}$$

B is a gap function for $\mathbb{E}(F, C)$, for any defined expression, e, either $\mathcal{V}(e) = 0$ or $|\mathcal{V}(e)| \geq B(w(e))$.

Unfortunately little is known about appropriate definitions of weights or the corresponding gap functions for most useful classes of expression. Van der Hoeven [vdH00] has discussed the forms some might have and in [SvdH01] analyses the complexity of zero testing by using them.

Bounds for polynomial expressions have been the main topic of preceding chapters. A case closer to the spirit of this chapter is that of nested radical expressions which were considered in section §4.4 (and also below). Richardson [Ric00] proposes a particular gap function for expressions involving $\log(\cdot)$ and $\exp(\cdot)$ operations which is the basis of the work reported here.

In this chapter, several classes of expression are investigated. A simple gap function can be found for rational expressions formed from natural numbers using field operations only. This gives an upper bound for a gap function for bigger classes of expression and suggests a possible structure for such functions.

Extending this to radical expressions by introducing an n^{th}-root function gives a result equivalent to those given earlier. For expressions involving arbitrary algebraic numbers result is obtained which is similar to Liouville's but an expo-

nentially better result is hypothesised.

Finally, the hardest case, that described by Richardson is considered. Very little is known but some empirical research is described and some structural properties suggested.

7.2 Gap Function Structures

It is convenient to consider expressions in $\mathbb{E}(F, C)$ as trees in the usual way. A tree may be an element of C (a *leaf* or *external node*) or an *(internal) node* of the form $f(e_1, \ldots, e_n)$ where $e_1, \ldots, e_n \in \mathbb{E}(F, C)$ are sub-trees. In practice only binary and unary nodes will be used.

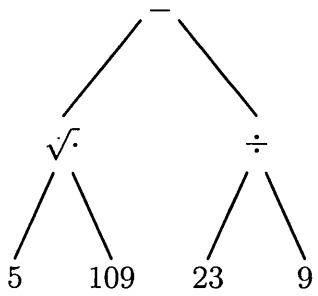

Figure 7.1: $\sqrt[5]{109} - \frac{23}{9}$ as a tree

The depth $d(e)$ of a tree (expression) is 0 if $e \in c$ and if $e = f(e_1, \ldots, e_n)$ it is $1 + \max\{d(e_1), \ldots d(e_n)\}$. The number of internal nodes of e is written $\#e$.

The value of any expression depends, in some way, on three factors: the size of the numbers appearing as leaves, the particular functions appearing at nodes, and the complexity of the tree structure.

Definition 15. $\mathcal{W} : \mathbb{E}(F, C) \to \mathbb{Z}^+$ *is a* weight function *if for all* $e \in \mathbb{E}(F, C)$

where $\mathcal{V}(e)$ is defined and non-zero

$$\mathcal{W}(e) \geq |e| \geq \mathcal{W}(e)^{-1} \tag{7.1}$$

and further, for all $f \in F$ of arity n the weight function can be written as

$$\mathcal{W}(f(e_1, \ldots, e_n) \equiv \mathcal{W}_f(\mathcal{W}(e_1), \ldots, \mathcal{W}(e_n))$$

for some function $W_f : \mathbb{Z}^n \to \mathbb{Z}^+$, i.e. the weight of an expression depends on the weights of its sub-expressions, not necessarily on their values. $\mathcal{W}(e)$ will be called the weight *of the expression.*

For any class $\mathbb{E}(F, C)$ and any $e \in \mathbb{E}(F, C)$, $\underline{\mathcal{W}}(e) = \prod_i w(c_i)$ where the c_i are the constants appearing in e, will be called the constant weight. *For $n \in \mathbb{Z}^+$ it will be assumed that $\mathcal{W}(n) = \underline{\mathcal{W}}(n) = \max(2, n)$*

The constant weight $\underline{\mathcal{W}}(\cdot)$ provides a way of separating the contribution to an expression of value of the numbers appearing in it from the complexity of the expression itself. Since a constant c will always have, in our definitions, $\mathcal{W}(c) = \underline{\mathcal{W}}(c) \geq 2$

The aim of this chapter will be to explore what weight functions might be appropriate for different sets of expressions. From here, the caveat that "$\mathcal{V}(e)$ is defined and non-zero" will be assumed.

7.3 Rational Expressions

Definition 16. *Define $\mathbb{E}_{\mathbb{Q}} \equiv \mathbb{E}(\{+, -, \times, -_u, /, /_u\}, \mathbb{N})$ where $-_u$ and $/_u$ are the unary minus and inversion operators $\mathcal{V}(-_u x) = -\mathcal{V}(x)$ and $\mathcal{V}(/_u x) = 1/\mathcal{V}(x)$.*

(The subscript will be dropped from the unary operators when the meaning is obvious.)

Proposition 21. *$\mathcal{W}$, defined below, is a weight function for $\mathbb{E}_{\mathbb{Q}}$*

$$\mathcal{W}(e) = \begin{cases} \max(2, e) & \text{if } e \in \mathbb{N} \\ \mathcal{W}(a)\,\mathcal{W}(b) & \text{if } e = a \pm b, e = a/b, e = a * b \\ \mathcal{W}(a) & \text{if } e = /a, e = -a \end{cases}$$

I.e. if $e \in \mathbb{E}_{\mathbb{Q}}$ and $\mathcal{V}(e)$ is defined then either $\mathcal{V}(e) = 0$ or

$$\underline{\mathcal{W}}(e)^{-1} \leq |e| \leq \underline{\mathcal{W}}(e)$$

Proof. The proof is by induction on the number of (internal) nodes in the expression tree.

The only trees with zero nodes correspond to some $e \in \mathbb{N}$ for which the result is trivial. Assume it is true for trees involving $< k$ nodes and consider any expression e with k nodes.

If e has the form $/f$ or $-f$ then the result holds for f and so for e.

If $e = f \times g$ then $|e| = |f||g|$ and, by hypothesis,

$$\mathcal{W}(e)^{-1} = \mathcal{W}(f)^{-1}\,\mathcal{W}(g)^{-1} \leq |e| = |f||g| \leq \mathcal{W}(f)\,\mathcal{W}(g) = \mathcal{W}(e)$$

If $e = f + g$, $|e| \leq |f| + |g| \leq \mathcal{W}(f)\,\mathcal{W}(g) = \mathcal{W}(e)$ since $\mathcal{W}(f) \geq 2$ and so the upper bound is correct. If either $|\mathcal{V}(f)|$ or $|\mathcal{V}(g)| \geq 1$ the lower bound is trivial;

this leaves only the case where both are < 1. If $\mathcal{V}(f) = a/b$ and $\mathcal{V}(g) = c/d$, then

$$\begin{aligned} |\mathcal{V}(f+g)| &= |(a/b + u/v)| \\ &= \left|\frac{ad+bc}{bv}\right| \\ &\geq 1/|b||d| \\ &\geq \frac{1}{\mathcal{W}(f)\,\mathcal{W}(g)} \end{aligned}$$

since $|b| \leq \mathcal{W}(f)$ and $|d| \leq \mathcal{W}(g)$.

This covers all cases and completes the proof. □

This result shows that a simple gap function exists and that the number of decimal places needed to distinguish a rational expression e from zero is $\log_{10}(\underline{\mathcal{W}}(e))$. Roughly this is the sum of number of decimal digits appearing in the leaves. It is convenient use $\mathbb{N}$ as the set of constants. The set of constants can also be taken to be $\mathbb{Q}$ with the weight of a constant $c = a/b$ taken as $\max(2, |a|, |b|)$, indeed the set can be extended to $\mathbb{Q}[\imath]$ where the weight of $a + \imath b$ is $\mathcal{W}(a)\,\mathcal{W}(b)$.

Only unary minus and division functions were included in the definition of $\mathbb{E}_{\mathbb{Q}}$. Clearly binary versions can be added without making any other changes since $f - g \equiv f + (-g)$ and $f/g \equiv f \times (/g)$ so $\mathcal{W}(f/g) = \mathcal{W}(f \times /g) = \mathcal{W}(f)\,\mathcal{W}(g)$ etc. In the proof above it is significant that only $+$ depended on the structure of the set of constants. The operations $\times, /_u, /$ and $-_u$ depend only on the magnitude of their operands for weight estimates. It is often convenient when bounds are considered to use $a \pm b$ to refer to either $a + b$ or $a - b$.

Other functions yielding rational results can be added as long as they are sufficiently well-behaved:

Lemma 16. *For a set of expressions $\mathbb{E}(F, C)$ with a weight function $\mathcal{W}(\cdot)$, let*

$V = \mathcal{V}(\mathbb{E}(F, C))$. *If for some arbitrary function* $g \notin F$ *with* $g : V^n \to V$ *then* $\mathbb{E}(F \cup \{g\}, C)$ *has a weight function iff there exists a function* $W : V^n \to V^+$ *such that given any* $e_1, \ldots, e_n \in E(F, C)$ *with* $w_i = \mathcal{W}(e_i)$ *and* $v_i = \mathcal{V}(e_i)$ *then*

$$\frac{1}{W(w_1, \ldots, w_n)} \leq |g(v_1, \ldots, v_n)| \leq W(w_1, \ldots, w_n) \tag{7.2}$$

Proof. The lemma is little more than a re-statement of definitions. However its significance is that g is independent of the structure of the expressions appearing as its arguments. Note that there is no need to specify further properties of g, for example it need not be smooth, only that its values satisfy Eq. 7.2.

Necessity is clear since $g(e_1, \ldots, e_n) \in \mathbb{E}(F \cup \{g\}, C)$ for all $e_1, \ldots, e_n \in \mathbb{E}(F, C)$.

For sufficiency, consider any expression in $\mathbb{E}(F \cup \{g\}, C)$ having sub-expressions for the form $g(\ldots)$. At least one of these sub-expressions must have no arguments which themselves contain sub-expressions involving g. For such a sub-expression, since $g : V^n \to V$, its value can be represented (in many ways) as an expression in $\mathbb{E}(F, C)$. By the hypothesis there must be a representation having least weight and it must satisfy Eq. 7.1. Thus sub-expressions in g can be successively removed from the expression structure to yield an expression in $\mathbb{E}(F, C)$ having the same value and satisfying the bounds. □

This lemma can be used in two ways, the simplest is when the new function is no more than an alternative expression whose structure includes skeletal leaf nodes which are to be filled with the expressions corresponding to arguments, for example starting from $\mathbb{E}_{\mathbb{Q}}$ a function $f(x) \equiv 2 \times x$ could be defined (and given a constant weight 2). A different case would be an operation x^n defined for $n \in \mathbb{Z} \setminus \{0\}$ which returns $\mathcal{V}(x)^n$. There is no single expression corresponding

to x^n which can be built in $\mathbb{E}_{\mathbb{Q}}$ though some expression does exist for each n (and, as an aside, no algorithm is known which generates the 'lightest' expression [Knu81, pp443-462]). A suitable, though clumsy, weight function for x^n would be $W(\mathcal{W}(x), \mathcal{W}(n)) \equiv \mathcal{W}(x)^{\mathcal{W}(n)-1} / \mathcal{W}(n)$.

7.4 Algebraic Expressions

Two different classes of algebraic expressions are considered:
$\mathbb{E}_{\mathbb{A}} \equiv \mathbb{E}(\{+, -, -_u, \times, /, /_u\}, \mathbb{A})$ the usual operators with constants being the algebraic numbers denoted by $\mathbb{A}$, and the class of nested radical expressions, $\mathbb{E}_r \equiv \mathbb{E}(\{+, -, -_u, \times, /, /_u, \sqrt[n]{\cdot}\}, \mathbb{N})$ where the constants are natural numbers but n^{th} roots may be taken.

7.4.1 General Algebraic Expressions

It is convenient to restrict the class $\mathbb{E}_{\mathbb{A}}$ initially by interpreting $\mathbb{A}$ as comprising only algebraic integers, equivalently a simple transformation will convert an expression, e in arbitrary algebraic numbers into e'/m where m is an integer and e' contains only algebraic integers.

The weight of an algebraic number α can be defined in a number of be defined in a number of ways as long as it satisfies

$$\frac{1}{\max(\lceil\overline{\alpha}\rceil, 2)} \leq \mathcal{W}(\alpha) \leq \max(\lceil\overline{\alpha}\rceil, 2)$$

where $\lceil\overline{\alpha}\rceil$ is the maximum absolute value of α and its conjugates. Any of the usual roots bounds, such as $1 + H(\alpha)$ or $\max(2, M(\alpha))$ can be used.

Lemma 17. *If* $e \in \mathbb{E}(\{+,-,-_u,\times\},\mathbb{A})$ *(i.e. expressions not involving division) and* $|e| \neq 0$ *then*

$$|e| > \underline{\mathcal{W}}^{1-\deg(e)}$$

where

$$\deg(e) = \begin{cases} n & \text{if } e \in \mathbb{A} \text{ with } \deg(e) = n \\ \deg(u) & \text{if } e = -u \\ \deg(u)\deg(v) & \text{if } e = u\{+,-,\times\}v \end{cases}$$

Proof. $\mathcal{V}(e)$ is an algebraic integer with $\deg(e) \leq d$ say. From the definition of weight $\mathcal{W}(e) \geq \lceil e \rceil$. If the minimal polynomial of $\mathcal{V}(e)$ is p and it's roots are $r_1, \ldots, r_{\deg(e)}$ with r_1 having smallest absolute value then

$$1 \leq |p(0)| = \prod |r_i| = |r_1| \prod_{i>1} |r_i| \leq |r_1| \lceil e \rceil^{\deg(e)-1}$$

$$\text{or} \quad |e| \geq |r_1| \geq \lceil e \rceil^{1-\deg(e)} \geq \mathcal{W}(e)^{1-\deg(e)}$$

□

In Example 1.3 $p_1 = x_1^2 - 2$, $p_2 = x_2^4 - 18x_2^2 + 49$, $q = x_2 - 2x_1 - 1$

Set $\mathcal{W}(x_1) = 2$, $\mathcal{W}(x_2) = 4(\lceil p_2 \rceil \approx 3.82)$.

$$|q| \geq 8^{-7} \approx 4.7 \times 10^{-7}$$

Liouville's bound gives $|q| \geq 1.6^{-10}$

The result above applies only to division free expressions. It is a small step to extend it.

Proposition 22. *If $e \in \mathbb{E}_{\mathbb{A}}$ (i.e. including division) then $|e| \neq 0$ implies*

$$|e| > \underline{\mathcal{W}}(e)^{-\deg(e)}$$

Proof. The expression e can be rewritten in the form f/g where f and g are division free and each is an algebraic integer with weight $\leq \underline{\mathcal{W}}(e)$. Thus $|g| \leq \underline{\mathcal{W}}(g) \leq \underline{\mathcal{W}}(e)$ and $|f| > \underline{\mathcal{W}}(f)^{1-\deg(f)} \geq \underline{\mathcal{W}}(e)^{1-\deg(e)}$. Multiplying the bounds gives the result. □

The result above is very crude for practical use where it would be better to obtain individual bounds for f and g by a recursive traversal of the expression.

It is interesting to note that the size of the bound depends on the number of occurrences of each algebraic number in the expression being tested.

Using the same example as Eq. 4.3

$$p_1 = x^3 + 11x^2 + 17x - 44$$

$$p_2 = 3x^4 + 21x^3 - 35x^2 - 14x + 22$$

with $\mathcal{W}(x) = 8$ where the best lower bound was $p_2 > \approx 1.2 * 10^{-9}$

---

p_2 as written: $\mathcal{W}(p_2) \approx 729 \times 10^{12}$

$$|p_2(\xi)| \geq 1.8 \times 10^{-30}$$

However writing $p_2 = 22 + x(-14 + x(-35 + x(21 + 3x)))$ reduces the weight to 278×10^7 and gives

$$|p_2(\xi)| \geq 1.2 \times 10^{-19}$$

7.4.2 Nested Radical Expressions

To some extent this section re-presents results given earlier but in terms of 'expressions' rather than polynomial maps.

Radical expressions are obtained by augmenting the usual field operations with $\sqrt[n]{\cdot}$, i.e. computing n^{th} roots. From a practical point of view even adding only a square root operator is valuable. Many problems in graphics involve the distance between rationally given points, square roots are naturally needed in calculations where there is a need to decide if lines cross or points coincide.

Perhaps because of its utility several methods exist for finding lower bounds of radical expressions using an argument similar to that above for more general algebraic expressions (see, eg, [Li01] or [BFMS00]; for a different approach see [Sch00]). The method seems to be frequently rediscovered (including by the author of this paper - though we are not aware of it being described elsewhere for general algebraic numbers).

To extend Prop. 22 to include radical expressions it is only necessary to define $\mathcal{W}(\sqrt[n]{e}) = \left\lceil \max(2, \sqrt[n]{w(e)} \right\rceil$ and add $\deg(\sqrt[n]{e}) = n \deg(e)$.

7.4.3 Conjectures and Non-constructive Results

The difference between the algebraic and rational case is the introduction of the degree into the lower bound. There is clear evidence to suggest that the form taken by the exponent is much too large. Prop. 22 can be proved by induction on the form of the expression as in the rational case. If this is done, the bound for

expressions in the operators $-_u, \times, /, /_u$ does not depend on the degree of their operands. The degree is only introduced in considering lower bounds for $a \pm b$. This reduces to finding lower bounds of $|a \pm b|$ where at least one of the terms is algebraic. This is a difficult and much studied problem.

For the case of rational $b = p/q$ the strongest result is the Thue-Siegel-Roth theorem [Wal00, p8] which states that, for any real number ϵ, there exists a constant $K = K(a, \epsilon) > 0$ such that

$$|a - p/q| > K/q^{2+\epsilon} \tag{7.3}$$

Unfortunately the constant K is not constructive. Nevertheless the power $2 + \epsilon$ doesn't depend on the degree of a (unlike the more familiar result due to Liouville [Wal00, p82]).

For the case where a and b are both algebraic even less is known. [Wal00] gives a number of Liouville type inequalities in which degrees appear explicitly, while Evertse [Eve00b] gives a summary of the current state of knowledge of the equivalent, for this case, of the Thue-Siegel-Roth theorem. The strongest 'result' is the unproved conjecture by Schmidt, given at Eq. 4.4 that for every $\delta > 0$ all but finitely many pairs of algebraic numbers α and β satisfy

$$|\alpha - \beta| \geq 1/(\max(M(\alpha), M(\beta)))^{2+\delta}$$

where $M(\alpha)$ is the Mahler measure of α. This suggests that there might be a stronger result for the lower bound of algebraic expressions of the form based on defining $\mathcal{W}(\alpha \pm \beta) = \max(\mathcal{W}(\alpha), \mathcal{W}(\beta))^k$ for some universal constant k. Unfortunately $\mathcal{W}$ cannot then be based solely on the Mahler measure, since $M(\sqrt[m]{2}) = 2$ and so $\sqrt[n]{2} - \sqrt[m]{2}$ would have a weight (and thus a lower bound) independent of n and m.

From this point on there are only very partial results and some linked conjectures. There, as here, the problem is in finding lower bounds for expressions $a \pm b$. This suggests that it is the key to bounding small expressions.

7.5 Exp-Log Expressions

7.5.1 The Uniformity Conjecture

The original motivation for this paper was work by Richardson on the *Uniformity Conjecture* ([Ric00], [Ric99b]). He considers the set of *expanded expressions*, $\mathbb{E}_x \equiv \mathbb{E}(\{+, -, -_u, \times, /, \ln, \exp, \sqrt[n]{\cdot}\}, \mathbb{Z})$ where $\exp(x) = e^x$ if $|x| \leq 1$ and is undefined elsewhere, $\ln(x)$ has its usual definition and $\sqrt[n]{m} = m^{1/n}$ for integer $n > 1$. Where values are complex they are chosen so that the imaginary part is in $(-\pi, \pi]$.

Expanded expressions are so called because general exponentiation has to be 'expanded' into repeated multiplication. A general exponentiation operation would allow very large and very small expressions to be created by its repeated application, requiring in turn a large weight function to bound expressions.

The conjecture states:

Conjecture 2 (Uniformity Conjecture (UC)). *For $e \in \mathbb{E}_x$ with $\mathcal{V}(e)$ defined and non-zero*

$$|e| \geq A^{-\mathcal{L}(e)}$$

where $A \leq 17$ *is a universal constant and* $\mathcal{L}(e)$ *is defined as*

$$\mathcal{L}(e) = \begin{cases} \lfloor \log_{10}(e) \rfloor + 1 & if\, e \in \mathbb{N} \\ \mathcal{L}(a) + \mathcal{L}(b) + 1 & if\, e = a \pm b, a \times b, a/b \text{ or } \sqrt[a]{b} \\ \mathcal{L}(a) + 1 & if\, e = \ln(a), \exp(a), \text{ or } -a \end{cases}$$

7.5.2 Counting Expressions

With hindsight the value $A \leq 17$ is suspect. The original formulation of the conjecture (in, among others, [RL02]) had $|e| \geq 1/N(\mathcal{L}(e))$ where $N(k)$ is the number of expressions of length $\leq k$ and it was claimed that $N(k) \leq 17^k$ - unfortunately the argument used was incorrect. A weaker, but justifiable, claim is: $N(k) \leq 18^{\lfloor (3k-1)/2 \rfloor}$.

Consider an expression of length k where the expression is written in prefix form, e.g. $1+22\times 3$ is written $+1 \times 22\ 3$. The original argument was that all expression were built from 17 symbols, $0 \ldots 9, +, -, \times, /, \exp, \log, \sqrt{\cdot}$ and thus there were at most 17^k possibilities. This is clearly false since a) it refers to expressions of length exactly k and b) different expressions may be indistinguishable if we ignore the 'spaces' between different numbers, for example $+1 \times 2\ 23$ is not the same as $+1 \times 22\ 3$. Thus the number of 'symbols' needs to be increased to indicate the dividing points between numbers. An expression of length k includes at most $\lfloor (k-1)/2 \rfloor + 1$ numbers (when there are d internal binary nodes, no unary nodes and $d+1$ one digit numbers, $k = 2 * d + 1$). If a symbol is introduced to make separate adjacent numbers, the length of the expression, including at most $\lfloor (k-1)/2 \rfloor$ separators, becomes $k + \lfloor (k-1)/2 \rfloor = \lfloor (3k-1)/2 \rfloor$. Thus the number of expressions is less than $18^{\lfloor (3k-1)/2 \rfloor}$. However, introducing an extra symbol allows it also to be used as 'padding' in shorter expressions and thus the estimate is large enough to include all expressions of length $\leq k$.

A better estimate is easy to obtain implicitly by considering the generating function $T(z) = \sum_{i \geq 0} t_i z^i$ where t_k is the number of expressions of length k. Since the result has (as far as we have been able to see) no closed form it is worth simplifying the bound by treating $\sqrt{\cdot}$ as a binary operation with operands which can both be arbitrary expressions. The $N(i)$ can be obtained from $T(z)/(1-z) = \sum_{i \geq 0} N(i) z^i$. $T(z)$ is obtained in the way commonly used to enumerate binary trees (c.f. [SF96, p224])

$$T(z) = 9z/(1-10z) + 5zT^2(z) + 3zT(z) \quad \text{or}$$
$$T(z) = \frac{1 - 3z \pm \sqrt{(3z-1)^2 - 180z^2/(1-10z)}}{10z}$$

($9z/(1-10z)$ is the generating function for integers > 10 and without leading zeros, the 5 and 3 for the number of binary and unary operators respectively.) Since $T(0) = 0$ (the number of expressions with length 0), application of Hôpitale's rule indicated that $\pm$ should be $-$ giving

$$T(z)/(1-z) = \frac{1 - 3z - \sqrt{(3z-1)^2 - 180z^2/(1-10z)}}{10z(1-z)}$$
$$= \frac{(1-3z)(1-10z) - \sqrt{900z^4 + 1020z^3 + 49z^2 - 26z + 1}}{10z(1-z)(1-10z)}$$

The first few terms are:

k	1	2	3	4	5	6	7	8
$N(k) \leq$	9	126	1782	26280	407259	6592536	110810817	1921572900

(and $N(20) > 17^{20}!$).

How good a predictor is the Uniformity Conjecture? Applying it to the two small but nonzero expressions and one identity gives (using $A = 17$)

Expression	Predicted Lower Bound	Actual Value
$\sqrt[8]{2} - 494/453$	$\geq 10^{-12}$	6.4×10^{-9}
$\sqrt[3]{\sqrt[5]{32/5} - \sqrt[5]{27/5}}$ $-(1 + \sqrt[5]{3} - \sqrt[5]{9})/\sqrt[5]{25}$	$\geq 10^{-29}$	0
$3\ln(640320)/\sqrt{163} - \pi$	$\geq 10^{-20}$	2.2×10^{-16}

A similar conjecture is given by Joris van der Hoeven for the set of expressions $\mathcal{E} = \mathbb{E}(\{+, -, \times, /, \exp, \log\}, \{0, 1\})$

Conjecture 3 ([vdH00]). *Given a rational number $N \geq 3$ denote by $\mathcal{E}_N$ the set of $x \in \mathcal{E}$ such that, for any subexpression x' of x, $N^{-1} \leq |x'| \leq N$. If a tree representation of x has σ nodes then there exists a function ϖ having one of the forms $\varpi(\sigma) = K\sigma$ or $\varpi(\sigma) = K^\sigma$ ($K \geq 1$ depends on N) such that $|x| \geq e^{\varpi(\sigma)}$*

Richardson's conjecture is more obviously close to the forms used earlier in this chapter. $\mathcal{L}(e)$ can be written as $k + \sum \mathcal{L}(c_i)$ where k is the number of operators in the expression and the c_i are the constants. This allows the bound to be rewritten as

$$\begin{aligned} |e| &\geq A^{-k} A^{-\sum \mathcal{L}(c_i)} \\ &\approx A^{-k} \prod (c_i + 1)^{-\delta} \\ &\approx A^{-k} \underline{\mathcal{W}}(e)^{-\delta} \end{aligned}$$

where $\delta = \log_{10}(A) \leq 1.24$. This suggests the first form of the conjecture used here

Conjecture 4. *For $e \in \mathbb{E}_x$ with $\mathcal{V}(e)$ defined and non-zero*

$$|e| \geq \mathcal{W}(e)^{-\delta} \quad \text{or}$$

$$|e| \geq \underline{\mathcal{W}}(e)^{-\delta} A^{-k}$$

where $1 < \delta < 1.24$ is a universal constant, k is the number of operators appearing in the expression, all of which have a constant weight of A

A more general conjecture would be that

$$|e| \geq \frac{1}{\underline{\mathcal{W}}(e)^{\delta} s(n)}$$

where n is the number of nodes in the tree representing the expression and $s(n)$ is the maximum contribution to weight of any structure with n nodes. It is tempting

to revisit the original form of the conjecture (§7.5.2 above) and wonder if $s(n)$ might be replaced by the number of 2-restricted trees (see e.g. [SF96, p289]) with n nodes with internal nodes labelled by operator names.

The set of expanded expressions is a suitable basis for modeling scientific programming. As before in Lemma 16 new functions and constants can be added as short hand for values which can already be computed in other ways. In particular, since $\imath = \sqrt{-1} \in \mathbb{E}_x$, $\imath$ can be used in expressions as a constant with $\mathcal{W}(\imath) = \mathcal{W}(\sqrt[2]{-1}) = 4A^2$. Lengths for additional elementary functions and constants, defined in terms of their definitions, can also be introduced. For example $\pi \equiv \ln(-1)/\imath$ giving $\mathcal{W}(\pi) = 8A^5$, and so on. Obviously these are 'upper bounds' for weights and mathematical criteria might suggest better ones.

A proof of any of these conjectures seems remote. It is known that the original formulation by Richardson is false and ways of finding counterexamples have been identified.

As counterexamples are found it is tempting to 'fix' the formula to take account of their properties. Instead, the following sections attempt to explore the "natural history" of such numbers and identify interesting properties.

A natural question to ask about $\mathbb{E}_x$ is, "Among expressions with a given weight, what is the form of the smallest expression?". For example, for all weights above some value W, it may be that the smallest expression of a given weight always has the form $a \pm b$ or $\log(a)$. *A priori* there is no reason to suppose that this is the case but some possibilities can be eliminated. If the conjecture is false it would be useful to know how and when it fails. We use $\mathcal{C} \subset \mathbb{E}_x$ for the set of counterexamples to the conjecture, i.e $e \in \mathcal{C}$ *iff* $|e| < \mathcal{W}(e)^{-\delta}$ or $|e| > \mathcal{W}(e)^{\delta}$.

Definition 17. *Writing $\#e$ for the number of interior nodes in the expression tree of any $e \in \mathbb{E}_x$ then a* minimal counterexample to the Uniformity Conjecture *is an element $c \in \mathcal{C}$ such that for all $e \in \mathcal{C}$, $\mathcal{W}(c) \leq \mathcal{W}(e)$ and $\#c \leq \#e$. Note that there may be no minimal counterexample and even, if there is, it need not be unique.*

Proposition 23. *If a minimal counterexample exists it has one of the forms $a \pm b$ or $\log(a)$ and further it is in $\mathcal{C}$ because it is smaller than the lower bound.*

Proof. Suppose that a minimal counterexample c exists. Then

i) $c \notin \mathbb{N}$ by the definition of expression weight.

ii) $c \neq \exp(a)$ since by the definition of a valid expression, $|a| \leq 1$ and so $\exp(-1) \leq |c| \leq \exp(1)$.

iii) $c \neq a \times b$ and $c \neq a/b$ since if it were one of a and b would be in $\mathcal{C}$.

iv) $c \neq -a$ and $c \neq /a$ trivially.

v) $e \neq \sqrt[r]{a}$ since for the upper bound, if $|a| \geq 1$ and $|a| \leq \mathcal{W}(a)$ then $|\sqrt[r]{a}| \leq \mathcal{W}(a) < \mathcal{W}(a)\,\mathcal{W}(n)A = \mathcal{W}(e)$. Since $\mathcal{W}(e) > 1$ the upper bound is trivial if $|a| < 1$. This only leaves the possibility that $|e| < 1/\mathcal{W}(e)$ but this would require $|\sqrt[r]{e}| \leq |e|$ which is impossible.

vi) $|e| = |\log(a)| \leq |a| \leq \mathcal{W}(a) \leq \mathcal{W}(\log(a))$.

vii) $|a \pm b| \leq \mathcal{W}(a)\,\mathcal{W}(b)$ as in the rational case.

This only leaves lower bounds of expressions $a \pm b$ or $\log(a)$ as candidates. $\square$

Thus any short/low-weight counterexample to the Uniformity Conjecture (in any of its forms) will be an expression of the form $a+b$ or $\log(a)$ whose absolute value is small. Even small values which are not counterexamples are of interest as they may give more evidence of the form of expressions with least value.

7.6 Empirical Research

Various experiments have been made to learn more about the nature of small expressions and their properties. The experiments have included 'brute force' searches of various forms and statistical sampling to identify distribution properties of $\mathbb{E}_x$. These are summarised in this section. For the 'length based' model, Richardson suggested $\rho(e) = |\log_{10}|e|\ |/\ \mathcal{L}(e)$ as a measure of smallness. $\rho(e) > A$ implies $e \in \mathcal{C}$ but none of the experiments produced even an expression with $\rho(e) > 1$. In the course of the experiments tens of millions of random expressions and every potential counterexample for all expressions lengths up to 12 have been generated. There are counterexamples, described below, but only for long expressions.

7.6.1 Continued Fractions

It can be shown[2] that if α is a real irrational closed form number, and there are arbitrarily large partial quotients in the continued fraction expansion for α, then there is an expression e of the form $\alpha - p/q$ for which $\rho(e) > 1$. For example,

[2]Discussion with Dr Richardson.

we could take $\alpha = \exp(1/9)$. The expressions constructed in this way are very large, of length at least a million in all cases we know, and are never, as far as has been determined, counterexamples. Although such expressions exist, none have actually been constructed, one of the problems being the very high precision floating point arithmetic which would be needed.

A smaller scale search has been done for $\alpha = n^{1/m}$, $\alpha = e^{n/m}$ and also for $\alpha = \log(n/m)$ with n and m ranging over all natural numbers between 2 and 100 and the first 50 partial quotients in the continued fraction approximation being computed. The results are reported in [Ric99b]. No counterexample was found but some small numbers are reported, for example, $2^{1/8} - 494/453$. An expression that was just missed is $109^{1/5} - 23/9$, found by Reyssat and mentioned in [Nit96] which discusses the relationship between this kind of approximation and the 'abc' Conjecture.

A natural extension of continued fractions would be to search among linear forms $a_1\alpha_1 + \cdots + a_n\alpha_n$ where $a_1, \ldots, a_n$ take integer values, and $\alpha_1, \ldots, \alpha_n$ are closed form numbers which are linearly independent over $\mathbb{Q}$. The LLL algorithm or PSLQ could be used for this. See [FBA99], [Weg87]. Some trials of this kind have been done by Richardson, but no counterexamples have been found. In the case of linear forms in logarithms, there is, again, a relationship with the 'abc' Conjecture. See [Phi99], [Phi00].

7.6.2 Exhaustive Search

This depends on having an algorithm for enumerating all expressions of a given length. The following recurrences relations were used where $\mathcal{E}(n)$ is the number

of expressions of length n, $\mathcal{N}(n)$ is the number of positive integers (base b) of length n and $\mathcal{U}(n)$, $\mathcal{B}(n)$ and $\mathcal{R}(n)$ are the number of expressions of length n in which the outer most operator is unary (one of $\exp, \log$ or $-$), binary (one of $+, -, \times, /$) or a root ($\sqrt{\cdot}$) respectively.

$$\mathcal{N}(n) = \begin{cases} 0 & \text{if } n = 0 \\ (b-1)b^{n-1} & \text{otherwise} \end{cases}$$

$$\mathcal{U}(n) = \begin{cases} 0 & \text{if } n < 2 \\ 3\mathcal{E}(n-1) & \text{otherwise} \end{cases}$$

$$\mathcal{B}(n) = \begin{cases} 0 & \text{if } n < 3 \\ 4\sum_{i=1}^{n-2} \mathcal{E}(i)\mathcal{E}(n-1-i) & \text{otherwise} \end{cases}$$

$$\mathcal{R}(n) = \begin{cases} 0 \text{ if } n < 3 \\ (b-2)\mathcal{E}(n-2) + \sum_{i=2}^{n-2} \mathcal{N}(i)\mathcal{E}(n-1-i) \end{cases}$$

The slightly different form of $\mathcal{R}(n)$ reflects the requirement that in $\sqrt[s]{e}$, $s > 1$ and so there are $(b-2)$ single digit roots. Combining these gives

$$\mathcal{E}(n) = \mathcal{N}(n) + \mathcal{U}(n) + \mathcal{B}(n) + \mathcal{R}(n)$$

The formula makes use of only syntactic information. Many expressions will be undefined or fail to be 'expanded' (i.e. in $\exp(a)$, $|a| \notin [0,1]$. However it is easy to compute the total number of expressions of a given type and map each expression of length n to an integer between 1 and $\mathcal{E}(n)$. Even for quite small n the number of expressions is very large, $O(18^n)$, and the program includes extensive checks: expressions containing sub-expressions which could be written in a shorter form are abandoned as soon as they are detected, as are expressions which trivially cause an undefined result.

Currently the program detects and discards any expression containing

1. integer only sub-expressions i.e. elements of $\mathbb{E}(\{-_u, +, -, \times\}, \mathbb{N}) \setminus \mathbb{N}$.

2. $\log(\exp(a))$, $\exp(\log(a))$, $\exp(-\log(e))$, $\exp(\log(e))$, $\exp(0)$, $\log(0)$ and $\log(1)$. Also no integer expression other than 1 and -1 is allowed as argument to exp.

3. $-a \odot -b$, $-(-a \odot b)$ and $-(a \odot -b)$ for $\odot \in \{+, -, \times, /\}$ as well as $--a$, $-\log(a/b)$, $-a + b$, $a + -b$ and $a - -b$.

4. $a + a$ (always at least as long as $2 \times a$), $a - a$ and a/a.

5. $\sqrt[n]{1}$.

6. $a \times 1$, $1 \times a$, $a/1$ (since 0 is never generated as an integer value, expressions such as $a + 0$ and so on are never produced).

If the final expression is explicitly wholly rational (i.e. it contains no applications of $\log(\cdot)$, $\exp(\cdot)$ or $\sqrt[n]{\cdot}$) it is also discarded.

An attempt is made to evaluate each expression using quad-precision floating point interval arithmetic (adequate for the accuracy needed here). Those expressions which appear to be zero or very small are output for checking using a computer algebra system (MuPAD). The standard expression simplification has always succeeded in identifying zeros. There have been no surprises and very few small expressions.

Richardson (using a diffent program) has reported results in [Ric99b] for $k = 4, 5$. We have extended it up to $k = 10$ for expressions of the form $a \pm b$ and $\log(a)$. With $k = 10$ the program ran for several days (on a 1GHz home PC).

7.6.3 Statistical Information

Exhaustive generation of all expressions, is feasible only for very short expression lengths. A second approach is generate many random expressions (typically 10^6/run) and investigate any interesting features of the distribution.

Generating expressions at random was done using the recurrence relations above to choose an expression number and map it to an actual expression. The resulting expressions were evaluated as above and three different distributions produced.

For lengths in the range 10 - 30, a typical run would produce 1,000,000 expressions in a few seconds. Usually about a quarter of these would contain subexpressions with form a/b or $\log(b)$ where $\mathcal{V}(b) = 0$ or fail to be in expanded form (i.e. including $\exp(e)$ where $|V(e)| > 1$).

Double precision complex numbers were used to evaluate the expressions and no attempt was made to adjust for rounding errors. Given the number of expressions generated it was assumed that any errors would be as randomly distributed as the values themselves.

There is a possibility that very small values will be just the points where floating point errors become significant. Also though double precision floating point values can be as small as $\approx 10^{-300}$, the machine-ϵ (the smallest x such that $1 + x \neq 1$) is only $\approx 10^{-16}$ so computing values mod 1 is suspect.

To check the significance of this, several tens of thousand of expressions which were small mod 1 were fed into MuPAD and symbolically simplified. No anomalies were found. Expressions with a floating point value of zero or close to

machine-ϵ were always found via MuPAD to be identically zero.

In developing and testing the program hundreds of small or probably zero expressions were checked manually. This is an un-exciting activity. Very few expressions required even pencil and paper to check and no even slightly interesting identities were found, a factor of $(1-1)$ in an expression probably accounts for most zeros.

7.6.3.1 Results

All experiments used numbers in base 10 and generated random sample expressions of length 7, 11, 13, 17, 19, and 23. Each run produced 10^6 expressions of which about 7×10^5 appeared to be valid (in expanded form and not containing subexpressions equivalent to either $1/0$ or $\log(0)$). (The percentage of valid expressions appears remarkably consistent, one sequence of six runs produced 700616, 700560, 700905, 700054, 700484 and 700053 valid expressions. This initially lead to doubt about the randomness of the expression generator but further statistical checks revealed no faults.) The results are similar for all sizes can be illustrated with a single example.

Fig. 7.2 shows the percentage of values mod 1 in each percentile for about 6×10^5 expressions (i.e. 1% on the Y scale is about 6000 values) of length 17.

This particular run produced an identity which was slightly less trivial than most: $\log(4) + \log(1/(6 \times \exp(\log(1 \times 2/\sqrt{9})))) = 0$.

This plot is typical (though, as the expression size increases, the number of 'spikes' increases while their relative height decreases). For very short expression lengths

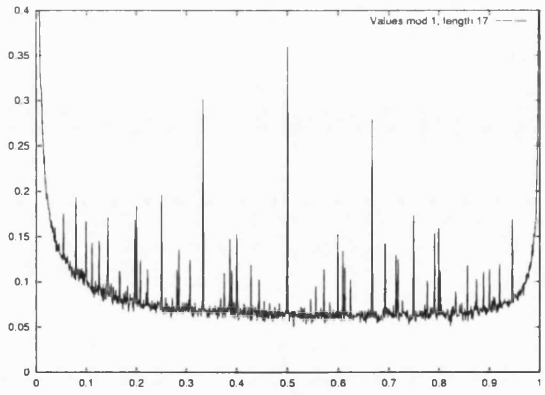

Figure 7.2: Distribution of values mod 1

(say < 5) there are zero height sections between the spikes.

The overall shape is not surprising. Consider the possible mod 1 values for expressions of length 3. You can form fractions (with single digit denominators), roots (from 2 to 9) of single digit values, log of two digit numbers, $-\exp(1)$ and $\exp(-1)$. These correspond to visible spikes on the graph. Now consider expressions of length 17 which can be written in the form $A \pm B$ where $\mathcal{L}(B) = 3, \mathcal{L}(A) = 13$ and $V(A) \in \mathbb{Z}$. There are $> 13^{13}$ integer expressions obtained by using only the operators $\{+, -, \times\}$, all divisions integers by their factors, numbers with exact roots, expressions of the form $\exp(\log(e))$ or $\log(\exp(e))$ where $\mathcal{V}(e) \in \mathbb{Z}$ and so on.

The spikes blot out other information which might be of interest. There are many ways of getting the value 0.5 - does the spike correspond to values equal to 0.5 or are many different values clustered nearby? From the sample we extracted all values between 0.45 and 0.55 and eliminated all duplicates (strictly, rational duplicates were removed and floating point values which differed by less than 10^{-50}). This left 530 numbers. The plot in Fig. 7.3 shows their distribution with the Y scale being percentage frequency (1% represents about 5 numbers).

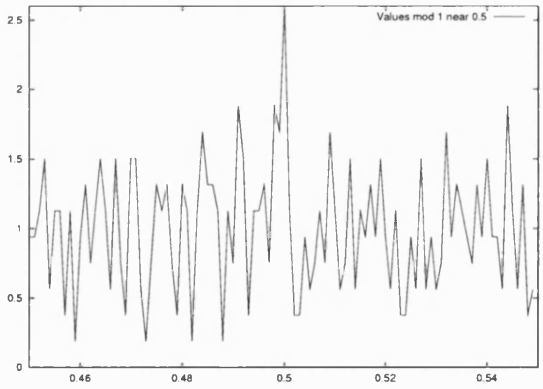

Figure 7.3: Distribution near 0.5 with duplicates removed

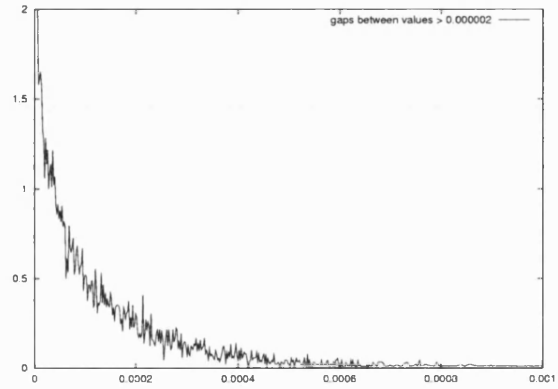

Figure 7.4: Intervals between values

The central peak still appears significant but statistical analysis is required to decide if the rest of the graph is random.

If the Uniformity Conjecture is correct, the difference of two values corresponding to expressions with length 17 should be $\geq 10^{-1.24\times 35} \approx 10^{-44}$. Using the first 10,000 expressions from the same data, sorted by expression value and with obvious duplicates removed, the differences between adjacent expressions were re-computed (to 50 decimal digit precision).

The smallest gap found was $\approx 5\times 10^{-20}$ and all differences after that were $\geq 10^{-14}$. About 5% of differences lay in the range $[10^{-14}, 2 \times 10^{-6}]$, larger gaps tail off rapidly after this (Fig. 7.4).

It seems that some more empirical investigation of the distribution of values might help to direct the search.

7.7 Refining the Uniformity Conjecture

7.7.1 Counterexamples

None of the searches found any counterexample to the Uniformity Conjecture. However a possible way of finding counterexamples is to adapt a traditional method of transcendental and algebraic number theory: construct a function $F(x)$ with a zero of high multiplicity at 0, $F(x)$ is then very small when x is close to zero. For example, one of the proofs of Lindemann's theorem begins by setting $F(x) = p_1(x)e^{\alpha_1 x} + p_2(x)e^{\alpha_2 x} + \cdots + p_m(x)e^{\alpha_m x}$, where $p_1(x), \ldots, p_m(x)$ are polynomials with unknown integral coefficients with degree bounded by n. Some method is used to determine integral values of these coefficients so that $F(x)$ is not identically zero but has a zero of multiplicity approximately $nm/2$ at zero. $F(q)$ for $q \in \mathbb{Q}$ with q small is a fational form in exponential with small absolute value. Richardson and van der Hoeven have used this (in [RE03]) as a starting point to disprove the Uniformity Conjecture. They used a length measure, $\mathcal{L}$, which has here been converted to weight, $\mathcal{W}$. The argument is unaffected.

Consider $\mathbb{F} \equiv \mathbb{E}(\{+, -, -_u, \times, /, \ln, \exp, \sqrt[n]{\cdot}\}, \mathbb{Z} \cup \{x\})$ where x is an indeterminate value. $f \in \mathbb{F}$ can be thought of as a function in x over $\mathbb{E}$ i.e. $f : \mathbb{E}_x \to \mathbb{E}_x$.

If f contains k occurrences of x then $f(f)$ is a function with k^2 occurrences of x. In general, defining $f_1 = f$ and $f_n = f(f_{n-1})$, f_n has k^n occurrences of x.

The weight of f now becomes a function which can be written $\mathcal{W}(f(x)) = w_f \mathcal{W}(x)^k$ where w_f corresponds to the part of the weight contributed by nodes in the expression other than the xs and $\mathcal{W}(x)^k$ reflects the k occurrences of the indeterminate. It follows that $\mathcal{W}(f_n) = w_f^r \mathcal{W}(x)^{k^n}$ where $r = \sum_{0 \le i \le n} k^i$.

If f, viewed as a function of x, has a zero at zero of multiplicity m, then $|f_n| = O(x^{m^n})$. This suggests a way of finding a counterexample: chose an integer $q > 1$ and consider the behavior of $f_n(1/q)$ as $n \to \infty$. If the Uniformity Conjecture were true we should expect that

$$\frac{q^{-k^n}}{K w_f^r} \le |f_n(1/q))| = O(q^{-m^n})$$

where K reflects the extra weight introduced by $1/$. But if $k < m$ then, for some n, the inequality cannot be true since the w_f^r factor also grows at $< O(w_f^{k^{n+1}})$. Joris Van Der Hoeven produced the first counterexample generator:

$$f = \log(1+x) - 2\log(1+\log(1+\frac{x}{2}))$$

This has only two occurrences of x, but is $O(x^3)$ at zero. This means that if $x = 10^{-N}$, then $f_n(x)$ has weight approximately 2^{nN}, but $|f_n(x)|$ is approximately $10^{-3^n N}$.

[RE03] shows how such expressions can be constructed, including differences of logarithms, exponentials, and radicals. Three more examples are

$$\sqrt{1+x} - \frac{25}{4} + \frac{21}{4}\sqrt{\frac{7}{5} - \frac{2}{5}\sqrt{-7+8\sqrt{1+\frac{5x}{21}}}} = -\frac{25}{111132}x^5 + O(x^6)$$

$$\ln(1+x) + 3\ln\left(1 - \frac{1}{2}\ln\left(1+\ln\left(1+\frac{2x}{3}\right)\right)\right) = -\frac{1}{1215}x^5 + O\left(x^6\right)$$

$$\frac{3}{2}\exp\left(\exp\left(\exp(-\frac{x}{3})+2\right)-1\right)-\frac{1}{2}-\exp(x)$$
$$=-\frac{1}{1215}x^5+O\left(x^6\right)$$

Though not discussed in [RE03] it is obvious that the construction can also produce counterexamples of other forms, for example $\ln(1+f_n)$ where f_n is a counterexample for the form above. From this it follows that there are counterexamples of the form $\log(e)$ for all forms of expression e, e.g. $\log(a*b)$ where a is any non-zero expression and $b=(1+f_n)/a$. There is still no evidence for a *minimal* counterexample of the form $\log(e)$ as identified in Prop. 23.

7.8 An Alternative Uniformity Conjecture

The results quoted above led Richardson to a new conjecture

Conjecture 5 (Revised Uniformity Conjecture). *For all* $e \in \mathbb{E}_x$,

$$|V(E)| \geq \max(W,2)^{-C\,2^d}$$

where W *is the maximum of the absolute values of the integers which occur in* e, d *is the depth of the expression representing* e, *and* C *is a universal constant independent of* E.

Even with $C=1$, no counterexamples have been found. As an example, and taking $C=1$, this would give a bound of H^{-8} for $|2^{1/n}-p/q|$, where $H=\max\{n,|p|,|q|\}$. For large n this is stronger than the Liouville inequality, but weaker than the Thue-Siegel-Roth theorem.

The decision taken here is slightly different: strengthen the conjecture to reflect the the special role of $a\pm b$ and $\log(a)$ and, at the same time, extend the range

of constants to $\mathbb{Q}[i]$. The latter choice is only to allow $\imath$ to be used in expressions without the circumlocution of writing $\mathcal{W}(\imath) = \mathcal{W}(\sqrt{-1})$.

Proposition 24. *For $e \in \mathbb{E}_x$, let $d(e)$ be the depth of an expression, let $s : \mathbb{N} \to \mathbb{R}$ be an unknown function and let the weight for any e be*

$$\mathcal{W}(e) = \begin{cases} \max(|e|, 2) & \text{if } e \in \mathbb{N} \\ \mathcal{W}(a)\,\mathcal{W}(b) & \text{if } e = a \times b \text{ or } e = a/b \\ (\mathcal{W}(a)\,\mathcal{W}(b))^{s(d(e))} & \text{if } e = a \pm b \\ \mathcal{W}(a)\,\mathcal{W}(n) & \text{if } e = \sqrt[n]{a} \\ \mathcal{W}(a) & \text{if } e = -a \text{ or } e = /a \\ 2\,\mathcal{W}(a) & \text{if } e = \exp(a) \\ 2\,\mathcal{W}(a)^{s(d(e))} & \text{if } e = \log(a) \end{cases}$$

If $s(n) \geq 1$ and monotonically increasing and if for every $a, b \in \mathbb{E}_x$

$$|a + b| \geq \mathcal{W}(a + b)^{-1}$$

then for all $e \in \mathbb{E}_x$

$$\mathcal{W}(e) > |e| \geq \mathcal{W}(e)^{-1}$$

Proof. The proof is by induction on the number of nodes in the expression and the argument is as used for rational expressions in Prop. 21 for the cases $e \in \mathbb{Q}[i]$, $a \times b$, a/b, $-a$, $/a$ and the upper bound to $a \pm b$ (since by hypothesis $s(d(a \pm b)) \geq 1$).

For $\exp(a)$, we require $|a| \leq 1$ and since $\mathcal{W}(a) \geq 2$ for all expressions the result follows.

The argument in Prop. 23 can be used for $\sqrt[n]{a}$. It might be tempting to use $\mathcal{W}(\sqrt[n]{e}) = \mathcal{W}(e)$, however no dependence on n would imply $|\sqrt[n]{2} - 1| > 2^{-s(2)}$ for all n. The Thue-Siegel-Roth Theorem and the symmetric Liouville conjecture suggest that dependence on degree in algebraic expressions is weak. It may be that $\mathcal{W}(a) + \lceil \log n \rceil$ is an adequate weight but nothing is definitely known.

For $\mathcal{V}(a) = |a|\exp(\imath\theta)$ with $|a| \geq 1$

$$|\log(a)| = |\log|a| + \imath\theta| \leq |a| + |\theta|$$
$$\leq 2\,\mathcal{W}(a) \leq 2\,\mathcal{W}(a)^{s(d(a))}$$

For the lower bound $\ln(1) = 0$ is the only zero of ln, so if $\ln(a) \neq 0$ how close to 1 can an expression of a given weight be? Since all expressions have weight ≥ 2 and the expressions '2' and '/2' have exactly that weight it can be assumed that if $\ln(a)$ is small then $1/2 \leq |a| \leq 2$. Expanding $\ln(a)$ about 1 in the annulus $D = \{z | z \in \mathbb{C},\ 1/2 \leq |z| \leq 2\}$

$$|\ln(a)| \geq \frac{|a-1|}{\max_{\alpha \in D}|\alpha|} \geq \frac{|a-1|}{2}$$

However, by hypothesis,

$$|a-1| \geq \frac{1}{\mathcal{W}(a-1)} = \mathcal{W}(a)^{-s(1+d(a))}$$

or

$$|\ln(a)| \geq \frac{\mathcal{W}(a)^{-s(d(e))}}{2}$$

□

Of course there may be no function s with the required qualities. Assuming that s exists what properties must it have? For example, using $\mathcal{W}(\imath) == 2$ and $\mathcal{W}(\pi) = \mathcal{W}(-\imath\log(-1)) = 2\,\mathcal{W}(\log(-1)) = 2^{s(2)+1}$

Expression	Actual Value	Implied Bound
$\sqrt[8]{2} - 494/453$	6.4×10^{-9}	$s(3) \geq 44$
$\sqrt[5]{109} - 23/9$	1.6×10^{-6}	$s(3) \geq 56$
$3\ln(640320)/\sqrt{163} - \pi$	2.3×10^{-16}	$s(10)s(2)^2 > 56204$

More importantly how does s perform with respect to the counterexamples above? The weakest form that s could take would be a constant value, S say; is this sufficient to preserve the conjecture?

As before assume f contains k occurrences of x and has the form $a \pm b$. $\mathcal{W}(f(x)) \geq (w_f \mathcal{W}(x)^k)^S$ where w_f reflects the weight of the unchanging part of the expression and it has been assumed that only one $\pm$ is present in the expression and no $\log(\cdot)$ sub-expressions (alternatively they are subsumed under w_f). $\mathcal{W}(f_2(x))$ is a product of terms including $(\mathcal{W}(x)^{k^2})^{k^2 S^2} = \mathcal{W}(x)^{(Sk^2)^2}$. Repeating this $\mathcal{W}(f_n(x)) > \mathcal{W}(x)^{S^n k^{n^2}}$ for any x. Though the analysis is crude it is clear that $\mathcal{W}(f_n(x))^{-1}$ is smaller than $O(k^{n^2})$ near zero and so the counterexamples fail.

7.9 Conclusions

The title of [RL02] is "Some Observations on Familiar Numbers", chosen to suggest the 'natural history' status of our current knowledge of bounds for exp-log expressions. All we really know at the moment is that very small valued expressions seem to be rare and are 'long' or 'weighty' depending on the model used.

No significant results have been proved. If any of the conjectures are true no proof is in sight. This is hardly surprising given the enormous effort expended in Diophantine approximation to prove results for much more restricted classes of expression. The ease with which the Uniformity Conjecture gives similar results suggests we shouldn't be too optimistic about proving it.

Nevertheless, if a bound could be found there would be a simple, quick (and entirely arithmetic) way of deciding if expressions are zero. The operators used in exp-log expressions can be implemented efficiently for arbitrary precision floating point calculation. Even where lower bounds are very small, computing sufficient decimal places to decide if an expression is zero would be much more effective

that algorithmic methods.

The existence of counterexamples to the earlier conjectures has led to modifications. Such tinkering is unsatisfactory in some ways but has produced conjectures which cannot be broken by any method we have been able to imagine. It is known that any counterexample must either be of the form $\log(a)$ or $a \pm b$ (unless it contains a counterexample sub-expression). Whether there are any true log-type counterexamples (i.e. not containing counterexample sub-expressions) to even the original conjecture remains unknown.

It may of course be that all the conjectures are false. One way of disproving them would be to show that they contradict some result from Diophantine approximation. So far none have been found though this is hardly surprising as most such results depend on detailed understanding of properties of very restricted subclasses of expression - the Uniformity Conjectures are much weaker in general.

Chapter 8

Conclusion

The initial aim of this project was to develop, if possible, usable algorithms for lazy exact real computation. As discussed in the introduction there are at least three facets to this: selecting subsets of the reals for which it is possible/practicable; using 'lazy' (i.e. general, non-canonical) representations of number fields; and, using 'lazy' arithmetic (i.e symbolic manipulation without reduction to canonical forms combined with numeric zero/sign testing).

8.1 Interval Arithmetic for Exact Arithmetic

8.1.1 Practical Issues

Interval arithmetic is only one way among many of obtaining guaranteed accuracy in calculations. However no other technique we have discovered has the maturity or breadth of users. This may not be of mathematical significance, but from the perspective of an implementer it is a great strength. Most of the corpus of interval arithmetic assumes hardware floating point arithmetic. A minor disadvantage is

that though many computer algebra packages provide arbitrary precision floating point, few are concerned with rounding issues and most provide a 'closest value' result. There is now an interval package available for GMP which is valuable for those using compiled languages. Even without GMP we have successfully implemented interval libraries in several packages (including Maple and MuPAD) where the rounding behaviour of floating point numbers is largely undocumented.

8.1.2 Validating Functions

Separating roots in a box has been a standard process in interval numerical methods for some time. Several products exist, some using much more sophisticated methods than described here. Though they can manipulate a wider range of functions, they have less power in one respect at least. In general they use hardware floating point, but even where they do not, they are not guaranteed to find all roots, but to return a list of boxes containing roots plus a list of (small) boxes which may or may not contain roots. By restricting our domain to boxes and restricting the allowable classes of function, the algorithm here does guarantee to decide if there is exactly 0, 1 or more roots and if there are a finite number, it can be adapted to count and/or separate them.

As shown in Propositions 6 and 7, the method can be extended to a wider range of functions. It is not clear whether it can be extended to, for example, elementary numbers (see §8.3) which can include terms of the form e^{e^x}.

8.2 Transforming Equations to Univariate Form

The complexities introduced by the use of multivariate systems of equations are immense compared with univariate systems. No doubt to a 'real' mathematician this increases the challenge and the interest but for a middle aged programmer like the author, the instinctive reaction is to do the one off conversion to an easier structure and carry out only the sign determination lazily but making use of the better bounds available for univariate systems (and since the Mahler measure is easy to compute in this case and only needs to be done once, Liouville's bound can be used).

Gap functions generally vastly under estimate the smallest size of roots. Few very small values encountered in the course of calculations are other than zero. When they appear to be zero it is often the case that a slight increase in precision will dispose of them. This suggest that one practical approach is to use the Schmidt conjecture from Eq. 4.4

$$|\alpha - \beta| \geq 1/(\max(M(\alpha), M(\beta)))^{2+\delta}$$

$q(\xi)$ could be written as $\alpha - \beta$ and a bound found for the Mahler measures and degrees of α and β. Lioville's bound would give $q(\xi) \neq 0$ implies

$$|q(\xi)| = |\alpha - \beta| \geq 2^{-d_\alpha d_\beta} M(\alpha)^{-d_\beta} M(\beta)^{-d_\alpha}$$

however choosing a small random number for δ there is a probability 1 that the previous equation holds. It is well worth checking since $q(\xi)$ may be shown to be non-zero by interval evaluation at the larger size.

Converting a general system to univariate form can be done in a number of ways. Two methods have been suggested, algebraically using u-resultants and

numerically using lattice reduction algorithms. In either if the estimated Bezout number of a system is D the computation will involve matrices of size $D \times D$.

The PSLQ algorithm requires floating point precision of order mD digits to detect a polynomial having coefficients of size $< 10^m$ [D B97]. The root bound from Eq. 6.4 also provides a bound for coefficients, implying $m = O(D^n \log H + (nD)^n \log D)$. The polynomial in the Ramanujan example Eq. 5.1 has $n = 5$, $D = 1874$ and $H = 2000$ which makes it impractical to solve in this way. Even if precision were not a problem, [FBA99] gives a bound for the number of iterations necessary to find a relation which in this case reduces to $O(D^n \log(nHD^n))$.

Of course, similar problems occur with Gröbner or resultant methods. If a Gröbner basis already exists, a companion matrix of a variable can be constructed in $O(nD)$ polynomial divisions from which the polynomal can be extracted in $O(D^3)$ integer operations. However we have no figure for the cost of a division which seems reliable, while the cost of producing a Gröbner basistends to make the method impracticable for all but the smallest systems.

8.3 Elementary Numbers

Our initial interest came from Dr Richardson's work on elementary numbers (in e.g. [Ric97, Ric96a, Ric96b]). His starting point is a set of equations $S = P \cup E$ comprising where

$$P = \{p_i \in \mathbb{Z}[x_1, \ldots, x_n] \ : \ 1 \leq i \leq k\} \text{ and}$$

$$E = \{w_i - \exp(z_i) \ : \ 1 \leq i \leq s, \{w_i, z_i\} \subset \{x_1, \ldots, x_n\}\}$$

which define a point ξ (i.e. $S(\xi) = 0$ is the only zero of S in some $\mathbf{B}$ and $J(S, \xi) \neq 0$).

To decide if $q(\xi) = 0$ for some $q \in \mathbb{Z}[x_1, \ldots, x_n]$ has two stages: decide if $P(\xi) = 0$ implies $q(\xi) = 0$ (i.e. this is essentially an algebraic problem); if not eliminate one variable and repeat.

The first step might plausibly be seen as amenable to the methods of this thesis. Unfortunately it fails for two reasons. Firstly, all the algebraic methods depend on having finite sets of solutions. Secondly, even if that condition is met, they reduce to determining the coefficients (or bounds on them) of $\chi(z) = \det M(z)$ where χ, by construction, has integral coefficients. In the elementary case the coefficients will be polynomials in the e^{x_i} for which, in general we have no lower bounds.

8.4 Gröbner and Resultant Methods

Since the original material for Chapter 5 was written about eight years ago the method described has become much more well known. The problem with Gröbner methods has always been the cost of producing the basis but faster methods are now available for zero-dimensional ideals [GV00].

The new bound from the Macaulay Resultant is an improvement on Canny result but still impractically small. Possibly using the Dixon resultant, at least as far as finding the Dixon polynomial and then using its size to estimate bounds may be more practical. The main alternative would be using one of the newer Gröbner basis algorithms (which offer the possibility of singly exponential time complexity) followed by esimation of the norm of the matrix representing the polynomial being tested.

8.5 The Uniformity Conjecture

The Uniformity Conjecture is an intriguingly different way of looking at gap functions. Unfortunately, despite many experiments and some rethinking of the conjecture, no proof of the conjecture is in sight. A paper, [RL02], on the conjecture by Dr Richardson and the the author of this report was entitled "Some Observations on Familiar Numbers" with the slightly whimsical aim of suggesting a slight offering by an 18^{th} Century naturalist. It is certainly the case that expanded expressions are a plausible model for scientific computing, and the title hoped to convey something of how little known such familiar objects are.

8.6 Finally

When Dr Richardson proposed finding algorithms for 'Lazy Exact Real Computation' he commented that, if any were found, "Several people would be quite interested" after a pause he added "Maybe a dozen, worldwide" and after another pause "Including you and me of course". What has been described in the preceding chapters are a range of algorithms, all of which are usable in theory though none are tractable and none are realistic for systems with more than single digit numbers of variables. It is unlikely that any of the putative dozen will be surprised by the outcome. Whether any of the other eleven will find results of interest is not for the twelfth to say - though he does retain some slight interest in the topic!

Appendix A

Pseudo-Code Notation

The algorithms are written in pseudo-code with:

1. Nesting indicated by indenting as in Python, or by use of explicit end keywords. I.e.

```
1    if a > b
2        c ⟵ d
3    x ⟵ y
```

   is equivalent to

```
1    if a > b
2        c ⟵ d
3    end if
4    x ⟵ y
```

   Single statements blocks are often written on the same line as the test.

```
1    if a > b   c ⟵ d
2    x ⟵ y
```

2. `loop` is an endless loop which must be terminated by `return` or `exit loop`.

`next loop` jumps to beginning of the loop.

```
1   loop
2      if a > b
3         next loop
4   end loop
5   x ⟵ y
```

3. `fail` is equivalent to throw/raise and terminates all levels of the algorithm.

```
1   try
2      if x = 0  fail
3      y ⟵ 1
4   on fail
5      ......
6   else
7      ......
```

4. $\mathbf{Y}(i : \mathbf{a}) \equiv (\mathbf{y_1}, \ldots, \mathbf{y_{i-1}}, \mathbf{a}, \mathbf{y_{i+1}}, \ldots, \mathbf{y_n})$.

5. If X is a variable in an algorithm, $\underset{\text{init}}{X}$ is its initial value, $\underset{\text{pre}}{X}$ and $\underset{\text{post}}{X}$ are its values at the beginning (end) of a section of code. X is used if the meaning is clear.

Appendix B

Algorithms

The *Validate* algorithms are repeated here for convenience.

```
1   Algorithm Validate
2   Input: Y                                  ▷ box to be validated/approximated
3   W ⟵ vol Y
4   R ⟵ {Y}
5   T ⟵ ∅
6   while R ≠ ∅
7      S ⟵ ∅                                  ▷ new boxes which may contain roots
8      for Y ∈ R
9         if dim Y = 0
10           if F(Y) = 0   T ⟵ T ∪ {Y}
11           next loop
12        end if
13        V ⟵ F(Y)
14        if 0 ∉ V  next loop                 ▷ Cannot contain a zero
15        X ⟵ Y
16        C ⟵ an approximation to J^{-1}(F, m(X))
17        U ⟵ CJ(F, X) − I
18        Y ⟵ (m(X) − Cm(V) − U(X − m(X))) ∩ X
19        if Y = ∅  next loop                 ▷ can't contain a zero
20        if Y ⊂ int(X)                       ▷ exactly one root in Y
21           T ⟵ T ∪ {Y}
22        else if ||U|| < 2                   ▷ at most one root
23           if vol Y < δ                     ▷ very small box
24              ▷ does it lie on the boundary?
```

```
25                  Z ⟵ Boundary(Y, X)
26                  if Z = ∅                    ▷ no root on boundary
27                     S ⟵ S ∪ Bisect(Y, true)
28                  else
29                     T ⟵ T ∪ {Z}              ▷ one root on boundary
30                  end if
31               else
32                  S ⟵ S ∪ Bisect(Y, true)
33               end if
34            else
35               S ⟵ S ∪ Bisect(Y, false)
36            end if
37         end for
38         if #T > 1  fail                       ▷ at least two roots
39         R ⟵ S
40      end while
41      if R = ∅
42         return ∅
43      else if R = {X}
44         if vol X > (2/3)W   X ⟵ Bisect(X, true)
45         return X
```

```
1    Algorithm Boundary                         ▷ at most one root on boundary
2    Input: Y                                   ▷ Krawczyk operator applied to X
3    Input: X                                   ▷ Y ⊂ X
4    ▷ At most one root, check boundary
5    for i from 1 to n                          ▷ all possible dimensions
6       if w(x_i) ≠ 0                           ▷ ignore point dimensions
7          if x_i = y_i                         ▷ coincident boundary
8             U ⟵ Validate(Y(i : [y_i, y_i]))
9             if U ≠ ∅  return U
10         else if x̄_i = ȳ_i
11            U ⟵ Validate(Y(i : [ȳ_i, ȳ_i]))
12            if U ≠ ∅  return U
13         end if
14      end if
15   end for
16   return ∅
```

```
1    Algorithm Bisect
2    Input: Y                                   ▷ Box
3    Input: singleRoot                          ▷ boolean - true = max of one root
```

```
4   m ⟵ w(Y)/2
5   i ⟵ index of widest dim of Y          ▷ bisect box
6   a_1 ⟵ y_i + m                         ▷ first try at about 1/2 length
7   Z_1 ⟵ Validate(Y(i : [a_1, a_1]))
8   if Z_1 = ∅
9      return {Y(i : [y_i, a_1]), Y(i : [a_1, ȳ_i])}
10  else if singleRoot
11     return Z_1
12  end if
13  ▷ Try again near first division
14  a_2 ⟵ a_1 ± +ε/2
15  Z_2 ⟵ Validate(Y(i : [a_2, a_2]))
16  if Z_2 ≠ ∅  fail                      ▷ More than one root
17  return {Y(i : [y_i, a_2]), Y(i : [a_2, ȳ_i])}
```

```
1   Algorithm Approximate
2   Input: X                              ▷ validated box
3   Input: W                              ▷ find approx. with width < W
4   while w(X) >= W
5      X ⟵ Validate(X)
6   end while
7   return X
```

References

[Abe94] O Aberth. Computation of Topological Degree using Interval Arithmetic, and Applications. *Mathematics of Computation*, 62(205):171–178, January 1994.

[Abe98] O Aberth. *Precise Numerical Methods Using C++*. Academic Press, 1998.

[ABRW96] M-E Alonso, E Becker, M-F Roy, and T Wörmann. Zeros, Multiplicities and Indempotents for Zero-Dimensional Systems. *Progress in Mathematics*, 143:1–15, 1996.

[ACM84] D Arnon, G Collins, and S McCallum. Cylindrical Algebraic Decomposition I: The Basic Algorithm. *SIAM Journal of Computing*, 13(4):865–877, 1984.

[AH83] G Alefeld and J Herzberger. *Introduction to Interval Computation.* Academic Press, 1983.

[AL94] W Adams and P Loustaunau. *An Introduction to Gröbner Bases.* American Mathematical Society, 1994.

[Alg] Algorithmic Solutions Group. The LEDA User Manual. Website at `http://www.mpi-sb.mpg.de/LEDA/MANUAL/MANUAL.html`.

[AM99] P Aubry and M Maza. Triangular Sets for Solving Polynomial Systems: a Comparative Implementation of Four Methods. *J. Symbolic Computation*, 28:125–154, 1999.

[Ax71] J Ax. On Schanuel's conjecture. *Ann. of Math.*, 93(2):252–268, 1971.

[BCSS98] L Blum, F Cucker, M Shub, and S Smale. *Complexity and Real Computation.* Springer–Verlag, 1998.

[BFMS00] C Burnikel, R Fleischer, K Melhorn, and S Schirra. A strong and easily computable separation bound for arithmetic expressions involving radicals. *Algorithmica*, 27:87–99, 2000.

[BP81] W Burnside and A Panton. *The Theory of Equations.* Dublin University Press, 1881.

[BP98] D Bailey and S Plouffe. Finding New Mathematical Identities via Numerical Computations. *ACM SIGNUM*, 33(1):17–22, January 1998.

[Bri02] K Briggs. XR - Exact Real Arithmetic. Website at `http://more.btexact.com/people/briggsk2/XR.html`, 2002.

[BSS89] L Blum, M Shub, and S Smale. On the Theory of Computation and Complexity over the Real Numbers: NP-completeness, Recursive Functions and Universal Machines. *Bull. Amer. Math. Soc.*, 21:1–46, 1989.

[Can88] J Canny. *The Complexity of Robot Motion Planning.* ACM Doctoral Dissertation Award. MIT, 1988.

[Can91] J Canny. An Improved Sign Determination Algorithm. In Mattson et al. [MMR91], pages 108–117.

[Cav70] B F Caviness. On canonical forms and simplification. *Journal of the ACM*, 17(2):385–396, 1970.

[Cho99] T Chow. What is a closed-form number? *American Mathematical Monthly*, 106(5):440–448, 1999.

[CL76] G Collins and R Loos. Polynomial real root isolation by differentiation. In *Proceedings of SYMSAC 76*, pages 15–25, New York, 1976. ACM Press.

[CLO92] D Cox, J Little, and D O'Shea. *Ideals, Varieties and Algorithms.* Springer–Verlag, 1992.

[CLO98] D Cox, J Little, and D O'Shea. *Using Algebraic Geometry.* Springer–Verlag, 1998.

[D B97] D Bailey and S Plouffe. Recognizing Numerical Constants. *Canadian Mathematical Society*, 20:73–88, 1997.

[DGY96] J-P Dedieu, X Gourdon, and J-C Yakoubsohn. Computing the Distance from a Point to an Algebraic Hypersurface. In J. Renegar, M. Shub, and S. Smale, editors, *Proceedings of the American Mathematical Society, seminar of Park City on Mathematics of Numerical Analysis: Real Number Algorithms*, July 1996.

[DST93] J Davenport, Y Siret, and E Tournier. *Computer Algebra.* Academic Press, 1993.

[EP97] A Edalat and P Potts. A New Representation for Exact Real Numbers. *Electronic Notes in Theoretical Computer Science 6*, 1997.

[Eve00a] J-H Evertse. Symmetric improvements of Liouville's inequality. *J. reine angew. Math.*, 527:69–95, 2000.

[Eve00b] J-H Evertse. Symmetric improvements of Liouville's inequality: A survey. In F Halter-Koch and R Tichy, editors, *Algebraic Number*

Theory and Diophantine Analysis, pages 129–141. Walter de Gruyter, 2000.

[FBA99] H Ferguson, D Bailey, and S Arno. Analysis of PSLQ, An Integer Relation Finding Algorithm. *Mathematics of Computation*, 68:351–269, January 1999.

[Fv93] S Fortune and C van Wyk. Efficient Exact Arithmetic for Computational Geometry. *Proc. of 9th Symp. on Computational Geometry*, pages 163–171, 1993.

[GH] M Giusti and J Heintz. Kronecker's Smart, Little Black Boxes.

[Gie95a] M Giesbrecht. Fast Computation of the Smith Normal Form of an Integer Matrix. In *International Symposium on Symbolic and Algebraic Computation*, pages 110–118, 1995.

[Gie95b] M Giesbrecht. Nearly Optimal Algorithms For Canonical Matrix Forms. *SIAM Journal on Computing*, 24(5):948–969, 1995.

[Gie01] M Giesbrecht. Fast Computation of the Smith Form of a Sparse Integer Matrix. *Comput. Complexity*, 10:41–69, 2001.

[GJ79] M Garey and D Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completness.* Freeman, 1979.

[GL96] G Golub and C Van Loan. *Matrix Computations (Third Edition).* Johns Hopkins University Press, 1996.

[GLS] M Giusti, G Lecerf, and B Salvy. A Gröbner Free Alternative for Polynomial System Solving.

[GLW99] T Gao, T Li, and X Wang. Finding All Isolated Zeros of Polynomial Systems in $\mathbb{C}^n$ via Stable Mixed Volumes. *J. Symbolic Computation*, 28:187–211, 1999.

[GS02] M Giesbrecht and A Storjohann. Computing Rational Forms of Integer Matrices. *Journal of Symbolic Computation*, 34(3), 2002.

[GV00] D Grigoriev and N Vorobjov. Bounds on Vectors of Multiplicities for Polynomials which are Easy to Compute. *ISSAC*, 2000.

[Ham70] S J Hamarling. *Latent Roots and Latent Vectors.* Adam Hilger, 1970.

[Han92] E Hansen. *Global Optimisation Using Interval Analysis.* Marcel Dekker, 1992.

[HD99] N Hur and J Davenport. An Exact Real Arithmetic with Equality Determination. *Draft, Bath University*, 1999.

[HS81] E Hansen and S Sengupta. Bounding Solutions of Systems of Equations using Interval Analysis. *BIT*, 21:203–211, 1981.

[Joh92] J Johnson. Real Algebraic Number Computation Using Interval Arithmetic. In *ISSAC '92*, 1992.

[KDN00] R Baker Kearfott, J Dian, and A Neumaier. Existence Verification for Singular Zeros of Complex Nonlinear Systems. *SIAM Journal of Numerical Analysis*, 38(2):360–379, 2000.

[Kea] R Baker Kearfott. Interval Notations. Book extract available at `www.cs.utep.edu/interval-comp/notations/suggestion.html`.

[Knu81] D Knuth. *Seminumerical Algorithms*, volume 2 of *The Art of Computer Programming.* Addison–Wesley, 1981.

[Kra86] R Krawczyk. A class of interval-Newton operators. *Computing*, 37:179–183, 1986.

[KS95] D Kapur and T Saxena. Comparison of Various Multivariate Formulations. In *Proc. 1995 Int. Symp. on Symbolic and Algebraic Computation.* ACM Press, 1995.

[KS97] D Kapur and T Saxena. Extraneous Factors in the Dixon Resultant Formulation. In *Proc. 1997 Int. Symp. on Symbolic and Algebraic Computation*, pages 141–148. ACM Press, 1997.

[KSY94] D Kapur, T Saxena, and L Yang. Algebraic and Geometric Reasoning using Dixon Resultants. In *Proc. 1994 Int. Symp. on Symbolic and Algebraic Computation*, pages 99–107. ACM Press, 1994.

[Lan65] S Lang. *Algebra.* Addison–Wesley, 1965.

[Li01] Chen Li. *Exact Geometric Computation.* PhD thesis, New York University, 2001.

[LiD] LiDIA Group at Technische Universität Darmstadt. LiDIA, A C++ Library for Computational Number Theory. Website at `http://www.informatik.tu-darmstadt.de/TI/LiDIA`.

[Mac16] F Macaulay. *The Algebraic Theory of Modular Forms.* Cambridge University Press, 1916. Reissued 1994.

[Mai98] A Maignan. Solving One and Two-dimensional Exponential Polynomial Systems. *J. of Complexity*, pages 215–221, 1998.

[Mai00] A Maignan. On Symbolic-Numeric Solving of Sine-Polynomial Equations. *J. of Complexity*, 16:274–285, 2000.

[MB01] K Makino and M Berz. Higher Order Verified Inclusions of Multidimensional Systems by Taylor Models. *Nonlinear Analysis*, 47:3503–3514, 2001.

[MB03] K Makino and M Berz. Taylor Models and Other Validated Functional Inclusion Methods. *International Journal of Pure and Applied Mathematics*, 4(4):379–456, 2003.

[Mig92a] M Mignotte. Identification of algebraic numbers. *Journal of Algorithms*, 3:197–204, 1992.

[Mig92b] M Mignotte. *Mathematics for Computer Algebra.* Springer–Verlag, 1992.

[MM97] K Meer and C Michaux. A Survey on Real Structural Complexity Theory. *Bulletin of the Belgian Math. Soc.*, 4:113–148, 1997.

[MMR91] H Mattson, T Mora, and T Rao, editors. *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes.* Springer–Verlag, 1991.

[MO91] H Müller and M Otte. Solving Algebraic Systems in Bernstein-Bézier Representation. In H Bieri and H Noltenmeier, editors, *Computer Geometry - Methods, Algorithms and Applications, Int. Workshop on Computational Geometry*, pages 162–169. Springer–Verlag, 1991.

[Möl93] H Möller. Systems of Algebraic Equations solved by means of Endomorphisms. In G Cohen, T Mora, and O Moreno, editors, *Applied Algebra, Algebraic Algorithms and Error Correcting Codes (AAECC-10).* Springer–Verlag, 1993.

[Moo79] R Moore. *Methods and Applications of Interval Analysis.* SIAM Studies in Applied Mathematics. SIAM, Philadelphia, 1979.

[MS87a] A Morgan and A Sommese. A homotopy for solving general polynomial systems that respects m-homogeneous structures. *Appl. Math. Comput.*, 24(2):101–113, 1987.

[MS87b] A Morgan and A Sommese. Computing all solutions to polynomial systems using homotopy continuation. . *Appl. Math. Comput.*, 24(2):115–138, 1987.

[MS95] H Möller and H Stetter. Multivariate Polynomial Equations with Multiple Zeros solved by Matrix Eigenproblems. *Numerische Mathematik*, 70:311–329, 1995.

[Mül00] N Müller. The iRRAM: Exact arithmetic in C++. In J Blanck, V Brattka, and P Hertling, editors, *Computability and Complexity in Analysis - CCA 2000*, volume 2064 of *Lecture Notes in Computer Science*, pages 222–252. Springer–Verlag, 2000.

[MW96] A Macintyre and A Wilkie. On the Decidability of the Real Exponential Field. In *Krieseliana: About and Around Georg Kriesel*, pages 441–467. A K Peters, 1996.

[Neu90] A Neumaier. *Interval Methods for Systems of Equations.* Encyclopedia of Mathematics and its Applications. Cambridge University Press, 1990.

[Neu02] A Neumaier. Taylor forms - use and limits. *Reliable Computing*, 9:43–79, 2002.

[Nit96] A Nitaj. La Conjecture *abc. Enseign. Math*, 42:3–24, 1996.

[Ped91a] P Pedersen. *Counting Real Zeroes.* PhD thesis, Courant Institute of Mathematical Sciences, New York University , 1991.

[Ped91b] P Pedersen. Multivariate Sturm Theory. In Mattson et al. [MMR91], pages 318–332.

[Phi99] P Philippon. Quelques remarques sur des questions d'approximation diophantine. *Bull Austral. Math. Soc.*, 59(2):323–334, 1999.

[Phi00] P Philippon. Addendum a quelques remarques sur des questions d'approximation diophantine. *Bull Austral. Math. Soc.*, 61(1):167–169, 2000.

[Poh93] M Pohst. *Computational Algebraic Number Theory.* Birkhauser Verlag, 1993.

[Pot96] P Potts. *Computable Real Arithmetic using Linear Fractional Transformations.* PhD thesis, Imperial College, 1996.

[PTVF92] W Press, S Teukolsky, W Vetterling, and B Flannery. *Numerical Recipes in C.* CUP, 1992.

[RE03] D Richardson and A Elsonbaty. Counterexamples to the uniformity conjecture. Preprint from the author, 2003.

[Ric54] H Rice. Recursive Real Numbers. *Proc. Amer. Math. Soc.*, 5(5):784–791, 1954.

[Ric68] D Richardson. Some Unsolvable Problems involving Elementary Functions of a Real Variable. *J. Symbolic Logic*, 33:514–520, 1968.

[Ric96a] D Richardson. Lazy Analysis and Elementary Numbers. *Lectures in Applied Mathematics*, Vol 32:665–676, 1996.

[Ric96b] D Richardson. Solutions of Elementary Systems of Equations in a Box in R^n. In Y Lakshman, editor, *Proc. 1996 Int. Symp. on Symbolic and Algebraic Computation*, pages 120–126. ACM Press, July 1996.

[Ric97] D Richardson. How to Recognise Zero. *J. Symbolic Computation*, 24(6):627–645, 1997.

[Ric99a] D Richardson. Computing with Nested Radical and Exponential-Logarithmic Expressions. *Draft, Bath University*, 1999.

[Ric99b] D Richardson. Testing the uniformity conjecture. Submitted for publication. Available from: `www.bath.ac.uk/~masdr/testu.dvi`,, 1999.

[Ric00] D Richardson. The uniformity conjecture. In *Proceedings of Computability and Complexity in Analysis 2000*, pages 253–272. Springer lecture notes in Computer Science, Volume 2064, 2000.

[RL02] D Richardson and S Langley. Some Observations on Familiar Numbers. In *Proc. 2002 Int. Symp. on Symbolic and Algebraic Computation*, pages 214–220. ACM Press, 2002.

[Rou99] F Rouillier. Solving zero-dimensional systems through the rational univariate representation. *Applicable Algebra in Engineering, Communications and Computing*, 9:433–461, 1999.

[Sch83] W Schmidt. Open Problems in Diophantine Approximation. *Prog. Maths.*, 31:271–287, 1983.

[Sch00] E Scheinerman. When close enough is close enough. *American Mathematical Monthly*, 107:489–499, 2000.

[SF96] R Sedgewick and P Flajolet. *An Introduction to the Analysis of Algorithms.* Adison-Wesley, 1996.

[SNar] H Schichl and A Neumaier. Exclusion regions for systems of equations. *SIAM J. Numer. Anal.*, to appear.

[Sto74] K Stolarsky. *Algebraic Numbers and Diophantine Approximation.* Marcel Dekker, 1974.

[Str97] A Strzeboński. Computing in the Field of Complex Algebraic Numbers. *J. Symbolic Computation*, 24:647–656, 1997.

[SvdH01] J Shackell and J van der Hoeven. Complexity Bounds for Zero-test Algorithms. Preprint from `http://www.math.u-psud.fr/~vdhoeven`, 2001.

[Tra98] Q-N Tran. A Symbolic-Numerical Method for Finding a Real Solution of an Arbitrary System of Nonlinear Algebraic Equations. *J. Symbolic Computation*, 26:739–760, 1998.

[Tur36] A Turing. On Computable Numbers with an Application to the Entscheidungsproblem. *Proc. of the London Math. Soc.*, 1936.

[Tur37] A Turing. On Computable Numbers with an Application to the Entscheidungsproblem, A Correction. *Proc. of the London Math. Soc.*, 1937.

[vdH00] J van der Hoeven. Zero-testing, witness conjectures and differential Diophantine approximation. Preprint from `http://www.math.u-psud.fr/~vdhoeven`, 2000.

[Vui90] J Vuillemin. Exact Real Computer Arithmetic with Continued Fractions. *IEEE Transactions on Computers*, 39:1087–1105, 1990.

[Wal00] M Waldschmidt. *Diophantine approximation on linear algebraic groups.* Number 326 in Grundlehren der Mathematischen Wissenschaften. Springer–Verlag, 2000.

[Wan] D Wang. Epsilon. Website at `http://www-calfor.lip6.fr/~wang/epsilon`.

[WBC01] D Wang, B Buchberger, and G Collins, editors. *Elimination Methods.* Telos Pr, 2001.

[Weg87] B De Weger. Solving exponential diophantine equations using lattice basis reduction algorithms. *J. Number Theory*, 26:325–367, 1987.

[Wu78] W Wu. On the Decision Problem and the Mechanization of Theorem-Proving in Elementary Geometry. *Sci. Sinica*, 21:159–172, 1978, 1978.

[XZW96] Zong-Ben Xu, Jiang-She Zhang, and Wei Wang. A Cell Exclusion Algorithm for Determining All the Solutions of a Nonlinear System of Equations. *Applied Mathematics and Computation*, 80:181–208, 1996.

[Yap] C Yap. The CORE Library Project. Website at `http://www.cs.nyu.edu/exact/core`.