

Pisot numbers and the Spectra of Real numbers

by

Kevin Hare

B.Math, University of Waterloo, 1997.

M.Sc, Simon Fraser University, 1999.

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
PH.D.

in the Department
of
Mathematics.

© Kevin Hare 2002

SIMON FRASER UNIVERSITY

June 2002

All rights reserved. This work may not be
reproduced in whole or in part, by photocopy
or other means, without the permission of the author.

APPROVAL

Name: Kevin Hare
Degree: Ph.D.
Title of thesis: Pisot numbers and the Spectra of Real numbers

Examining Committee: Dr. M. Trummer
Chair

Dr. P. Borwein
Senior Supervisor

Dr. P. Lisonek
Supervisory Committee

Dr. I. Chen
Supervisory Committee

Dr. S. Choi

Dr. C. Smyth
External Examiner

Date Approved: _____

Abstract

It's all nonsense, that stuff they fill your heads with, the Academy and all of those books and primers and philosophy – all that five-times-six – a curse on the lot of it!

Taras Bulba – Nikolai Gogol

Pisot numbers have a long history, being studied as early as 1912 by Thue [38]. Some simple examples of Pisot numbers are the golden ratio (approximately 1.6180339), all integers greater than or equal to 2, and the real root of $x^3 - x - 1$ (approximately 1.324717957). Salem shows that the set of Pisot numbers is infinite and very structured [31]. Pisot numbers appear in a variety of different areas of mathematical research, such as: β -expansions [24], disjoint coverings of the natural numbers [2], robotics [11], quasilattices and quasicrystals [9], exceptional sets in harmonic analysis [31], and Salem numbers [31, 32]. A recent line of inquiry, initiated by Erdős, Joó and Komornik [13], is the determination of $l^1(q)$ for Pisot numbers q . Here $l^m(q) := \inf\{|y| : y = \epsilon_n q^n + \epsilon_{n-1} q^{n-1} + \dots + \epsilon_0, \epsilon_i \in \{\pm m, \pm(m-1), \dots, \pm 1, 0\}, y \neq 0\}$. This line of inquiry is generalized by considering the spectra of any set of polynomials evaluated at q , where the coefficients of the polynomials are restricted to a finite set of integers. Some common generalizations include restricting the coefficients to $\{0, 1\}$ or $\{\pm 1\}$.

An algorithm for computing $l^m(q)$ and its generalizations is given in this thesis. Using this algorithm a systematic investigation of the spectra of $\{\pm 1\}$ polynomials is done. This investigation results in the discovery of non-Pisot numbers with discrete

spectra. Furthermore a complete description of $l^m(q)$ for all unit quadratic Pisot numbers is given. A similar description appears to be possible for some cubic Pisot numbers, but so far no proof is known. This class of cubic Pisot numbers is studied.

Acknowledgments

Particular people and organizations that I would like to thank include:

- My Supervisory Committee, for a large number of useful comments and correction.
- Warren Hare, for such useful comments as: “This sentence is nonsensical.”
- Jeff Graham, for editing my thesis for grammar.
- Peter Borwein, for pointing out that proof reading comes naturally for some people, and that I am not one of these people.
- Jason Loepky, for reading a single sentence, finding a single misprint, and then whining until he got listed in the acknowledgements.
- Idris Mercer, for complaining about a comma.
- NSERC, for funding (PGS A and PGS B).
- MITACS for funding as an RA, the Math Department as funding as a TA/SI, SFU for funding in the form of fellowships and scholarships.
- The Weekly World News, for being a wonderful source of honest and informative news.

Contents

| | |
|--|-----|
| Abstract | iii |
| Acknowledgments | v |
| List of Tables | ix |
| List of Figures | x |
| 1 Introduction and Background | 1 |
| 1.1 Algebraic numbers | 1 |
| 1.2 Spectra | 6 |
| 1.3 Generalizations of Spectra | 10 |
| 2 An Algorithm for Computing Spectra | 11 |
| 2.1 Background to the Algorithm | 11 |
| 2.2 The Algorithm | 12 |
| 2.3 Implementation | 16 |
| 2.4 Running Time of the Algorithm | 17 |
| 3 Explorations of Spectra | 22 |
| 3.1 Spectra of $\Lambda(q)$ and $A(q)$ for Pisot Numbers q | 22 |
| 3.2 $A(q)$ for Non-Pisot Numbers q | 26 |
| 3.3 Salem Numbers and $A(q)$ | 31 |

| | | |
|------------|--|----|
| 3.4 | Quadratic and Cubic Pisot numbers | 34 |
| 4 | Unit Quadratic Pisot Numbers | 35 |
| 4.1 | Background on Quadratic Pisot Numbers | 35 |
| 4.2 | Description of the Proof | 36 |
| 4.3 | Main Theorem | 39 |
| 4.4 | Finding Height m Polynomials | 46 |
| 4.5 | Non-Unit Quadratic Pisot Numbers | 48 |
| 4.6 | Further Research | 48 |
| 5 | Cubic Pisot Numbers | 51 |
| 5.1 | Background on Cubic Pisot Numbers | 51 |
| 5.2 | Upper Bound for a Sequence of Quadratics | 56 |
| 5.3 | Lower Bound for all Quadratics | 61 |
| 5.4 | Proof that $D(q) < 1$ | 64 |
| 5.5 | Bounds for the Height with Respect to ϵ | 68 |
| 5.6 | Further Research | 70 |
| 6 | Some Conclusions and Open Questions | 72 |
| 6.1 | Open questions | 72 |
| 6.2 | Generalizations | 74 |
| Appendices | | |
| A | Code | 77 |
| A.1 | Data Types | 77 |
| A.2 | Spectrum Algorithm | 87 |
| A.3 | Top Level Code | 90 |
| A.4 | Compiling Stuff | 96 |

| | | |
|-----|------------------------------|----|
| A.5 | GNU Public License | 99 |
|-----|------------------------------|----|

List of Tables

| | | |
|-----|--|----|
| 3.1 | Pisot numbers where $l(q) = a(q)$ | 24 |
| 3.2 | Pisot numbers that do not satisfy ± 1 polynomials | 25 |
| 3.3 | Successful calculations of $l(q)$ with a spectrum over 20 million | 27 |
| 3.4 | Successful calculations of $a(q)$ with a spectrum over 20 million | 28 |
| 3.5 | Polynomials with non-uniformly discrete spectrum $A(q)$ | 30 |
| 3.6 | Known Salem numbers q of degree ≤ 10 , where $A(q)$ is discrete | 31 |
| 4.1 | Range for m when $l^m(q) = F_{k-1}q - F_k $ | 45 |
| 4.2 | Range for m when $l^m(q) = E_{k-1}q - E_k $ or $ G_{k-1}q - G_k $ | 46 |
| 5.1 | Relationship between $P_n(q)$ and $l^m(q)$ for the first cubic Pisot number | 53 |
| 5.2 | Relationship between $P_n(q)$ and $l^m(q)$ for the second cubic Pisot number | 54 |
| 5.3 | $D(q)$ for various q | 65 |

List of Figures

| | | |
|-----|---|----|
| 1.1 | Roots of $x^3 - x - 1$ | 3 |
| 1.2 | Roots of $x^6 - x^5 + x^2 - x^4 - 1$ | 4 |
| 1.3 | Roots of $x^4 - 2x^3 + x - 1$ | 4 |
| 2.1 | Algorithm for finding the spectrum of a Pisot number q | 14 |
| 4.1 | (s, t) where $sq + t \in \Lambda^2(q)$, q satisfies $x^2 - 3x + 1$, $ s \leq 20$ | 37 |
| 4.2 | (s, t) where $sq + t \in \Lambda^2(q)$ and bounding lines | 38 |
| 4.3 | (s, t) where $sq + t \in \Lambda^2(q)$, q satisfies $x^2 - 3x + 1$, $s \leq 20$, and lines $sr + t = \pm \frac{2}{1- r }$ | 38 |
| 4.4 | (s, t) where $sq + t \in \Lambda^3(q)$ | 49 |
| 5.1 | $T_1(x, y)$ on the region of $x \geq 0$ | 67 |
| 5.2 | $T_2(x, y)$ on the region of $x \leq 0$ | 68 |
| 5.3 | Regions where $D_1(q)$ and $D_2(q)$ are minimal | 69 |
| 6.1 | $(x, y) \in \Lambda^S(\tau)$ for $S \subset \mathbb{R}^2$, $\sqrt{x^2 + y^2} \leq 10$ | 76 |

Chapter 1

Introduction and Background

I was at the mathematical school, where the master taught his pupils after a method scarce imaginable to us in Europe. The proposition and demonstration were fairly written on a thin wafer, with ink composed of a cephalic tincture. This the student was to swallow upon a fasting stomach, and for three days following eat nothing but bread and water. As the wafer digested, the tincture mounted to his brain, bearing the propositions along with it.

Gulliver's Travels – Jonathan Swift

1.1 Algebraic numbers

Pisot numbers have a long history, being studied as early as 1912 by Thue [38]. Salem first became interested in Pisot numbers q because of their property that $q^n \rightarrow 0 \pmod{1}$ as $n \rightarrow \infty$ [32]. Moreover, Salem shows that Pisot numbers are the only algebraic numbers that have this property. Recall that an *algebraic number* is a root of a polynomial with integer coefficients, and an *algebraic integer* is a root of a monic polynomial with integer coefficients. The *conjugates* of an algebraic number

are the other roots of the algebraic number's minimal polynomial. (Note, this is non-standard, as typically the conjugates of an algebraic number include itself.) Salem gives a straightforward description of Pisot numbers that we give here as the definition:

Definition 1.1 (Pisot number). *A Pisot number is a positive real algebraic integer greater than 1, all of whose conjugates are of modulus strictly less than 1.*

For completeness, we give definitions for two related sets of algebraic integers, namely the Salem numbers and cyclotomic numbers.

Definition 1.2 (Salem number). *A Salem number is a positive real algebraic integer greater than 1, all of whose conjugates are of modulus less than or equal to 1. At least one of the conjugates must be of modulus 1.*

Definition 1.3 (Cyclotomic number). *A cyclotomic number is an algebraic integer α such that $\alpha^n = 1$ for some $n \in \mathbb{Z}$, $n \neq 0$.*

In [32] Salem shows that the set of Pisot numbers is closed. Furthermore, Salem shows that every Pisot number is a two sided limit of Salem numbers.

Recall:

Definition 1.4 (Mahler measure). *The Mahler measure of a polynomial with integer coefficients of the form:*

$$a_n x^n + a_{n-1} x^{n-1} + \cdots + a_0 = a_n \prod_{i=1}^n (x - \alpha_i)$$

is:

$$|a_n| \prod_{i=1}^n \max(1, |\alpha_i|).$$

The Mahler measure of an algebraic number is the Mahler measure of the algebraic number's minimal polynomial.

Further recall:

Definition 1.5. The reciprocal of a polynomial $P(x)$ is defined as

$$P^*(x) := x^d P\left(\frac{1}{x}\right),$$

where d is the degree of P .

Smyth shows that if p is irreducible, and $p(x) \neq \pm p^*(x)$ then the minimal Mahler measure is that for $x^3 - x - 1$ [35].

Example 1. We consider three examples. These examples are $x^3 - x - 1$ (Figure 1.1), $x^6 - x^5 - x^4 + x^2 - 1$ (Figure 1.2) and $x^4 - 2x^3 + x - 1$ (Figure 1.3). The locations of the roots of these polynomials are plotted, along with the unit circle. Notice that for each example, all but one root is within the unit circle. The one root that is not within the unit circle is a positive real algebraic integer, and hence a Pisot number.

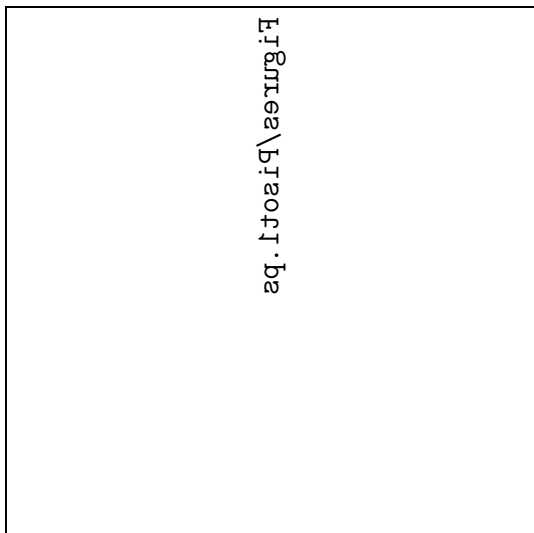


Figure 1.1: Roots of $x^3 - x - 1$

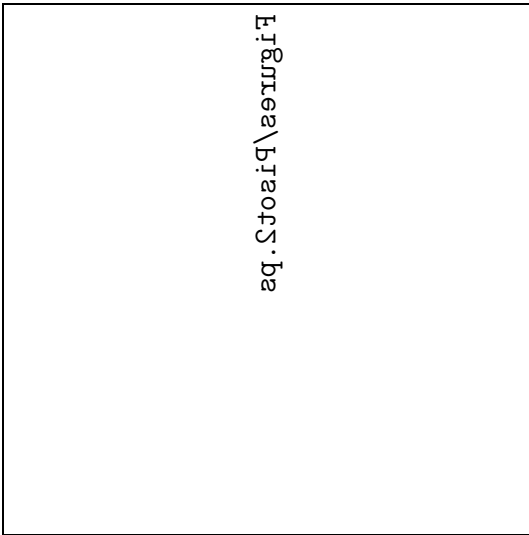


Figure 1.2: Roots of $x^6 - x^5 + x^2 - x^4 - 1$

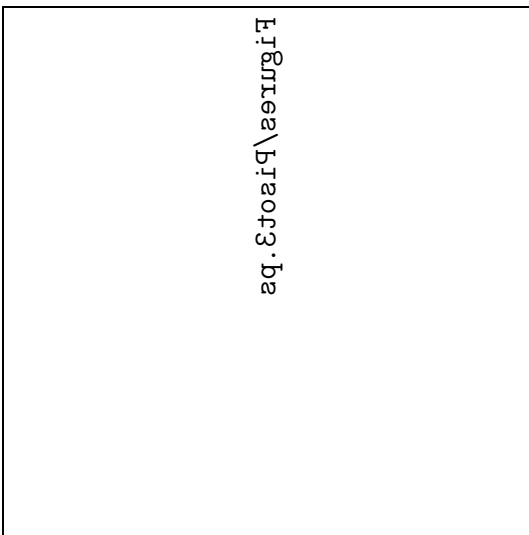


Figure 1.3: Roots of $x^4 - 2x^3 + x - 1$

We define:

Definition 1.6 (Height). Let $p(x) = a_n x^n + \cdots + a_0$. The height of $p(x)$ is denoted:

$$H(p(x)) = \max\{|a_0|, \dots, |a_n|\}.$$

A classic result, [8, 14, 18], shows that if q is a Pisot number, and $p(x)$ is any height m polynomial with integer coefficients, then either $p(q) = 0$ or $p(q)$ is bounded away from zero (where the bound is independent of $p(x)$). For completeness we include this theorem here, using the presentation of Bugeaud [8] and Garsia [18].

Theorem 1.1 (Bugeaud/Garsia). *If q is a Pisot number, and $p(x)$ is a height m polynomial with integer coefficients, then there exists a constant $c(q, m) > 0$ such that either $p(q) = 0$, or $|p(q)| \geq c(q, m)$.*

Proof. Assume $p(q) \neq 0$. Let q_2, \dots, q_n be the conjugates of q . As q is an algebraic integer, we have that:

$$|p(q)p(q_2) \cdots p(q_n)| \geq 1.$$

From this it follows that:

$$|p(q)| \geq \frac{1}{|p(q_2) \cdots p(q_n)|} \tag{1.1}$$

$$\geq \frac{1}{\prod_{i=2}^n |p(q_i)|} \tag{1.2}$$

$$\geq \frac{1}{\prod_{i=2}^n \sum_{j=0}^{\infty} m |q_i|^j} \tag{1.3}$$

$$\geq \frac{1}{\prod_{i=2}^n m \frac{1}{1-|q_i|}} \tag{1.4}$$

$$\geq \frac{\prod_{i=2}^n (1 - |q_i|)}{m^{n-1}}. \tag{1.5}$$

So on setting $c(q, m) := \frac{\prod_{i=2}^n (1 - |q_i|)}{m^{n-1}} > 0$ we are done. It is worth noting that the inequality in on line 1.3 is strict, except if the product is empty (i.e. the Pisot number is an integer).

■

1.2 Spectra

The proof for the existence of $c(q, m)$ in Theorem 1.1 is constructive, but the construction is not the best possible. In fact, it is only the best possible for the case when the Pisot number is also an integer. This raises the question of what is the best possible construction of $c(q, m)$ and how to calculate it. To determine an optimal value for $c(q, m)$ we need to perform a minimization of $p(q)$ over all height m polynomials $p(x)$ with integer coefficients, where $p(q) > 0$.

The *spectrum* of q is the set of $p(q)$ where $p(x)$ ranges over all polynomials whose coefficients are restricted to a finite set of integers. The structure of these spectra are interesting beyond simply the study of their minimal positive values.

Erdős, Joó, and Komornik [13] are among the first to study the spectra of real numbers. The spectra they study are with respect to the class of polynomials with coefficients restricted to 0 and 1. Consider, for $q \in (1, 2)$:

$$Y(q) := \{\epsilon_n q^n + \dots + \epsilon_0 : n \in \mathbb{N}, \epsilon_i \in \{0, 1\}\} = \{0, 1, q, q + 1, q^2, q^2 + 1, q^2 + q, \dots\}.$$

Order the set $Y(q)$ as $\{y_0 = 0 < y_1 < y_2 < y_3 < \dots\}$. Erdős, Joó, and Komornik are concerned with the value of $y_{k+1} - y_k$. They show that for $q > \tau$ (where τ is the *golden ratio*, the larger root of $x^2 - x - 1$), there exist infinitely many k where $y_{k+1} - y_k = 1$. As well, if $q < \tau$ and q is a Pisot number they show that $y_{k+1} - y_k \not\rightarrow 0$. It is further shown by Erdős, Joó, and Joó [16] that $y_{k+1} - y_k \leq 1$ for all k . If q is the Pisot number satisfying $x^n - x^{n-1} - \dots - 1$ then $\liminf\{y_{k+1} - y_k\} = \frac{1}{q}$. Erdős, Joó, and Joó ask: for which other q is this infimum strictly greater than 0?

There is considerable study of this infimum, and it has its own notation:

$$l(q) := \liminf\{y_{k+1} - y_k\}.$$

We define a wider class of spectra as:

$$Y^m(q) := \{\epsilon_n q^n + \cdots + \epsilon_1 q + \epsilon_0, n \in \mathbb{N}, \epsilon_i \in \{0, \dots, m\}\}.$$

Order the set $Y^m(q)$ as $\{y_0^m = 0 < y_1^m < y_2^m < y_3^m < \cdots\}$. We extend the definition of $l(q)$ in the obvious way as:

$$l^m(q) := \liminf\{y_{k+1}^m - y_k^m\}.$$

It is clear that $l(q) = l^1(q) \geq l^2(q) \geq l^3(q) \geq \cdots \geq 0$.

The first partial answer to Erdős, Joó, and Joó's question of when $l(q) > 0$ is given by Erdős, Joó and Schnitzer [15]. Erdős, Joó and Schnitzer show that when $q < \tau$ then $l^2(q) > 0$ if and only if q is a Pisot number.

The question of when $l(q) > 0$ is answered more completely by Bugeaud [8] who shows for $q \in (1, 2)$ that $l^m(q) > 0$ for all m if and only if q is a Pisot number. Bugeaud also studies a problem related to $l^m(q)$, that of $\limsup\{y_{k+1} - y_k\}$. For this we introduce the notation:

$$L(q) := \limsup\{y_{k+1} - y_k\}$$

and extended this in the obvious way to:

$$L^m(q) := \limsup\{y_{k+1}^m - y_k^m\}.$$

Clearly $L(q) = L^1(q) \geq L^2(q) \geq \cdots \geq 0$ and $L^m(q) \geq l^m(q)$. Bugeaud shows that $L(q) < 1$ for all $q < \tau$ and $L^2(q) < 1$ for all $q \geq \tau$. Also, Bugeaud shows that if q does not satisfy a polynomial of height 1, then $l(q) = 0$ by a pigeon hole argument.

A good overview of problems relating to $l(q)$ is provided in Joó and Schnitzer [25]. They list a number of problems, two of which we include here. Both of these problems, at the time this thesis was written, are still open. These problems are:

1. For $q \in (1, 2)$, is $l(q) > 0$ if and only if q is a Pisot number?
2. For $1 < q < \tau$ does $l(q) = 0$ imply $L(q) = 0$?

If we restrict our attention to $l^m(q)$, we notice that we are trying to find the minimal positive value in $Y^m(q) - Y^m(q)$. Since $Y^m(q) - Y^m(q)$ is the set of all height m polynomials evaluated at q , we are led to the definitions:

$$\Lambda(q) := \{\epsilon_0 + \epsilon_1 q^1 + \cdots + \epsilon_n q^n : \epsilon_i \in \{\pm 1, 0\}\}$$

and

$$\Lambda^m(q) := \{\epsilon_0 + \epsilon_1 q^1 + \cdots + \epsilon_n q^n : \epsilon_i \in \{\pm m, \pm(m-1), \dots, \pm 1, 0\}\}.$$

From this we can re-define $l(q)$ and $l^m(q)$ as:

$$l(q) := \inf\{|y| : y \in \Lambda(q), y \neq 0\}$$

and

$$l^m(q) := \inf\{|y| : y \in \Lambda^m(q), y \neq 0\}.$$

It is an easy exercise to show that these two definitions, $l(q)$ and $l^m(q)$, are equivalent to those given earlier. Further results by Erdős and Komornik [17] are:

1. If q is not a Pisot number and $m \geq q - q^{-1}$ then $\Lambda^m(q)$ has a finite accumulation point.
2. If q is not a Pisot number, then $l^m(q) = 0$ for all $m \geq \lceil q - q^{-1} \rceil + \lceil q - 1 \rceil$.
3. If $q \in (1, 2^{1/4}]$ and if q^2 is not the first or second Pisot number, then $l^m(q) = 0$ for all m .

Note that for q in $(1, 2)$, Erdős and Komornik's second result implies that $l^3(q) > 0$ if and only if q is a Pisot number. Erdős, Joó and Komornik [14] show that if q is a Pisot number then $l(q) \geq (1 + q)^{-1} q^{(\log(d-1) \log(1+q) \log(1-Q)) / \log(Q)} > 0$ where d is the degree of the minimal polynomial satisfied by q , and Q is the modulus of q 's largest conjugate.

Komornik, Loreti and Pedicini [26] show that if q is the Pisot number satisfying $x^3 - x^2 - 1$ then $l(q) = q^2 - 2$. For general m , and τ the golden ratio, they give a complete description for $l^m(\tau)$. If F_k is the k th *Fibonacci number* ($F_0 = 0, F_1 = 1, F_n = F_{n-1} + F_{n-2}$), and $\tau^{k-2} < m \leq \tau^{k-1}$ then $l^m(\tau) = |F_k\tau - F_{k+1}|$.

Another spectrum that is studied is the class of ± 1 polynomials evaluated at q , defined as:

$$A(q) := \{\epsilon_0 + \epsilon_1 q^1 + \cdots + \epsilon_n q^n : \epsilon_i \in \{\pm 1\}\}$$

with the minimal value $a(q)$ defined as:

$$a(q) := \inf\{|y| : y \in A(q), y \neq 0\}.$$

How and when $A(q)$ is discrete is of interest. Some formal definitions are required:

Definition 1.7 (Discrete). *A spectrum Λ is discrete if for any finite interval $[a, b]$ of the real line, $\Lambda \cap [a, b]$ has only a finite number of elements. Equivalently Λ is discrete if it has no accumulations points.*

Definition 1.8 (Uniformly discrete). *A spectrum is uniformly discrete if there exists an ϵ greater than zero such that any two distinct values in the spectrum are at least ϵ apart.*

Definition 1.9 (Non-uniformly discrete). *A spectrum is non-uniformly discrete if it is discrete, and it is not uniformly discrete.*

It is clear that $A(q) \subseteq \Lambda(q)$, hence results about $\Lambda(q)$ extend to $A(q)$. Peres and Solomyak [29] show that if q is a Pisot number then $A(q)$ is uniformly discrete. However, examples of q where $A(q)$ is discrete, but not uniformly discrete are given in Chapter 3. Peres and Solomyak also show that $A(q)$ is dense in \mathbb{R} for almost every $q \in (\sqrt{2}, 2)$. Furthermore, if $q \in (1, \sqrt{2})$ and q^2 is not the root of a height 1 polynomial, then $A(q)$ is dense.

1.3 Generalizations of Spectra

In light of the large variety of spectra that we study, we make the following general definitions:

Definition 1.10 ($\Lambda^S(q)$). *Let S is a finite set of integers and q is a real number. Define the spectrum of q with respect to S as:*

$$\Lambda^S(q) := \{\epsilon_0 + \epsilon_1 q^1 + \cdots + \epsilon_n q^n : \epsilon_i \in S\}.$$

We notice that using this definition we have that:

1. $\Lambda^m(q) = \Lambda^{\{-m, -m+1, \dots, m-1, m\}}(q)$,
2. $\Lambda(q) = \Lambda^{\{-1, 0, 1\}}(q)$,
3. $A(q) = \Lambda^{\{\pm 1\}}(q)$,
4. $Y^m(q) = \Lambda^{\{0, 1, \dots, m\}}(q)$,
5. $Y(q) = \Lambda^{\{0, 1\}}(q)$.

We further make the general definition:

Definition 1.11 ($l^S(q)$). *Define:*

$$l^S(q) := \inf\{|y| : y \in \Lambda^S(q), y \neq 0\}.$$

Similarly, we note that:

1. $l^m(q) = l^{\{-m, -m+1, \dots, m-1, m\}}(q)$,
2. $l(q) = l^{\{-1, 0, 1\}}(q)$,
3. $a(q) = l^{\{\pm 1\}}(q)$.

Chapter 2

An Algorithm for Computing Spectra

... mathematicians stay away from actual, specific numbers as much as possible. We like to talk about numbers without actually exposing ourselves to them - that's what computers are for.

Cryptonomicon – Neal Stephenson

2.1 Background to the Algorithm

Recall from Chapter 1 that we are interested in determining $\Lambda^S(q)$ for various sets $S \subset \mathbb{Z}$. Determining $\Lambda^S(q)$ is useful for calculating $l(q)$, $a(q)$ or general $l^S(q)$. In the case of $l(q)$ we would take S to be $\{-1, 0, 1\}$ and determine the smallest positive value in $\Lambda^S(q)$. An algorithm to determine $\Lambda(q) \cap \left[\frac{-1}{q-1}, \frac{1}{q-1} \right]$ is given by Ka-Sing Lau in [27]. In Lau's paper the values of the size of the spectra in $\left[\frac{-1}{q-1}, \frac{1}{q-1} \right]$ are determined for the Pisot numbers satisfying the polynomials $x^3 - x^2 - x - 1$, $x^3 - 2x^2 + x - 1$, $x^2 - x - 1$, $x^3 - x^2 - 1$, $x^4 - x^3 - 1$ and $x^3 - x - 1$. This algorithm is generalized in Section 2.2. Section 2.3 discusses the implementation of this algorithm in C++. Lastly, Section 2.4 gives an estimate of the running time of this algorithm.

2.2 The Algorithm

In this section an algorithm is given to find the spectrum of a real number in a particular range. That is, this algorithm determines $\Lambda^S(q) \cap [a, b]$ for a real number q and a finite range $[a, b]$. If q is a Pisot number then this algorithm terminates. Moreover, if this algorithm terminates then the spectrum is discrete over the entire real line. In the case of $\Lambda(q)$, this algorithm is similar to that given by Ka-Sing Lau [27]. First, a lemma:

Lemma 2.1. *Let S be a finite set of integers. Let $p(x)$ be a degree n polynomial with coefficients in S . Let $s_l := \min\{S\}$, $s_u := \max\{S\}$, and let $q > 1$. Denote $\alpha_u := \frac{s_l}{1-q}$ and $\alpha_l := \frac{s_u}{1-q}$. If $p(q) \notin [\alpha_l, \alpha_u]$ then $q \times p(q) + s \notin [\alpha_l, \alpha_u]$ for all $s \in S$.*

Proof. Assume that $p > \alpha_u$. Then:

$$\begin{aligned}
 qp + s &\geq qp + s_l \\
 &> q\alpha_u + s_l \\
 &\geq \frac{-s_l q}{q-1} + \frac{s_l q - s_l}{q-1} \\
 &\geq \frac{-s_l}{q-1} \\
 &\geq \alpha_u.
 \end{aligned}$$

A similar result shows that if $p < \alpha_l$ the $qp + s < \alpha_l$ for all $s \in S$. ■

Consider a calculation of $\Lambda^S(q) \cap [\alpha_l, \alpha_u]$. From Lemma 2.1 it follows that, if $p(q) \notin [\alpha_l, \alpha_u]$, then the polynomial $q \times p(q) + s$ need not be considered, as it cannot contribute to the spectrum in this range. Furthermore, if $\alpha_l^* \leq \alpha_l$ and $\alpha_u^* \geq \alpha_u$, then the same result holds that $p(q) \notin [\alpha_l^*, \alpha_u^*]$ implies $q \times p(q) + s \notin [\alpha_l^*, \alpha_u^*]$ for all $s \in S$.

The next lemma ensures that if q is a Pisot number, then an exhaustive search for all elements in a finite range for a given spectrum terminates. The algorithm will

keep running until it finds no more new elements, and if there are a finite number of elements in a given range, then this condition will eventually be satisfied.

Lemma 2.2. *If q is a Pisot number, S a finite set of integers and $[a, b]$ a finite interval, then $|\Lambda^S(q) \cap [a, b]|$ is finite.*

Proof. Let $r = \max\{|s| : s \in S\}$. Let $y_1, y_2 \in \Lambda^S(q), y_1 \neq y_2$. Then $y_1 - y_2 \in \Lambda^{2r}(q)$ hence $|y_1 - y_2| \geq l^{2r}(q) > 0$, where the last inequality comes from [8, 13, 17]. Thus $|\Lambda^S(q) \cap [a, b]| \leq \frac{b-a}{l^{2r}(q)} + 1 < \infty$.

■

In Figure 2.1 (page 14) an exhaustive search to determine $\Lambda^S(q) \cap [\alpha_l, \alpha_u]$ is given. If we wish to find the spectra in a range $[a, b]$ other than $[\alpha_l, \alpha_u]$, we use in the algorithm a lower bound of $\min\{a, \alpha_l\}$ and an upper bound of $\max\{b, \alpha_u\}$, and then restrict our attention to $[a, b]$.

```

Spec(S, q)
  alpha[u] := -min(s:s in S)/(q-1);
  alpha[l] := -max(s:s in S)/(q-1);
  L[0] := S;
  d := 0;
  repeat
    L[d+1] := L[d];
    for p in (L[d] minus L[d-1]), s in S do
      if q * p + s in [alpha[l], alpha[u]] then
        L[d+1] = L[d+1] union {q * p + s}
      end if
    end do
    d := d + 1;
  until L[d-1] = L[d];
  RETURN(L[d]);
end;

```

Figure 2.1: Algorithm for finding the spectrum of a Pisot number q

Example 2. Let us compute $\Lambda(q) \cap \left[\frac{-1}{q-1}, \frac{1}{q-1} \right]$ for the Pisot number q that satisfies $x^3 - 2x^2 + x - 1$. We have $q = 1.754877666$, $\alpha_l = -1.3247179$ and $\alpha_u = 1.3247179$. Printed below are the various values of L_d , where L_d is defined in Figure 2.1.

$$L_0 = [-1, 0, 1],$$

$$L_1 = [-1, -0.7548776, 0, 0.7548776, 1],$$

$$L_2 = [-1.3247179, -1, -0.7548776, -0.3247179, 0, 0.3247179, \\ 0.7548776, 1, 1.3247179],$$

$$L_3 = [-1.3247179, -1, -0.7548776, -0.5698402, -0.4301597,$$

$$\begin{aligned}
& -0.3247179, 0, 0.3247179, 0.4301597, 0.5698402, 0.7548776, \\
& 1, 1.3247179], \\
L_4 = & [-1.3247179, -1, -0.7548776, -0.5698402, -0.4301597, \\
& -0.3247179, -0.2451223, 0, 0.2451223, 0.3247179, 0.4301597, \\
& 0.5698402, 0.7548776, 1, 1.3247179], \\
L_5 = & [-1.3247179, -1, -0.7548776, -0.5698402, -0.4301597, \\
& -0.3247179, -0.2451223, 0, 0.2451223, 0.3247179, 0.4301597, \\
& 0.5698402, 0.7548776, 1, 1.3247179], \\
= & L_4.
\end{aligned}$$

Since $L_5 = L_4$ we see that the algorithm has terminated. From this we find the minimal element in the spectrum, $l(q)$, is 0.245122334.

We know the spectrum of a Pisot number q is uniformly discrete [17, 18], and hence $|\Lambda^S(q) \cap [\alpha_l, \alpha_u]| < \infty$. The natural question is, do we know anything about the converse?

Theorem 2.1. *Let $q > 1$. Let α_l and α_u be defined as in Lemma 2.1. If $|\Lambda^S(q) \cap [\alpha_l, \alpha_u]| < \infty$ then $\Lambda^S(q)$ is discrete.*

Notice that in Theorem 2.1 q is not assumed to be a Pisot number. It is worth noting that Theorem 2.1 only proves that $\Lambda^S(q)$ is discrete, and does not prove that $\Lambda^S(q)$ is uniformly discrete. In fact there exist examples of spectra, described later in Section 3.2, that satisfy Theorem 2.1 and are provably non-uniformly discrete.

Proof (Theorem 2.1). Let s_l, s_u, α_l and α_u be defined as in Lemma 2.1. Define $\lambda_1 = \Lambda^S(q) \cap [\alpha_l, \alpha_u]$. Define $\alpha_{u,1} = \min\{q\beta + s > \alpha_u : \beta \in \lambda_1, s \in S\}$, and define $\alpha_{l,1}$ similarly to be the maximal element in the spectrum less than α_l . From this, define $\alpha_{u,n} = q \times \alpha_{u,n-1} + s_l$ and $\alpha_{l,n} = q \times \alpha_{l,n-1} + s_u$. Next, define $\lambda_n = [\alpha_{l,n-1}, \alpha_{u,n-1}] \cap \Lambda^S(q)$.

Clearly, $\alpha_{u,n} \rightarrow \infty$ and $\alpha_{l,n} \rightarrow -\infty$ as $n \rightarrow \infty$. By assumption λ_1 has only a finite number of elements. Noticing that $\lambda_n = \lambda_{n-1} \cup (\{q\beta + s : \beta \in \lambda_{n-1}, s \in$

$S\} \cap [\alpha_{l,n-1}, \alpha_{u,n-1}]$ gives by induction that λ_n has only a finite number of elements. Thus, $\Lambda^S(q)$ is discrete, as required.

■

2.3 Implementation

The algorithm described in Section 2.2 has been implemented in C++ [5]. This implementation is found at [23]. A complete list of the C++ code is also found in Appendix A. This section summarizes some of the techniques used to improve the efficiency of this code.

When calculating $\Lambda^S(q)$ a list of all polynomials examined must be kept. As the degrees of these polynomials can be quite large, the list of polynomials examined can take up a large amount of memory. To reduce space requirements, this C++ implementation stores the remainders of the polynomials when they are divided by the minimal polynomial of q . This is advantageous because the degrees of the remainders are bounded above. Furthermore, through experimentation it is noted that the heights of these polynomials being stored do not grow larger than a “short int” in C++. (The software tests whether the coefficients are small enough, and raises an error if they grow larger than a “short int”.)

The storing of the remainders of the polynomials has a second advantage, if $p_1(q) = p_2(q)$, where p_1 and p_2 are polynomials, then the remainders of p_1 and p_2 , when divided by the minimal polynomial of q , are equal. Thus duplication within the spectrum is easily recognized.

Keeping a list of all polynomials examined is done by storing these polynomials in a red-black tree with a lexicographical order on the coefficients. The lexicographical order is used, as it allows for easy comparisons to previously examined polynomials. Any height regulating tree with any order would give similar results for time comparisons of duplication recognition. For more information on red-black trees see [12].

The next observation to be made is that if S is symmetric, (i.e. $s \in S$ implies that $-s \in S$), then $\Lambda^S(q)$ is symmetric. Utilizing this symmetry eliminates half of the calculations needed.

There exists further refinements to this algorithm, described in [5], that allow the calculation of the spectra to be partitioned into multiple sub-calculations.

2.4 Running Time of the Algorithm

To analyze the running time of this algorithm, we need to make some estimates about the size of $\Lambda^S(q) \cap [\alpha_l, \alpha_u]$. We assume that q is a Pisot number, as this is the only known case when the algorithm must terminate for all finite subsets S of the integers. If we set $m = \max(|s| : s \in S)$ then $\Lambda^S(q) \subseteq \Lambda^m(q)$. Hence the running time for $\Lambda^S(q)$ is less than or equal to the running time for $\Lambda^m(q)$. So without loss of generality, we assume that we are determining the running time for spectra of the form $\Lambda^m(q)$.

Let q be a Pisot number, with conjugates $q = q_1, q_2, \dots, q_n$. We use q and q_1 interchangeably throughout this section. Let the minimal polynomial of q be written as $x^n - b_{n-1}x^{n-1} - \dots - b_0$. Define $\mathcal{P}(x) := \{P(x)\}$ to be the unique set of polynomials of degree $\leq n - 1$ such that $\mathcal{P}(q) = \Lambda(q) \cap \left[\frac{-1}{q-1}, \frac{1}{q-1}\right]$. This set exists because if $p_1(q) = p_2(q)$ then upon division by the minimal polynomial of q , the remainders of $p_1(x)$ and $p_2(x)$ are equal, and of degree at most $n - 1$.

Example 3. Consider the Pisot number q satisfying $x^3 - 2x^2 + x - 1$. In Example 2 (page 14), we found that:

$$\begin{aligned} \Lambda(q) \cap \left[\frac{-1}{q-1}, \frac{1}{q-1} \right] &= [-1.3247179, -1, -0.7548776, -0.5698402, \\ &\quad -0.4301597, -0.3247179, -0.2451223, 0, \\ &\quad 0.2451223, 0.3247179, 0.4301597, 0.5698402, \\ &\quad 0.7548776, 1, 1.3247179]. \end{aligned}$$

This gives:

$$\begin{aligned} \Lambda(q) \cap \left[\frac{-1}{q-1}, \frac{1}{q-1} \right] = & [-q^2 + q, -1, -q + 1, -q^3 + q^2 + q, \\ & q^3 - q^2 - q - 1, -q^2 + q + 1, \\ & -q^4 + q^3 + q^2 + q - 1, 0, q^4 - q^3 - q^2 - q + 1, \\ & q^2 - q - 1, -q^3 + q^2 + q + 1, \\ & q^3 - q^2 - q, q - 1, 1, q^2 - q]. \end{aligned}$$

Thus, we write:

$$\begin{aligned} \mathcal{P}(q) = & [-q^2 + q, -1, -q + 1, -q^2 + 2q - 1, q^2 - 2q, -q^2 + q + 1, \\ & q - 2, 0, -q + 2, q^2 - q - 1, -q^2 + 2q, q^2 - 2q + 1, q - 1, \\ & 1, q^2 - q]. \end{aligned}$$

An upper bound for the number of elements in $\mathcal{P}(x)$ is needed to give an upper bound to the running time of this algorithm.

Define $a_{i,j}$ such that:

$$x^k \equiv a_{k,n-1}x^{n-1} + \cdots + a_{k,0} \pmod{p(x)}.$$

Now consider the polynomial $a_{k,n-1}x^{n-1} + \cdots + a_{k,0}$ as a point in n -space denoted by $\vec{a} := (a_{k,n-1}, \dots, a_{k,0})$. The $a_{k,j}$ follow a recurrence of the form:

$$a_{k,j} = b_{n-1}a_{k-1,j} + \cdots + b_0a_{k-n,j}$$

and thus we write each $a_{k,j}$ as:

$$a_{k,j} = \alpha_{j,1}q_1^k + \cdots + \alpha_{j,n}q_n^k.$$

Example 4. Let us again consider the example of the Pisot number q satisfying $x^3 - 2x^2 + x - 1$ (Example 2, page 14). We see that:

$$\begin{aligned}
x^0 &\equiv 0x^2 + 0x + 1 \pmod{x^3 - 2x^2 + x - 1}, \\
x^1 &\equiv 0x^2 + 1x + 0 \pmod{x^3 - 2x^2 + x - 1}, \\
x^2 &\equiv 1x^2 + 0x + 0 \pmod{x^3 - 2x^2 + x - 1}, \\
x^3 &\equiv 2x^2 - 1x + 1 \pmod{x^3 - 2x^2 + x - 1}, \\
&\vdots \\
x^k &\equiv a_{k,2}x^2 + a_{k,1}x + a_{k,0} \pmod{x^3 - 2x^2 + x - 1},
\end{aligned}$$

where:

$$\begin{aligned}
a_{k,2} &= 2a_{k-1,2} - a_{k-2,2} + a_{k-3,2}, \\
a_{k,1} &= 2a_{k-1,1} - a_{k-2,1} + a_{k-3,1}, \\
a_{k,0} &= 2a_{k-1,0} - a_{k-2,0} + a_{k-3,0},
\end{aligned}$$

and has initial values:

$$\begin{aligned}
a_{0,2} &= 0, \quad a_{1,2} = 0, \quad a_{2,2} = 1, \\
a_{0,1} &= 0, \quad a_{1,1} = 1, \quad a_{2,1} = 0, \\
a_{0,0} &= 1, \quad a_{1,0} = 0, \quad a_{2,0} = 0.
\end{aligned}$$

Thus, we have:

$$\begin{aligned}
a_{k,2} &= \frac{q_1^k}{(q_1 - q_3)(q_1 - q_2)} - \frac{q_2^k}{(-q_3 + q_2)(q_1 - q_2)} + \frac{q_3^k}{-q_1 q_3 + q_2 q_1 - q_2 q_3 + q_3^2}, \\
a_{k,1} &= -\frac{(q_2 + q_3) q_1^k}{(q_1 - q_3)(q_1 - q_2)} + \frac{(q_1 + q_3) q_2^k}{(-q_3 + q_2)(q_1 - q_2)} - \frac{(q_1 + q_2) q_3^k}{-q_1 q_3 + q_2 q_1 - q_2 q_3 + q_3^2}, \\
a_{k,0} &= \frac{q_2 q_3 q_1^k}{(q_1 - q_3)(q_1 - q_2)} - \frac{q_1 q_3 q_2^k}{(-q_3 + q_2)(q_1 - q_2)} + \frac{q_2 q_1 q_3^k}{-q_1 q_3 + q_2 q_1 - q_2 q_3 + q_3^2}.
\end{aligned}$$

We see that $\vec{a} = (a_{k,n-1}, \dots, a_{k,0})$ is $(\alpha_{n-1,1}q_1^k + \dots + \alpha_{n-1,n}q_n^k, \dots, \alpha_{0,1}q_1^k + \dots + \alpha_{0,n}q_n^k)$. As q_2, \dots, q_n are of modulus less than 1, we see that for large k that $\vec{a} \approx$

$q_1^k(\alpha_{n-1,1}, \dots, \alpha_{0,1})$. So x^k is roughly on the line $q_1^k(\alpha_{n-1,1}, \dots, \alpha_{0,1})$. The maximum deviation of \vec{a} from this line $\vec{\alpha} := (\alpha_{n-1,1}, \dots, \alpha_{0,1})$ is:

$$\sum_{j=0}^{n-1} \sum_{i=2}^n |\alpha_{j,i}| |q_i|^k.$$

So the maximum deviation for any term in $\Lambda(q)$ from this line is:

$$R := \sum_{k=0}^{\infty} \sum_{j=0}^{n-1} \sum_{i=2}^n |\alpha_{j,i}| |q_i|^k, \quad (2.1)$$

$$= \sum_{j=0}^{n-1} \sum_{i=2}^n |\alpha_{j,i}| \frac{1}{1 - |q_i|}. \quad (2.2)$$

Notice that any polynomial $p(x) \in \mathcal{P}(x)$ for $\Lambda^m(q)$, is such that $p(q)$ is in the range $\left[\frac{-m}{q-1}, \frac{m}{q-1}\right]$. Further, any polynomial $p(x) \in \mathcal{P}(x)$ for $\Lambda^m(q)$, when considered as a point in n -space, is within a radius of mR from $\vec{\alpha}$. Thus it is sufficient to determine the number of integer points a radius mR from $\vec{\alpha}$ in the range $\left[\frac{-m}{q-1}, \frac{m}{q-1}\right]$. This gives the number of terms in $\Lambda^m(q) \cap \left[\frac{-m}{q-1}, \frac{m}{q-1}\right]$ being bounded by:

$$\frac{2m^n}{q-1} R^{n-1}.$$

An insertion into a set requires $\mathcal{O}(\log(\frac{2m^n}{q-1} R^{n-1}))$ time. For each element in Λ we must examine $qp + s$ for $s \in \{-m, \dots, m\}$, (hence $2m + 1$ possible neighbours). So the running time of the algorithm is of order

$$\mathcal{O}\left(\frac{2m^n}{q-1} R^{n-1} \log\left(\frac{2m^n}{q-1} R^{n-1}\right) (2m + 1)\right) = \mathcal{O}\left(\frac{m^{n+1}}{q-1} R^{n-1} n \log(mR)\right).$$

Example 5. Let us again consider the example of the Pisot number q satisfying $x^3 - 2x^2 + x - 1$ (Example 2, page 14). We have:

$$\begin{aligned} \alpha_{0,2} &= .4956083622, & \alpha_{0,3} &= .4956083622, \\ \alpha_{1,2} &= .7556553486, & \alpha_{1,3} &= .7556553486, \\ \alpha_{2,2} &= .3741236838, & \alpha_{2,3} &= .3741236838. \end{aligned}$$

This gives a value of $R = 13.26184660$. Thus in $\Lambda(q) \cap \left[\frac{-1}{q-1}, \frac{1}{q-1} \right]$ the number of elements is bounded by:

$$\frac{2}{q_1 - 1} R^{n-1} = 465.$$

This is an overestimate, given that enumerating this set gives 15 elements.

We have just proved the following theorem:

Theorem 2.2. *Let q be a degree n Pisot number. The algorithm to determine $\Lambda^m(q)$ runs in $\mathcal{O}(m^n \log m)$ time.*

In practice memory is the constraint, not time. Any calculation that could be completed in less than 2 gigabytes of RAM took less than 90 minutes to complete.

Chapter 3

Explorations of Spectra

The device also functioned as an ordinary calculator, but only to a limited degree. It could handle any calculation which returned an answer of anything up to “4”.

“1 + 1” it could manage (“2”), and “1 + 2” (“3”) and “2+2” (“4”) or “tan 74” (“3.4874145”) but anything above “4” it represented as “A Suffusion of Yellow”. Dirk was not certain if this was a programming error or an insight beyond his ability to fathom...

The Long Dark Tea-Time of the Soul – Douglas Adams

3.1 Spectra of $\Lambda(q)$ and $A(q)$ for Pisot Numbers q

Specific values of $l^m(q)$ are calculated for some Pisot numbers q . If q is the Pisot number satisfying $x^3 - x^2 - 1$, then $l(q) = q^2 - 2$ [26]. If q is the Pisot number satisfying $x^n - x^{n-1} - \dots - 1$ then $l(q) = q^{n-1} - q^{n-2} - \dots - 1 = \frac{1}{q}$ [16, 26]. If τ is the golden ratio, then $l^2(\tau) = \tau^3 - 2\tau^2 + 2\tau - 2 = 2\tau - 3$ (this corrects a misprint in [8], which used the notation u^2 for $l^2(\tau)$).

The algorithm in Figure 2.1 (page 14) is used to calculate $l(q)$ for all Pisot numbers between 1 and 2, up to and including degree 9 (and most Pisot numbers of degree

10) and $a(q)$ for Pisot numbers up to and including degree 10. The methods of David Boyd are used [7] to determine the Pisot numbers up to and including degree 10. There are 232 Pisot numbers of degree less than or equal to 10 between 1 and 2 and thus we do not include all of the results here. All of the results concerning $l(q)$ and $a(q)$, as well as the code used to determine these results, are found at [23].

In the calculations of $a(q)$ and $l(q)$, there are only a few cases where $a(q) = l(q)$. Some of these Pisot numbers are enumerated in Table 3.1. The last column of this table does not give the ± 1 polynomial for $a(q)$, this column instead gives the polynomial's canonical degree $n - 1$ representation, as described in Section 2.4 for $\mathcal{P}(x)$ (see page 17). This shall be the practice with all tables within this thesis. We know from [16, 26] that if q is the Pisot number satisfying $x^n - x^{n-1} - \dots - 1$ then $l(q) = q^{n-1} - q^{n-2} - \dots - 1$. Thus for Pisot numbers q satisfying $x^n - x^{n-1} - \dots - 1$ it is clear that $a(q) = l(q)$. Hence the set of Pisot numbers q , where $l(q) = a(q)$, is infinite.

A question of interest is for which Pisot numbers q does q satisfy a ± 1 polynomial. This is equivalent to asking for which Pisot numbers q is $0 \in A(q)$. Computationally most Pisot numbers q are such that $0 \in A(q)$. Table 3.2 summarizes all failures found. Thus, all of the Pisot numbers given in Table 3.2 provably do not satisfy a ± 1 polynomial. The first known failure is a Pisot number with minimal polynomial of degree 6. It is interesting to note that all of the failures found are such that q is greater than 1.95. It would be interesting to know if this must always be the case.

Table 3.3 lists large calculations of $l(q)$, and Table 3.4 lists large calculations of $a(q)$. Due to potential floating point error, a range slightly larger than $[\alpha_l, \alpha_u]$ was chosen. Hence an exact size of the spectrum in the range $[\alpha_l, \alpha_u]$ cannot be given, but only an approximation. Any calculation of $l(q)$ or $a(q)$ where $\Lambda(q) \cap \left[\frac{-1}{q-1}, \frac{1}{q-1} \right]$ or $A(q) \cap \left[\frac{-1}{q-1}, \frac{1}{q-1} \right]$ has an approximate spectrum size over 20 million is listed. Due to the memory requirements of this program, any spectra of approximate size over 48 million could not be computed. The calculation is successful for all $a(q)$ tested, and for $l(q)$, all but 16 of the 232 Pisot numbers tested. The timings listed in these tables are for a MIPS R10000 Processor Chip Revision: 3.4 (Main memory size: 4096

| Minimal polynomial | Pisot number q | $l(q)$ | Exact representation of $\pm l(q)$ |
|-----------------------------|------------------|-----------|---|
| $x^2 - x - 1$ | 1.618034 | 0.618034 | $q - 1$ |
| $x^3 - 2x^2 + x - 1$ | 1.754878 | 0.245122 | $q - 2$ |
| $x^3 - x^2 - x - 1$ | 1.839286 | 0.543689 | $q^2 - q - 1$ |
| $x^4 - x^3 - 1$ | 1.380278 | 0.008993 | $q^3 - 4q^2 + 5$ |
| $x^4 - 2x^3 + x - 1$ | 1.866760 | 0.13324 | $q - 2$ |
| $x^4 - x^3 - 2x^2 + 1$ | 1.905166 | 0.068706 | $q^3 - 3q^2 + q + 2$ |
| $x^4 - x^3 - x^2 - x - 1$ | 1.927562 | 0.518790 | $q^3 - q^2 - q - 1$ |
| $x^5 - x^4 - x^3 + x^2 - 1$ | 1.443269 | 0.002292 | $4q^2 - 3q - 4$ |
| $x^5 - x^3 - x^2 - x - 1$ | 1.534158 | 0.002155 | $2q^4 - 3q^3 + q^2 - 3q + 2$ |
| $x^5 - x^4 - x^2 - 1$ | 1.570147 | 0.006992 | $q^4 - 2q^2 - 2q + 2$ |
| $x^5 - 2x^4 + x^3 - x^2$ | 1.673649 | 0.009705 | $q^4 - q^3 - q^2 - 2q + 3$ |
| $+x - 1$ | | | |
| $x^5 - x^4 - x^3 - 1$ | 1.704903 | 0.030844 | $2q^3 - 3q^3 - 2$ |
| $x^5 - x^4 - x^3 - x^2$ | 1.965948 | 0.508660 | $q^4 - q^3 - q^2 - q - 1$ |
| $-x - 1$ | | | |
| $x^6 - x^5 - x^4 + x^2 - 1$ | 1.501595 | 0.0003491 | $q^5 + 2q^4 - 4q^3 - 3q^2$ $+3q - 2$ |
| $x^6 - 2x^5 + x - 1$ | 1.967168 | 0.032831 | $q - 2$ |
| $x^6 - x^5 - x^4 - x^3$ | 1.983583 | 0.504138 | $q^5 - q^4 - q^3 - q^2$ $-q - 1$ |
| $-x^2 - x - 1$ | | | |
| $x^7 - x^5 - x^4 - x^3$ | 1.590005 | 0.0001137 | $4q^6 - 5q^5 - q^4 - q^3$ $+q - 6$ |
| $-x^2 - x - 1$ | | | |
| $x^7 - x^6 - x^4 - x^2 - 1$ | 1.601347 | 0.0004642 | $2q^5 - q^4 - 3q^3 - 1q^2$ $-q + 2$ |
| $x^7 - 2x^6 + x^5 - x^4$ | 1.640728 | 0.0003030 | $2q^6 - 2q^5 - 2q^4$ $-2q^2 + q + 3$ |
| $+x^3 - x^2 + x - 1$ | | | |
| $x^7 - 2x^6 + x^5 - 2x^4$ | 1.790223 | 0.0006021 | $q^6 - 3q^5 + 5q^4 - 4q^3$ $-4q + 1$ |
| $+2x^3 - x^2 + x - 1$ | | | |
| $x^7 - 2x^6 + x - 1$ | 1.983861 | 0.016138 | $q - 2$ |
| $x^7 - x^6 - x^5 - x^4$ | 1.991964 | 0.502017 | $q^6 - q^5 - q^4 - q^3$ $-q^2 - q - 1$ |
| $-x^3 - x^2 - x - 1$ | | | |

Table 3.1: Pisot numbers where $l(q) = a(q)$

| Minimal polynomial | Pisot number |
|---|--------------|
| $x^6 - x^5 - 2x^4 + x^2 - x - 1$ | 1.979476326 |
| $x^6 - 3x^5 + 3x^4 - 2x^3 + x - 1$ | 1.955451068 |
| $x^8 - x^7 - x^6 - x^5 - 2x^4 + 1$ | 1.995777793 |
| $x^9 - x^8 - x^7 - 2x^6 + x^3 - x^2 - x - 1$ | 1.997784254 |
| $x^9 - 2x^8 + x^5 - 2x^4 + x - 1$ | 1.996283920 |
| $x^9 - x^8 - x^7 - x^6 - x^5 - x^4 - x^3 - x^2 - 1$ | 1.994016415 |
| $x^9 - 2x^7 - 3x^6 - 2x^5 + x^3 - x - 1$ | 1.992483962 |
| $x^9 - x^8 - x^7 - x^6 - x^5 - x^4 - x^3 - x - 1$ | 1.989944545 |
| $x^9 - x^8 - x^7 - x^6 - x^5 - x^4 + 1$ | 1.963515789 |
| $x^{10} - x^9 - x^8 - x^7 - x^6 - 2x^5 + 1$ | 1.998987762 |
| $x^{10} - x^9 - 2x^8 + x^6 - x^5 - 2x^4 + x^2 - x - 1$ | 1.998772685 |
| $x^{10} - 2x^9 + x^7 - 2x^6 + x^4 - 2x^3 + x - 1$ | 1.998277927 |
| $x^{10} - 2x^9 + x^5 - x^4 - x + 1$ | 1.969456013 |
| $x^{10} - x^9 - 2x^8 + x^6 - x^5 - x^4 + x^3 + x^2 - x - 1$ | 1.966884957 |
| $x^{10} - x^9 - x^8 - x^7 - x^6 - x^5 + 1$ | 1.964715641 |
| $x^{10} - 2x^8 - 3x^7 - x^6 - x^3 + x + 1$ | 1.954062236 |

Table 3.2: Pisot numbers that do not satisfy ± 1 polynomials

Mbytes). The code to perform these calculations is written in C++, using the GNU compiler. (See Section 2.3 for a discussion of the code, Appendix A for a listing of the code, and [23] for a copy of the code.) Precise values of $l(q)$ and $a(q)$ in terms of their polynomial evaluations at q can be found at [23].

3.2 $A(q)$ for Non-Pisot Numbers q

Peres and Solomyak ask in [29], for which $q \in (1, 2)$ is $A(q)$ dense. In [29] they say that the only known examples of q with the property that $A(q)$ is not dense are the Pisot numbers. This section finds examples that have this property, that are not Pisot numbers.

A search of 1868 non-Pisot numbers q is done, to find examples where $A(q)$ is discrete. An additional test is made of 578 different Salem numbers. To explain how these non-Pisot numbers are chosen consider the following theorem:

Theorem 3.1. *Let $q \in (1, 2)$. If q does not satisfy a polynomial of the form $\epsilon_n x^n + \dots + \epsilon_m x^m + \beta_{m-1} x^{m-1} + \dots + \beta_0$ where $\epsilon_i \in \{\pm 1\}$ and $\beta_i \in \{\pm 2, 0\}$, then $A(q)$ is not discrete.*

Proof. Take $P_0 = 1$. If $q \times P_{n-1} > 1$ then take $P_n = q \times P_{n-1} - 1$, and if $q \times P_{n-1} < 1$ then take $P_n = 1 - q \times P_{n-1}$. Clearly $P_i \in A(q)$ for all i and $0 \leq P_i \leq 1$.

If this sequence of P_i repeats, then q satisfies the difference of two ± 1 polynomials, which is of the form described above. If this sequence does not repeat, then the sequence of P_i is an infinite non-repeating sequence in $[0, 1]$, and thus $A(q)$ is not discrete. ■

Corollary 3.1. *Let $q \in (1, 2)$. If q does not satisfy a height 2 polynomial, then $A(q)$ is not discrete.*

Lemma 3.1. *If $A(q)$ is discrete, then $A(q^n)$ is discrete for all n .*

| Minimal polynomial | Pisot number q | $l(q)$ | Approximate size of spectrum in $[\alpha_l, \alpha_u]$ | CPU secs |
|--|------------------|-------------|--|----------|
| $x^{10} - x^9 - x^8 - x^7 + x^6 - x^3 + 1$ | 1.742975573 | 1.18668e-07 | 26973910 | 39m50s |
| $x^{10} - x^9 - x^7 - x^6 - x^5 - x^4 - x^3 - x^2 - 1$ | 1.746541923 | 7.04603e-08 | 41498130 | 58m41s |
| $x^{10} - x^9 - x^8 - x^7 + x^5 - x^3 + 1$ | 1.795572823 | 3.5123e-08 | 43357472 | 1h1m7s |
| $x^{10} - x^9 - x^8 - x^7 - x^3 + 1$ | 1.852234868 | 8.17922e-08 | 25981420 | 34m38s |
| $x^{10} - x^9 - x^8 - x^7 - x^5 + x^4 + 1$ | 1.860952864 | 3.80874e-07 | 24944436 | 35m22s |
| $x^{10} - 2x^9 + x^8 - 2x^7 + x^6 + x^3 - x^2 + x - 1$ | 1.870250440 | 4.44816e-08 | 46252634 | 1h4m56s |
| $x^{10} - 2x^9 + x^7 - x^6 - x^3 + x^2 - 1$ | 1.881601063 | 2.57611e-07 | 27513576 | 35m35s |
| $x^{10} - 2x^8 - 3x^7 - x^6 + x^5 + 2x^4 + x^3 - x^2 - 2x - 1$ | 1.890027098 | 2.67873e-07 | 20923016 | 29m43s |
| $x^{10} - 2x^9 + x^8 - x^7 - x^6 - x^2 + x - 1$ | 1.903832902 | 2.22525e-07 | 22738454 | 28m42s |
| $x^{10} - x^9 - x^8 - x^7 - x^5 - x^4 - x^2 - x - 1$ | 1.921407084 | 3.12296e-08 | 41511868 | 57m5s |
| $x^{10} - 2x^9 + x^8 - 2x^7 + x^6 - x^5 - x^2 - 1$ | 1.957362809 | 2.22214e-07 | 22336604 | 29m7s |
| $x^{10} - 2x^9 + x^7 - 2x^6 + x^4 - 2x^3 + x - 1$ | 1.998277927 | 2.447e-08 | 46943484 | 1h3m54s |

Table 3.3: Successful calculations of $l(q)$ with a spectrum over 20 million

| Minimal polynomial | Pisot number q | $a(q)$ | Approximate size of spectrum in $[\alpha_l, \alpha_u]$ | CPU secs |
|---|------------------|-------------|--|----------|
| $x^{10} - x^9 - x^8 + x^2 - 1$ | 1.601755862 | 1.59445e-07 | 33921896 | 30m38s |
| $x^{10} - x^9 - x^8 - x^2 + 1$ | 1.632690733 | 1.03354e-07 | 21835702 | 17m30s |
| $x^{10} - 2x^9 + x^8 - x^7 + x^3 - x^2 + x - 1$ | 1.735143707 | 8.28149e-08 | 32342934 | 29m18s |

Table 3.4: Successful calculations of $a(q)$ with a spectrum over 20 million

Proof. Let $\alpha = \sum_{i=1}^m \pm q^{ni}$ be an element in $A(q^n)$. Then $(q^{n-1} + q^{n-2} + \dots + q + 1)\alpha$ is in $A(q)$. Thus $\frac{q^n - 1}{q - 1}A(q^n) \subset A(q)$. Thus $A(q^n)$ is discrete. ■

Lemma 3.1 is of theoretical interest, but is not of much practical use, because other than the two Pisot numbers, no $q < \sqrt{2}$, has been found where $A(q)$ is discrete. With the limits imposed by Theorem 3.1, the search is restricted to:

1. All irreducible polynomials dividing a height 1 polynomial of degree ≤ 7 .
2. All polynomials dividing any ± 1 polynomial up to degree 10.
3. All polynomials dividing a polynomial of the form $\epsilon_n x^n + \epsilon_{n-1} x^{n-1} + \dots + \epsilon_m x^m + \beta_{m-1} x^{m-1} + \dots + \beta_0$ where $\epsilon_i \in \{\pm 1\}$ and $\beta_i \in \{\pm 2, 0\}$ up to degree 7.
4. A list of 578 Salem numbers, between 1 and 2, of degree less than or equal to 14.

Some observations made on the basis of this search are:

1. All examples found of q where $A(q)$ is discrete are *Perron numbers* (all conjugates are of modulus less than q).

2. There are 125 examples found of non-Pisot numbers q with discrete spectra $A(q)$.
3. There are 12 Salem numbers found, with discrete spectra. Some of these are listed in Table 3.6.
4. The only non-Pisot numbers q found whose Mahler measure is less than 2 and where $A(q)$ is discrete are these 12 Salem numbers.
5. The smallest non-Pisot number found with discrete spectrum is the Salem number q satisfying $x^4 - x^3 - x^2 - x + 1$ (approximately 1.72208).
6. The largest root of $x^n - x^{n-1} - \dots - x + 1$ is a Salem number with discrete spectrum (Theorem 3.3 and Theorem 3.4), and the only Salem numbers found of degree 14 or less with discrete spectrum satisfy a polynomial of this type.
7. All q found where $A(q)$ is discrete and where q does not satisfy a height 1 polynomial, do not have zero in the spectrum.
8. The smallest degree minimal polynomial found of a non-Pisot number q such that $A(q)$ is discrete is $x^3 - 2x - 2$.

It is worth noting here the distinction between discrete spectra and uniformly discrete spectra. The examples found of non-Pisot numbers with discrete spectra $A(q)$ are “most probably” non-uniformly discrete, and some provably non-uniformly discrete.

Theorem 3.2. *If $l(q) = 0$ then $A(q)$ is not uniformly discrete, and if $A(q)$ is not uniformly discrete, then $l^2(q) = 0$.*

Proof. The first part follows by noticing that:

$$2\Lambda(q) = \Lambda^{\{\pm 2, 0\}}(q) \subseteq A(q) - A(q),$$

| Minimal polynomial | Non-Pisot number |
|---|------------------|
| $x^3 - 2x - 2$ | 1.769292354 |
| $x^4 - x^3 - 2x - 2$ | 1.873708564 |
| $x^4 - 2x^2 - 2x - 2$ | 1.899321089 |
| $x^5 - x^4 - 2x^2 - 2$ | 1.803707279 |
| $x^5 - x^4 - x^3 - 2x^2 + 2$ | 1.917514202 |
| $x^5 - x^4 - 2x^2 - 2x - 2$ | 1.942887561 |
| $x^5 - 2x^3 - 2x^2 - 2x - 2$ | 1.953501637 |
| $x^6 - 2x^4 - 2x^3 - 2$ | 1.813277575 |
| $x^6 - x^5 - x^4 - 2x^3 + 2x + 2$ | 1.859080768 |
| $x^6 - 2x^4 - 2x^3 - 2x^2 + 2$ | 1.865843123 |
| $x^6 - x^5 - x^4 - 2x^3 + 2$ | 1.961038629 |
| $x^6 - 2x^4 - 2x^3 - 2x^2 - 2x - 2$ | 1.977807115 |
| $x^6 - x^5 - x^4 - x^3 - 2x^2 + 2$ | 1.963984556 |
| $x^7 - x^6 - x^5 - x^4 + x^3 - 2x^2 + 2$ | 1.815396315 |
| $x^7 - x^6 - x^5 - 2x^4 + 2x^2 + 2$ | 1.888840344 |
| $x^7 - x^6 - x^5 - x^4 - x^3 + 2$ | 1.903972308 |
| $x^7 - x^6 - x^5 - 2x^4 + 2x + 2$ | 1.937730036 |
| $x^7 - x^6 - x^5 - x^4 - x^3 - 2x^2 + 2x + 2$ | 1.945197233 |
| $x^7 - x^6 - x^5 - 2x^4 + 2$ | 1.981204104 |
| $x^7 - x^6 - x^5 - x^4 - 2x^3 + 2$ | 1.982546502 |
| $x^7 - x^6 - x^5 - x^4 - x^3 - 2x^2 + 2$ | 1.983151826 |

Table 3.5: Polynomials with non-uniformly discrete spectrum $A(q)$

and the second part follows by noticing that:

$$A(q) - A(q) \subseteq \Lambda^2(q).$$

■

So if the conjecture is true that for $q \in (1, 2)$, $l(q) > 0$ if and only if q is a Pisot number (see for example [5, 25]), then all the non-Pisot numbers q with discrete $A(q)$ must be non-uniformly discrete.

The non-Pisot numbers in Table 3.5 are known to have non-uniformly discrete spectra. By a simple calculation these numbers have discrete spectra. It is seen that

| Minimal polynomial | Salem number q | $a(q)$ | Size of spectrum in $[\alpha_l, \alpha_u]$ |
|---|------------------|----------|--|
| $x^4 - x^3 - x^2 - x + 1$ | 1.722083806 | 0.243489 | 11 |
| $x^4 - 2x^3 + x^2 - 2x + 1$ | 1.883203506 | 0.249038 | 13 |
| $x^6 - x^5 - x^4 - x^3 - x^2 - x + 1$ | 1.946856268 | 0.249814 | 15 |
| $x^6 - 2x^5 + x^4 - 2x^3 + x^2 - 2x + 1$ | 1.974818708 | 0.249959 | 17 |
| $x^6 - 2x^4 - 3x^3 - 2x^2 + 1$ | 1.987793167 | 0.249991 | 19 |
| $x^8 - 2x^7 + x^6 - 2x^5 + x^4 - 2x^3 + x^2 - 2x + 1$ | 1.994004199 | 0.249998 | 21 |
| $x^{10} - x^9 - x^8 - x^7 - x^6 - x^5 - x^4 - x^3 - x^2 - x + 1$ | 1.997032367 | 0.249999 | 23 |
| $x^{10} - 2x^9 + x^8 - 2x^7 + x^6 - 2x^5 + x^4 - 2x^3 + x^2 - 2x + 1$ | 1.998524670 | 0.25 | 25 |

Table 3.6: Known Salem numbers q of degree ≤ 10 , where $A(q)$ is discrete

$l(q) = 0$ for these numbers as they do not satisfy a height 1 polynomial [8, 14]. Thus by Theorem 3.2 these spectra are non-uniformly discrete.

3.3 Salem Numbers and $A(q)$

Consider the Salem numbers in Table 3.6. By noticing that $x^6 - 2x^4 - 3x^3 - 2x^2 + 1$ divides $x^8 - x^7 - x^6 - x^5 - x^4 - x^3 - x^2 - x + 1$, and that $x^{2n} - 2x^{2n-1} + x^{2n-2} - \dots - 2x + 1$ divides $x^{2n+1} - x^{2n} - x^{2n-1} - \dots - x + 1$, it is noticed that all of these Salem numbers satisfy a polynomial of the form $x^n - x^{n-1} - x^{n-2} - \dots - x + 1$ for $n \geq 4$.

In this section we prove that if q is the positive real algebraic integer greater than 1 satisfying $x^n - x^{n-1} - x^{n-2} - \dots - x + 1$, then q is a Salem number. We further prove that if q is a Salem number of this form, then $A(q)$ is discrete. It is still unknown

if all Salem numbers q where $A(q)$ is discrete satisfy a polynomial of this form, nor is it known if these Salem numbers are the only non-Pisot numbers q where $A(q)$ is discrete and where their Mahler measure is less than 2.

Theorem 3.3. *The root of the polynomial $x^n - x^{n-1} - \dots - x + 1$ between 1 and 2 is a Salem number, for $n \geq 4$.*

Proof. By [31] if $P(x)$ is the minimal polynomial of a Pisot number and $P^*(x) := P(\frac{1}{x})x^{\deg(P(x))}$ is the *reciprocal* of $P(x)$, then provided that $(x^n P(x) - P^*(x))/(x - 1)$ has a root greater than 1, then this root is a Salem number. Here we take $P(x) = x - 2$. To see that there is a root between 1 and 2, we use the intermediate value theorem. ■

Theorem 3.4. *If $1 < q$ satisfies the polynomial $x^n - x^{n-1} - x^{n-2} - \dots - x + 1$, then $A(q)$ is discrete.*

Proof. To see that $A(q)$ is discrete, simply consider the algorithm and notice that it must terminate. The following observations are needed.

1. $q^m - q^{m-1} - \dots - q + 1 > \frac{1}{q-1}$ for $m < n$.
2. $q^n - q^{n-1} - \dots - q - 1 < \frac{-1}{q-1}$.

Thus at each step of the algorithm there is only one choice, and the algorithm must terminate after n steps. (The case of polynomials with negative lead coefficients is equivalent by symmetry.)

It remains to prove these two observations.

1. First notice that the Salem numbers q_n that satisfy $x^n - x^{n-1} - \dots - x + 1$ form an increasing sequence bounded below by 1 and above by 2. (This follows as $q_n^{n+1} - q_n^n - \dots - q_n + 1 = q_n^{n+1} - 2q_n^n = q_n^n(q_n - 2) < 0$ and $2^{n+1} - 2^n - \dots - 2 + 1 = 3 > 0$.)

Thus for $m \leq n - 2$:

$$q_n^{m+2} - q_n^{m+1} - \dots - q_n + 1 \geq 0,$$

which implies:

$$q_n^{m+2} - q_n^{m+1} - \dots - q_n^2 \geq q_n - 1.$$

Dividing by q_n^2 gives:

$$\begin{aligned} q_n^m - q_n^{m-1} - \dots - 1 &\geq \frac{q_n - 1}{q_n^2} \\ &> 0. \end{aligned}$$

Adding 2 to each side gives:

$$\begin{aligned} q_n^m - q_n^{m-1} - \dots + 1 &> 2 \\ &> \frac{1}{q-1}. \end{aligned}$$

The last inequality, that $2 > \frac{1}{q-1}$, follows as the smallest Salem number of this form is approximately 1.72.

For $m = n - 1$ and $q_n = q > \frac{3+\sqrt{17}}{4} \approx 1.780$ it follows that:

$$0 > -2q^2 + 3q + 1.$$

Dividing by $q - 1$ gives:

$$\begin{aligned} 0 &> \frac{-2q^2 + 2q}{q-1} + \frac{q+1}{q-1}, \\ &> -2q + \frac{q}{q-1} + \frac{1}{q-1}, \\ &> \frac{q}{q-1} - 2q + 1. \end{aligned}$$

Moving $\frac{q}{q-1}$ and flipping the sides gives:

$$\begin{aligned}
\frac{q}{q-1} &< 2q - 1 \\
&< (q^n - q^{n-1} - \dots - q + 1) + 2q - 1 \\
&< q^n - q^{n-1} - \dots - q^2 + q \\
&< q(q^{n-1} - \dots - q + 1) \\
&< q^{n-1} - \dots - q + 1.
\end{aligned}$$

For the case of $q < 1.780$, we simply note that this is covered in Table 3.6.

2. Notice $q^n - q^{n-1} - \dots - q - 1 = q^n - q^{n-1} - \dots - q + 1 - 2 = -2 < \frac{-1}{q-1}$

■

From this, we deduce:

Corollary 3.2. *If q is the Salem number satisfying $x^n - x^{n-1} - x^{n-2} - \dots - x + 1$, then $a(q) = \frac{q-1}{q^2} \approx \frac{1}{4}$.*

3.4 Quadratic and Cubic Pisot numbers

For τ the golden ratio, there is a nice description of $l^m(\tau)$. If F_k is the k th Fibonacci number and $\tau^{k-2} < m \leq \tau^{k-1}$ then $l^m(\tau) = |F_k\tau - F_{k+1}|$ [26]. The algorithm of Chapter 2 calculates $l^m(q)$ for any Pisot number q and any integer m , limited only by the memory of the computer. Although this method makes calculations for any given q or m , it has no predictive power for general m , or classes of Pisot numbers q . Using this method of calculation, an examination of other Pisot numbers is performed in the hopes of finding a similarly nice description of $l^m(q)$ for other q . The results of this search are described in Chapters 4 and 5.

Chapter 4

Unit Quadratic Pisot Numbers

This made him a grad student, and grad students existed not to learn things but to relieve the tenured faculty members of tiresome burdens such as educating people or doing research.

Cryptonomicon – Neal Stephenson

4.1 Background on Quadratic Pisot Numbers

Chapter 3 explores various applications of the algorithm given in Chapter 2. Some applications looked at include calculations of $l(q)$ and $a(q)$ for Pisot numbers q of degree 10 or less. Thus we explore $l(q)$ or $a(q)$ as q ranges over various Pisot (or other algebraic) numbers. In this chapter we instead fix q and explore $l^m(q)$ as m varies over the positive integers. We base this exploration on the model of Komornik, Loreti and Pedicini [26]. They give a complete description of all $l^m(\tau)$ where τ is the golden ratio. If F_k is the k th Fibonacci number and $\tau^{k-2} < m \leq \tau^{k-1}$ then $l^m(\tau) = |F_k\tau - F_{k+1}|$.

The algorithm in Chapter 2 calculates $l^m(q)$ for any Pisot number q and any integer $m > 0$, limited only by the memory of the computer. Although this algorithm makes calculations for any given combination of m and q , it has no predictive power for general m , or classes of Pisot numbers q . Using this method of calculation, examples

are found that suggest a relationship between all unit quadratic Pisot numbers and their best approximations, similar to the relationship found by Komornik, Loreti and Pedicini for the golden ratio. This chapter proves this relationship.

Recall that an algebraic integer α is a *unit* if the product of α with all of its conjugates is ± 1 . It is convenient to adopt the following notation:

Definition 4.1 (Unit quadratic Pisot number). *Let Ω be the set of Pisot numbers, such that these Pisot numbers are units, and they have minimal polynomials of degree 2.*

We see that these numbers satisfy polynomials of the form $x^2 - cx - 1$ for $c \in \{1, 2, 3, \dots\}$, or $x^2 - cx + 1$ for $c \in \{3, 4, 5, \dots\}$.

Let $q \in \Omega$. This chapter shows that $l^m(q) = |Dq - C|$ where $\frac{C}{D}$ is a best approximation of q . A better description of which best approximations $l^m(q)$ is equal to is given in Theorem 4.1, of. Section 4.3.

Section 4.2 gives a description of the proof of Theorem 4.1. The formal proof is given in Section 4.3. An algorithmic implementation of one of the lemmas of Section 4.3 is given in Section 4.4. Section 4.5 demonstrates why Theorem 4.1 does not easily extend to non-unit quadratic Pisot numbers. Section 4.6 gives some direction where this research might be expanded upon.

4.2 Description of the Proof

The next section gives a description of $l^m(q)$ for all $q \in \Omega$. However, we first give a heuristic argument as to why the description in Section 4.3 should be true.

Example 6. *Let $q \in \Omega$ be the Pisot number satisfying $x^2 - 3x + 1$ (approximately 2.61803). Notice, for all $y \in \Lambda^2(q)$ there exists $\epsilon_i \in \mathbb{Z}$, $|\epsilon_i| \leq 2$ such that $y = \sum \epsilon_i q^i$. Choose s and t such that $sq + t \equiv \sum \epsilon_i x^i \pmod{x^2 - 3x + 1}$. Figure 4.1 graphs the ordered pair (s, t) where $sq + t \in \Lambda^2(q)$, and the line $sq + t = 0$. (Points close to the line $sq + t = 0$ are small values in the spectra, and hence of interest.)*

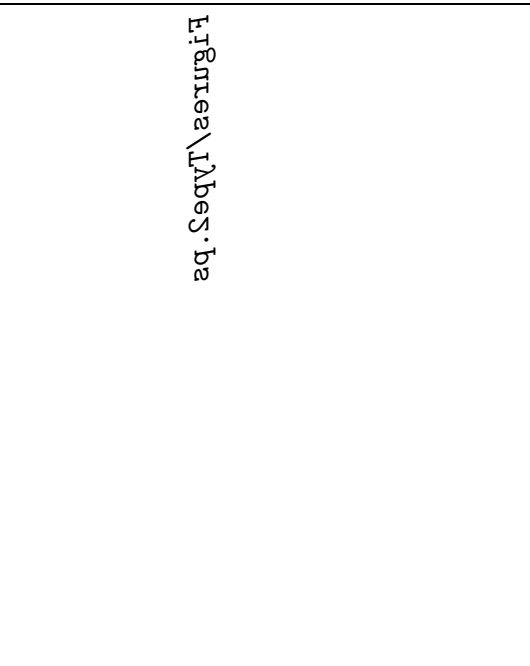
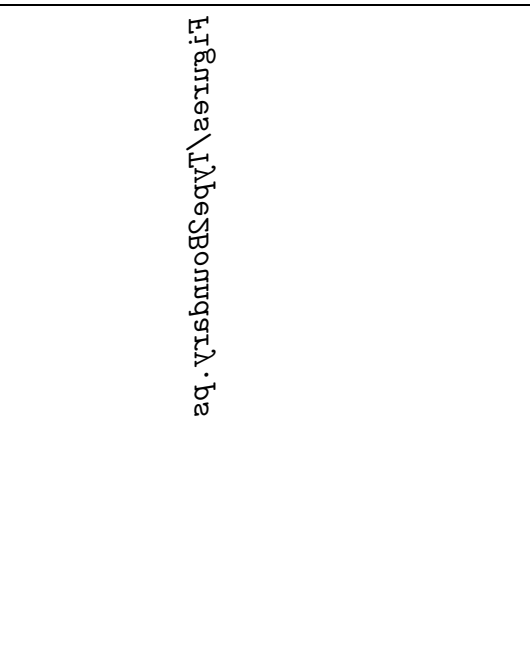


Figure 4.1

Figure 4.1: (s, t) where $sq + t \in \Lambda^2(q)$, q satisfies $x^2 - 3x + 1$, $|s| \leq 20$



Figures\labesBomunqrd\ba

Figure 4.3: (s, t) where $sq + t \in \Lambda^2(q)$, q satisfies $x^2 - 3x + 1$, $s \leq 20$, and lines $sr + t = \pm \frac{2}{1-|r|}$

We notice in Example 6 for $m = 2$ that all integer pairs (s, t) such that $sq + t \in \Lambda^m(q)$ appear to be between two parallel lines. This is indeed the case, as is shown in Lemma 4.2. Furthermore, all integer pairs (s', t') between these two parallel lines are such that $s'q + t' \in \Lambda^m(q)$, as is shown in Corollary 4.1.

Example 7. Consider Example 6 (page 36) again, and let q be the Pisot number satisfying $x^2 - 3x + 1$. Let r be the conjugate of q . Figure 4.3 gives (s, t) where $sq + t \in \Lambda^2(q)$ as well as the two bounding lines from Lemma 4.2, $sr + t = \pm \frac{2}{1-|r|}$.

By Lemmas 4.2 and 4.3 we have a description of $\Lambda^m(q)$, as all $sq + t$ where the integer points (s, t) are bounded by the lines $sr + t = \pm \frac{m}{1-|r|}$. Next it is an easy matter to show that the minimal values of $sq + t$ given the restrictions are the best approximations (Definition 4.4). This then gives us the complete description of $\Lambda^m(q)$

(Theorem 4.1).

4.3 Main Theorem

In this section we give a complete description of $l^m(q)$ for $q \in \Omega$. Unfortunately this description does not work for non-unit quadratic Pisot numbers, as shown in Section 4.5. Before proceeding we need a few lemmas and definitions. Here and throughout this section let a and b be any two fixed integers.

Definition 4.2 (A_n, B_n). Define the sequences $\{A_n\}_{n=0}^\infty$ and $\{B_n\}_{n=0}^\infty$ as:

1. $A_0 = 0, A_1 = 1, A_n = aA_{n-1} + bA_{n-2},$
2. $B_0 = 1, B_1 = 0, B_n = aB_{n-1} + bB_{n-2}.$

Lemma 4.1. Using the notation of Definition 4.2:

$$\det \left(\begin{bmatrix} A_n & A_{n-1} \\ B_n & B_{n-1} \end{bmatrix} \right) = (-b)^{n-1}.$$

Proof. Notice that:

$$\begin{aligned} \det \left(\begin{bmatrix} A_n & A_{n-1} \\ B_n & B_{n-1} \end{bmatrix} \right) &= \det \left(\begin{bmatrix} aA_{n-1} + bA_{n-2} & A_{n-1} \\ aB_{n-1} + bB_{n-2} & B_{n-1} \end{bmatrix} \right), \\ &= \det \left(\begin{bmatrix} bA_{n-2} & A_{n-1} \\ bB_{n-2} & B_{n-1} \end{bmatrix} \right), \\ &= -b \det \left(\begin{bmatrix} A_{n-1} & A_{n-2} \\ B_{n-1} & B_{n-2} \end{bmatrix} \right). \end{aligned}$$

Further, notice that:

$$\begin{bmatrix} A_1 & A_0 \\ B_1 & B_0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

The result follows by induction. ■

Lemma 4.2. *Let $sq + t \in \Lambda^m(q)$ where $q \in \Omega$ and r is the conjugate of q . Then:*

$$|sr + t| \leq \pm \frac{m}{1 - |r|}.$$

Proof. Let $p(x) = x^2 - ax - b$ be the minimal polynomial of q . Notice by induction that for $n \geq 0$ we have

$$x^n \equiv A_n x + B_n \pmod{p(x)}. \quad (4.1)$$

Thus, from a standard result of recurrence relations (see for instance [22]), we have:

$$A_n = \frac{1}{q - r} q^n + \frac{1}{r - q} r^n \quad (4.2)$$

and

$$B_n = \frac{r}{r - q} q^n + \frac{q}{q - r} r^n. \quad (4.3)$$

Combining equations (4.1), (4.2) and (4.3) gives:

$$x^n \equiv \frac{1}{q - r} q^n (x - r) + \frac{1}{r - q} r^n (x - q) \pmod{p(x)}.$$

If $sq + t \in \Lambda^m(q)$ then there exist $\epsilon_n \in \mathbb{Z}$ with $|\epsilon_n| \leq m$, such that

$$sx + t \equiv \sum_{n=0}^k \epsilon_n x^n \pmod{p(x)}$$

and thus:

$$sx + t \equiv \sum_{n=0}^k \frac{\epsilon_n}{q - r} (q^n (x - r) - r^n (x - q)) \pmod{p(x)}.$$

By evaluating at r we get:

$$\begin{aligned}
|sr + t| &= \left| \sum_{n=0}^k \frac{\epsilon_n}{q-r} (q^n(r-r) - r^n(r-q)) \right|, \\
&= \left| \sum_{n=0}^k \frac{-\epsilon_n}{q-r} r^n(r-q) \right|, \\
&\leq \left| \sum_{n=0}^{\infty} mr^n \right|, \\
&\leq \frac{m}{1-|r|}.
\end{aligned}$$

This is the desired result. ■

The next lemma is similar to the ones described in Section 3 of [26], but the presentation is different. For this lemma we need the observation that because $\lim \left| \frac{A_n}{B_n} \right| < \infty$ there exists an m such that $|A_n| \leq 2m|B_n|$ for all $n \geq 2$. For this lemma we also need the following definition:

Definition 4.3 (\mathcal{R}). Define $\mathcal{R} := \{\rho_n x^n + \dots + \rho_0 : n \in \mathbb{N}, \rho_i \in \mathbb{R}, |\rho_i| \leq 1\}$.

Lemma 4.3. Let $q \in \Omega$, let $p(x) = x^2 - ax \pm 1$ be the minimal polynomial of q , and let r be the conjugate of q . Let $m \in \mathbb{N}$ be such that $|A_n| \leq 2m|B_n|$ for all $n \geq 2$. Let $y \in m\mathcal{R}$ such that $y \equiv cx + d \pmod{p(x)}$ where $c, d \in \mathbb{Z}$. Then there exists a $y' \in m\mathcal{R} \cap \mathbb{Z}[x]$ such that $y' \equiv cx + d \pmod{p(x)}$.

Proof. Assume that $y \equiv cx + d \pmod{p(x)}$ with $c, d \in \mathbb{Z}$. We show here how to find y' such that $y' \equiv y \pmod{p(x)}$ and $y' \in \mathbb{Z}[x] \cap m\mathcal{R}$. Let:

$$y = a_n x^n + \dots + a_0$$

where $a_i \in \mathbb{R}$. If $a_n \in \mathbb{Z}$ then continue recursively on $a_{n-1}x^{n-1} + \dots + a_0$. If $a_0 \in \mathbb{Z}$ then continue recursively on $a_n x^{n-1} + \dots + a_1$. If neither a_0 nor a_n is an integer, then use the

identity that $x^n - A_n x - B_n \equiv 0 \pmod{p(x)}$ along with the fact that $|A_n| \leq m|B_n|$ to solve for α where $a_n + \alpha \in \mathbb{Z}$ or $a_0 - \alpha B_n \in \mathbb{Z}$, and $|a_n + \alpha|, |a_1 - A_n \alpha|, |a_0 - B_n \alpha| \leq m$. Solving for α such that $a_0 - \alpha B_n = \lceil a_0 \rceil$ or $\lfloor a_0 \rfloor$ gives us two possible values, one negative and one positive. These possible values have the property that their difference is equal to $\frac{1}{|B_n|}$. Similarly, by solving for α such that $a_n + \alpha = \lceil a_n \rceil$ or $\lfloor a_n \rfloor$ we again get two possible values, one negative and one positive. Now let α_l be the maximal of the two negative values, and α_u be the minimal of the two positive values. So for both $\alpha = \alpha_l$ and $\alpha = \alpha_u$ we have that $a_n + \alpha \in \mathbb{Z}$ or $a_0 - \alpha B_n \in \mathbb{Z}$ and further $|a_n + \alpha|, |a_0 - B_n \alpha| \leq m$. By assumption $|A_n| \leq 2m|B_n|$ and further $\alpha_u - \alpha_l \leq \frac{1}{|B_n|}$ which gives that $|\alpha_u A_n - \alpha_l A_n| \leq 2m$ and hence at least one of these two values, $|a_1 - \alpha A_n| \leq m$. Continue recursively on $a_n x^n + \dots + a_0 + \alpha(x^n - A_n x - B_n)$.

By application of this recursion we see that y' is such that all of the a_i can be made integers, with the possible exception of two consecutive terms, a_j and a_{j-1} . Notice that:

$$a_n x^n + \dots + a_{j+1} x^{j+1} + a_{j-2} x^{j-2} + \dots + a_0 \equiv c' x + d' \pmod{p(x)}$$

for some $d', c' \in \mathbb{Z}$. Thus we see that:

$$a_j x^j + a_{j-1} x^{j-1} \equiv (c - c')x + (d - d') \pmod{p(x)},$$

where $c - c', d - d' \in \mathbb{Z}$. By Lemma 4.2 we know that:

$$a_j x^j + a_{j-1} x^{j-1} \equiv a_j A_j x + a_{j-1} A_{j-1} x + a_j B_j + a_{j-1} B_{j-1} \pmod{p(x)}.$$

Thus we have that:

$$\begin{bmatrix} A_j & A_{j-1} \\ B_j & B_{j-1} \end{bmatrix} \begin{bmatrix} a_j \\ a_{j-1} \end{bmatrix} = \begin{bmatrix} c - c' \\ d - d' \end{bmatrix}.$$

Noticing that the determinant of $\begin{bmatrix} A_j & A_{j-1} \\ B_j & B_{j-1} \end{bmatrix}$ is ± 1 , gives that the inverse of this matrix has integer entries, and thus $a_j, a_{j-1} \in \mathbb{Z}$.

■

What is interesting here is that this proof is constructive, and a computer algorithm can be designed to perform the steps of the induction in this proof. This is

described in Section 4.4.

Corollary 4.1. *Let $q \in \Omega$. Let m be such that $|A_n| \leq 2m|B_n|$ for all $n \geq 2$. Let r be the conjugate of q . Then $sq + t \in \Lambda^m(q)$ if and only if*

$$|sr + t| < \pm \frac{m}{1 - |r|}$$

Proof. From Lemma 4.2 we see that $sq + t \in \Lambda^m(q)$ implies $|sr + t| \leq \pm \frac{m}{1 - |r|}$.

If $|sr + t| < \pm \frac{m}{1 - |r|}$ we see from the proof of Lemma 4.2 that we can find a choose real numbers ϵ_n , where $|\epsilon| \leq m$ such that $sx + t \equiv \sum_{n=0}^k \epsilon_n x^n$. From Lemma 4.3 we can find an integer polynomial $p(x)$, of height m such that $sq + t = p(q)$. From this it follows that $sq + t \in \Lambda^m(q)$ and the result follows. ■

Lemma 4.4. *Let $q \in \Omega$ satisfy a polynomial $x^2 - ax \pm 1$. If $|A_n| > 2m|B_n|$ for some $n \geq 2$ then $l^m(q) = 1$.*

Proof. Notice that $A_2 = a$ and $B_2 = b$. Further notices that $\frac{|A_n|}{|B_n|}$ are best approximations of q . So we notice that if $|A_n| > 2m|B_n|$ for some $n \geq 2$ then $|A_2| > 2m|B_2|$. This would happen if $2m < q$. In this case $l^m(q) = 1$. ■

We adopt the definition of [10] for best approximation:

Definition 4.4 (Best Approximation). *A fraction $\frac{C}{D}$, $C, D \in \mathbb{Z}$, $C > 0$ gives a best approximation to q if*

$$|D'q - C'| > |Dq - C| \text{ for all } 0 < D' < D.$$

It is noted in [10] (page 5, Theorem II) than $\frac{C}{D}$ are the convergents to q .

These results combine to give the final theorem:

Theorem 4.1. *Let $q \in \Omega$ have a conjugate r . Let q have best approximations $\left\{ \frac{C_k}{D_k} \right\}$ and let k be the maximal integer such that:*

$$|D_k r - C_k| \leq m \frac{1}{1 - |r|}$$

then:

$$l^m(q) = |D_k q - C_k|.$$

It is worth noting this is a generalization of Theorem 3.1 in [26], and gives the equivalent result when it is restricted to the case when q is the golden ratio.

Proof (Theorem 4.1). By Lemma 4.4 we see that the Theorem is trivially true for when $|A_n| > 2m|B_n|$ for some $n \geq 2$. Hence we may assume that $|A_n| \leq 2m|B_n|$ for all $n \geq 2$. Consider the best approximations of q , $\left\{ \frac{C_k}{D_k} \right\}$. By Definition the best linear approximations to zero are of the form $D_k q - C_k$. Corollary 4.1 indicates that if $D_k x - C_k \in m\mathcal{R} \pmod{p(x)}$ then there exists a $y \in \mathbb{Z}[x] \cap m\mathcal{R}$ such that $y \equiv D_k x - C_k \pmod{p(x)}$. It follows that $l^m(q) = |D_k q - C_k|$ when $D_k x - C_k \in m\mathcal{R} \pmod{p(x)}$ with k maximal. Thus if k is maximal such that $|D_k r - C_k| \leq \frac{m}{1 - |r|}$ then $l^m(q) = |D_k q - C_k|$ which is the desired result. ■

Corollary 4.2. *Let $F_n = aF_{n-1} + F_{n-2}$ with $F_0 = 0$ and $F_1 = 1$ and let q be the Pisot number satisfying $x^2 - ax - 1$. If $q^{k-1}(q-1) \leq m < q^k(q-1)$ then $l^m(q) = |F_k q - F_{k+1}|$.*

Proof. It is easy to verify that $\left\{ \frac{F_{k+1}}{F_k} \right\}$ are the best approximations of q . A simple calculation shows that:

$$F_k = \frac{1}{q - r}(q^k - r^k)$$

and that $r = -\frac{1}{q}$. This yields that $l^m(q) = |F_k q - F_{k+1}|$ when k is maximal such that:

$$\begin{aligned} \frac{m}{1 - |r|} &\geq |F_k r - F_{k+1}| \\ &\geq \left| \frac{1}{q - r}((q^k - r^k)r - (q^{k+1} - r^{k+1})) \right| \end{aligned}$$

$$\begin{aligned} &\geq \left| \frac{1}{q-r}(q^k r - q^{k+1}) \right| \\ &\geq \left| \frac{r-q}{q-r} q^k \right| \\ &\geq q^k, \end{aligned}$$

which implies:

$$(q-1)q^{k-1} \leq m,$$

and the result follows. ■

Table 4.1 gives the ranges that m is in, for $l^m(q) = F_{k-1}q - F_k$.

| | Minimal polynomial | | | |
|----------------|--------------------|----------------|----------------|----------------|
| $l^m(q)$ | $x^2 - x - 1$ | $x^2 - 2x - 1$ | $x^2 - 3x - 1$ | $x^2 - 4x - 1$ |
| $ F_0q - F_1 $ | | [1,1] | [1,2] | [1,3] |
| $ F_1q - F_2 $ | [1,1] | [2,3] | [3,7] | [4,13] |
| $ F_2q - F_3 $ | | [4,8] | [8,25] | [14,58] |
| $ F_3q - F_4 $ | [2,2] | [9,19] | [26,82] | [59,245] |
| $ F_4q - F_5 $ | [3,4] | [20,48] | [83,274] | [246,1042] |
| $ F_5q - F_6 $ | [5,6] | [49,115] | [275,904] | [1043,4413] |
| $ F_6q - F_7 $ | [7,11] | [116,280] | [905,2989] | [4414,18698] |
| $ F_7q - F_8 $ | [12,17] | [281,675] | [2990,9871] | [18699,79205] |

Table 4.1: Range for m when $l^m(q) = |F_{k-1}q - F_k|$

With a proof similar to that of Corollary 4.2, we get:

Corollary 4.3. Define $E_n = aE_{n-1} - E_{n-2}$ with $E_0 = 0$ and $E_1 = 1$. Define $G_n = aG_{n-1} - G_{n-2}$ with $G_0 = 1$ and $G_1 = 1$. Let q be the Pisot number satisfying

$x^2 - ax + 1$. If $q^{k-3}(q-1)^2 \leq m < q^{k-2}(q-1)$ then $l^m(q) = |G_{k-1}q - G_k|$ and if $q^{k-2}(q-1) \leq m < q^{k-1}(q-1)$ then $l^m(q) = |E_{k-1}q - E_k|$.

Table 4.2 gives the ranges that m is in, for $l^m(q) = |G_{k-1}q - G_k|$ and $l^m(q) = |E_{k-1}q - E_k|$.

| | Minimal polynomial | | | |
|----------------|--------------------|----------------|----------------|----------------|
| $l^m(q)$ | $x^2 - 3x + 1$ | $x^2 - 4x + 1$ | $x^2 - 5x + 1$ | $x^2 - 6x + 1$ |
| $ G_0q - G_1 $ | | | | |
| $ E_0q - E_1 $ | [1,1] | [1,2] | [1,2] | [1,3] |
| $ G_1q - G_2 $ | | | [3,3] | [4,4] |
| $ E_1q - E_2 $ | [2,2] | [3,7] | [4,14] | [5,23] |
| $ G_2q - G_3 $ | [3,4] | [8,10] | [15,18] | [24,28] |
| $ E_2q - E_3 $ | [5,6] | [11,27] | [19,68] | [29,135] |
| $ G_3q - G_4 $ | [7,11] | [28,38] | [69,87] | [136,164] |
| $ E_3q - E_4 $ | [12,17] | [39,103] | [88,329] | [165,791] |

Table 4.2: Range for m when $l^m(q) = |E_{k-1}q - E_k|$ or $|G_{k-1}q - G_k|$

4.4 Finding Height m Polynomials

For $q \in \Omega$ we have $l^m(q) = |Dq - C|$ for some integers C and D where $\frac{C}{D}$ is a best approximation of q . However, both C and D are often greater than m . This section is interested in finding a particular height m polynomial equal to $l^m(q)$. We notice that Lemma 4.3 can be implemented as an algorithm. Thus, if $p(x)$ is the minimal polynomial of q , then it is sufficient to find a $t(x) \in m\mathcal{R}$ such that $t(x) \equiv Dx - C \pmod{p(x)}$. For this we can use the simplex method [34]. Write:

$$Dx - C + \left(\sum_{i=0}^n a_i x^i \right) p(x) = \sum_{k=0}^{n+2} b_k x^k$$

for unknowns a_k . We wish $-m \leq b_k \leq m$ for all $k = 0, \dots, (n+2)$. So for the correct value of n we minimize h with:

$$-h \leq b_k \leq h \quad (4.4)$$

and solve for the a_k (and hence the b_k). If $h \leq m$ then we are done. A careful calculation (via Maple) yields the minimal value for n for which we can get $h \leq m$ is:

$$n = \left\lceil \ln \left(\frac{m + |Dr - C||r| - |Dr - C|}{m|r|} \right) (\ln(|r|))^{-1} \right\rceil. \quad (4.5)$$

Using the simplex method in this way works for any polynomial, although the value for n given in equation (4.5) is specifically for $q \in \Omega$. Thus, if we wish to implement this algorithm for polynomials that are not minimal polynomials for some $q \in \Omega$, we simply increase n until we find one that works.

Example 8. Let q be the Pisot number satisfying $p(x) = x^2 - 3x + 1$. A simple calculation demonstrates that $l^7(q) = 5q - 13$. Using equation (4.5) we discover that the minimal value for n is 3. So minimizing h with respect to the constraints in equation (4.4) and with $n = 3$ gives $h = \frac{305}{44} < 7$. This gives the polynomial:

$$\frac{17}{4}x^5 - \frac{305}{44}x^4 - \frac{305}{44}x^3 - \frac{305}{44}x^2 - \frac{305}{44}x - \frac{305}{44}.$$

Here we use the techniques in Lemma 4.3 iteratively. Notice that at any step, only three coefficients are altered.

$$\begin{array}{rcccccccc} \frac{17}{4}x^5 & - & \frac{305}{44}x^4 & - & \frac{305}{44}x^3 & - & \frac{305}{44}x^2 & - & \frac{305}{44}x & - & \frac{305}{44} & \equiv & 5x - 13, \\ \frac{327}{77}x^5 & - & \frac{305}{44}x^4 & - & \frac{305}{44}x^3 & - & \frac{305}{44}x^2 & - & \frac{520}{77}x & - & 7 & \equiv & 5x - 13, \\ \frac{371}{88}x^5 & - & \frac{305}{44}x^4 & - & \frac{305}{44}x^3 & - & \frac{553}{88}x^2 & - & 7x & - & 7 & \equiv & 5x - 13, \\ 4x^5 & - & \frac{305}{44}x^4 & - & \frac{229}{44}x^3 & - & \frac{305}{44}x^2 & - & 7x & - & 7 & \equiv & 5x - 13, \\ 4x^5 & - & 7x^4 & - & 5x^3 & - & 7x^2 & - & 7x & - & 7 & \equiv & 5x - 13. \end{array}$$

Thus $l^7(q) = 5q - 13 = 4q^5 - 7q^4 - 5q^3 - 7q^2 - 7q - 7$.

4.5 Non-Unit Quadratic Pisot Numbers

Note that Theorem 4.1 does not work for all quadratic Pisot numbers. The problem is that Lemma 4.3 doesn't work for Pisot numbers satisfying $x^2 - ax - b$ when $b \neq \pm 1$. This is demonstrated in Example 9.

Example 9. *If q is the Pisot number satisfying $x^2 - 2x - 2$ (approximately 2.7320508) then we see that $\frac{7}{16}x^8 - 2x^7 + 3x^6 - 3x^5 + 3x^4 - 3x^3 + 3x^2 - 3x + 3 \equiv 8 - 3x \pmod{x^2 - 2x - 2}$ is in $3\mathcal{R}$ but is not in $\mathbb{Z}[x] \cap 3\mathcal{R}$. Further we have that $|8 - 3q| = 0.196152424 < 0.267949192 = l^3(q)$.*

Figure 4.4 is a picture of (s, t) where $sq + t \in \Lambda^3(q)$. We have also included the lines $sr + t = \pm \frac{m}{1-|r|}$, and the line $sq + t = 0$. Notice that $\Lambda^3(q)$ does not contain all $sq + t$ such that the integer pairs (s, t) are between these two lines, as would be expected from Corollary 4.1, and as would be necessary for this theorem to work. Further note that the edges of $\Lambda^3(q)$ are irregular enough so that it is not possible to find alternate lines that bound all the pairs (s, t) , and such that all $sq + t$ are included in the spectrum.

4.6 Further Research

The counter example of Section 4.5 shows that Theorem 4.1 does not work for all quadratic Pisot numbers. Despite this, the spirit of Theorem 4.1 appears to be true. Let $\left\{ \frac{C_k}{D_k} \right\}$ be the best approximations of a quadratic Pisot number q , with conjugate r , not necessarily a unit. If Theorem 4.1 could be extended to this case, then $l^m(q)$ would equal $|D_k q - C_k|$, where k is maximal such that $|D_k r - C_k| \leq \frac{m}{1-|r|}$. Computationally, it appears that $l^m(q) = |D_k q - C_k|$ or $|D_{k-1} q - C_{k-1}|$. It would be interesting to know if $l^m(q)$ must be one of these two values.

It would also be of interest if a lemma similar to Lemma 4.3 could be found that would work for all polynomials p where $p(0) = \pm 1$, regardless of the degree of $p(x)$. If something like this could be found then this could be used to prove: for $q \in (1, 2)$,

Figure 4.4

Figure 4.4: (s, t) where $sq + t \in \Lambda^3(q)$

$l(q) > 0$ if and only if q is a Pisot number, (see Chapter 6 for details). This is believed to be true by a number of people, see for example [5, 25]. The second part of this Lemma easily extends to an arbitrary degree, but it is not clear that there is an algorithm that forces all but d consecutive terms to be integers (where d is the degree of $p(x)$).

Chapter 5

Cubic Pisot Numbers

MOSCOW – Five top Russian scientists have cooked up a bold, ambitious plan to transform the entire Earth into a paradise and banish their country’s harsh winters forever – by blowing the moon to smithereens!

...

Dr. Khruinsky and his colleagues aren’t the first to tout the potential benefits of blowing up the moon. As far back as 1991, Iowa State University mathematics professor Alexander Abian proposed the idea, as Weekly World News reported then.

New plan to blow up the moon! – Vickie York

5.1 Background on Cubic Pisot Numbers

Chapter 4 looks at $l^m(q)$ for unit quadratic Pisot numbers q . Chapter 4 shows that $l^m(q)$ is $|Dq - C|$ where $\frac{C}{D}$ is a best approximation of q . Using the algorithm from Chapter 2 other patterns are searched for, similar to the pattern for the unit quadratic Pisot numbers.

A pattern is found for the first and second cubic Pisot numbers satisfying $x^3 - x - 1$ and $x^3 - x^2 - 1$ respectively. It appears that $l^m(q) = P_n(q) := \frac{1}{q^n}$ for some n dependent on m and q . Here, $P_n(q)$ is defined in Theorem 5.1. This relationship is verified up to $m = 39$ for the first cubic Pisot number and up to $m = 47$ for the second cubic Pisot number. These results are summarized in Tables 5.1 and 5.2. Both the first and second cubic Pisot numbers are part of a larger class of Pisot numbers, called Υ , (see Definition 5.1). This chapter does not show that $l^m(q) = P_n(q)$ for q in this larger class of Pisot numbers, but it does show some very special properties of $P_n(q)$ for these q , which may be useful in showing $l^m(q) = P_n(q)$.

Much as the study of quadratic Pisot numbers requires knowledge of good linear approximations (i.e., best approximations), the study of cubic Pisot numbers requires knowledge of good quadratic approximations. For an irrational number q , the idea of finding good linear approximations to zero of the form $a_nq - b_n$, [3, 10], can be extended to the idea of finding good quadratic approximations to zero of the form $A_nq^2 + B_nq + C_n$. Clearly, if q is a quadratic number, then A_n , B_n and C_n can be solved exactly as integers. Here we consider the case when q is a particular type of cubic Pisot number. The class of cubic Pisot numbers that we consider is:

Definition 5.1. *Let Υ be the set of Pisot numbers with minimal polynomials $p(x)$ such that:*

1. $p(0) = -1$,
2. $\deg(p(x)) = 3$,
3. $p(x)$ has exactly one real root, and two non-real roots.

We see that requirement 3 does not follow directly from requirements 1 and 2, as $x^3 - 3x^2 - 4x - 1$ satisfies the first two requirements, yet has three real roots of approximately -0.6920214716, -0.3568958679 and 4.048917340. We easily see that this class of Pisot numbers is non-trivial and infinite. Both the first and second cubic Pisot numbers are members of Υ .

| n | $P_n(q)$ | $l^m(q)$ where $l^m(q) = P_n(q)$ |
|----|---------------------|----------------------------------|
| 0 | 1 | |
| ⋮ | ⋮ | ⋮ |
| 10 | $-3q^2 + q + 4$ | $l^1(q)$ |
| ⋮ | ⋮ | ⋮ |
| 15 | $-7q^2 + 4q + 7$ | $l^2(q)$ |
| ⋮ | ⋮ | ⋮ |
| 18 | $-4q^2 - 3q + 11$ | $l^3(q)$ |
| ⋮ | ⋮ | ⋮ |
| 21 | $10q^2 - 14q + 1$ | $l^4(q)$ |
| 22 | $q^2 + 10q - 15$ | $l^5(q)$ |
| 23 | $-15q^2 + q + 25$ | $l^6(q)$ |
| 24 | $25q^2 - 15q - 24$ | $l^7(q)$ |
| 25 | $-24q^2 + 25q + 9$ | |
| 26 | $9q^2 - 24q + 16$ | $l^8(q), l^9(q)$ |
| 27 | $16q^2 + 9q - 40$ | $l^{10}(q), l^{11}(q)$ |
| 28 | $-40q^2 + 16q + 49$ | $l^{12}(q), l^{13}(q)$ |
| 29 | $49q^2 - 40q - 33$ | |
| 30 | $-33q^2 + 49q - 7$ | $l^{14}(q), l^{15}(q)$ |
| 31 | $-7q^2 - 33q + 56$ | $l^{16}(q), \dots, l^{18}(q)$ |
| 32 | $56q^2 - 7q - 89$ | $l^{19}(q), \dots, l^{22}(q)$ |
| 33 | $-89q^2 + 56q + 82$ | $l^{23}(q), \dots, l^{25}(q)$ |
| 34 | $82q^2 - 89q - 26$ | $l^{26}(q)$ |
| 35 | $-26q^2 + 82q - 63$ | $l^{27}(q), \dots, l^{32}(q)$ |
| 36 | $63q^2 + 26q - 145$ | $l^{23}(q), \dots, l^{39}(q)$ |

Table 5.1: Relationship between $P_n(q)$ and $l^m(q)$ for the first cubic Pisot number

| n | $P_n(q)$ | $l^m(q)$ where $l^m(q) = P_n(q)$ |
|----------|---------------------|----------------------------------|
| 0 | 1 | |
| \vdots | \vdots | \vdots |
| 5 | $q^2 - 2$ | $l^1(q)$ |
| \vdots | \vdots | \vdots |
| 10 | $-3q^2 + q + 5$ | $l^2(q)$ |
| 11 | $5q^2 - 8q + 1$ | |
| 12 | $q^2 + 4q - 8$ | $l^3(q)$ |
| 13 | $-8q^2 + 9q + 4$ | $l^4(q)$ |
| 14 | $4q^2 - 12q + 9$ | |
| 15 | $9q^2 - 5q - 12$ | $l^5(q), l^6(q)$ |
| 16 | $-12q^2 + 21q - 5$ | |
| 17 | $-5q^2 - 7q + 21$ | $l^7(q), l^8(q)$ |
| 18 | $21q^2 - 26q - 7$ | $l^9(q), l^{10}(q)$ |
| 19 | $-7q^2 + 28q - 26$ | $l^{11}(q), l^{12}(q)$ |
| 20 | $-26q^2 + 19q + 28$ | $l^{13}(q), \dots, l^{16}(q)$ |
| 21 | $28q^2 - 54q + 19$ | $l^{17}(q)$ |
| 22 | $19q^2 + 9q - 54$ | $l^{18}(q), \dots, l^{23}(q)$ |
| 23 | $-54q^2 + 73q + 9$ | $l^{24}(q), l^{25}(q)$ |
| 24 | $9q^2 - 63q + 73$ | $l^{26}(q), \dots, l^{32}(q)$ |
| 25 | $73q^2 - 64q - 63$ | $l^{33}(q), \dots, l^{40}(q)$ |
| 26 | $63q^2 - 136q + 64$ | $l^{31}(q), \dots, l^{47}(q)$ |

Table 5.2: Relationship between $P_n(q)$ and $l^m(q)$ for the second cubic Pisot number

Let us again consider the case of linear approximations to zero as it arises in the study of best approximations. It is known [3] that if α is an algebraic number and A and B are integers such that $|B\alpha - A| \leq \frac{1}{2B}$, then A/B is one of the best approximations of α . From this it follows that for $0 < b < B$ we have $|B\alpha - A| < |a\alpha - b|$. In other words, $P(x) = Bx - A$ is the best non-trivial linear approximation to zero at α bounded by height $\max\{A, B\}$. We prove a similar result here for quadratic approximations to zero of $q \in \Upsilon$.

A classic result known as Liouville's inequality, (see for example [28]), shows that if α is an algebraic number of degree $d \geq 2$, then there exists a $D(\alpha)$ such that:

$$\left| \alpha - \frac{p}{q} \right| \geq \frac{D(\alpha)}{q^d}, \quad p, q \in \mathbb{Z}, \quad q \geq 1,$$

or equivalently:

$$|q\alpha - p| \geq \frac{D(\alpha)}{q^{d-1}}.$$

Another result worth noting, by Schmidt [33], shows that for every $\epsilon > 0$, and for any algebraic number q where $1, q$ and q^2 are independent, there are only a finite number of integer quadratics $P(x)$ such that:

$$|P(q)| \leq \frac{1}{H(P(x))^{2+\epsilon}},$$

where $H(P(x))$ is the height of the polynomial $P(x)$. What Schmidt shows is actually more general than this, demonstrating that if $1, \alpha$ and β are linearly independent algebraic numbers, then there are only a finite number of integer solutions to

$$|a\alpha + b\beta + c| < \max\{a, b\}^{-2-\epsilon}.$$

Roth's classic result [30] states that if α is an irrational algebraic number, then for all $\epsilon > 0$ the equation:

$$\left| \alpha - \frac{p}{q} \right| \leq \frac{1}{q^{2+\epsilon}}$$

has only a finite number of integer solutions. The main open question with respect to this result is whether an effective upper bound for q (dependent on ϵ) can be found.

This chapter answers the equivalent question for quadratic approximations for $q \in \Upsilon$. An explicit upper bound is found for the height of $P(x)$, dependent on ϵ , after which there are no solutions to:

$$|P(q)| \leq \frac{1}{H(P(x))^{2+\epsilon}}.$$

In Section 5.2 we prove the existence of a sequence of good quadratic approximations to zero for $q \in \Upsilon$. Section 5.3 proves that the sequence described in Section 5.2 is the best possible. In Section 5.4 we show that $D(q) < 1$ for all $q \in \Upsilon$, where $D(q)$ is a factor that indicates how good an approximation the sequence of quadratics gives. Section 5.5 determines a bound on the height of a quadratics $P(x)$ (dependent on ϵ) after which there are no solutions to $P(q) \leq \frac{1}{H(P(x))^{2+\epsilon}}$. Lastly Section 5.6 gives some conclusions and lists some open problems.

5.2 Upper Bound for a Sequence of Quadratics

This section proves the existence of a sequence of quadratics that give good approximations to zero when evaluated at q . First we need to define $D(q)$. Here τ is the golden ratio.

Definition 5.2 ($D(q)$). *Let $q \in \Upsilon$, with conjugates q_1 and q_2 . If $q > \tau$ then:*

$$D(q) = \min \left\{ \frac{q^4}{(q_1q + q + q_1)(q_2q + q + q_2)(q - q_1)(q - q_2)}, \frac{q^4}{(q_1q - q - q_1)(q_2q - q - q_2)(q - q_1)(q - q_2)} \right\}$$

and if $q < \tau$ then:

$$D(q) = \min \left\{ \frac{q^4}{(q_1q + q + q_1)(q_2q + q + q_2)(q - q_1)(q - q_2)}, \frac{q^4}{(q_1q - q - q_1)(q_2q - q - q_2)(q - q_1)(q - q_2)}, \frac{q^2}{(q_1q + 1)(q_2q + 1)(q - q_1)(q - q_2)} \right\}.$$

It is easy to observe that $D(q)$ is always a positive real number. For convenience we denote:

$$D_1(q) = \frac{q^4}{(q_1q + q + q_1)(q_2q + q + q_2)(q - q_1)(q - q_2)}$$

and

$$D_2(q) = \frac{q^4}{(q_1q - q - q_1)(q_2q - q - q_2)(q - q_1)(q - q_2)}.$$

Theorem 5.1. *If $q \in \Upsilon$ then for all $\epsilon > 0$ there exists a sequence of integer quadratics $P_n(x) = A_nx^2 + B_nx + C_n$ such that:*

$$|P_n(q)| \leq \frac{D(q) + \epsilon}{H(P_n(x))^2}.$$

Proof. Let $q \in \Upsilon$ and $p(x)$ be the minimal polynomial of q . Notice that $p(x)$ is of the form $x^3 + k_2x^2 + k_1x - 1$. We notice in $\mathbb{Z}[x]/p(x)$, that $x(x^2 + k_2x + k_1) = 1$, or equivalently $\frac{1}{x}$ is in $\mathbb{Z}[x]/p(x)$. So we can define $P_n(x) := A_nx^2 + B_nx + C_n := \frac{1}{x^n}$ in the ring $\mathbb{Z}[x]/p(x)$. From a standard result of recurrence relations, (see for instance [22]), A_n, B_n and C_n can be written as:

$$A_n = a_1\beta^n + a_2\gamma^n + a_3\bar{\gamma}^n \tag{5.1}$$

$$B_n = b_1\beta^n + b_2\gamma^n + b_3\bar{\gamma}^n \tag{5.2}$$

$$C_n = c_1\beta^n + c_2\gamma^n + c_3\bar{\gamma}^n \tag{5.3}$$

where $\beta = \frac{1}{q}$ and where γ and $\bar{\gamma}$ are the conjugates of β . Explicitly solving for a_2, a_3, b_2, b_3, c_2 and c_3 in equations (5.1), (5.2) and (5.3) gives:

$$a_2 = \frac{-1}{(q_1 - q_2)(q - q_1)} \tag{5.4}$$

$$a_3 = \frac{1}{(q_1 - q_2)(q - q_2)} \tag{5.5}$$

$$b_2 = \frac{(q + q_2)}{(q_1 - q_2)(q - q_1)} \tag{5.6}$$

$$b_3 = \frac{-(q + q_1)}{(q_1 - q_2)(q - q_2)} \tag{5.7}$$

$$c_2 = \frac{-qq_2}{(q_1 - q_2)(q - q_1)} \quad (5.8)$$

$$c_3 = \frac{qq_1}{(q_1 - q_2)(q - q_2)} \quad (5.9)$$

where q_1 and q_2 are the conjugates of q . It is worth noting that $a_2 = \bar{a}_3$, $b_2 = \bar{b}_3$ and $c_2 = \bar{c}_3$. As $\beta < 1$, for any $\epsilon > 0$ there exists an N , such that for all $n \geq N$ we have $|a_1\beta^n|, |b_1\beta^n|, |c_1\beta^n| < \epsilon$. (Thus, we can now ignore β .) Notice that if $\left(\frac{\gamma}{|\gamma|}\right)^n = 1$ then $\left(\frac{\bar{\gamma}}{|\gamma|}\right)^n = 1$ which would imply $\gamma^n = (\bar{\gamma})^n \in \mathbb{R}$. As the number of conjugate of γ^n is strictly less than the number of conjugate of γ , and the number of conjugates of γ^n must divide the number of conjugates of γ we see that $\gamma^n \in \mathbb{Z}$. But we know that the norm of γ is 1 so γ^n must have norm 1. This contradicts $|\gamma| > 1$, which comes from the definition of γ . Thus we know that $\frac{\gamma}{|\gamma|}$ is not a cyclotomic number, and hence we can get arbitrarily close to any angle in the complex plane with powers of γ . Thus for $n \geq N$ the height of $P_n(x)$ can be written as:

$$\begin{aligned} H(P_n(x)) &= \max\{|A_n|, |B_n|, |C_n|\} \\ &\leq \max\{|a_2\gamma^n + a_3\bar{\gamma}^n|, |b_2\gamma^n + b_3\bar{\gamma}^n|, |c_2\gamma^n + c_3\bar{\gamma}^n|\} + \epsilon \\ &\leq |\gamma^n| \max\left\{\left|a_2\left(\frac{\gamma}{|\gamma|}\right)^n + a_3\left(\frac{\bar{\gamma}}{|\gamma|}\right)^n\right|, \right. \\ &\quad \left. \left|b_2\left(\frac{\gamma}{|\gamma|}\right)^n + b_3\left(\frac{\bar{\gamma}}{|\gamma|}\right)^n\right|, \left|c_2\left(\frac{\gamma}{|\gamma|}\right)^n + c_3\left(\frac{\bar{\gamma}}{|\gamma|}\right)^n\right|\right\} + \epsilon \\ &\leq |\gamma^n| \max\{|a_2\alpha^n + a_3\bar{\alpha}^n|, |b_2\alpha^n + b_3\bar{\alpha}^n|, |c_2\alpha^n + c_3\bar{\alpha}^n|\} + \epsilon. \end{aligned}$$

Here $\alpha := \frac{\gamma}{|\gamma|}$ is a non-cyclotomic complex number of modulus 1. Thus for any $\epsilon > 0$ we can choose a subsequence of the n , say n_i , such that

$$H(P_{n_i}(x)) \leq |\gamma^{n_i}| \left(\min_{|\theta|=1} \{ \max\{|a_2\theta + a_3\bar{\theta}|, |b_2\theta + b_3\bar{\theta}|, |c_2\theta + c_3\bar{\theta}|\} \} + \epsilon \right)$$

Define $U(q) := \min_{|\theta|=1} \{ \max\{|a_2\theta + a_3\bar{\theta}|, |b_2\theta + b_3\bar{\theta}|, |c_2\theta + c_3\bar{\theta}|\} \}^2$. (We later show that $U(q) \leq D(q)$.) On noticing that $|\gamma|^2 = q$ and that $U(q)$ is a finite number, we can redefine ϵ to get:

$$H(P_{n_i}(x))^2 \leq (U(q) + \epsilon)|\gamma|^{2n_i},$$

$$\begin{aligned} H(P_{n_i}(x))^2 &\leq (U(q) + \epsilon)q^{n_i}, \\ \frac{1}{q^{n_i}} &\leq \frac{U(q) + \epsilon}{H(P_{n_i}(x))^2}, \\ |P_{n_i}(q)| &\leq \frac{U(q) + \epsilon}{H(P_{n_i}(x))^2}. \end{aligned}$$

Consider the minimization:

$$\min_{|\theta|=1} \{\max\{|a_2\theta + a_3\bar{\theta}|, |b_2\theta + b_3\bar{\theta}|, |c_2\theta + c_3\bar{\theta}|\}\}^2. \quad (5.10)$$

Consider a typical term $|a_2\theta + a_3\bar{\theta}|$. Let us consider this term as a function of θ . By noticing that $a_2 = \bar{a}_3$ we can rewrite this as $|2\Re(a_2\theta)|$. If we define the argument of a_2 as $e^{\arg(a_2)i} = \frac{a_2}{|a_2|}$. Then we can rewrite this term as $2|\cos(\arg(a_2) + \arg(\theta))|$. We see that this function has two local minima for $-\pi < \arg(\theta) \leq \pi$. At both of these minima, the function has a value of 0. As $\arg(a_2)$, $\arg(b_2)$ and $\arg(c_2)$ are all distinct, we have that this minimization in equation 5.10 cannot occur at a local minimum of any of these terms. This means that the minimization in equation 5.10 will be satisfied for some θ when two of the values within the maximization are equal.

We now show that for $q > \tau$ the solution to

$$\min_{|\theta|=1} \{\max\{|a_2\theta + a_3\bar{\theta}|, |b_2\theta + b_3\bar{\theta}|, |c_2\theta + c_3\bar{\theta}|\}\}^2$$

occurs when $|b_2\theta + b_3\bar{\theta}| = |c_2\theta + c_3\bar{\theta}|$ and $|a_2\theta + a_3\bar{\theta}| < |b_2\theta + b_3\bar{\theta}|$. This is done by solving for θ such that $|b_2\theta + b_3\bar{\theta}| = |c_2\theta + c_3\bar{\theta}|$ and showing that $|a_2\theta + a_3\bar{\theta}| < |b_2\theta + b_3\bar{\theta}|$ is a consequence of this. We see that $a_2\theta + a_3\bar{\theta}$, $b_2\theta + b_3\bar{\theta}$ and $c_2\theta + c_3\bar{\theta}$ are all real. Thus there are two cases to solve for when $|b_2\theta + b_3\bar{\theta}| = |c_2\theta + c_3\bar{\theta}|$. The first is $b_2\theta + b_3\bar{\theta} = c_2\theta + c_3\bar{\theta}$ and the second is $b_2\theta + b_3\bar{\theta} = -c_2\theta - c_3\bar{\theta}$.

Consider the case when $b_2\theta + b_3\bar{\theta} = c_2\theta + c_3\bar{\theta}$. We use the values for a_2, a_3, b_2, b_3, c_2 and c_3 given in equations (5.4) through (5.9). Solving for $(b_2\theta + b_3\bar{\theta})^2$ (via Maple [19]) given that $b_2\theta + b_3\bar{\theta} = c_2\theta + c_3\bar{\theta}$ with $|\theta| = 1$ yields:

$$(b_2\theta + b_3\bar{\theta})^2 = \frac{q^4}{(q_1q + q + q_1)(q_2q + q + q_2)(q - q_1)(q - q_2)} = D_1(q).$$

Furthermore, for this θ we arrive at:

$$E_1(q) := (a_2\theta + a_3\bar{\theta})^2 = \frac{(q+1)^2}{(q_1q + q + q_1)(q_2q + q + q_2)(q - q_1)(q - q_2)}.$$

Next from:

$$\frac{E_1(q)}{D_1(q)} = \frac{(q+1)^2}{q^4}$$

we see that for $q > \tau$ that $D_1(q) > E_1(q)$ and hence $D_1(q) \geq U(q)$.

Now consider the case when $b_2\theta + b_3\bar{\theta} = -c_2\theta - c_3\bar{\theta}$. We use the values for a_2, a_3, b_2, b_3, c_2 and c_3 given in equations (5.4) through (5.9). Solving for $(b_2\theta + b_3\bar{\theta})^2$ (again via Maple [19]) given that $b_2\theta + b_3\bar{\theta} = -c_2\theta - c_3\bar{\theta}$ where $|\theta| = 1$ yields:

$$(b_2\theta + b_3\bar{\theta})^2 = \frac{q^4}{(q_1q - q - q_1)(q_2q - q - q_2)(q - q_1)(q - q_2)} = D_2(q).$$

Furthermore, for this θ we arrive at:

$$E_2(q) := (a_2\theta + a_3\bar{\theta})^2 = \frac{(q-1)^2}{(q_1q - q - q_1)(q_2q - q - q_2)(q - q_1)(q - q_2)}.$$

On noticing that:

$$\frac{E_2(q)}{D_2(q)} = \frac{(q-1)^2}{q^4}$$

we see that $D_2(q) > E_2(q)$ for $q > 1$ and hence $D_2(q) \geq U(q)$.

So for $q > \tau$ we have $D_1(q) \geq U(q)$ and $D_2(q) \geq U(q)$, hence:

$$U(q) \leq \min\{D_1(q), D_2(q)\} = D(q).$$

For $q < \tau$ there are only two cases to consider. If q is the first cubic Pisot number, satisfying $x^3 - x - 1$, then:

$$U(q) = D(q) = \frac{q^4}{(q_1q + q + q_1)(q_2q + q + q_2)(q - q_1)(q - q_2)} = 0.3003453559.$$

If q is the second cubic Pisot number in Υ , satisfying $x^3 - x^2 - 1$, then:

$$U(q) = D(q) = \frac{q^2}{(q_1q + 1)(q_2q + 1)(q - q_1)(q - q_2)} = 0.3429099932.$$

Thus we see that $U(q) \leq D(q)$ and the result follows. ■

5.3 Lower Bound for all Quadratics

This section shows that the sequence of P_n described in Section 5.2 is the best possible.

Theorem 5.2. *If $q \in \Upsilon$ and $\epsilon > 0$, then $|P(q)| \geq \frac{D(q)-\epsilon}{H(P(x))^2}$, except for a finite number of quadratic polynomials with integer coefficients $P(x)$, where H is the height function, and $D(q)$ is given in Definition 5.2 .*

Proof. Let $P(x)$ be a quadratic polynomial with integer coefficients, $q \in \Upsilon$, and q_1 and $q_2 = \bar{q}_1$ the two conjugates of q . As $P(x)$ is a quadratic polynomial and q is of degree 3 we know that $P(q) \neq 0$. As q is an algebraic integer, $|P(q)P(q_1)P(q_2)| \geq 1$ and hence:

$$|P(q)| \geq \left| \frac{1}{P(q_1)P(q_2)} \right|.$$

Since $P(q_1) = \overline{P(q_2)}$ we see that $P(q_1)P(q_2)$ is a positive real number, hence:

$$|P(q)| \geq \frac{1}{P(q_1)P(q_2)}.$$

We can rewrite this as:

$$|P(q)| \geq \frac{1}{H(P(x))^2 \max\{S(q_1)S(q_2) : H(S(x)) \leq 1\}}$$

where $S(x)$ is a quadratic polynomial with real coefficients. We know from Theorem 5.1 that we can find quadratic polynomials that evaluate arbitrarily close to zero at q . Thus, except for finitely many exceptions:

$$|P(q)| \geq \frac{1 - \epsilon_0}{H(P(x))^2 \max\{S(q_1)S(q_2) : H(S(x)) \leq 1, S(q) = 0\}}, \quad (5.11)$$

where $S(x)$ is a quadratic polynomial with real coefficients.

The problem of finding the maximum in equation (5.11) is the same as the problem of finding the maximum magnitude on a convex polytope, and thus the maximal value is attained on an extreme point (i.e. a corner point). We examine the equation:

$$\max\{S(q_1)S(q_2) : H(S(x)) \leq 1, S(q) = 0, S(x) \text{ is an extreme point}\}, \quad (5.12)$$

more carefully. We see that $S(x)$ is an extreme point of this convex polytope if and only if $S(x)$ is of the form $S(x) = ax^2 + bx + c$ where:

1. at least two of a , b and c are ± 1 ,
2. $S(q) = 0$,
3. $-1 \leq a, b, c \leq 1$.

We can eliminate half of the cases by noting the symmetry $|S(q)| = |-S(q)|$. We examine the six remaining cases for a , b and c , when at least two of these three variables are of ± 1 .

Case 1: Let $S(x) = x^2 + x + c$. By requirement 2 we deduce that $c = -q^2 - q$. Next we see that $-q^2 - q < -1$ which violates requirement 3. Hence, this is not a possible value for $S(x)$.

Case 2: Let $S(x) = x^2 - x + c$. By requirement 2 we deduce that $c = -q^2 + q$. By requirement 3 we deduce that $-1 \leq -q^2 + q \leq 1$. Thus, we have that $q \leq \tau$. Hence, for $q > \tau$, this is not a possible value for $S(x)$.

Case 3: Let $S(x) = x^2 + bx + 1$. By requirement 2 we deduce that $b = -q - \frac{1}{q}$. Next we see that $-q - \frac{1}{q} < -1$ which violates requirement 3. Hence, this is not a possible value for $S(x)$.

Case 4: Let $S(x) = x^2 + bx - 1$. By requirement 2 we deduce that $b = -q + \frac{1}{q}$. By requirement 3 we have that $-1 \leq -q + \frac{1}{q} \leq 1$. Thus, we have that $q \leq \tau$. Hence, for $q > \tau$, this is not a possible value for $S(x)$.

Case 5: Let $S(x) = ax^2 + x + 1$. By requirement 2 we deduce that $a = -\frac{1}{q} - \frac{1}{q^2}$. By requirement 3 we have that $-1 \leq -\frac{1}{q} - \frac{1}{q^2} \leq 1$. Thus, we have that $q \geq \tau$. Hence, for $q < \tau$, this is not a possible value for $S(x)$.

Case 6: Let $S(x) = ax^2 + x - 1$. By requirement 2 we deduce that $a = -\frac{1}{q} + \frac{1}{q^2}$. By requirement 3 we have that $-1 \leq -\frac{1}{q} + \frac{1}{q^2} \leq 1$. This is always true if $q > 1$.

If $q > \tau$ then cases 5 and 6 give that equation (5.12) is equal to:

$$\max \left\{ \left| \left(-\frac{1}{q} + \frac{1}{q^2} \right) q_1^2 + q_1 - 1 \right|^2, \left| \left(-\frac{1}{q} - \frac{1}{q^2} \right) q_1^2 + q_1 + 1 \right|^2 \right\}.$$

If $q \leq \tau$ then cases 2, 4 and 6 give that equation (5.12) is equal to:

$$\max \left\{ \left| \left(-\frac{1}{q} + \frac{1}{q^2} \right) q_1^2 + q_1 - 1 \right|^2, \left| q_1^2 + \left(-q + \frac{1}{q} \right) q_1 - 1 \right|^2, \right. \\ \left. |q_1^2 - q_1 + (-q^2 + q)|^2 \right\}.$$

Or equivalently, taking $\epsilon_0 = \frac{\epsilon}{D(q)}$ and $q > \tau$ we have:

$$\begin{aligned} |P(q)| &\geq \frac{1 - \epsilon_0}{H(P(x))^2 \max\{S(q_1)S(q_2) : H(S(x)) \leq 1, S(q) = 0\}}, \\ &= \frac{1}{\max \left\{ \left| \left(-\frac{1}{q} + \frac{1}{q^2} \right) q_1^2 + q_1 - 1 \right|^2, \left| \left(-\frac{1}{q} - \frac{1}{q^2} \right) q_1^2 + q_1 + 1 \right|^2 \right\}}, \\ &\quad \times \frac{1 - \epsilon_0}{H(P(x))^2}, \\ &= \frac{1 - \epsilon_0}{H(P(x))^2} \min \left\{ \frac{q^4}{(q_1q - q - q_1)(q_2q - q - q_2)(q - q_1)(q - q_2)}, \right. \\ &\quad \left. \frac{q^4}{(q_1q + q + q_1)(q_2q + q + q_2)(q - q_1)(q - q_2)} \right\}, \\ &= \frac{D(q) - \epsilon}{H(P(x))^2}. \end{aligned}$$

Using a similar method, if $q \leq \tau$ then:

$$\begin{aligned} |P(q)| &\geq \frac{1 - \epsilon_0}{H(P(x))^2 \max\{S(q_1)S(q_2) : H(S(x)) \leq 1, S(q) = 0\}}, \\ &= \frac{1 - \epsilon_0}{H(P(x))^2} \min \left\{ \frac{q^4}{(q_1q - q - q_1)(q_2q - q - q_2)(q - q_1)(q - q_2)}, \right. \\ &\quad \frac{q^2}{(q_1q + 1)(q_2q + 1)(q - q_1)(q - q_2)}, \\ &\quad \left. \frac{1}{(q + q_1 - 1)(q + q_2 - 1)(q - q_1)(q - q_2)} \right\}, \\ &= \frac{D(q) - \epsilon}{H(P(x))^2}. \end{aligned}$$

Thus:

$$|P(q)| \geq \frac{D(q) - \epsilon}{H(P(x))^2}$$

for all but a finite number of quadratic polynomials with integer coefficients $P(x)$, which is the desired result. ■

It is worth pointing out that when $q > \tau$, cases of $D(q) = D_1(q)$ and $D(q) = D_2(q)$ both occur, as demonstrated in Table 5.3. This table lists all $q \in \Upsilon$, $q \in (\tau, 5)$, along with their minimal polynomials, $D(q)$, and whether $D(q) = D_2(q)$. These Pisot numbers are found using the techniques of David Boyd, as described in [7]. In the next section we discuss when $D(q) = D_1(q)$ and when $D(q) = D_2(q)$, as well as showing that $D(q) < 1$.

5.4 Proof that $D(q) < 1$

From Table 5.3 we make the observation that $D(q) < 1$ for all $q \in \Upsilon$, $q \in (1, 5)$. This section proves for all $q \in \Upsilon$ that $D(q) < 1$.

Theorem 5.3. *If $q \in \Upsilon$ then $D(q) < 1$. Furthermore, $D(q) \rightarrow 1$ as $q \rightarrow \infty$.*

Proof. There are only two cases of $q \in \Upsilon$ where $q < \tau$. We compute these cases explicitly. For the first cubic Pisot number q , satisfying $x^3 - x - 1$, we get $D(q) = 0.3003453559 < 1$. For the second cubic Pisot number q , satisfying $x^3 - x^2 - 1$, we get $D(q) = 0.3429099932 < 1$.

So, without loss of generality, we assume that $q > \tau$. Thus we have that:

$$\begin{aligned} D(q) &= \min \left\{ \frac{q^4}{(q_1q + q + q_1)(q_2q + q + q_2)(q - q_1)(q - q_2)}, \right. \\ &\quad \left. \frac{q^4}{(q_1q - q - q_1)(q_2q - q - q_2)(q - q_1)(q - q_2)} \right\} \\ &= \min\{D_1(q), D_2(q)\} \end{aligned}$$

| Minimal polynomial | Pisot number q | $D(q)$ | $D_1(q)$ or $D_2(q)$ |
|-----------------------|------------------|--------------|----------------------|
| $x^3 - 2x^2 + x - 1$ | 1.754877666 | 0.3429800030 | $D_1(q)$ |
| $x^3 - x^2 - x - 1$ | 1.839286755 | 0.4133318671 | $D_2(q)$ |
| $x^3 - x^2 - 2x - 1$ | 2.147899036 | 0.3501354747 | $D_2(q)$ |
| $x^3 - 2x^2 - 1$ | 2.205569430 | 0.5080747995 | $D_1(q)$ |
| $x^3 - 3x^2 + 2x - 1$ | 2.324717957 | 0.4453345199 | $D_1(q)$ |
| $x^3 - 2x^2 - x - 1$ | 2.546818277 | 0.5309353500 | $D_2(q)$ |
| $x^3 - 3x^2 + x - 1$ | 2.769292354 | 0.5232405572 | $D_1(q)$ |
| $x^3 - 2x^2 - 2x - 1$ | 2.831177207 | 0.4435746902 | $D_2(q)$ |
| $x^3 - 2x^2 - 3x - 1$ | 3.079595623 | 0.3847011683 | $D_2(q)$ |
| $x^3 - 3x^2 - 1$ | 3.103803403 | 0.6573022517 | $D_1(q)$ |
| $x^3 - 4x^2 + 3x - 1$ | 3.147899036 | 0.4911183769 | $D_1(q)$ |
| $x^3 - 3x^2 - x - 1$ | 3.382975768 | 0.6198187695 | $D_2(q)$ |
| $x^3 - 4x^2 + 2x - 1$ | 3.511547142 | 0.5392925792 | $D_1(q)$ |
| $x^3 - 3x^2 - 2x - 1$ | 3.627365085 | 0.5238048354 | $D_2(q)$ |
| $x^3 - 4x^2 + x - 1$ | 3.806300717 | 0.6215030654 | $D_1(q)$ |
| $x^3 - 3x^2 - 3x - 1$ | 3.847322102 | 0.4565756315 | $D_2(q)$ |
| $x^3 - 4x^2 - 1$ | 4.060647028 | 0.7429688162 | $D_1(q)$ |
| $x^3 - 5x^2 + 4x - 1$ | 4.079595623 | 0.5016713072 | $D_1(q)$ |
| $x^3 - 4x^2 - x - 1$ | 4.287625262 | 0.6821435669 | $D_2(q)$ |
| $x^3 - 5x^2 + 3x - 1$ | 4.365230013 | 0.5427050340 | $D_1(q)$ |
| $x^3 - 4x^2 - 2x - 1$ | 4.494492837 | 0.5872601546 | $D_2(q)$ |
| $x^3 - 5x^2 + 2x - 1$ | 4.613470268 | 0.6026317629 | $D_1(q)$ |
| $x^3 - 4x^2 - 3x - 1$ | 4.685779526 | 0.5175073389 | $D_2(q)$ |
| $x^3 - 5x^2 + x - 1$ | 4.835975919 | 0.6842306666 | $D_1(q)$ |
| $x^3 - 4x^2 - 4x - 1$ | 4.864536512 | 0.4639992754 | $D_2(q)$ |

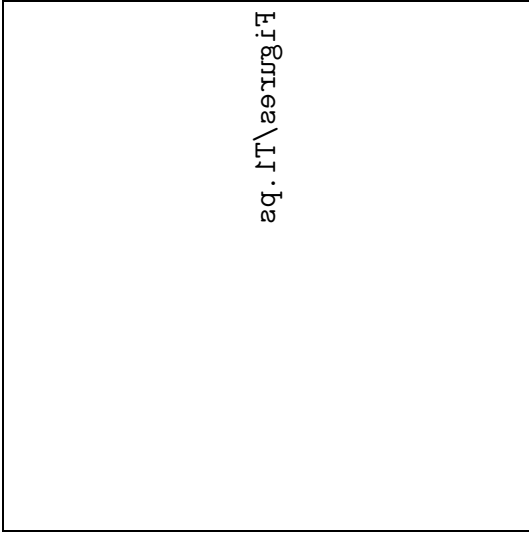
Table 5.3: $D(q)$ for various q

Next let us consider $T(x, y) := \frac{1}{D(q)}$, where q_1 is $x + yi$, q_2 is $x - yi$ and q is $\frac{1}{q_1 q_2} = \frac{1}{x^2 + y^2}$. Simplifying $T(x, y)$ (via Maple) gives:

$$\begin{aligned}
T(x, y) := & \max \{ x^{12} + 6x^{10}y^2 + 15x^8y^4 + 20x^6y^6 + 15x^4y^8 + 6x^2y^{10} \\
& + y^{12} + 2x^{10} + 10x^8y^2 + 20x^6y^4 + 20x^4y^6 + 10x^2y^8 + 2y^{10} \\
& + x^8 + 4x^6y^2 + 6x^4y^4 + 4x^2y^6 + y^8 - 2x^7 - 6x^5y^2 - 6x^3y^4 \\
& - 2xy^6 - 2x^6 - 2x^4y^2 + 2x^2y^4 + 2y^6 - 2x^5 - 4x^3y^2 - 2xy^4 \\
& - 2x^4 + 2y^4 + x^2 + y^2 + 2x + 1, \\
& x^{12} + 6x^{10}y^2 + 15x^8y^4 + 20x^6y^6 + 15x^4y^8 + 6x^2y^{10} + y^{12} \\
& - 2x^{10} - 10x^8y^2 - 20x^6y^4 - 20x^4y^6 - 10x^2y^8 - 2y^{10} + x^8 \\
& + 4x^6y^2 + 6x^4y^4 + 4x^2y^6 + y^8 + 2x^7 + 6x^5y^2 + 6x^3y^4 + 2xy^6 \\
& - 2x^6 - 2x^4y^2 + 2x^2y^4 + 2y^6 - 2x^5 - 4x^3y^2 - 2xy^4 + 2x^4 \\
& - 2y^4 + x^2 + y^2 - 2x + 1 \}.
\end{aligned}$$

For convenience label the first polynomial as $T_1(x, y) = \frac{1}{D_1(q)}$ and the second polynomial as $T_2(x, y) = \frac{1}{D_2(q)}$. As we are assuming that $q > \tau$, we have that $-\sqrt{\frac{1}{\tau}} \leq x \leq \sqrt{\frac{1}{\tau}}$ and $-\sqrt{\frac{1}{\tau}} \leq y \leq \sqrt{\frac{1}{\tau}}$.

Consider $T_1(x, y)$ on the region $0 \leq x \leq \sqrt{\frac{1}{\tau}}$ and $-\sqrt{\frac{1}{\tau}} \leq y \leq \sqrt{\frac{1}{\tau}}$. On the line $x = \sqrt{\frac{1}{\tau}}$, there is a local minimum at $y = 0$ of approximately 1.364498234. On the line $y = -\sqrt{\frac{1}{\tau}}$, there are no local minima or maxima, but the boundary value of $x = 0$ gives approximately 3.236067980, and the other boundary value of $x = \sqrt{\frac{1}{\tau}}$ gives approximately 17.70723545. The line $y = \sqrt{\frac{1}{\tau}}$ is symmetric to the line $y = -\sqrt{\frac{1}{\tau}}$. On the line $x = 0$, there is only one local minimum at $y = 0$, giving the value of 1. On the interior of this region there are no local minima or maxima (there is a saddle point at $y = 0$, $x = 0.5550360821$ of approximately 2.047461577). Thus, $T_1(x, y)$ is always greater than or equal to 1 on this region, and is equal to 1 only at $x = y = 0$. (See Figure 5.1.)

Figure 5.1: $T_1(x, y)$ on the region of $x \geq 0$

Consider $T_2(x, y)$ on the region $-\sqrt{\frac{1}{\tau}} \leq x \leq 0$ and $-\sqrt{\frac{1}{\tau}} \leq y \leq \sqrt{\frac{1}{\tau}}$. On the line $x = -\sqrt{\frac{1}{\tau}}$, there is a local minimum at $y = 0$ of approximately 3.732814929. On the line $y = -\sqrt{\frac{1}{\tau}}$, there are no local minima or maxima (in the correct range), but the boundary value of $x = 0$ gives approximately 1.347524158, and the other boundary value of $x = -\sqrt{\frac{1}{\tau}}$ gives approximately 3.371362832. The line $y = \sqrt{\frac{1}{\tau}}$ is symmetric to the line $y = -\sqrt{\frac{1}{\tau}}$. On the line $x = 0$, there is only one local minimum at $y = 0$, giving the value of 1. On the interior of this region there are no local minima or maxima. Thus, $T_2(x, y)$ is always greater than or equal to 1 on this region, and is equal to 1 only at $x = y = 0$. (See Figure 5.2.)

Thus, $T_1(x, y)$ and $T_2(x, y)$ are strictly greater than 1 on their respective regions, except at the point $x = y = 0$. Thus, as we take the maximum of $T_1(x, y)$ and $T_2(x, y)$, $T(x, y) > 1$ as x and y cannot both equal 0. Furthermore, we see that if $q \rightarrow \infty$, then $q_1 \rightarrow 0$, which implies that $x \rightarrow 0$ and $y \rightarrow 0$. Thus, as $q \rightarrow \infty$ we have $T_1(x, y) \rightarrow 1$ and $T_2(x, y) \rightarrow 1$, and hence $T(x, y) \rightarrow 1$.

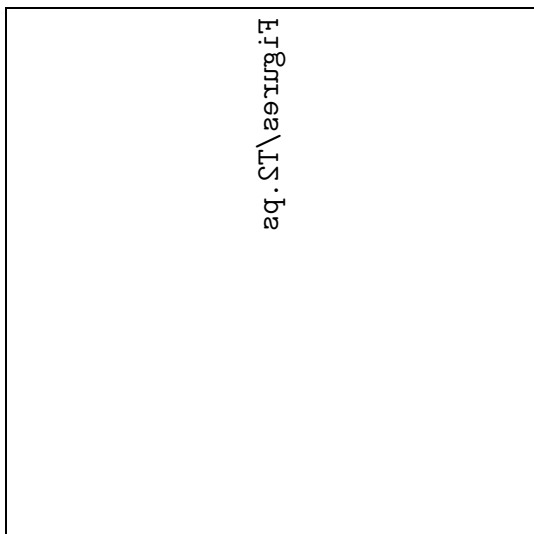


Figure 5.2: $T_2(x, y)$ on the region of $x \leq 0$

Therefore $D(q) < 1$, and further as $q \rightarrow \infty$ then $D(q) \rightarrow 1$.

■

In Table 5.3 we indicate when $D(q) = D_2(q)$. (When $D(q) \neq D_2(q)$ and $q > \tau$ then $D(q) = D_1(q)$.) Figure 5.3 gives a graphical interpretation of when $D(q) = D_2(q)$. The value $D(q)$ takes depends on the location of its conjugate q_1 in the complex plane. If q_1 is in the interior of the convex curve (where the real part is less than zero), then $D(q) = D_1(q)$, otherwise $D(q) = D_2(q)$.

5.5 Bounds for the Height with Respect to ϵ

Let $q \in \Upsilon$. We know from [33] that:

$$|P(q)| \leq \frac{1}{H(P(x))^{2+\epsilon}}$$

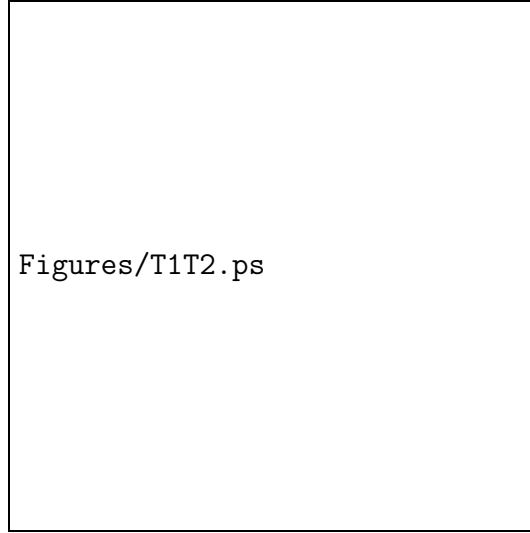


Figure 5.3: Regions where $D_1(q)$ and $D_2(q)$ are minimal

has only finitely many integer quadratic solutions. In this section we find an upper bound for $H(P(x))$ (dependent on ϵ) for all of these solutions.

We see from Theorem 5.2 that $|P(q)| \geq \frac{D(q)-\epsilon}{H(P(x))^2}$ in all but finitely many cases. Let us re-examine the proof of Theorem 5.2 to find a lower bound for $|P(q)|$ that holds in all cases. By not binding $S(q)$ to zero, we can re-write equation (5.11) (page 61) as:

$$|P(q)| \geq \frac{1}{H(P(x))^2 \max\{S(q_1)S(q_2) : H(S(x)) \leq 1, \deg(S(x)) \leq 2\}}.$$

This is true for *all* integer quadratics $P(x)$. Again, we notice $S(x)$ must be an extreme point, hence this can be rewritten as

$$|P(q)| \geq \frac{1}{H(P(x))^2 \max\{S(q_1)S(q_2) : S(x) = \pm x^2 \pm x \pm 1\}}.$$

So we see that there is a solution only if:

$$\begin{aligned} \frac{1}{H(P(x))^{2+\epsilon}} &\geq |P(q)|, \\ &\geq \frac{1}{H(P(x))^2 \max\{S(q_1)S(q_2) : S(x) = \pm x^2 + \pm x + \pm 1\}}. \end{aligned}$$

From which it follows that:

$$\frac{1}{H(P(x))^\epsilon} \geq \frac{1}{\max\{S(q_1)S(q_2) : S(x) = \pm x^2 + \pm x + \pm 1\}}.$$

By taking the reciprocal we get:

$$H(P(x))^\epsilon \leq \max\{S(q_1)S(q_2) : S(x) = \pm x^2 + \pm x + \pm 1\},$$

which yields:

$$H(P(x)) \leq (\max\{S(q_1)S(q_2) : S(x) = \pm x^2 + \pm x + \pm 1\})^{1/\epsilon}.$$

This proves the following theorem:

Theorem 5.4. *If $q \in \Upsilon$ with $q_1 = \bar{q}_2$ the conjugates of q and*

$$H(P(x)) > (\max\{S(q_2)S(q_1) : S(x) = \pm x^2 \pm x \pm 1\})^{1/\epsilon}$$

then there are no integer quadratic solutions $P(x)$ to

$$|P(q)| \leq \frac{1}{H(P(x))^{2+\epsilon}}.$$

5.6 Further Research

Recall from Section 5.1 that it appears that $l^m(q) = P_n(q)$ for the first and second cubic Pisot numbers. Even though this chapter has demonstrated a simple relationship between $P_n(q)$ and the minimal integer quadratic approximation to 0 for $q \in \Upsilon$, the initial reason for this investigation is still open. Some further questions to consider are:

1. Does $l^m(q) = P_n(q)$ for the first and second cubic Pisot numbers?

2. If so, what is the relationship between q , n and m in the equation $P_n(q) = l^m(q)$?
3. Is this pattern true to a limited extent for any of the other Pisot numbers q in Υ ? (This is suggested by some calculations.)

Chapter 6

Some Conclusions and Open Questions

Please don't take it amiss, good sirs, if there are more mistakes in this little book than there are grey hairs on my old head. What can I do? I've never had much to do with book-learning and the like before. May the fellow who dreamed it all up choke on his porridge! As you stare at those letters they start to look the same. Your eyes cloud over, just like someone had scattered grain all over the page.

See how many misprints I've found! All I ask, if you find any of them, is that you pay no attention, and read them as if they were spelt correctly.

Village Evenings near Dikanka – Nikolai Gogol

6.1 Open questions

This thesis answers a number of questions concerning various spectra of real numbers, with particular emphasis on Pisot numbers. Many of these answers lead to new questions. The main unproven conjecture in this area of research is:

Conjecture 1. For $q \in (1, 2)$, $l(q) > 0$ if and only if q is a Pisot number.

As mentioned in Chapter 4, if a lemma similar to Lemma 4.3 (page 41) could be found that would work for all polynomials $p(x)$ where $p(0) = \pm 1$, regardless of the degree of $p(x)$ or whether $p(x)$ has a Pisot number as a root or not, then this lemma could be used to prove Conjecture 1. The second part of this lemma, which comes from Lemma 4.1, easily extends to an arbitrary degree polynomial, but it is not clear that there exists an algorithm that forces all but d consecutive terms to be integers (where d is the degree of $p(x)$).

To elaborate, let q be a unit quadratic Pisot number with conjugate r and minimal polynomial $p(x)$. Recall Lemma 4.2 (page 40) shows that if $sx + t \equiv \sum \epsilon_i x^i \pmod{p(x)}$ with $\epsilon_i \in \{\pm 1, 0\}$, then the integer pair (s, t) is bounded by the lines $sr + t = \pm c$. Equivalently, we could say that the integer pair (s, t) must be within a finite distance, (dependant on c) from the line $s(1, -r)$ for $s \in \mathbb{R}$. A general result holds true that if $p(x)$ is a irreducible degree d polynomial and $a_{d-1}x^{d-1} + \dots + a_0 \equiv \sum \epsilon_i x^i \pmod{p(x)}$ with $\epsilon_i \in \{\pm 1, 0\}$ then the integer d -tuple (a_{d-1}, \dots, a_0) is bounded to some line $s(b_{d-1}, \dots, b_0)$ for $s \in \mathbb{R}$ if and only if $p(x)$ has a Pisot number as a root.

Thus, if a lemma like Lemma 4.3 could be shown to be true for all polynomials, regardless of degree, or of whether they have a Pisot number as a root or not, it would follow that for any $a_{d-1}q^{d-1} + \dots + a_0 \in \Lambda(q)$ there would be restrictions on a_{d-1}, \dots, a_0 if and only if q is a Pisot number. Hence, it would follow that $l(q) > 0$ if and only if q is a Pisot number.

There are also numerous other questions that are raised:

1. Does there exist an $\alpha \approx 1.95$ such that if $q < \alpha$, and q is a Pisot number, then $0 \in A(q)$? (Page 23.)
2. If $q \in (1, 2)$ and $A(q)$ is discrete, is q a Perron number? (Page 28.)
3. For $q \in (1, 2)$, if $A(q)$ is discrete and the Mahler measure of q is less than 2, then is q either a Salem or Pisot number? (Page 29.)

4. Do the only Salem numbers q , where $A(q)$ is discrete satisfy a polynomial of the form $x^n - x^{n-1} - \dots - x + 1$? (Page 29.)
5. Does there exist an $\alpha \approx 1.72$ such that if $q < \alpha$ and q is not a Pisot number then $A(q)$ is not discrete? (Page 29.)
6. If q is any quadratic Pisot number (including non-unit quadratic Pisot numbers), and $\left\{ \frac{C_k}{D_k} \right\}$ are the best approximations of q , does $l^m(q) = |D_k q - C_k|$, and if so, what is the relationship between k and m ? (Page 48.)
7. For $P_n(q)$ defined in Theorem 5.1 (page 57), does $l^m(q) = P_n(q)$ for the first or second cubic Pisot number? (Page 70.) If so, what is the relationship between n , m and q ? (Page 71.)
8. Is this pattern that $l^m(q) = P_n(q)$ true to a limited extent for any of the other Pisot numbers $q \in \Upsilon$? (Page 71.)

6.2 Generalizations

Consider a ring R and define a Pisot number q over R to be the root of a monic polynomial with coefficients in R , where $|q|$ is greater than 1, and where all the conjugates of q are of modulus strictly less than 1. An example of this would be the root of $x^3 - ix^2 - 1$ (approximately $-0.426114 + 1.31001i$). There are conditions on R that guarantee that all Pisot numbers of this form have discrete spectra. Very little is known about the problem in this setting though, and would an obvious direction for further exploration.

A second way that these results can be generalized is to consider S to be a finite subset of \mathbb{R}^n instead of \mathbb{Z} . In this case, we define

$$\Lambda^S(q) := \left\{ \sum_{i=0}^n q^i s_i : n \in \mathbb{N}, s_i \in S \subset \mathbb{R}^n \right\}$$

where the s_i are vectors and the q^i are scalars. There are conditions for S that guarantee that the spectra are discrete. These conditions though are still not well

understood. Considering S of this form has applications to quasicrystals, as well as to robotics [9, 11]. For example, let:

$$S = \left\{ [1, 0], \left[\cos\left(\frac{2\pi}{5}\right), \sin\left(\frac{2\pi}{5}\right) \right], \left[\cos\left(\frac{4\pi}{5}\right), \sin\left(\frac{4\pi}{5}\right) \right], \right. \\ \left. \left[\cos\left(\frac{6\pi}{5}\right), \sin\left(\frac{6\pi}{5}\right) \right], \left[\cos\left(\frac{8\pi}{5}\right), \sin\left(\frac{8\pi}{5}\right) \right] \right\}.$$

Let τ be the golden ratio. It is provable that $\Lambda^S(\tau)$ has uniformly discrete spectra. Figure 6.1 is a plot of all the points in $\Lambda^S(\tau)$ of norm less than 10.



Figure 6.1: $(x, y) \in \Lambda^S(\tau)$ for $S \subset \mathbb{R}^2$, $\sqrt{x^2 + y^2} \leq 10$

Appendix A

Code

A.1 Data Types

File name: Polynomial.h

```
/*
This is the base class for polynomials.

Copyright (C) 2000 Kevin G Hare

This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
*/

#ifndef _INT_POLY
#define _INT_POLY

#define true 1
#define false 0

#include <cmath>
#include <stream.h>
#define POLY_IS_SHORT true

#ifdef POLY_IS_SHORT
#define PolyType short int
#define MaxPolyType 16384
```

```

#else
#define PolyType int
#define MaxPolyType 1073741824

#endif

// #define PolyType long long
// #define MaxPolyType 9223372036854775807

inline double max(double a, double b) {
    if (a < b) { return(b); } return(a);
};

inline double abs(double a) {
    return(max(a, -a));
};

inline double min(double a, double b) {
    if (a > b) { return(b); } return(a);
};

#ifdef POLY_IS_SHORT
inline int max(int a, int b) {
    if (a < b) { return(b); } return(a);
};
#endif

/*
inline int abs(int a) {
    return(max(a, -a));
};
*/

#ifdef POLY_IS_SHORT
inline int min(int a, int b) {
    if (a > b) { return(b); } return(a);
};
#endif

inline PolyType max(PolyType a, PolyType b) {
    if (a < b) { return(b); } return(a);
};

#ifdef POLY_IS_SHORT
inline PolyType abs(PolyType a) {
    return(max((PolyType)a, (PolyType)-a));
};
#endif

inline PolyType min(PolyType a, PolyType b) {
    if (a > b) { return(b); } return(a);
};

class intPoly {
protected:
    PolyType *poly;
    int deg;

public:
    inline PolyType* coeff() {return(poly);};

```



```

inline int degree() {return(deg);};

/*
intPoly operator=(intPoly poly2);
int operator==(intPoly poly);
int operator>(intPoly poly);
int operator<(intPoly poly);
*/
inline int operator<(intPoly* poly) {
    if (poly->degree() < this->degree()) {
        return(false);
    } else if (poly->degree() > this->degree()) {
        return(true);
    }

    if ((poly->degree() < 0) && (this->degree() < 0)) {
        return(false);
    }

    for(int i = 0; i <= poly->degree(); i++) {
        if (poly->coeff()[i] < this->coeff()[i]) {
            return(false);
        } else if (poly->coeff()[i] > this->coeff()[i]) {
            return(true);
        }
    }
    return(false);
};

inline void read() {
    int i;

    if (deg >= 0) {
        delete poly;
    }
    cout << "What degree is the polynomial" << endl;
    cin >> deg;

    poly = new PolyType[deg+1];

    for(i=0; i <= deg; i++) {
        cout << "Term " << i << ": ";
        cin >> poly[i];
    }
};

inline intPoly() {deg = -1;};

inline intPoly(int degr, PolyType* polyn) {
    int i;
    deg = degr;
    if (deg < 0) {return; }

    poly = new PolyType[(deg) + 1];
    for(i = 0; i <= deg; i++) {
        poly[i] = polyn[i];
    }
    for(i = degr; i >= 0; i--) {
        if (poly[i] == 0) {
            deg = i-1;
        } else {
            break;
        }
    }
};

```

```

    }
}
};

inline int zero() {
    if (deg == 0) {
        if (poly[deg] == 0) {
            return(true);
        }
    } else if (deg < 0) {
        return(true);
    } return(false);
};

inline PolyType height() {
    int i;
    PolyType h;
    PolyType t;
    h = 0;
    for(i = 0 ; i <= deg; i++) {
        t = abs(poly[i]);
        h = max(t, h);
    }
    return(h);
};

inline double eval(double beta) {
    double alpha, betai;
    int i;
    alpha = 0;
    betai = 1;
    for (i=0; i <= deg; i++) {
        alpha = alpha + betai * ((double)(coeff()[i]));
        betai = betai * beta;
    }
    return(alpha);
};

inline int eval(int beta) {
    int alpha, betai;
    int i;
    alpha = 0;
    betai = 1;
    for (i=0; i <= deg; i++) {
        alpha = alpha + betai * ((int)(coeff()[i]));
        betai = betai * beta;
    }
    return(alpha);
};

inline void print() {
    int i;
    if (deg < 0) {
        cout << 0 ;
        return;
    }
    for (i = 0; i <= deg-1; i++) {
        cout << "(" << poly[i] << ")*x" << i << "+";
    };
    cout << "(" << poly[i] << ")*x" << i ;
};

/*

```

```

nextPoly();
*/

inline ~intPoly () {
    if (deg >= 0) {
        delete poly;
    }
};

};

intPoly* divide(intPoly *poly1, intPoly* poly2);
intPoly* add(intPoly* poly1, intPoly* poly2);
intPoly* mult(intPoly* poly1, intPoly* poly2);
intPoly* diff(intPoly* poly1);

//int operator<(intPoly& poly1, intPoly& poly2) {
//    cout << "Did a comparison (YEAH)" << endl;
//    return(true);
//};
#endif

File name: Polynomial.cc

/*
This is the base data structure of polynomials.

Copyright (C) 2000 Kevin G Hare

This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
*/

#include <stream.h>
#include "Polynomial.h"

#define true 1
#define false 0

/*
inline double max(double a, double b) {
    if (a < b) { return(b); } return(a);
};

inline double abs(double a) {
    return(max(a, -a));
};

inline double min(double a, double b) {
    if (a > b) { return(b); } return(a);
};

```

```

inline int max(int a, int b) {
    if (a < b) { return(b); } return(a);
};

inline int abs(int a) {
    return(max(a, -a));
};

inline int min(int a, int b) {
    if (a > b) { return(b); } return(a);
};

inline PolyType max(PolyType a, PolyType b) {
    if (a < b) { return(b); } return(a);
};

inline PolyType abs(PolyType a) {
    return(max(a, -a));
};

inline PolyType min(PolyType a, PolyType b) {
    if (a > b) { return(b); } return(a);
};
*/

/*
intPoly intPoly::operator=(intPoly poly1) {
    intPoly* poly = new intPoly(poly1.degree(), poly1.coeff());
    cout << "It actually called the assignment operator" << endl;
    return(*poly);
}
*/

/*
int intPoly::operator==(intPoly poly) {
    if (poly.degree() != this->degree()) {
        return(false);
    }

    for(int i = 0; i <= poly.degree(); i++) {
        if (poly.coeff()[i] != this->coeff()[i]) {
            return(false);
        }
    }
    return(true);
}

int intPoly::operator>(intPoly poly) {
    return((poly < *(this)));
}
*/

/*
int intPoly::operator<(intPoly* poly) {
    if (poly->degree() < this->degree()) {
        return(false);
    } else if (poly->degree() > this->degree()) {
        return(true);
    }

    if ((poly->degree() < 0) && (this->degree() < 0)) {
        return(false);
    }
}

```

```

    }

    for(int i = 0; i <= poly->degree(); i++) {
        if (poly->coeff()[i] < this->coeff()[i]) {
            return(false);
        } else if (poly->coeff()[i] > this->coeff()[i]) {
            return(true);
        }
    }
    return(false);
}
*/

/*
PolyType* intPoly::coeff() {
    return(poly);
};
*/

/*
int intPoly::degree() {
    return(deg);
};
*/

/*
PolyType intPoly::height() {
    int i;
    PolyType h;
    h = 0;
    for(i = 0 ; i <= deg; i++) {
        h = max(abs(poly[i]), h);
    }
    return(h);
};
*/

/*
double intPoly::eval(double beta) {
    double alpha, betai;
    int i;
    alpha = 0;
    betai = 1;
    for (i=0; i <= deg; i++) {
        alpha = alpha + betai * ((double)(coeff()[i]));
        betai = betai * beta;
    }
    return(alpha);
};
*/

/*
intPoly::read() {
    int i;

    if (deg >= 0) {
        delete poly;
    }
    cout << "What degree is the polynomial" << endl;
    cin >> deg;

    poly = new PolyType[deg+1];
}

```

```

    for(i=0; i <= deg; i++) {
        cout << "Term " << i << ": ";
        cin >> poly[i];
    }
}
*/

/*
intPoly::print() {
    int i;
    if (deg < 0) {
        cout << 0 ;
        return(0);
    }
    for (i = 0; i <= deg-1; i++) {
        cout << "(" << poly[i] << ")*x^" << i << "+" ;
    };
    cout << "(" << poly[i] << ")*x^" << i ;
};
*/

/*
int intPoly::zero() {
    if (deg == 0) {
        if (poly[deg] == 0) {
            return(true);
        }
    } else if (deg < 0) {
        return(true);
    } return(false);
};
*/

/*
intPoly::~intPoly () {
    if (deg >= 0) {
        delete poly;
    }
}
*/

/*
intPoly::intPoly() {
    deg = -1;
}
*/

/*
intPoly::intPoly(int degr, PolyType* polyn) {
    int i;
    deg = degr;
    if (deg < 0) {return; }

    poly = new PolyType[(deg) + 1];
    for(i = 0; i <= deg; i++) {
        poly[i] = polyn[i];
    }
    for(i = degr; i >= 0; i--) {
        if (poly[i] == 0) {
            deg = i-1;
        } else {
            break;
        }
    }
}

```

```

    }
};
*/

/*
intPoly::nextPoly() {
    int i;
    for(i = 0; i <= deg; i++){
        if (poly[i] == 1) {
            poly[i] = -1;
            break;
        } else if(poly[i] == -1) {
            poly[i] = 1;
        };
    };
}
*/

intPoly* divide(intPoly *poly1, intPoly* poly2){

//    cout << "Poly is " << poly2->eval(1) << endl;
//    cout << "Called divide with degree " << poly1->degree()
//        << " and " << poly2->degree()
//        << endl;
    if (poly1->degree() > poly2->degree()) {
        return(new intPoly(poly2->degree(), poly2->coeff()));
    }

// Find the lead coefficient of the two polynomials.
//    cout << "Find the lead term" << endl;
    int hdeg1, hdeg2;
    hdeg1 = poly1->coeff()[poly1->degree()];
    hdeg2 = poly2->coeff()[poly2->degree()];

// Create a scalar polynomial opposite of the original
    PolyType *coeff1 = new PolyType[1];
// This had better be an integer or all hell will break loose
    coeff1[0] = 1;
    intPoly* scalar1 = new intPoly(0, coeff1);

    delete coeff1;

    PolyType *coeff2 = new PolyType[poly2->degree() - poly1->degree() + 1];

    for(int i=0; i < poly2->degree() - poly1->degree(); i++) {
        coeff2[i] = 0;
    }

//    cout << "Check lead term condition" << endl;
    if (!(hdeg2 % hdeg1) == 0) {
        cout << "ARGH, THIS WON'T DIVIDE PROPERLY" << endl;
        return(poly1);
    }
    coeff2[poly2->degree() - poly1->degree()] = -hdeg2/hdeg1;

    intPoly *scalar2 = new intPoly(poly2->degree() - poly1->degree(), coeff2);
    intPoly *poly3 = mult(poly2, scalar1);
    intPoly *poly4 = mult(poly1, scalar2);

    intPoly *poly5 = add(poly3, poly4);
//    cout << "The new degree is " << poly5->degree() << endl;;

    intPoly *ans = divide(poly1, poly5);

```

```

delete scalar1;
delete scalar2;
delete coeff2;
delete poly3;
delete poly4;
if (ans != poly5) {
    delete poly5;
}
return(ans);
}

intPoly* add(intPoly* poly1, intPoly* poly2) {
    int deg1;
    int deg2;
    PolyType* coeff;
    PolyType* coeff1;
    PolyType* coeff2;
    int i;

    deg1 = poly1->degree();
    deg2 = poly2->degree();

    coeff1 = poly1->coeff();
    coeff2 = poly2->coeff();
    coeff = new PolyType[max(deg1, deg2)+1];

    for(i=0; i <= min(deg1, deg2); i++) {
        coeff[i] = coeff1[i] + coeff2[i];
    }
    if (deg1 == max(deg1, deg2)) {
        for(i = deg2 + 1; i <= deg1; i++) {
            coeff[i] = coeff1[i];
        }
    } else {
        for(i = deg1 + 1; i <= deg2; i++) {
            coeff[i] = coeff2[i];
        }
    }

    intPoly *poly = new intPoly(max(deg1, deg2), coeff);
    delete coeff;
    return(poly);
}

intPoly* mult(intPoly* poly1, intPoly* poly2) {
    int i, j;
    int deg;

    deg = poly1->degree() + poly2->degree();
    PolyType* coeff = new PolyType[deg+2];

    for (i=0; i <= deg; i++) {
        coeff[i] = 0;
    }

    for (i=0; i <= poly1->degree(); i++) {
        for (j=0; j <= poly2->degree(); j++) {
            coeff[i+j] = coeff[i+j] + poly1->coeff()[i] * poly2->coeff()[j];
        }
    }

    intPoly *poly = new intPoly(deg, coeff);
    delete coeff;
}

```



```

    return(poly);
}

intPoly* diff(intPoly* poly1) {
    PolyType *coeff = new PolyType[poly1->degree()];

    int j;

    for (j=1; j <= poly1->degree(); j++) {
        coeff[j-1] = j * poly1->coeff()[j];
    }

    intPoly * poly = new intPoly(poly1->degree() - 1, coeff);

    delete coeff;

    return(poly);
}

```

A.2 Spectrum Algorithm

File name: Spec.h

```

/*
This is the generic algorithm to compute l(q), a(q) or l^m(q)

Copyright (C) 2000 Kevin G Hare

This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
*/

#ifndef _SPEC
#define _SPEC

#include <CC/stack.h>
#include <CC/set.h>
#define true 1
#define false 0
#include "Polynomial.h"

queue<intPoly*>*
Spec(intPoly* poly, double beta, double upper, int numStart, int* Start,
    int numIterate, int* Iterate, int sym = false,
    int Zero = false);

#endif

```

File name: Spec.cc

```

/*
This is the generic algorithm to compute l(q), a(q), or l^m(q)

Copyright (C) 2000 Kevin G Hare

This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
*/

#define true 1
#define false 0
#include "Spec.h"

class ptnPoly {
public:
    intPoly * Poly;
    ptnPoly(intPoly*poly) { Poly = poly; };
    ptnPoly() {};
    intPoly * poly() const {return(Poly);};

    ~ptnPoly() {
};

    int operator<(const ptnPoly poly1) const {
        intPoly *P1, *P2;
        P1 = Poly;
        P2 = (poly1.Poly);

        int a;
        a = P1->operator<(P2);
        return(a);
};
};

// bool less<ptnPoly>::operator()(const class ptnPoly & p1,
// const class ptnPoly & p2) const {
//     int a = p1.poly()->operator<(p2.poly());
//     return(true);
// }

queue<intPoly*>*
Spec(intPoly* poly, double beta, double upper, int numStart, int* Start,
    int numIterate, int* Iterate, int sym,
    int Zero)
{
    queue<intPoly*> ToLookAt;
    queue<intPoly*>* LookedAt = new queue<intPoly*>();
    set<ptnPoly> Pisot;

    double pisot;
    int counter;

```

```

intPoly* poly2;
intPoly* poly3;
intPoly* poly4;
intPoly* poly5;
PolyType* coeffs;
coeffs = new PolyType[2];
coeffs[0] = 0;
coeffs[1] = 1;
intPoly* x      = new intPoly(1, coeffs);
coeffs[0] = -1;
intPoly* mone   = new intPoly(0, coeffs);
delete coeffs;

ptnPoly *iterate;
iterate = new ptnPoly[numIterate+1];

PolyType maxAllow = MaxPolyType /(1+poly->height());

counter = 0;

int i;
coeffs = new PolyType[1];
for(i = 1; i <= numIterate; i++) {
    coeffs[0] = Iterate[i-1];
    iterate[i] = ptnPoly(new intPoly(0, coeffs));
}

for(i=1; i<= numStart; i++) {
    coeffs[0] = Start[i-1];
    intPoly* startPoly = new intPoly(0, coeffs);
    ToLookAt.push(startPoly);
};
delete coeffs;

while (!(ToLookAt.empty())) { while(true) {
    if ((counter % 500000) == 0) {
        cout << "To look at:" << ToLookAt.size() <<
            " Looked at:" << LookedAt->size() << endl;
    }
    counter = counter + 1;

    poly2 = ToLookAt.front();
    ToLookAt.pop();
    pisot = poly2->eval(beta);

    if ((pisot >= upper) || (pisot <= -upper)) {
        delete poly2;
        break;
    }

    if (poly2->height() > maxAllow) {
        cout << "%% The height is too high, and is not believable at ";
        cout << poly2->height() << endl;
        return((queue<intPoly*>*)0);
    }

    if (sym) {
        if (poly2->degree() >= 0) {
            if (poly2->coeff()[poly2->degree()] < 0) {
                poly3 = mult(poly2, mone);
                delete poly2;
                poly2 = poly3;
            }
        }
    }
}
}

```

```

    }
}

if (Zero&& poly2->zero()) {
    cout << "%% FOUND ZERO " << endl;
    return(LookedAt);
}

int size1 = Pisot.size();
ptnPoly ptn(poly2);
Pisot.insert(ptn);
if (Pisot.size() == size1) {
    delete poly2;
    break;
}

LookedAt->push(poly2);

poly3 = mult(poly2, x);
for(i = 1; i <= numIterate; i++) {

    poly4 = add(poly3, iterate[i].poly());

    poly5 = divide(poly, poly4);
    if ((poly5 != poly4) {
        delete poly4;
    }
    ToLookAt.push(poly5);
};
delete poly3;
break;
}; };

return(LookedAt);
}

```

A.3 Top Level Code

File name: LittleL.cc

```

/*
This will calculate  $l(q)$  for a real number  $q$ .
(The minimal non-zero value of  $p(q)$ , where  $p$  ranges over all
height 1 polynomials).
It will take as input the polynomial, as well as the root of the
polynomial.

```

Copyright (C) 2000 Kevin G Hare

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

```

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
*/

#include "Spec.h"

inline double dmax(double a, double b) {
    if (a < b) { return(b); } return(a);
};

inline double dabs(double a) {
    return(dmax(a, -a));
};

inline double dmin(double a, double b) {
    if (a > b) { return(b); } return(a);
};

int main() {
    intPoly* poly1 = new intPoly();
    poly1->read();
    cout << "The actually polynomial "; poly1->print(); cout << endl;

    double beta;
    cout << "What is the value of beta? ";
    cin >> beta;
    double upper;
    upper = 1/(beta-1)+0.001;
    cout << "The upper bound is " << upper << endl;

    int* coeff1 = new int[3];
    int* coeff2 = new int[3];

    coeff1[0] = -1; coeff1[1] = 0; coeff1[2] = 1;
    coeff2[0] = -1; coeff2[1] = 0; coeff2[2] = 1;

    queue<intPoly*>* q = Spec(poly1, beta, upper, 3, coeff1, 3,coeff2, true);

    cout << "%% The size of Lambda(q) in the range ["
        << 0 << ", " << upper << "] is (approximately) " << 2*(q->size())-1
        << endl;

    double Z;
    double Ztemp;
    Z = 10;
    intPoly* poly;
    intPoly* ZPoly = new intPoly();
    while (!q->empty()) {
        poly = q->front();
        Ztemp = dabs(poly->eval(beta));

        if ((Ztemp > 0) && ( Ztemp < Z)) {
            Z = Ztemp;
            delete ZPoly;
            ZPoly = poly;
            cout << Z << endl;
            cout << "From polynomial ";
            poly->print();
            cout << endl;
        } else {
            delete poly;

```

```

    }

    q->pop();
};
cout << "%% The value of l(q) is " << Z << endl;
cout << "%% This value is from the polynomial: "
    << endl << "%% ";
ZPoly->print();
cout << endl;
};

```

File name: LittleLm.cc

```

/*
This calculates  $l^m(q)$ , the minimal non-zero value of  $p(q)$  as
 $p$  ranges over all height  $m$  polynomials. It takes as input
the minimal polynomial of  $q$ ,  $q$ , and  $m$ .

Copyright (C) 2000 Kevin G Hare

This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
*/

#include "Spec.h"

inline double dmax(double a, double b) {
    if (a < b) { return(b); } return(a);
};

inline double dabs(double a) {
    return(dmax(a, -a));
};

inline double dmin(double a, double b) {
    if (a > b) { return(b); } return(a);
};

int main() {
    intPoly* poly1 = new intPoly();
    poly1->read();
    cout << endl <<
        "%% Examining the root of the polynomial ";
    poly1->print(); cout << endl;

    double beta;
    cout << "What is the value of beta? ";
    cin >> beta;

    int m;
    cout << "What is the value of m? ";

```

```

cin >> m;

double upper;
upper = m/(beta-1);

int* coeff1 = new int[2*m+1];
int* coeff2 = new int[2*m+1];

for(int k = 0; k <= 2*m; k++) {
    coeff1[k] = -m+k;
    coeff2[k] = -m+k;
}

queue<intPoly*>* q = Spec(poly1, beta, upper, 2*m+1, coeff1, 2*m+1,coeff2,
    true);

cout << "%% The size of the spectrum is " << q->size() << endl;

double Z;
double Ztemp;
Z = 10;
intPoly* poly;
intPoly* ZPoly = new intPoly();
while (!q->empty()) {
    poly = q->front();
    Ztemp = dabs(poly->eval(beta));

    if ((Ztemp > 0) && ( Ztemp < Z)) {
        Z = Ztemp;
        ZPoly = poly;
        cout << Z << endl;
        cout << "From polynomial ";
        poly->print();
        cout << endl;
    } else {
        delete poly;
    }

    q->pop();
};
cout << "%% l^" << m << " value is " << Z ;
cout << " from " ;
ZPoly->print();
cout << endl;
};

```

File name: LittleWoodL.cc

```

/*
This will calculate a(q), the minimal non-zero value of p(q)
as p ranges over all +1 polynomials. It takes as input the
minimal polynomial of q, as well as q

```

Copyright (C) 2000 Kevin G Hare

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of

```

MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
*/

#include "Spec.h"

inline double dmax(double a, double b) {
    if (a < b) { return(b); } return(a);
};

inline double dabs(double a) {
    return(dmax(a, -a));
};

inline double dmin(double a, double b) {
    if (a > b) { return(b); } return(a);
};

int main() {
    intPoly* poly1 = new intPoly();
    poly1->read();
    cout << endl << "%%%" The value of q being examined is the root of:"
        << endl << "%%%" "
        poly1->print(); cout << endl;

    double beta;
    cout << "What is the value of beta? ";
    cin >> beta;
    double upper;
    cout << endl << "%%%" So the approximate value of q is " << beta << endl;
    upper = 1/(beta-1)+0.001;
    cout << "The upper bound is " << upper << endl;

    int* coeff1 = new int[2];
    int* coeff2 = new int[2];

    coeff1[0] = -1; coeff1[1] = 1;
    coeff2[0] = -1; coeff2[1] = 1;

    queue<intPoly*>* q = Spec(poly1, beta, upper, 2, coeff1, 2 ,coeff2, true);
    cout << "%%%" << endl;

    cout << "%%%" The size of A(q) in the range [" << 0 << ","
        << upper << "] is (approximately) " << 2*(q->size()) << endl;

    double Z;
    double Ztemp;
    Z = 10;
    int Zero;
    Zero = false;
    intPoly* poly;
    intPoly* ZPoly = new intPoly();
    while (!q->empty()) {
        poly = q->front();
        Ztemp = dabs(poly->eval(beta));

        if ((Ztemp > 0) && ( Ztemp < Z) ) {
            Z = Ztemp;
            delete ZPoly ;

```



```

        ZPoly = poly;
        cout << Z << endl;
        cout << "From polynomial ";
        poly->print();
        cout << endl;
    } else if (poly->zero()) {
        Zero = true;
        delete poly;
    } else {
        delete poly;
    }
}

q->pop();
};

cout << "%% The value of a(q) is " << Z << endl;
cout << "%% This value is from the polynomial: " << endl
    << "%% ";
ZPoly->print();
cout << endl;

if (Zero) {
    cout << "%% Zero is in A(q)" << endl;
} else {
    cout << "%% Zero is not in A(q)" << endl;
};
cout << "%%" << endl;
};

File name: PisotDivideLittlewood.cc

/*
This checks to see if q is a root of a +-1 polynomial.
It takes as input the minimal polynomial of q, and q.

Copyright (C) 2000 Kevin G Hare

This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
*/

#include "Spec.h"

int main() {
    intPoly* poly1 = new intPoly();
    poly1->read();
    cout << "The actual polynomial "; poly1->print(); cout << endl;

    double beta;
    cout << "What is the value of beta? ";
    cin >> beta;

```

```

int* coeff1 = new int[2];
int* coeff2 = new int[2];
double upper;
upper = 1/(beta-1);

coeff1[0] = -1; coeff1[1] = 1;
coeff2[0] = -1; coeff2[1] = 1;

queue<intPoly*>* q = Spec(poly1, beta, upper, 3, coeff1, 3 ,coeff2);

intPoly* poly;
while (!q->empty()) {
    poly = q->front();
    if (poly->zero()) {
        cout << "0 is in the spectrum" << endl;
        return(0);
    }
    q->pop();
    delete poly;
};
cout << "Guess it is not in the spectrum " << endl;
};

```

A.4 Compiling Stuff

File name: Makefile

```

all: LittleLPar
test: Test CheckSpec
ll: LittleL LittleWoodL
all2: LittleL Test GExpand PisotDivideLittlewood LittleLm LittleWoodL
thesis: LittleL LittleLm LittleWoodL PisotDivideLittlewood

#DEBUG = -ggdb
#DEBUG = -debug
#DEBUG = -pg -g
OPTIMIZE = -IPA -INLINE
#OPTIMIZE = -O3 -finline-functions
#-Winline
#LIB = -l /usr/include/ -l /usr/include/CC/
# -l /usr/freeware/lib/gcc-lib/mips-sgi-irix6.2/2.8.1/mabi=64/libgcc.a
#LIB = /usr/include/
#CCC = g++
CCC = CC -64

.SUFFIXES: .cc .c,v .h .h,v

.h,v.h:
co $*.h

.cc,v.cc:
co $*.cc

VBound.o: VBound.cc VBound.h Polynomial.h makefile
$(CCC) -c $(DEBUG) $(OPTIMIZE) VBound.cc -o VBound.o

Spec.o: Spec.cc Spec.h Polynomial.h makefile
$(CCC) -c $(DEBUG) $(OPTIMIZE) Spec.cc -o Spec.o

```

```

SpecPar.o: SpecPar.cc SpecPar.h PolynomialPar.h makefile
$(CCC) -c $(DEBUG) $(OPTIMIZE) SpecPar.cc -o SpecPar.o

SpecControlPar.o: SpecControlPar.cc SpecControlPar.h SpecPar.h \
    PolynomialPar.h makefile
$(CCC) -c $(DEBUG) $(OPTIMIZE) SpecControlPar.cc -o SpecControlPar.o

GExpand.o: GExpand.cc Polynomial.h makefile
$(CCC) -c $(DEBUG) $(OPTIMIZE) GExpand.cc -o GExpand.o

GExpand: GExpand.o Polynomial.o Polynomial.h
$(CCC) $(DEBUG) $(OPTIMIZE) GExpand.o Polynomial.o -o GExpand

PisotDivideLittlewood.o: PisotDivideLittlewood.cc Polynomial.h Spec.h makefile
$(CCC) -c $(DEBUG) $(OPTIMIZE) PisotDivideLittlewood.cc -o \
    PisotDivideLittlewood.o

PisotDivideLittlewood: PisotDivideLittlewood.o Spec.o Spec.h Polynomial.o \
    Polynomial.h
$(CCC) $(DEBUG) $(OPTIMIZE) PisotDivideLittlewood.o Spec.o \
    Polynomial.o -o PisotDivideLittlewood

LittleLm.o: LittleLm.cc Polynomial.h Spec.h makefile
$(CCC) -c $(DEBUG) $(OPTIMIZE) LittleLm.cc -o LittleLm.o

LittleLm: LittleLm.o Spec.o Spec.h Polynomial.o Polynomial.h
$(CCC) $(DEBUG) $(OPTIMIZE) LittleLm.o Spec.o Polynomial.o -o LittleLm

CounterSpec.o: CounterSpec.cc VBound.h Polynomial.h Spec.h makefile
$(CCC) -c $(DEBUG) $(OPTIMIZE) CounterSpec.cc -o CounterSpec.o

CounterSpec: CounterSpec.o Spec.o Spec.h Polynomial.o Polynomial.h VBound.o \
    VBound.h
$(CCC) $(DEBUG) $(OPTIMIZE) VBound.o CounterSpec.o Spec.o \
    Polynomial.o -o CounterSpec

CheckSpec.o: CheckSpec.cc VBound.h Polynomial.h Spec.h makefile
$(CCC) -c $(DEBUG) $(OPTIMIZE) CheckSpec.cc -o CheckSpec.o

CheckSpec: CheckSpec.o Spec.o Spec.h Polynomial.o Polynomial.h VBound.o VBound.h
$(CCC) $(DEBUG) $(OPTIMIZE) VBound.o CheckSpec.o Spec.o Polynomial.o \
    -o CheckSpec

LittleWoodL.o: LittleWoodL.cc Polynomial.h Spec.h makefile
$(CCC) -c $(DEBUG) $(OPTIMIZE) LittleWoodL.cc -o LittleWoodL.o

LittleWoodL: LittleWoodL.o Spec.o Spec.h Polynomial.o Polynomial.h
$(CCC) $(DEBUG) $(OPTIMIZE) LittleWoodL.o Spec.o Polynomial.o -o \
    LittleWoodL

LittleLDDiff.o: LittleLDDiff.cc Polynomial.h SpecDDiff.h makefile
$(CCC) -c $(DEBUG) $(OPTIMIZE) LittleLDDiff.cc -o LittleLDDiff.o

LittleLDDiff: LittleLDDiff.o SpecDDiff.o SpecDDiff.h Polynomial.o Polynomial.h
$(CCC) $(DEBUG) $(OPTIMIZE) LittleLDDiff.o SpecDDiff.o \
    Polynomial.o -o LittleLDDiff

LittleLDiff.o: LittleLDiff.cc Polynomial.h SpecDiff.h makefile
$(CCC) -c $(DEBUG) $(OPTIMIZE) LittleLDiff.cc -o LittleLDiff.o

LittleLDiff: LittleLDiff.o SpecDiff.o SpecDiff.h Polynomial.o Polynomial.h
$(CCC) $(DEBUG) $(OPTIMIZE) LittleLDiff.o SpecDiff.o \

```

```

Polynomial.o -o LittleLDiff

LittleWoodDiff.o: LittleWoodDiff.cc Polynomial.h SpecDiff.h makefile
$(CCC) -c $(DEBUG) $(OPTIMIZE) LittleWoodDiff.cc -o LittleWoodDiff.o

LittleWoodDiff: LittleWoodDiff.o SpecDiff.o SpecDiff.h Polynomial.o Polynomial.h
$(CCC) $(DEBUG) $(OPTIMIZE) LittleWoodDiff.o SpecDiff.o \
    Polynomial.o -o LittleWoodDiff

ZOPolyDiff.o: ZOPolyDiff.cc Polynomial.h SpecDiff.h makefile
$(CCC) -c $(DEBUG) $(OPTIMIZE) ZOPolyDiff.cc -o ZOPolyDiff.o

ZOPolyDiff: ZOPolyDiff.o SpecDiff.o SpecDiff.h Polynomial.o Polynomial.h
$(CCC) $(DEBUG) $(OPTIMIZE) ZOPolyDiff.o SpecDiff.o \
    Polynomial.o -o ZOPolyDiff

ZOPoly.o: ZOPoly.cc Polynomial.h Spec.h makefile
$(CCC) -c $(DEBUG) $(OPTIMIZE) ZOPoly.cc -o ZOPoly.o

ZOPoly: ZOPoly.o Spec.o Spec.h Polynomial.o Polynomial.h
$(CCC) $(DEBUG) $(OPTIMIZE) ZOPoly.o Spec.o Polynomial.o -o ZOPoly

LittleLPar.o: LittleLPar.cc PolynomialPar.h SpecPar.h SpecControlPar.h makefile
$(CCC) -c $(DEBUG) $(OPTIMIZE) LittleLPar.cc -o LittleLPar.o

LittleLPar: LittleLPar.o SpecPar.o SpecPar.h PolynomialPar.o PolynomialPar.h \
    SpecControlPar.o SpecControlPar.h
$(CCC) $(DEBUG) $(OPTIMIZE) LittleLPar.o SpecPar.o PolynomialPar.o \
    SpecControlPar.o -o LittleLPar

LittleL.o: LittleL.cc Polynomial.h Spec.h makefile
$(CCC) -c $(DEBUG) $(LIB) $(OPTIMIZE) LittleL.cc -o LittleL.o

LittleL: LittleL.o Spec.o Spec.h Polynomial.o Polynomial.h
$(CCC) $(DEBUG) $(LIB) $(OPTIMIZE) LittleL.o Spec.o Polynomial.o -o LittleL

Polynomial.o: Polynomial.cc makefile Polynomial.h
$(CCC) -c $(DEBUG) $(OPTIMIZE) Polynomial.cc -o Polynomial.o

PolynomialPar.o: PolynomialPar.cc makefile PolynomialPar.h
$(CCC) -c $(DEBUG) $(OPTIMIZE) PolynomialPar.cc -o PolynomialPar.o

#Test.o: Test.cc SpecPar.h PolynomialPar.h makefile
Test.o: Test.cc
$(CCC) -c $(DEBUG) $(OPTIMIZE) Test.cc -o Test.o

temp: temp.o
$(CCC) $(DEBUG) $(OPTIMIZE) temp.o -o temp

temp.o: temp.cc
$(CCC) -c $(DEBUG) $(OPTIMIZE) temp.cc -o temp.o

#Test: Test.o PolynomialPar.o PolynomialPar.h SpecPar.o SpecPar.h
Test: Test.o
# $(CCC) $(DEBUG) $(OPTIMIZE) Test.o PolynomialPar.o -o Test
$(CCC) $(DEBUG) $(OPTIMIZE) Test.o -o Test

checkout:
co -l GExpand.cc LittleL.cc LittleLm.cc \
    PisotDivideLittlewood.cc Polynomial.cc \
    Polynomial.h Spec.cc Spec.h Test.cc

clean:

```

```
rm LittleL LittleWoodL LittleLm ZOPoly ZOPolyDiff LittleWoodDiff\
    LittleLDiff LittleLDDiff CheckSpec LittleLPar CounterSpec *.o\
ii_files/*
```

A.5 GNU Public License

File name: gpl.txt

GNU GENERAL PUBLIC LICENSE
Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.
59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free

program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If

identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

- a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
- b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
- c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such

parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the program's name and a brief idea of what it does.>
Copyright (C) 19yy <name of author>
```

```
This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
(at your option) any later version.
```

```
This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
```

GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this
when it starts in an interactive mode:

```
Gnomovision version 69, Copyright (C) 19yy name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type 'show w'.
This is free software, and you are welcome to redistribute it
under certain conditions; type 'show c' for details.
```

The hypothetical commands 'show w' and 'show c' should show the appropriate
parts of the General Public License. Of course, the commands you use may
be called something other than 'show w' and 'show c'; they could even be
mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your
school, if any, to sign a "copyright disclaimer" for the program, if
necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright interest in the program
'Gnomovision' (which makes passes at compilers) written by James Hacker.
```

```
<signature of Ty Coon>, 1 April 1989
Ty Coon, President of Vice
```

This General Public License does not permit incorporating your program into
proprietary programs. If your program is a subroutine library, you may
consider it more useful to permit linking proprietary applications with the
library. If this is what you want to do, use the GNU Library General
Public License instead of this License.

Bibliography

- [1] Douglas Adams, *A long dark tea-time of the soul*, Pan Books Ltd., London, 1988.
- [2] Shiro Ando and Teluhiko Hilano, *A disjoint covering of the set of natural numbers consisting of sequences defined by a recurrence whose characteristic equation has a Pisot number root*, *Fibonacci Quart.* **33** (1995), no. 4, 363–367.
- [3] J. M. Borwein and P. B. Borwein, *Pi and the AGM*, John Wiley & Sons Inc., New York, 1998, A study in analytic number theory and computational complexity, Reprint of the 1987 original, A Wiley-Interscience Publication.
- [4] P. Borwein and K.G. Hare, *Non-trivial quadratic approximations to zero of a family of cubic Pisot numbers*, (manuscript).
- [5] ———, *Some computations on the spectra of Pisot and Salem numbers*, *Math. Comp.* **71** (2002), 767–780.
- [6] ———, *General forms for minimal spectral values for a class of quadratic Pisot numbers*, *Journal of the London Math Society* (to appear).
- [7] David W. Boyd, *Pisot and Salem numbers in intervals of the real line*, *Math. Comp.* **32** (1978), no. 144, 1244–1260.
- [8] Y. Bugeaud, *On a property of Pisot numbers and related questions*, *Acta Math. Hungar.* **73** (1996), no. 1-2, 33–39.
- [9] Č. Burdík, Ch. Frougny, J. P. Gazeau, and R. Krejcar, *Beta-integers as natural counting systems for quasicrystals*, *J. Phys. A* **31** (1998), no. 30, 6449–6472.

- [10] J. W. S. Cassels, *An introduction to Diophantine approximation*, Cambridge University Press, New York, 1957, Cambridge Tracts in Mathematics and Mathematical Physics, No. 45.
- [11] Y. Chitour and B. Piccoli, *Controllability for discrete systems with a finite control set*, Math. Control Signals Systems **14** (2001), no. 2, 173–193.
- [12] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to algorithms*, MIT Press, Cambridge, MA, 1990.
- [13] P. Erdős, I. Joó, and V. Komornik, *Characterization of the unique expansions $1 = \sum_{i=1}^{\infty} q^{-n_i}$ and related problems*, Bull. Soc. Math. France **118** (1990), no. 3, 377–390.
- [14] P. Erdős, I. Joó, and V. Komornik, *On the sequence of numbers of the form $\epsilon_0 + \epsilon_1 q + \dots + \epsilon_n q^n$, $\epsilon_i \in \{0, 1\}$* , Acta Arith. **83** (1998), no. 3, 201–210.
- [15] P. Erdős, I. Joó, and F. J. Schnitzer, *On Pisot numbers*, Ann. Univ. Sci. Budapest. Eötvös Sect. Math. **39** (1996), 95–99 (1997).
- [16] P. Erdős, M. Joó, and I. Joó, *On a problem of Tamás Varga*, Bull. Soc. Math. France **120** (1992), no. 4, 507–521.
- [17] P. Erdős and V. Komornik, *Developments in non-integer bases*, Acta Math. Hungar. **79** (1998), no. 1-2, 57–83.
- [18] A. M. Garsia, *Arithmetic properties of Bernoulli convolutions*, Trans. Amer. Math. Soc. **102** (1962), 409–432.
- [19] K.O. Geddes, G. Labahn, M. B. Monagan, and S. Vorketter, *The Maple programming guide*, Springer-Verlag, New York, 1996.
- [20] Nikolai Gogol, *Taras Bulba*, Village Evenings near Dikanka and Mirgorod, Oxford University Press, Great Britain, 1994, Translated from the Russian by Christopher English, (first published 1833).

- [21] ———, *Village evenings near Dikanka*, Village Evenings near Dikanka and Mirgorod, Oxford University Press, Great Britain, 1994, Translated from the Russian by Christopher English, (first published 1832).
- [22] R. L. Graham, D. E. Knuth, and O. Patashnik, *Concrete mathematics*, second ed., Addison-Wesley Publishing Company, Reading, MA, 1994, A foundation for computer science.
- [23] K. G. Hare, *Home page*, <http://www.cecm.sfu.ca/~kghare>, 1999.
- [24] Shunji Ito and Yuki Sano, *On periodic β -expansions of Pisot numbers and Rauzy fractals*, Osaka J. Math. **38** (2001), no. 2, 349–368.
- [25] I. Joó and F. J. Schnitzer, *On some problems concerning expansions by noninteger bases*, Anz. Österreich. Akad. Wiss. Math.-Natur. Kl. **133** (1996), 3–10 (1997).
- [26] V. Komornik, P. Loreti, and M. Pedicini, *An approximation property of Pisot numbers*, J. Number Theory **80** (2000), no. 2, 218–237.
- [27] Ka-Sing Lau, *Dimension of a family of singular Bernoulli convolutions*, J. Funct. Anal. **116** (1993), no. 2, 335–358.
- [28] Maurice Mignotte, *Mathematics for computer algebra*, Springer-Verlag, New York, 1992, Translated from the French by Catherine Mignotte.
- [29] Y. Peres and B. Solomyak, *Approximation by polynomials with coefficients ± 1* , J. Number Theory **84** (2000), no. 2, 185–198.
- [30] K. F. Roth, *Rational approximations to algebraic numbers*, Mathematika **2** (1955), 1–20; corrigendum, 168.
- [31] R. Salem, *Power series with integral coefficients*, Duke Math. J. **12** (1945), 153–172.
- [32] Raphaël Salem, *Algebraic numbers and Fourier analysis*, D. C. Heath and Co., Boston, Mass., 1963.

- [33] Wolfgang M. Schmidt, *On simultaneous approximations of two algebraic numbers by rationals*, Acta Math. **119** (1967), 27–50.
- [34] A. Schrijver, *Theory of linear and integer programming*, John Wiley & Sons Ltd., Chichester, 1986, A Wiley-Interscience Publication.
- [35] C. J. Smyth, *On the product of the conjugates outside the unit circle of an algebraic integer*, Bull. London Math. Soc. **3** (1971), 169–175.
- [36] Neal Stephenson, *Cryptonomicon*, Perennial, New York, 1999.
- [37] Jonathan Swift, *Gulliver's travels*, Houghton Mifflin Company, Boston, 1960, (first published 1726).
- [38] Axel Thue, *Über eine Eigenschaft, die keine transcendente Grosse haben kann*, Videnskapsselskapets Skrifter. I Mat.-naturv. Klasse, Kristiania (1912), no. 20.
- [39] Vickie York, *New plan to blow up the moon!*, Weekly World News (April 2, 2002).

Index

- algebraic integer, 1
- algebraic number, 1
- best approximation, 36, 39, 43, 44, 46, 48, 51, 52, 55, 73
- C++, 16, 26
- conjugate, 1
- convex polytope, 61, 69
- cyclotomic number, 2, 58
- discrete, iv, 9, 12, 13, 15, 16, 26, 28, 29, 31, 32, 72, 73
 - non-uniform, 9, 15, 29–31
 - uniform, 9, 15, 29, 74
- Fibonacci, 9, 34, 35
- golden ratio, iii, 6–9, 11, 22, 23, 34–36, 44, 56, 59, 60, 62–64, 66–68, 74
- graduate studies, 35
- grammar, 71
- height, 5, 7–9, 16, 26, 28, 29, 31, 43, 46, 55–59, 61, 63, 64, 69, 70
- linear approximation, 44, 52, 55
- Louville’s inequality, 55
- Mahler measure, 2, 29, 32, 72
- Maple, 47, 59, 60, 66
- mathematical pedagogy, iii, 1, 11
- moon, 51
- Perron number, 28, 72
- Pisot number, 2
 - cubic, iv, 34, 52–70, 73
 - first, iii, 3, 8, 11, 52, 53, 60, 64, 70, 73
 - second, 8, 9, 11, 22, 52, 54, 60, 64, 70, 73
 - quadratic, iv, 34–42, 44–49, 51, 52, 72, 73
- programming errors, 22
- quadratic approximation, 52–70, 73
- reciprocal, 3, 32
- red-black tree, 16
- Salem number, iii, 2, 26, 29, 31–33, 72, 73
- simplex method, 46, 47
- spectra, iii, iv, 6–15, 17, 18, 20–23, 26, 28–32, 34–40, 44–46, 48–54, 70, 72–74

algorithm, 14

Ka-Sing Lau, 11

unit, 36

Glossary

- $A(q)$ - ± 1 polynomials evaluated at q ,
9, 10, 22, 23, 26, 28–32, 72, 73
- A_n - For fixed integers a, b , $A_0 = 0$,
 $A_1 = 1$, $A_n = aA_{n-1} + bA_{n-2}$,
39–42
- B_n - For fixed integers a, b , $B_0 = 1$,
 $B_1 = 0$, $B_n = aB_{n-1} + bB_{n-2}$,
39–42
- $D(q)$ - See definition (page 56), 56–61,
63–66, 68, 69
- $D_1(q)$ - See definition (page 57), 57, 59,
60, 64, 66, 68, 69
- $D_2(q)$ - See definition (page 57), 57, 60,
64, 65
- F_k - Fibonacci number, ($F_0 = 0, F_1 =$
 $1, F_n = F_{n-1} + F_{n-2}$), 9, 34, 35
- $H(P(x))$ - The height of a polynomial
 $P(x)$, 5, 55–59, 61, 63, 64, 69,
70
- $L(q)$ - $\limsup(y_{k+1} - y_k)$, $y_k \in Y(q)$, 7,
8
- $L^m(q)$ - $\limsup(y_{k+1} - y_k)$, $y_k \in Y^m(q)$,
7, 8
- $P^*(x)$ - $P(\frac{1}{x})x^{\deg(P(x))}$ the reciprocal of
 $P(x)$, 3, 32
- $P_n(x)$ - $\frac{1}{x^n}$, 52–54, 57, 58, 61, 70, 73
- $Y(q)$ - 0,1 polynomials evaluated at q ,
6, 10
- $Y^m(q)$ - 0, \dots , m polynomials evaluated
at q , 6–8, 10
- $\Lambda(q)$ - -1,0,1 polynomials evaluated at
 q , 8–12, 14, 17, 18, 20–23, 30,
36, 72
- $\Lambda^S(q)$ - polynomials with coefficients from
 S evaluated at q , 8–12, 14, 15,
17, 18, 20–23, 30, 36–38, 40, 48,
49, 72, 74
- $\Lambda^m(q)$ - $-m, \dots, m$ polynomials evalu-
ated at q , 8–12, 14, 17, 18, 20–
23, 30, 36–38, 40, 48, 49, 72
- τ - The golden ratio, root of $x^2 - x - 1$,
6–9, 22, 34, 35, 56, 59, 60, 62–
64, 66–68, 74
- $a(q)$ - Minimal ± 1 polynomial at q , 9–
11, 23, 26, 31, 34, 35
- $c(q, m)$ - Lower bound of height m poly-
nomials evaluated at q , 5, 6
- $l(q)$ - Minimal height 1 polynomial at q ,
iii, 7–11, 15, 22, 23, 26, 29–31,
35, 50, 72

- $l^S(q)$ - Minimal polynomial with coefficients from S evaluated at q , iii, iv, 7–11, 13, 15, 22, 23, 26, 29–31, 34–36, 39, 44–46, 48, 50–54, 70, 72, 73
- $l^m(q)$ - Minimal height m polynomial at q , iii, iv, 7–11, 13, 15, 22, 23, 26, 29–31, 34–36, 39, 44–46, 48, 50–54, 70, 72, 73
- Ω - Unit quadratic Pisot numbers, 36, 39–41, 44, 46, 47
- Υ - Non-totally real unit cubic Pisot numbers, 52, 55–57, 60, 61, 64, 68, 70, 73
- $\mathcal{O}(f(x))$ - Running time of a function is bounded asymptotically by $f(x)$, 20, 21
- $\mathcal{P}(x)$ - See definition (page 17), 17, 18, 23
- \mathcal{R} - Height 1 real polynomials, 41, 44, 46, 48