Algorithmic aspects of generalized number systems

Péter Burcsi

PhD Thesis
Department of Computer Algebra
Eötvös Loránd University

Supervisor: Dr. Attila Kovács PhD
Associate Professor

PhD School of Informatics
Dr. János Demetrovics
PhD Program of Information Systems
Dr. András Benczúr
Budapest, 2008

Contents

Preface

About this thesis

This thesis contains algorithmic results in the theory of generalized number systems. Most of these already appeared in the journal papers [10], [12], [14]. These articles are joint work with Attila Kovács and ([14]) with Zsuzsanna Papp-Varga.

The text aims to be self-contained but I tried to avoid a lengthy survey-like introduction to the topic followed by my own contributions. There are excellent surveys in the field (see e.g. [1] or [20]), so I will only recall the most relevant definitions and theorems. Also, I tried to keep the thesis compact and coherent. The journal paper [11], although it is connected to computational number theory and its ideas are potentially applicable to a fast computer search of number system constructions, would not fit here. I also left out still unpublished and/or not algorithmic results, like [13].

In some cases, when it would bring in unnecessary formalism and complications but no new ideas, I have chosen not to put statements in their most general form. So, for example, we will always consider the standard lattice $\mathbb{Z}^n \subseteq \mathbb{R}^n$ and integer matrices, though we could generalize to arbitrary lattices with real matrices. But all our results are applicable through a change of base. Similarly, in chapter 3, we look for monic polynomials, but essentially the same methods would give polynomials with any fixed leading coefficient.

The results shown in the thesis are of algorithmic nature. All algorithms have been programmed either in Maple[1] or in C/C++. The focus is not only on *how* and *why* these methods work, but also on *how fast* they run.

[1]Maple is a trademark of Waterloo Maple Inc.

Beside the theory, there are also some computational and performance measurements.

The structure of the thesis

The thesis is built up as follows. Chapter 2 describes an algorithm for deciding the expansivity of polynomials. Since expansivity of the characteristic polynomial is a necessary condition for a number system base, this algorithm can be the first step of deciding the number system property. Chapter 3 contains two algorithms for finding monic integer expansive polynomials with prescribed constant term. This search is the key to a *complete* list of canonical number system polynomials with a given degree and constant term. Chapter 4 contains two algorithms for deciding the number system property and listing all periods (which we call the classification problem for (M, D)).

Acknowledgments

I would like to thank my supervisor, Attila Kovács for his guidance, help and patience. Before I started working with him, I found that mathematics was rather a lonely activity (at least for me). We spent many hours thinking together, which proved not only useful and instructive, but also an enjoyable intellectual adventure. I am also grateful for his practical way of thinking and his talent for asking the right questions.

I also thank the many people who taught me (the love of) mathematics. It would not be possible to name them all, so the following chronological list is definitely incomplete: my father, who helped me with the first steps, László Spissich, who showed me the beauty of maths and helped me acquire some self-confidence with problem solving, all my teachers at the Eötvös University, Gábor Tardos, from whom I first learnt algorithmic thinking, Antal Járai and Henri Cohen, who opened my eyes on algorithmic number theory, and Imre Kátai, who was one of the initiators of the research of generalized number systems.

Last but not least I thank my wife and my whole family for their support and endless patience.

Chapter 1

Introduction

1.1 Basic definitions

This thesis is concerned with a certain generalization of our ordinary decimal number system. Recall that each positive integer can be uniquely written in the form

$$d_0 + 10d_1 + 100d_2 + \cdots + 10^k d_k \,,$$

if we require that $d_k \neq 0$ and $d_i \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$. This is called the expansion of the number in base 10, and the d_i are called digits. If we replace the powers of 10 with powers of 2, and choose d_i from $\{0, 1\}$, we get the binary number system. If we use -2, we can also represent negative numbers. We can also play with the digits, like allowing $\{-4, -3, -2, \ldots, 5\}$ for the base 10. There are several ways to generalize the notion of number systems. We will only consider the one we call *generalized number systems*.

Definition 1.1.1. *Let $M \in GL(n, \mathbb{Z})$, called the base, and $D \subseteq \mathbb{Z}^n$ a finite set, called the digit set. The pair (M, D) is called a generalized number system (GNS) if every $x \in \mathbb{Z}^n$ has a unique finite representation in the form $x = \sum_{i=0}^{k} M^i d_i$ with $d_i \in D$, and $d_k \neq 0$ if $k > 0$.*

Note that the ordinary decimal system is not a GNS, because negative numbers are not represented. We will also need the following definitions:

Definition 1.1.2. *$x, y \in \mathbb{Z}^n$ are congruent modulo M if*

$$\exists p \in \mathbb{Z}^n : \quad x - y = Mp \,.$$

This is an equivalence relation and the equivalence classes are called congruence classes modulo M.

Definition 1.1.3. *D is a complete residue system (CRS) modulo M if it contains exactly one element of each congruence class of $\mathbb{Z}^n$ modulo M.*

Clearly, if M is invertible, there are exactly $|\det M|$ congruence classes and D is a CRS if and only if it consists of $|\det M|$ pairwise incongruent elements. From now on, if not specified otherwise, D will denote a complete residue system modulo M.

If D is a CRS then for all $x \in \mathbb{Z}^n$, there is a unique $d \in D$ that is congruent to x. We define a map $\varphi : \mathbb{Z}^n \to \mathbb{Z}^n$ by

$$\varphi(x) = M^{-1}(x - d)\,.$$

Intuitively, this function "cuts off" the last digit (d) in the expansion of x. So one can analyze expansions through φ. We recall the following standard definition.

Definition 1.1.4. *The φ-orbit or simply orbit of $x \in \mathbb{Z}^n$ is the series $\varphi^k(x)$, $k = 0, 1, \ldots$.*

In order to formulate some necessary conditions for the number system property, we need the definition of expansiveness.

Definition 1.1.5. *A matrix M is expansive (expanding) if $|\lambda| > 1$ for every eigenvalue $\lambda \in \mathbb{C}$ of M. A polynomial $p \in \mathbb{Z}[x]$ is expansive (expanding) if $|z| > 1$ for every root $z \in \mathbb{C}$ of p.*

Thus, a matrix is expansive if and only if its characteristic polynomial is, and a polynomial is expansive if and only if its companion matrix is.

Special digit sets are the following.

Definition 1.1.6. *D is called canonical digit set if $D = \{(i, 0, 0, \ldots, 0) \mid 0 \leq i < |\det M|\}$.*

Definition 1.1.7. *A monic integer polynomial is called a canonical number system polynomial (CNS polynomial), if its companion matrix with the canonical digit set forms a GNS.*

1.2 Former results about number systems

We give some classic results, most of which are easy to show. We only give the key idea of the proofs. For details, see [17] or [25].

Theorem 1.2.1. *If (M, D) is a generalized number system then M is expansive and D is a complete residue system modulo M.*

Only the first statement is not trivial. The difficult part is to show that the absolute values of the eigenvalues are strictly larger than 1.

Theorem 1.2.2. *If M is expansive then for all x in $\mathbb{Z}^n$, the orbit of x is ultimately periodic.*

This is elementary. The key observation is that the inverse of M is a contraction in some norm. The other important fact is that $\varphi(x)$ is "almost" $M^{-1}(x)$. Since they only differ by $M^{-1}(d)$, which is bounded, φ behaves also like a contraction outside a sufficiently large ball. Thus each orbit eventually reaches this large (but finite) ball.

Theorem 1.2.3. *If (M, D) is a generalized number system then $0 \in D$.*

The finiteness of expansions together with unicity yields this result.

Theorem 1.2.4. *Let M be expansive, D a complete residue system modulo M, containing 0 as a digit. Then (M, D) is a generalized number system if and only if $0 \in D$ and the only period is that of 0.*

Trivially, a nonzero periodic element has no finite expansion.

1.3 The discussed problems

We are now ready to pose the problems we address in the 3 main chapters of the thesis.

Problem 1.3.1. *The expansivity decision problem is the following:*
Input: *a monic polynomial p.*
Output: *Yes, if p is expansive, otherwise No.*

This is discussed in chapter 2 where we ask how one can decide the expansivity of M, or a polynomial p. There are many ways to do this. The most immediate is to numerically determine the eigenvalues and look at their absolute values. We present an algorithm that avoids the unnecessary numerical solution of a polynomial equation. It is also favorable because it gives negative answers quickly.

Problem 1.3.2. *The expansivity search problem is the following:*
Input: $2 \leq n \in \mathbb{N}$, $c \in \mathbb{Z}$.
Output: *All monic expansive integer polynomial of degree n, with constant term c.*

This is analyzed in chapter 3. We present algorithms for an exhaustive search of expansive polynomials. We fix a degree and a constant term, and then ask how one can find all monic expansive polynomials with the desired values. We use this e.g. for the determination of "binary" number systems, i.e. systems with $|\det M| = 2$.

Problem 1.3.3. *The GNS decision problem is the following:*
Input: (M, D).
Output: *Yes, if (M, D) is a number system, otherwise No.*

Problem 1.3.4. *The GNS classification problem is the following:*
Input: (M, D).
Output: *The periods of φ.*

In chapter 4 we ask how one can decide if (M, D) is a generalized number system, once we know M is expansive. The analysis of the CRS property and the effective computation of φ are presented in [23]. We suppose that these are already known. We also examine how the periods can be determined if (M, D) is not a generalized number system.

Chapter 2

Checking expansivity

In this chapter we investigate from an algorithmic point of view one of the three necessary conditions – the expansivity of the base – of the number system property. In order to check the expansivity of M the method of Lehmer-Schur [29] was suggested and used in [24]. Below, we present another method.

2.1 Stability of linear systems

Consider the systems described by sets of linear differential equations, or linear difference equations of the form

$$\dot{x}(t) = A\,x(t), \quad t \geq 0,$$

and

$$x(k+1) = B\,x(k), \quad k = 0, 1, 2, \ldots,$$

respectively, where $x(t)$ and $x(k)$ denote the n-dimensional state vector. These representations arise in control theory, where the fundamental question is the *stability* of the systems.

The origin of linear control systems emerged at the end of the XIX. century in connection of centrifugal regulator, theoretically treated by Maxwell, Hermite and Liapunov. They observed that the stability nature of equilibrium points depends upon the sign of the real parts of the eigenvalues of A. A continuous-time linear control system is said to be *asymptotically stable* if

all the eigenvalues $\lambda_1, \ldots, \lambda_n$ of A have negative real parts. In the discrete-time case it is equivalent with $|\mu_k| < 1, k = 1, \ldots, n$, where $\mu_1, \ldots, \mu_n$ are the eigenvalues of B.

There is a linear rational function that creates a correspondence between the left half-plane and the open unit disk (in the case of differential equations it is usually achieved by the exponential map, but it is not one to one and less appropriate for our purpose):

$$Re(\lambda_i) < 0 \Leftrightarrow |\mu_i| < 1$$

and this is achieved by the standard bilinear mapping

$$\lambda = \frac{\mu + 1}{\mu - 1}, \quad \mu = \frac{\lambda + 1}{\lambda - 1}. \tag{2.1}$$

It is equivalent to taking

$$A = (B + I)(B - I)^{-1}, \quad B = (A - I)^{-1}(A + I). \tag{2.2}$$

The asymptotic stability criterion means now that all the eigenvalues of B must lie inside the unit disc $|\mu| < 1$ in the complex plane. Obviously, in this case $B^k \to 0$ as $k \to \infty$.

A direct method of testing a given system for asymptotic stability is to apply one of the standard numerical algorithms for determining the eigenvalues. But we are interested in checking the stability condition (1) without finding the eigenvalues exactly, (2) we are looking for a method in which the operator may contain symbolic coefficients. We consider the stability problem in terms of the characteristic polynomial of A or B.

Definition The polynomial

$$p(z) = p_0 + p_1 z + p_2 z^2 + \ldots p_n z^n \in \mathbb{R}[z] \tag{2.3}$$

is said to be *stable* if all its roots lie in the open left half of the complex plane. In the complementary case the polynomial is said to be *unstable*.

Fortunately, the theory of polynomial stability is well-researched. Many people were involved in this area, including Routh, Stodola, Hurwitz etc., see [18, 19, 37]. Now, we describe a method which decides the stability of a

given polynomial.

There is a simple necessary condition for a polynomial to be stable.

Theorem 2 [Stodola condition] The polynomial $p(z)$ in (2.3) can only be stable if all its coefficients are of the same sign.

In what follows, we give a necessary and sufficient condition for stability. Some real functions can be written in an m-terminating continued fraction of form

$$\cfrac{b_1}{1+\cfrac{b_2 z}{1+\cfrac{b_3 z}{\ddots \\ 1+b_m z}}}, \tag{2.4}$$

where b_k $(k = 1, 2, \ldots, m)$ are appropriate real numbers. If we consider the rational function

$$r(z) = \frac{c_0 + c_1 z + \cdots + c_s z^s}{d_0 + d_1 z + \cdots d_t z^t}$$

having an m-terminating continued fraction form, then it can be proved ([19]) that the numbers b_k in (2.4) satisfy the following recursion:

$$\begin{aligned} c_n^{(-1)} &:= d_n, \\ c_n^{(0)} &:= c_n, \\ b_{k+1} &:= \frac{c_0^{(k)}}{c_0^{(k-1)}}, \\ c_n^{k+1} &:= b_{k+1} c_{n+1}^{(k-1)} - c_{n+1}^{(k)}, \end{aligned}$$

where $k = 0, 1, \ldots$, $n = 0, 1, \ldots$ and $c_n = 0$ if $n > s$, $d_n = 0$ if $n > t$. The proof of the next theorem can be found in [19].

Theorem 3 The polynomial (2.3) of degree n is stable if and only if the rational function

$$h(z) = \frac{p_1 + p_3 z + p_5 z^2 + \cdots}{p_0 + p_2 z + p_4 z^2 + \cdots} = \frac{\sum_{k=0}^{\lfloor (n-1)/2 \rfloor} p_{2k+1} z^k}{\sum_{k=0}^{\lfloor n/2 \rfloor} p_{2k} z^k}$$

(the Hurwitz alternant of p) can be represented by an n-terminating continued fraction, in which every number b_k $(k = 1, 2, \ldots, n)$ is positive.

2.2 The algorithm

Let M be an invertible $n \times n$ matrix and let

$$b^*(\mu) = b_0 + b_1\mu + \cdots + b_n\mu^n \in \mathbb{Z}[\mu]$$

be the characteristic polynomial of M. The characteristic polynomial of M^{-1} is

$$b(\mu) = b_n + b_{n-1}\mu + \cdots + b_0\mu^n \in \mathbb{Z}[\mu],\ b_0 > 0. \tag{2.5}$$

Applying the linear transformation (2.1) we get the polynomial

$$a(\lambda) = a_n + a_{n-1}\lambda + \cdots + a_0\lambda^n \in \mathbb{Z}[\lambda]$$

Now, using Theorem 2 and 3 the stability of $a(\lambda)$ can easily be decided.

Summarizing our result, the operator M is expansive if and only if $a(\lambda)$ is stable.

The algorithm is presented in pseudocode below. The input is a list of coefficients of the characteristic polynomial of M.

Algorithm 1: IS-EXPANSIVE($[f_0, f_1, \ldots, f_n]$)

```
1  p(x) ← Σ_{i=0}^{n} f_i (x+1)^{n-i} (x-1)^i ;
2  [p_0, p_1, ..., p_n] ← the coefficients of p(x) ;
3  if p_0 = 0 then
       // -1 is a root of f(x)
4      return false ;
5  if p_0 < 0 then
6      [p_0, p_1, ..., p_n] ← [-p_0, -p_1, ..., -p_n] ;
7  for i ← 1 to n do
8      if p_i < 0 then
           // Stodola condition violated
9          return false ;
10 end
11 m ← 1 ;
12 denom ← [p_0, p_2, ..., p_{2⌊n/2⌋}] ;
13 numer ← [p_1, p_3, ..., p_{2⌊(n-1)/2⌋+1}] ;
14 while numer[0] > 0 and m ≤ n do
15     b[m] ← numer[0]/denom[0] ;
16     s ← length(numer)-1 ;
17     t ← length(denom)-1 ;
18     if numer[s] = 0 and denom[t] = 0 then
           // Truncate the lists
19         numer ← [numer[0], numer[1], ..., numer[s-1]] ;
20         s ← s - 1 ;
21         denom ← [denom[0], denom[1], ..., denom[t-1]] ;
22         t ← t - 1 ;
23     temp ← numer ;
24     for i ← 0 to s - 1 do
25         numer[i] ← b[m]denom[i+1] - numer[i+1] ;
26     end
27     numer[s] ← 0 ;
28     denom ← temp ;
29     m ← m + 1 ;
30 end
31 if m ≤ n then
32     return false ;
33 else
34     return true ;
```

The same algorithm works for symbolic values. One cannot make comparisons in this case, so the while loop is governed by $m \leq n$. Then the

answer is positive if and only if every element of the list b is positive. The correctness of the algorithm follows from the statements in the previous section. The running time is analyzed below. An efficient way of performing the first two lines is also discussed below.

Let us see some examples.

(1) Let the characteristic polynomial of the operator M be

$$a(x) = -18 + 9x - 8x^2 + 4x^3 \in \mathbb{Z}[x]. \tag{2.6}$$

Now, IS-EXPANSIVE works as follows. After the linear transformation the algorithm computes the list $p = [-39, -43, -49, -13]$. Then, since the first element of p is negative, the algorithm produces the list $l = [39, 43, 49, 13]$. Observing that all the element of p have the same sign, the Stodola condition is not violated. Then, the algorithm computes the continued fraction coefficients b_i, which are $b_1 = 43/39, b_2 = 1600/1677$, and $b_3 = 13/43$. Since all b_i-s are positive, the algorithm returns with true, so the operator M is expansive.

(2) Let the characteristic polynomial of M be

$$a(x) = a_0 + a_1x + a_2x^2 + x^3 \in \mathbb{Z}[x].$$

The algorithm IS-EXPANSIVE$([a_0, a_1, a_2, 1])$ gives the following:

Expansive, if the following expressions are all positive:

$$\frac{3a_0 - a_1 - a_2 + 3}{a_0 - a_1 + a_2 - 1}, \frac{8(a_0^2 - a_0a_2 + a_1 - 1)}{(a_0 - a_1 + a_2 - 1)(3a_0 - a_1 - a_2 + 3)}, \frac{a_0 + a_1 + a_2 + 1}{3a_0 - a_1 - a_2 + 3}.$$

(3) Let the characteristic polynomial of M be

$$a(x) = -4 - 2x + ex^2 + x^3 + x^4 \in \mathbb{Z}[x].$$

The algorithm gives that M is expansive if and only if the numbers

$$\frac{14}{2 - e}, \frac{(100 + 27e)}{7(2 - e)}, \frac{16(69 + 25e)}{7(100 + 27e)}, \frac{7(4 - e)}{100 + 27e} \tag{2.7}$$

are all positive. Simplifying (2.7) we have that $-3 < \lfloor -69/25 \rfloor < e < 2$,

2.3 Bilinear transformation

In this section we give a method for applying the (2.1) bilinear transformation to the (2.5) polynomial, which uses only $n(n+1)$ additions and n binary shifts.

Clearly,

$$b(\mu) = b\left(\frac{\lambda+1}{\lambda-1}\right) = \frac{1}{(\lambda-1)^n}(a_0\lambda^n + \cdots + a_n) = \frac{1}{(\lambda-1)^n}a(\lambda) \qquad (2.8)$$

for an appropriate $a(\lambda)$ polynomial. The polynomial $a(\lambda)$ has the same distribution of roots relative to the imaginary axis as does $b(\mu)$ relative to the unit circle. Our task is to obtain the coefficients a_i of $a(\lambda)$ in terms of the coefficients b_i of $b(\mu)$. Since

$$\left(\frac{2}{\lambda-1}+1\right) = \frac{\lambda+1}{\lambda-1},$$

therefore if we let $\mu = 1+\sigma$ then

$$b(1+\sigma) = b_0(1+\sigma)^n + \cdots + b_n = c_0\sigma^n + \cdots + c_n = c(\sigma) \qquad (2.9)$$

for appropriate c_i-s. The coefficients c_i in (2.9) can be obtained using a sequence of Horner's scheme. Then let $\sigma = 2/(\lambda-1)$, so we have that

$$b\left(\frac{\lambda+1}{\lambda-1}\right) = c(\sigma) = \frac{1}{(\lambda-1)^n}(2^n c_0 + 2^{n-1}c_1(\lambda-1) + \ldots + c_n(\lambda-1)^n). \quad (2.10)$$

Comparing (2.8) with (2.10) the required a_i-s can be obtained by a second application of Horner's rule to the coefficients $c_n, 2c_{n-1}, 2^2c_{n-2}, \ldots, 2^nc_0$. All together we performed $n(n+1)$ additions plus n binary shifts.

2.4 Summary

Deciding the expansivity of an operator $M \in GL(n,\mathbb{Z})$ by our algorithm requires $\Theta(n^2)$ arithmetic operations in $\mathbb{Q}$. To see what this theoretical speed means in practice we performed computer tests. In our experiment 100000 polynomials of degree n were generated with random integer coefficients be-

tween -100 and 100. This was repeated 8 times, changing the fixed degree of the polynomials from 3 up to 10.

For each random polynomial we checked if it defines an expansive operator, both by the Lehmer-Schur method and by our method (they gave the same results). We counted the number of arithmetic operations (additions and multiplications) needed to decide the expansivity. The figures show the cumulated number of all operations performed when *expansive* operators were found. We present the number of additions, multiplications and all operations as functions of the degree, showed on a logarithmic scale.

The number of expansive cases decreases as the degree increases. The decreasing number of operations is due to this fact. From the concave shape of the curves we can read that this decrease is even faster than exponential. What is more interesting is the approximately constant vertical difference between curves in all three figures. The tests show that our method needs about twice as much additions as the Lehmer-Schur method. On the other hand it only needs one third of the number of multiplications of the latter. The number of all operations is almost the same, slightly in favor of our algorithm. If addition is faster than multiplication it is worth choosing our test based on stability checking rather than the Lehmer-Schur method.

We conclude that in the case of expansive operators our method is a good alternative of the Lehmer-Schur method. We should choose it if we know in advance that the operator is likely to be expansive. This can happen if we know that another operator "close" to it is expansive. We use this method for an extensive search of *all* expansive operators in a parameter space where, in a certain sense, expansive operators are close to each other.

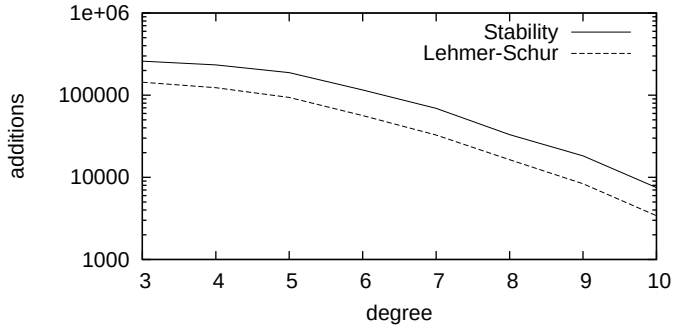


Figure 2.1: The number of additions performed by the two algorithms

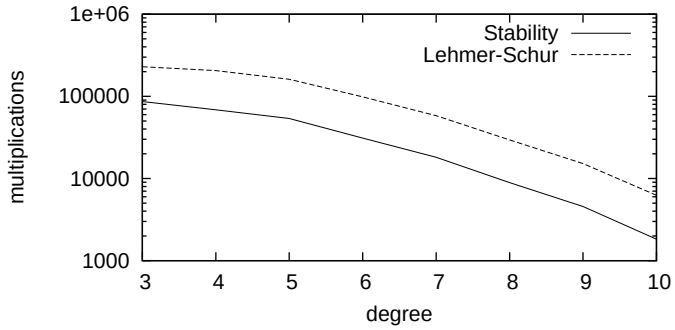


Figure 2.2: The number of multiplications performed by the two algorithms

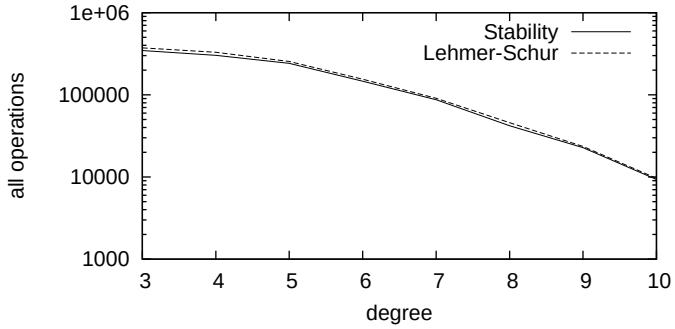


Figure 2.3: The number of all arithmetic operations performed by the two algorithms

Chapter 3

Searching for expansive polynomials

In this chapter we present algorithms for the exhaustive search of expansive monic integer polynomials with fixed degree and constant term. These algorithms are then used for finding all binary canonical number system (CNS) polynomials up to degree 11. Since all CNS polynomials are expansive, the list is complete.

The binary case, when $|\det M| = 2$, has special interest since, via binary CNS, the existence of number systems with two digits can be characterized. To be more precise, let M be an expanding operator in $\mathbb{Z}^k$ with $|\det(M)| = 2$. Then there is a digit set D for which $(\mathbb{Z}^k, M, D)$ is a number system if and only if M is $\mathbb{Z}$-similar to the companion matrix of the characteristic polynomial of M, and the characteristic polynomial is a CNS polynomial (Barbé, von Haeseler [6]).

The chapter is built up as follows. In section 3.1, we give bounds on the coefficients of expansive polynomials. In sections 3.2 and 3.3, algorithms for finding all expansive polynomials with a fixed degree and constant term are presented. In section 3.4, we define semi-CNS polynomials, a possible generalization of CNS polynomials. We also give some sufficient conditions for an expansive polynomial to be a semi-CNS polynomial, and briefly describe the GNS decision algorithms used for the cases not covered by these conditions. In section 3.5, we give our computational results.

Below all polynomials have integer coefficients, unless otherwise stated.

3.1 Bounds on the coefficients

We give upper bounds on the coefficients of complex expansive monic polynomials with fixed constant term. In [24] similar bounds were given. We also show that our results are sharp. The case of degree 1 is trivial, so we only deal with $n \geq 2$.

Lemma 3.1.1. *For every complex number c_0 with $|c_0| > 1$, and integers $n \geq 2$, $1 \leq k \leq n-1$, there exists a positive real number $B(c_0, n, k)$ such that*

- *if $c_0 + c_1 x + \ldots + c_{n-1}x^{n-1} + x^n$ is an expansive polynomial with complex coefficients then $|c_k| < B(c_0, n, k)$, and*
- *for all $\varepsilon > 0$ there exists an expansive polynomial $c_0 + c_1 x + \ldots + c_{n-1}x^{n-1} + x^n$ such that $|c_k| > B(c_0, n, k) - \varepsilon$.*

That is, $B(c_0, n, k)$ is the supremum (but not maximum) of the set of possible absolute values of c_k of monic expansive polynomials of degree n with constant term c_0.

Lemma 3.1.1 has the following immediate corollary, which can be used for finding all expansive integer monic polynomials with fixed degree and constant term.

Corollary 3.1.2. *Let c_0 be a fixed integer with $|c_0| \geq 2$. Then the number of integer-coefficient expansive polynomials of the form $c_0 + c_1 x + \ldots + c_{n-1}x^{n-1} + x^n$ is finite, and it is at most*

$$\prod_{i=1}^{n-1} (2 \cdot \lfloor B(c_0, n, k) \rfloor + 1) \, .$$

Proof. (of lemma 3.1.1)

We can explicitly give the value of $B(c_0, n, k)$ by

$$B(c_0, n, k) = \binom{n-1}{k-1} + |c_0| \cdot \binom{n-1}{k} \, .$$

Let $v_i = |z_i|$, where z_i $(i = 1, \ldots, n)$ are the roots of the polynomial.

Then we have:

$$\begin{aligned} |c_k| &= \left| (-1)^{n-k} \sum_{1 \le i_1 < i_2 < \ldots < i_{n-k} \le n} z_{i_1} z_{i_2} \cdots z_{i_{n-k}} \right| \\ &\le \sum_{1 \le i_1 < i_2 < \ldots < i_{n-k} \le n} v_{i_1} v_{i_2} \cdots v_{i_{n-k}} \end{aligned}$$

We denote this symmetrical polynomial with $f = f_k(v_1, v_2, \ldots, v_n)$. We have to show that the given bound is a supremum for f on the set

$$G = \{(v_1, v_2, \ldots, v_n) \in \mathbf{R}^n \mid 1 < v_i \text{ and } v_1 v_2 \cdots v_n = c_0\}.$$

This set is the interior of

$$H = \{(v_1, v_2, \ldots, v_n) \in \mathbf{R}^n \mid 1 \le v_i \text{ and } v_1 v_2 \cdots v_n = c_0\}.$$

Since H is compact, f takes its maximum on H. We claim that f takes this maximum only in the following n points of the boundary: $(c_0, 1, 1, \ldots, 1)$, $(1, c_0, 1, 1, \ldots, 1)$, $\ldots$, $(1, 1, \ldots, 1, c_0)$. It is easy to verify that in these points the value of f is indeed $B(c_0, n, k)$.

Any other point $v = (v_1, v_2, \ldots, v_n) \in H$ has at least two coordinates, say v_1, v_2 with $1 < v_{1,2} < c_0$. Then, for the point $v' = (1, v_1 v_2, v_3, \ldots, v_n)$, we have $f(v') > f(v)$. This inequality is obtained with some calculation using $v'_1 + v'_2 = 1 + v_1 v_2 > v_1 + v_2$. Since the maximum places lie outside G, we proved the inequality of the lemma.

For the sharpness we choose $n - 1$ roots of the polynomial close to -1, say $-1 - \delta$, the last root is then $-c_0/(1 + \delta)^{n-1}$. For a sufficiently small $\delta > 0$, these roots provide an expansive polynomial with $|c_k|$ arbitrarily close to $B(c_0, n, k)$. □

We list the exact values of $B(2, n, k)$ for $3 \le n \le 12$ and $1 \le k < n$. The last column shows the values of $\prod_{k=1}^{n-1}(2B(2, n, k) - 1)$. This product tells us how many polynomials of degree n fulfill the bound conditions on the coefficients.

We can see that the products increase extremely fast as n grows. If we were to find all expansive polynomials given n and c_0 by checking expansivity

	k=1	2	3	4	5	6	7	8	9	10	11	*product*
n=3	5	4	–	–	–	–	–	–	–	–	–	6.3e+01
4	7	9	5	–	–	–	–	–	–	–	–	2.0e+03
5	9	16	14	6	–	–	–	–	–	–	–	1.6e+05
6	11	25	30	20	7	–	–	–	–	–	–	3.1e+07
7	13	36	55	50	27	8	–	–	–	–	–	1.5e+10
8	15	49	91	105	77	35	9	–	–	–	–	1.9e+13
9	17	64	140	196	182	112	44	10	–	–	–	6.1e+16
10	19	81	204	336	378	294	156	54	11	–	–	5.0e+20
11	21	100	285	540	714	672	450	210	65	12	–	1.1e+25
12	23	121	385	825	1254	1386	1122	660	275	77	13	5.9e+29

Table 3.1: The values of $B(2, n, k)$.

for every polynomial fulfilling these bound conditions, we could hopefully succeed in the cases $n \leq 8$. But we would need more than a year in case of $n = 9$ even if a polynomial is checked in one nanosecond. For larger values of n this is hopeless. Either we should improve the bounds in the case of *integer* coefficient polynomials or we should find a more sophisticated algorithm.

Slight improvements on the bounds can be made in the case of integer coefficients, but the complete lists we obtain in smaller dimensions suggest that even the improved bounds are far from optimal. In order to find all expansive polynomials in higher dimensions, we developed an algorithm that uses dynamic bounds on the coefficients. This algorithm is described in the following section.

3.2 The first search algorithm

Our search algorithm is based on the following three statements and their corollaries. We recall Newton's formula on elementary symmetrical functions and sums of powers of numbers.

Lemma 3.2.1. *(Newton-Girard) Let $z_1, z_2, \ldots, z_n$ be complex numbers. Let $s_k = \sum_{i=1}^{n} z_i^k$, and σ_k be the kth elementary symmetric polynomial in z_1, z_2,...,z_n. Then for $1 \leq k \leq n$ we have*

$$s_k - s_{k-1}\sigma_1 + s_{k-2}\sigma_2 + \ldots + (-1)^k k\sigma_k = 0$$

Applied to roots and coefficients of a polynomial it takes the following equivalent form:

Corollary 3.2.2. *Let $z_1, z_2, \ldots, z_n$ denote the complex roots of the polynomial $a_0 + a_1 x + \ldots + a_{n-1}x^{n-1} + a_n x^n$ ($a_n \neq 0$). Let $s_k = \sum_{i=1}^{n} z_i^k$. Then for $1 \leq k \leq n$ we have*

$$s_k a_n + s_{k-1} a_{n-1} + s_{k-2} a_{n-2} + \ldots + k a_{n-k} = 0$$

Note that this means that s_k only depends on coefficients $a_{n-k}, \ldots, a_n$.

The second statement is the base of the iteration step in the Lehmer–Schur algorithm (that can be found in [29], or many textbooks on numerical analysis, like [31], p. 355). In the case of real coefficients we may use this simpler form:

Lemma 3.2.3. *(Lehmer-Schur [29],[31]) Let $f(x) = \sum_{i=0}^{n} a_i x^i$ be a polynomial with real coefficients. This polynomial is expansive if and only if $|a_0| > |a_n|$ and the polynomial*

$$Tf(x) = \sum_{i=0}^{n-1} b_i x^i = \sum_{i=0}^{n-1} (a_0 a_i - a_n a_{n-i}) x^i$$

of degree at most $n-1$ is expansive as well.

We have the following corollary.

Corollary 3.2.4. *With the notations of the preceding lemma, if $f(x) = a_0 + a_1 x + \ldots + a_{n-1}x^{n-1} + a_n x^n$ is an expansive polynomial, and $k < n/2$ an integer, then $T^k f(x)$ is an expansive polynomial as well, $T^k f(x) = \sum_{i=0}^{n-k} b_i x^i$ has degree at most $n-k$. The coefficient b_{n-k} is linear in terms of a_{n-k} and a_k. The constant term b_0 is independent of the coefficients a_l with $k \leq l \leq n-k$. Let $k < n/2$ and let us fix the numerical values of $a_0, \ldots, a_{k-1}$ and $a_{n-k+1}, \ldots, a_n$. Then there exist constants U_1, U_2 and U_3 such that*

$$\begin{aligned} b_0 &= U_1 \,, \\ b_{n-k} &= U_2 a_k + U_3 a_{n-k} \,, \end{aligned}$$

where $U_2 \neq 0$ and $U_1 > 0$

This is easily proved by induction. Using the notations of the previous corollary we can give bounds on a_k.

Corollary 3.2.5. *If* $a_0+a_1x+\ldots+a_{n-1}x^{n-1}+a_nx^n$ *is an expansive polynomial and* U_1, U_2, U_3 *are as above, then expansivity yields* $|b_0| > |b_{n-k}|$. *Thus,* a_k *is in the open interval between the two numbers (in some order)*

$$\frac{-U_1 - U_3 a_{n-k}}{U_2} \quad \textit{and} \quad \frac{U_1 - U_3 a_{n-k}}{U_2}\,.$$

Finally, we have

Lemma 3.2.6. *Let* $z_1, z_2, \ldots, z_n$ *denote the roots of the expansive polynomial* $c_0 + c_1x + \ldots + c_{n-1}x^{n-1} + x^n$, $1 \le k \le n$ *an integer. Then*

$$\left|\sum_{i=1}^{n} z_i^k\right| < n + |c_0|^k - 1\,.$$

The proof is easy since if we replace the roots by their respective kth powers we get another monic expansive polynomial with constant term $\pm c_0^k$. Then we apply lemma 3.1.1.

As a consequence, we have for any expansive polynomial $a_0+a_1x+\ldots+a_{n-1}x^{n-1}+a_nx^n$ the following

Corollary 3.2.7.

$$\left|\sum_{i=1}^{n} z_i^k\right| < n + \left|\frac{a_0}{a_n}\right|^k - 1\,.$$

Now, putting together corollaries 3.2.2 and 3.2.7 we have a bound on a_{n-k} in terms of $a_{n-k+1}, \ldots, a_n$ $(k \le n/2)$.

Corollary 3.2.8.

$$-\frac{1}{k}\left(s_{k-1}a_{n-1} + \ldots + s_1 a_{n-k+1}\right) - \frac{|a_n|}{k}\left(n + \left|\frac{a_0}{a_n}\right|^k - 1\right) < a_{n-k} <$$
$$< -\frac{1}{k}\left(s_{k-1}a_{n-1} + \ldots + s_1 a_{n-k+1}\right) + \frac{|a_n|}{k}\left(n + \left|\frac{a_0}{a_n}\right|^k - 1\right)\,.$$

Note that equality cannot hold on either side of the formula.

One of the two key points used in the algorithm is corollary 3.2.5, which gives bounds on the "low" coefficient a_k in terms of $a_0, \ldots, a_{k-1}$ and $a_{n-k+1}, \ldots, a_n$, where $k < n/2$. The other one is corollary 3.2.8, which gives bounds

on the "high" coefficient a_{n-k} in terms of $a_{n-k+1},\ldots,a_n$ $(k \leq n/2)$. These are dynamic bounds, which are calculated within the algorithm.

We now give the pseudocode for our algorithm. The function EXPANSIVE-SEARCH(n, tc) outputs all expansive monic polynomials with integer-coefficients of degree n with constant term tc.

Function EXPANSIVE-SEARCH(n, tc)

```
1 coeff ← zero-vector[n + 1] ;
2 coeff[0] ← tc ;
3 coeff[n] ← 1 ;
4 ITERATION-HIGH-COEFFICIENT(coeff, 0)
```

The function ITERATION-HIGH-COEFFICIENT(*coeff*,k) takes the elements of *coeff* at indices $0, 1, \ldots, k$ and $n-k, \ldots, n-1, n$ $(k < n/2)$, as input. It outputs all expansive polynomials $c_0 + c_1x + \ldots + c_{n-1}x^{n-1} + x^n$ with $c_0 = coeff[0]$, ..., $c_k = coeff[k]$ and $c_{n-k} = coeff[n-k]$, ..., $c_n = coeff[n]$.

Function ITERATION-HIGH-COEFFICIENT(*coeff*, k)

```
1  n ← sizeof(coeff) − 1 ;
2  if n = 2k + 1 then
3      if IS-EXPANSIVE(coeff) then
4          output coeff
5  else
6      min,max ← DETERMINE-HIGH-RANGE(coeff, k) ;
7      for i← min to max do
8          coeff[n − k − 1] ← i ;
9          ITERATION-LOW-COEFFICIENT(coeff, k)
10     end
```

The function ITERATION-LOW-COEFFICIENT(*coeff*, k) takes the elements of *coeff* at indices $0, 1, \ldots, k$ and $n-k-1, \ldots, n-1, n$, as input. It outputs all expansive polynomials $c_0 + c_1x + \ldots + c_{n-1}x^{n-1} + x^n$ with $c_0 = coeff[0]$, ..., $c_k = coeff[k]$ and $c_{n-k-1} = coeff[n-k-1]$, ..., $c_n = coeff[n]$.

Function ITERATION-LOW-COEFFICIENT(*coeff*, k)

```
1  n ← sizeof(coeff) − 1 ;
2  if n = 2k + 2 then
3      if IS-EXPANSIVE(coeff) then
4          output coeff
5  else
6      min,max← DETERMINE-LOW-RANGE(coeff, k) ;
7      for i← min to max do
8          coeff[k + 1] ← i ;
9          ITERATION-HIGH-COEFFICIENT(coeff, k+1)
10     end
```

We take benefit of corollary 3.2.8 and in the function DETERMINE-HIGH-RANGE.

Function DETERMINE-HIGH-RANGE(*coeff*, k)

```
1  Use corollary 3.2.8 to get bounds lower-bound, upper-bound ;
2  return ⌊lower-bound + 1⌋, ⌈upper-bound − 1⌉
```

In the function DETERMINE-LOW-RANGE we use corollary 3.2.4.

Function DETERMINE-LOW-RANGE(*coeff*, k)

```
1  Perform k steps of the Lehmer–Schur algorithm, computing
   numerically with the fixed coefficients, and computing symbolically
   with the unfixed coefficients ;
2  Use corollary 3.2.5 to get bounds lower-bound, upper-bound ;
3  return ⌊lower-bound + 1⌋, ⌈upper-bound − 1⌉
```

Finally, the function IS-EXPANSIVE(*coeff*) can be any algorithm that decides the expansivity of a polynomial. This can be the Lehmer–Schur algorithm[29], or the algorithm given in chapter 2 and [10].

The pseudocode above should only be considered as the skeleton of the algorithm. The implementation of the algorithm used several small ideas to improve performance, like for example dynamic programming and further bounds derived from chapter 2 and [10].

The correctness of the algorithm is implied by the statements at the beginning of the section. The running time of the algorithm is hard to analyze theoretically. The number of polynomials checked is bounded by the product $\prod_{k=1}^{n-1}(2B(2,n,k)-1)$ using the notations of the previous section.

Experiments show that the actual number is a lot smaller, but increases sharply with n. So the unpleasant asymptotic behavior remains, but even so, we managed to reach the degree 11 instead of 8.

3.3 The second search algorithm

The search algorithm presented in the previous section performs well when the degree is small. But even for constant term 2, the degrees beyond 11 seem inaccessible.

In order to extend the search to larger constant terms and higher degrees, a different algorithm has to be applied. Prior application of the algorithm presented below for finding CNS polynomials is unknown to me, although similar algorithms already appeared in numeration research: for finding Pisot and Salem numbers in intervals of the real line, Boyd [7] used a similar method.

The proof that the algorithm is correct is essentially the same as in [16] and [15]. The method originated in two papers by Schur [33], [34], in which he examined power series of bounded holomorphic functions in the unit disk. His methods were generalized by Dufresnoy and Pisot [16] for meromorphic functions that have a single pole in the unit disk, and by Chamfy [15] for the case of several poles. In the description of the algorithm, we follow Boyd [7].

We will denote the reciprocal polynomial of a polynomial P by P^*. Take a monic integer polynomial $P(z) = c_0 + c_1 z + \cdots + z^n$. The key idea is to consider $P^*(z) = 1 + c_{n-1}z + \ldots + c_1 z^{n-1} + c_0 z^n$ and the quotient $f(z) = \pm P(z)/P^*(z)$, where the sign is chosen so that $f(0) > 0$. The quotient has modulus 1 on the unit circle, and has a power series $u_0 + u_1 z + u_2 z^2 + \cdots$ in 0 with integer coefficients, with constant term $\pm c_0$. The search algorithm essentially works by giving lower and upper bounds on u_n that depend on $u_0, u_1, \ldots, u_{n-1}$. The following theorem holds.

Theorem 3.3.1. *Let $f(z) = \pm P(z)/P^*(z) = u_0 + u_1 z + \cdots + u_{n-1}z^{n-1} + u_n z^n + \cdots$, where P is monic expansive of degree at least n, and the sign is chosen so that $u_0 > 0$.*

- *There exists a unique monic expansive polynomial $Q(z)$ of degree n and a $v_n = v_n(u_0, u_1, \ldots, u_{n-1}) \in \mathbb{R}$ so that the power series expansion of*

$Q(z)/Q^*(z)$ *begins with* $u_0 + u_1 z + \cdots + u_{n-1}z^{n-1} + v_n z^n$.

- *There exists a unique monic expansive polynomial* $R(z)$ *of degree* n *and* $w_n = w_n(u_0, u_1, \ldots, u_{n-1}) \in \mathbb{R}$ *with* $-R(z)/R^*(z) = u_0 + u_1 z + \cdots + u_{n-1}z^{n-1} + w_n z^n + \cdots$.

We have $v_n \leq u_n \leq w_n$. *Equality holds on the left if and only if* $P = Q$ *and equality holds on the right if and only if* $P = R$.

Given $u_0, u_1, \ldots u_{n-1}$, the coefficients of Q and R can be determined by a system of linear equations. For the calculations it is more convenient to use the following relations:

$$\begin{aligned} Q_{n+1}(z) &= (1+z)Q_n(z) - \frac{u_n - v_n}{u_{n-1} - v_{n-1}} z Q_{n-1}(z), \\ R_{n+1}(z) &= (1+z)R_n(z) - \frac{u_n - w_n}{u_{n-1} - w_{n-1}} z R_{n-1}(z). \end{aligned}$$

For the proof of the theorem and the recurrence relations we refer to [16] or [15], where analogous statements are proved.

Fix an integer u_0. Using the theorem, one can build a rooted tree of Taylor-polynomials. The root is u_0, and the children of a node $(u_0 + u_1 z + \cdots + u_{n-1}z^{n-1})$ are $(u_0 + u_1 z + \cdots + u_n z^n)$ for $v_n(u_0, u_1, \ldots, u_{n-1}) \leq u_n \leq w_n(u_0, u_1, \ldots, u_{n-1})$, $u_n \in \mathbb{Z}$, if $v_n < w_n$. If $v_n \geq w_n$, then the node is a leaf. This tree can be built using the recurrence relations above, and traversed to any depth n using a tree traversal algorithm (the authors used depth first search). Every monic expansive polynomial $P(z) \in \mathbb{Z}[z]$ of degree at most n and constant term $\pm u_0$ can be found by solving $P(z)/P^*(z) = f(z)$ for some leaf f of the tree up to level n.

We illustrate the algorithm for the binary case in figure 1. The nodes of the tree are polynomials of form $u_0 + u_1 z + \cdots + u_n z^n$ that are truncations of the power series of P/P^* for some P monic expansive polynomial with integer coefficients and constant term $\pm u_0$. Expansive polynomials of degree n can be found by looking at leaves at level n. Linear and quadratic expansive polynomials P are shown in parentheses below the corresponding leaves P/P^*.

We observed that when c_0 becomes large, the tree becomes extremely wide, which slows down the algorithm. Although asymptotically (with the

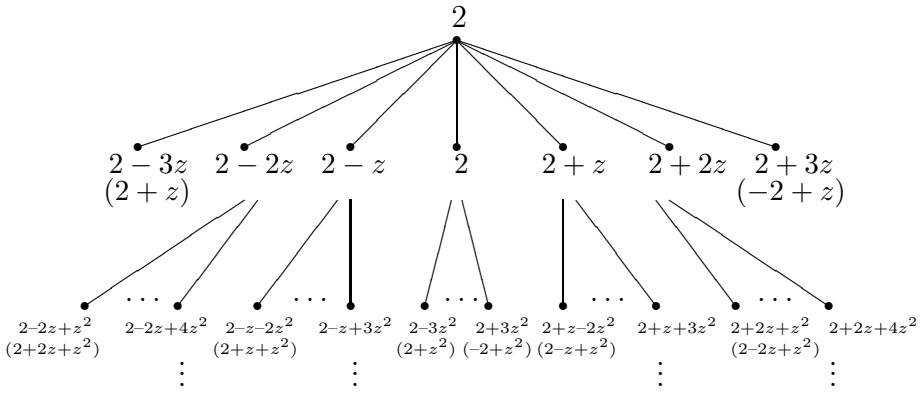


Figure 3.1: The first two levels of the tree of Taylor-polynomials, $u_0 = 2$.

degree) the tree traversal algorithm performs faster than the one in section 3.2 , empirical data suggest that for large c_0 and small degrees the original algorithm is faster.

3.4 CNS and semi-CNS polynomials

3.4.1 CNS polynomials

Given an expansive polynomial, there are several ways of deciding if it is a CNS polynomial or not. We used Brunotte's algorithm and an enhanced version of a method that performs an exhaustive search for possible cycles in a region. The detailed description of these decision algorithms can be found in chapter 4. Experiments show that the time complexity of both decision algorithms grows rapidly with the degree, and for Brunotte's algorithm this is also true for space complexity. It can be observed however that negative answers are obtained quickly by both algorithms.

Let $c(x) = c_0 + c_1x + \cdots + c_{n-1}x^{n-1} + x^n$ be given. If the dominant condition $c_0 > |c_1| + \cdots + |c_{n-1}|$ holds then there are sets of conditions, by which $c(x)$ is CNS. For $n = 3$ and $n = 4$ see [5, 32], for $n = 5$ and higher see [5]. Under the dominant condition both of the mentioned decision algorithms are fast. Moreover, there are other known methods as well ([4], [5], Theorem 4.3, Corollary 4.4). Without the dominant condition we know the following results:

- The theorem of B. Kovács, proved in [27], see the introduction of chapter 4.

- H. Brunotte characterized CNS trinomials in [9]. Let $n > 2$.

  (i) The polynomial $x^n + bx + c$ is a CNS polynomial if and only if $-1 \leq b \leq c - 2$.

  (ii) Let $1 < q < n$, $q \nmid n$. The polynomial $x^n + bx^q + c$ is a CNS polynomial if and only if $0 \leq b \leq c - 2$.

The next theorem shows that CNS polynomials of degree n can be "lifted" to yield polynomials of degree nk. This is also proved by Brunotte [9] in a different way.

Theorem 3.4.1. *Let $c(x) = c_0 + c_1x + c_2x^2 + \cdots + c_nx^n$ be a CNS polynomial. Then, for any integer $k \geq 1$, $c(x^k) = c_0 + c_1x^k + c_2x^{2k} + \cdots + c_nx^{nk}$ is also a CNS polynomial.*

Proof. Identify $\mathbb{Z}^{nk}$ with the quotient ring $\mathbb{Z}[x]/c(x)\mathbb{Z}[x]$. Every element p of this ring can be written in the form

$$p = \sum_{i=0}^{nk-1} a_i x^i = \sum_{i=0}^{k-1} x^i \sum_{j=0}^{n-1} a_{kj+i} x^{kj+i} .$$

From the CNS property of $c(x)$ one has the radix expansions

$$\sum_{j=0}^{n-1} a_{kj+i} x^j = \sum_{j=0}^{l_i} d_j^{(i)} x^j \quad 0 \leq i \leq k-1 ,$$

where $l_i \in \mathbb{N}$ and $d_j^{(i)}$ are digits. But then

$$p = \sum_{i=0}^{k-1} \sum_{j=0}^{l_i} d_j^{(i)} x^{kj+i}$$

is a radix expansion for $c(x^k)$. Unicity of the expansion can be proved analogously. □

3.4.2 Semi-CNS polynomials

It is well-known and easy to show that a monic polynomial with negative constant term cannot be a CNS polynomial. A classical example is $x - 10$, meaning that the ordinary decimal number system is not a CNS, since negative numbers have no representation. We introduce the following definition:

Definition 3.4.2. *Let c_0 be an integer. A polynomial $P(x) = c_0 + c_1 x + \cdots + x^n$ is called a semi-CNS polynomial if for the digit set $D = \{0, 1, \ldots |c_0| - 1\}$ the finite expansions $\left\{ \sum_{i=0}^{l} d_i x^i \mid l \in \mathbb{N}, d_i \in D \right\}$ form an additive semigroup.*

With this definition, $x - c_0$ becomes a semi-CNS polynomial for $c_0 > 1$. With a small modification of Brunotte's algorithm ([5],[8]) it is easy to decide the semi-CNS property.

Suppose that D is a complete residue system for an expansive $n \times n$ matrix M. For $p \in \mathbb{Z}^n$, we denote with $\tau(p)$ the unique element $q \in \mathbb{Z}^n$ for which there exists $d \in D$ so that $p = Mq + d$.

Theorem 3.4.3. *(Brunotte's algorithm for semi-CNS) Let $c(x)$ be an expansive polynomial, M its companion matrix and D the canonical digit set. Let τ be the above map. If there exists a set $E \subseteq \mathbb{Z}^n$ such that*

(i) $(1, 0, \ldots, 0) \in E$,

(ii) *For all $e \in E$ and $d \in D$ we have $\tau(e + d) \in E$,*

(iii) *For all $e \in E$ there exists a positive integer k so that $\tau^k(e) = 0$,*

then $c(x)$ is a semi-CNS polynomial.

The proof is identical to Brunotte's proof in [8]. Note that for the CNS property $(\pm 1, 0, \ldots, 0) \in E$ is needed in (i). The construction of such a set E (or the proof of its non-existence) is usually done by choosing an initial set E_0, and enlarging it until it satisfies (ii). Then one checks whether (iii) holds. Below we give some properties of semi-CNS polynomials.

It is noted in [2] that for CNS candidate polynomials the final set E is the same for $E_0 = \{(1, 0, \ldots, 0)\}$ and $E_0 = \{(\pm 1, 0, \ldots, 0)\}$. This means that when the constant term of a polynomial is positive, the semi-CNS and CNS properties are equivalent.

For negative constant terms one has the following condition.

Theorem 3.4.4. *Let* $c_0 + c_1x + \ldots + c_{n-1}x^{n-1} + x^n$ *be an expansive polynomial with* $c_0 < 0$*. If no other coefficient is negative, then it is a semi-CNS polynomial.*

Proof. Expansivity implies $c_1 + c_2 + \cdots c_{n-1} + 1 < |c_0|$, otherwise there would be a real root between 0 and 1. Using Brunotte's basis (see [8]), the set $E = \{(\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n) \mid \varepsilon_i = 0 \text{ or } 1 \text{ for } 1 \leq i \leq n\}$ satisfies the conditions of theorem 3.4.3. □

Statement 3.4.1. *Let* k, n *be positive integers. The number of polynomials of degree* n *and constant term* $-k$ *satisfying the conditions of theorem 3.4.4 is* $\binom{n+k-3}{k-2}$.

Proof. It is easy to see that $c_1 + c_2 + \cdots c_{n-1} + 1 < |c_0| = k$ is also a sufficient condition of expansivity. We have to determine the number of non-negative tuples $(c_1, c_2, \ldots, c_{n-1})$ that sum up to at most $k - 2$. That is equal to the number of ordered partitions of $k - 2$ into n parts, the expression above. □

3.5 Computational Results

The search algorithm was implemented in C/C++, using multi-precision arithmetic. It can be parallelized using the inherent parallel nature of the algorithm. Parts of the search were performed on the Desktop Grid of the Hungarian Academy of Sciences [35, 26]. The decision algorithms were implemented in C++, much attention being paid for memory-efficiency in the case of Brunotte's algorithm. They were performed on desktop computers.

Since the results for degrees up to 8 were known earlier ([24]), we extend this list with the binary CNS polynomials for degrees 9, 10 and 11 in appendix A.

There are altogether 192 expansive polynomials of degree 9 with constant term 2. 12 of them are CNS polynomials. There are altogether 623 expansive polynomials of degree 10 with constant term 2. 42 of them are CNS polynomials. There are altogether 339 expansive polynomials of degree 11 with constant term 2. 11 of them are CNS polynomials.

There are 1085 expansive polynomials of degree 12 with constant term 2. The number of CNS polynomials among them is between 56 and 66. For

c_0\Degree	2	3	4	5	6	7	8
2	5/4	7/4	29/12	29/7	105/25	95/12	309/32
3	7/5	25/13	131/47	310/75	1413/242	2619/322	10273/816
4	9/6	51/26	327/108	1240/286	6749/1033	20129/2194	
5	11/7	85/43	655/200	3369/735	21671/3010		
6	13/8	127/63	1155/332	7468/1546	55785/7106		
7	15/9	177/88	1829/509	14411/2876	122633/14606		
8	17/10	235/115	2747/742	25265/4887	241391/27263		
9	19/11	301/147	3905/1025	41331/7802			
10	21/12	375/182	5379/1378	63959/11824			

Table 3.2: The number of expansive and CNS polynomials, ordered by degree and constant term. (The missing cells are being computed.)

$-c_0$\Degree	2	3	4	5	6	7	8
2	1/1	7/1	7/1	29/1	23/1	95/1	57/1
3	3/2	25/3	55/4	310/5	563/6	2619/7	4091/8
4	5/3	51/6	179/10	1240/15	3605/21	20129/28	
5	7/4	85/10	421/20	3369/35	13501/56		
6	9/5	127/15	795/35	7468/70	37853/126		
7	11/6	177/21	1353/56	14411/126	88501/252		
8	13/7	235/28	2099/84	25265/210	182235/462		
9	15/8	301/36	3083/120	41331/330			
10	17/9	375/45	4349/165	63959/495			

Table 3.3: The number of expansive and semi-CNS polynomials, ordered by degree and constant term. (The missing cells are being computed.)

degree 13 the number of CNS polynomials is between 14 and 15 out of 526 expansives, and for degree 14 the number is between 29 and 45 out of 1283. The authors find it very likely that the precise answers are 66, 14 and 45. The uncertainty is due to the fact that the set of witnesses in Brunotte's algorithm becomes very large and does not fit into computer memory.

In tables 1 (resp. 2) we list the number of expansive/CNS (resp. semi-CNS) polynomials by degree n and constant term c_0 (resp. $-c_0$).

We presented a fast algorithm for finding all monic expansive polynomials with fixed degree and constant term. In our future work we wish to extend the results shown in tables 1 and 2. We also plan to examine the obtained expansive polynomials for number system property using symmetrical digit set and other kinds of non-canonical digit sets. For a characterization of matrices rather than just polynomials, the determination of similarity classes of matrices with fixed characteristic polynomials is needed. That is also one of our computational challenges in the future.

Chapter 4

GNS Decision and GNS Classification

We present algorithms for the decision and classification of generalized number systems. In the first part of the chapter, an algorithm using an enclosing parallelepiped for the set of fractions is considered. We mainly focus on minimizing the number of lattice points in the parallelepiped by choosing the basis optimally. In the second part we generalize Brunotte's canonical number system decision algorithm for generalized systems and we extend the results for the classification problem. Finally, we compare the algorithms by their performances in practice.

Throughout the chapter we will assume that the pair (M, D) meets the following conditions: M is expansive and D is a complete residue system modulo M.

The decision problem for (M, D) asks if they form a GNS or not. The more general classification problem means finding all periods (sometimes also called cycles in the sequel).

No general fast algorithm is known for these problems. Special cases have been treated with success. Important results are known for the canonical case. Quadratic CNS polynomials were classified by I. Kátai and B. Kovács [22, 21] and independently by W.J. Gilbert [17]. Cubic and quartic CNS polynomials were investigated by S. Akiyama, H. Brunotte, A. Pethő [3], H. Brunotte [8], and K. Scheicher, J.M. Thuswaldner [32].

The following was discovered by B. Kovács for irreducible polynomials

[27], and slightly generalized by A. Pethő [30]. Let $c(x) = c_0 + c_1 x + \ldots + c_{n-1}x^{n-1} + x^n \in \mathbb{Z}^n$. If $c_0 \geq 2$, $c_{n-1} \leq \cdots \leq c_1 \leq c_0$ and $c(x)$ is not divisible by a cyclotomic polynomial, then $c(x)$ is a CNS polynomial. S. Akiyama, A. Pethő [4], S. Akiyama, H. Rao [5] and K. Scheicher, J.M. Thuswaldner [32] showed characterization results under the "dominant" condition $c_0 > |c_1| + \cdots + |c_{n-1}|$.

Below, we give two algorithms for the decision/classification problem in the general case. The original version of the first method, using a covering of the set of fractions, was first applied in [23]. It gives lower and upper bounds on the coordinates of periodic points. The method is combined here with a base transformation using a randomized algorithm in order to improve the bounds.

The second method is a generalization of Brunotte's CNS decision algorithm [8]. We show how one can modify it so that it handles arbitrary digit sets and the classification problem. We also give detailed pseudocode implementation of the algorithm.

Finally, we compare the methods by their practical performance.

4.1 Covering the set of fractions

The method is based on a covering construction that appeared in [23]. The set of fractions is defined as

$$H = \left\{ \sum_{i=1}^{\infty} M^{-i} d_{j_i} \mid d_{j_i} \in D \right\}.$$

Let x be a periodic point, with period length k. Then $\phi^k(x) = x$, thus, from the definition of ϕ, $M^{-k}x = x + \sum_{i=1}^{k} M^{-i} d_{j_i}$ for some j_i. If we denote the

$n \times n$ identity matrix by I, we have

$$\begin{aligned} -x &= \left(I - M^{-k}\right)^{-1} \left(\sum_{i=1}^{k} M^{-i} d_{j_i}\right) \\ &= \left(I + M^{-k} + M^{-2k} + \cdots\right) \left(\sum_{i=1}^{k} M^{-i} d_{j_i}\right) \\ &= \sum_{l=0}^{\infty} \sum_{i=1}^{k} M^{-lk-i} d_{j_i} \in H\,, \end{aligned}$$

where absolute convergence and the validity of the calculation follows from the expansivity of M.

If we find lower and upper bounds $l_m, u_m, (m = 1, \ldots, n)$ such that $-H \subseteq T = \{(x_1, x_2, \ldots, x_n) \mid l_i \leq x_i \leq u_i\}$, then all periodic points lie in the brick T. We briefly describe a method for finding such bounds.

Let $\|\cdot\|$ denote the maximum norm in $\mathbb{R}^n$. Choose a positive real number $c < 1$ and $k \in \mathbb{N}$ with $\left\|M^{-k}\right\| = \delta \leq c$.

Let $p_m : \mathbb{R}^n \to \mathbb{R}$ denote the projection in the mth coordinate for $m = 1, \ldots, n$. Let

$$\begin{aligned} a_{m,j} &= \min_{d \in D} p_m(M^{-j} d) \text{ and} \\ b_{m,j} &= \max_{d \in D} p_m(M^{-j} d) \end{aligned}$$

for $j = 1, \ldots, k$. Let

$$W = \left\{ (x_1, \ldots, x_n)^T \,\middle|\, -\sum_{j=1}^{k} b_{m,j} \leq x_m \leq -\sum_{j=1}^{k} a_{m,j} \right\}.$$

Clearly, $-\sum_{i=1}^{k} M^{-i} d_i \in W$ for an arbitrary sequence $d_i \in D$. So $-H \subseteq W + M^{-k}W + M^{-2k}W + \cdots$. Now let A be the largest absolute value among the $\sum a_{m,j}$ and $\sum b_{m,j}$. We have $\|x\| \leq A$ for every $x \in W$, and $\left\|M^{-ik}\right\| \leq \delta^i$

for $i \geq 1$. Let $\gamma = A(\delta + \delta^2 + \cdots) = A\frac{\delta}{1-\delta}$. Then

$$l_m = -\sum_{j=1}^{k} b_{m,j} - \gamma \ , \text{ and}$$

$$u_m = \sum_{j=1}^{k} a_{m,j} + \gamma$$

are appropriate lower and upper bounds. The number of periodic lattice points is bounded by $\mathrm{Vol}(M, D) = \prod_{m=1}^{n} \lfloor u_m - l_m + 1 \rfloor$. Classification can be done by examining the points in the brick determined by these bounds. The running time of the classification is roughly proportional to $\mathrm{Vol}(M, D)$. Choosing δ sufficiently small (0.1 seemed reasonable in our experiments) guarantees that the bounds obtained from the covering brick are almost always sharp. We cannot directly reduce the value of $\mathrm{Vol}(M, D)$. Clearly, a basis transformation (applied to both M and D) does not affect the periodicity of points. Hence we can perform a basis transformation *before* calculating $\mathrm{Vol}(M, D)$. Below, we give an algorithm for choosing a basis that yields significant decrease in the value of $\mathrm{Vol}(M, D)$. If T is a transformation matrix, we will simply denote $\mathrm{Vol}(TMT^{-1}, TD)$ by $\mathrm{Vol}(T)$ (M and D are fixed). For a matrix U, we denote by $incr(U, i, j)$ and $decr(U, i, j)$ the matrix that differs from U only at position (i, j) by $+1$ and -1, respectively. The main algorithm uses the following routine:

Function CHANGE-AT-POSITION(M, D, i, j, T)

```
1  U ← T ;
2  OldVol ← Vol(T) ;
3  while Vol(incr(U, i, j)) < OldVol do
4      U ← incr(U, i, j) ;
5      OldVol ← Vol(U) ;
6  end
7  V ← T ;
8  OldVol ← Vol(T) ;
9  while Vol(decr(V, i, j)) < OldVol do
10     V ← decr(V, i, j) ;
11     OldVol ← Vol(V) ;
12 end
13 return [U, V]
```

This function increases (decreases) $T_{i,j}$ as long as smaller volumes are received. The best matrix is put into U (resp. V).

Algorithm FIND-BASIS-TRANSFORMATION(M, D)

Input: M, D, *CandNum, ProbeNum, NoImprLim, TargetVol*
Output: Upper triangular basis transformation matrix T

```
1  Candidates ← CandNum sized list of identity matrices ;
2  NoImpr ← 0 ;
3  while NoImpr < NoImprLim and Vol(Candidates[1]) > TargetVol do
4      NewCands ← [] ;
5      forall T ∈ Candidates do
6          for k ← 1 to ProbeNum do
7              Choose random position (i, j) above the diagonal ;
8              [U, V] ← CHANGE-AT-POSITION(M, D, i, j, T) ;
9              NewCands ← NewCands + [U, V] ;
10         end
11     end
12     Sort NewCands increasingly by values of Vol(T)
13     if Vol(NewCands[1]) < Vol(Candidates[1]) then
14         NoImpr ← 0 ;
15     else
16         NoImpr ← NoImpr+1 ;
17     Copy first CandNum elements of NewCands into Candidates ;
18 end
19 return Candidates[1]
```

The computation goes along several branches. It keeps track of the best *CandNum* transformation matrix candidates and, in each iteration, it tries to improve them *probeNum* times. The computation ends when no improvement occurs for *NoImprLim* consecutive iterations, or when a sufficiently small volume, lower than *TargetVol* is reached.

The running time depends on these parameters. Practically, it is worth letting this algorithms run for a longer time, since the main computation in GNS classification is the exhaustive examination of the covering set.

Below we give a detailed example of the algorithm.

We use the shorthand $((i_1, j_1, x_1), ..., (i_k, j_k, x_k))$ to denote a matrix that differs from the identity matrix at positions (i_h, j_h), holding the values x_h, $(h = 1 \dots k)$.

We input

$$M = \begin{pmatrix} 0 & 0 & 0 & 0 & -7 \\ 1 & 0 & 0 & 0 & 6 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix},$$

D is canonical, $CandNum = 2$, $ProbeNum = 2$, $NoImprLim = 3$, $TargetVol = 5000$. The initial volume is 17500.

After the first iteration of the while loop (lines 3–18) we have $Candidates = [((3, 4, -1)), ((4, 5, -1))]$, $\mathrm{Vol}(Candidates) = [10500, 10500]$.

This means that the random method found two distinct positions where a change in the matrix results in a smaller volume.

In the next iteration it tries to improve the first element of *Candidates*. No improvement is found for this matrix. For the second one, however, positions $(3, 4)$ and $(1, 4)$ were chosen, both positions are incremented and decremented. Incrementing at position $(3, 4)$ does not improve, but decrementing it to -1 yields volume 6525, incrementing it further gives 10500. Decrementing at position $(1, 4)$ does not improve (12375 for -1), but incrementing it yields the volumes 9000, 7875, 6750, 6000 and 4500, after the next iteration the volume becomes 6750. After sorting, we have $Candidates = [((4, 5, -1)(1, 4, 5)), ((4, 5, -1), (3, 4, -1))]$, $\mathrm{Vol}(Candidates) = [4500, 6525]$.

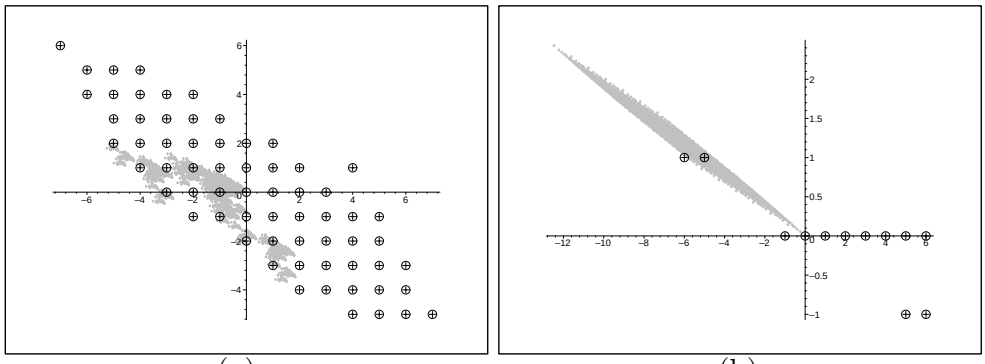
Since 4500 is smaller than *TargetVol*, we stop. We succeeded in reducing the size of the covering set to 4500 from the original volume 17500. The transformed matrix is

$$\begin{pmatrix} 0 & 0 & 5 & 0 & -7 \\ 1 & 0 & 0 & -5 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & -1 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix},$$

and the digit set remains canonical. We note that letting the algorithm run a bit longer (through playing with the parameters) gives an even smaller volume, below 1000.

We give two more examples in Figure 1.

In order to test the algorithm on a large data set, we used algorithms in [12] for the generation of random expansive polynomials. Figure 2 shows



(a) (b)

Figure 4.1: Decreasing the volume of the covering set
Example (a) shows $M = \left(\begin{smallmatrix} 1 & -2 \\ 1 & 3 \end{smallmatrix}\right)$ with $D = \{(0,0),\ (1,0),\ (0,1),\ (4,1),\ (-7,6)\}$. Changing the basis to $\{(1,0),(1,1)\}$ decreases the volume from 42 to 24. Example (b) shows $M = \left(\begin{smallmatrix} 0 & -7 \\ 1 & 6 \end{smallmatrix}\right)$ with canonical digit set. Replacing basis vector $(0,1)$ with $(-5,1)$ gives volume 4 instead of 64. In both pictures, grey areas represent $-H$. The circles represent the set E in Brunotte's algorithm (cf. function CONSTRUCT-SET-E).

the average improvement in orders of magnitude, as the constant term takes values $6, \ldots, 100$, and dimension is changed from 3 to 8.

A possible way to improve the algorithm is to perform the algorithm for the transformed system with a lower triangular transformation matrix as well, and combine the two transformations. We performed it in four different ways: only upper, only lower, upper followed by lower and lower followed by upper triangular transformation matrices. The average reduction in orders of magnitude were $3.218, 0.603, 3.417$ and 3.168, respectively. The smaller figure at lower triangular matrices may be due to the special form of test cases: for their importance only companion matrices were considered. We will continue our experiments with arbitrary matrices to see if this caused the difference. Generating random expansive matrices seems difficult. One can apply an integer basis transformation to the companion matrix of a polynomial, but we know from [28] and [36] that this method generates all expansive matrices only if the class number of the order corresponding to the polynomial is 1.

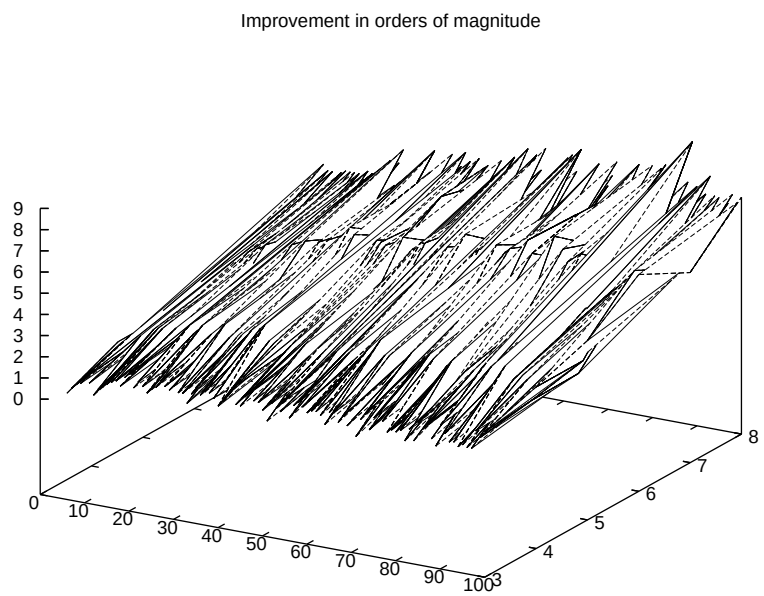


Figure 4.2: The average improvement in the volume of the covering set after algorithm FIND-BASIS-TRANSFORMATION, expressed in orders of magnitude (base 10).

4.2 Generalization of Brunotte's algorithm

4.2.1 The decision algorithm

Brunotte's canonical number system decision algorithm first appeared in [8]. An alternative treatment of the algorithm can be found in [5]. We reformulate and prove Brunotte's results on decision in the case of arbitrary generalized number systems. The statement of the theorem and the proof idea are slightly modified compared to the original version. The generalized algorithm starts with the construction of a set E. We use the function ϕ defined in the introduction.

Function CONSTRUCT-SET-E(M, D)

```
E ← D , E' ← ∅ ;
while E ≠ E' do
    E' ← E ;
    forall e ∈ E and d ∈ D do
        put φ(e + d) into E ;
    end
end
return E
```

Proposition 4.2.1. *The above algorithm terminates.*

Proof. Let $x \in \mathbb{Z}^n$, and let $d \in D$ be the digit with $\phi(x) = M^{-1}(x - d)$. Since M^{-1} is contractive, there exists a norm in $\mathbb{R}^n$ with induced norm $\|M^{-1}\| = r < 1$. Let $m = \max_{d,d' \in D} \|d - d'\|$. If $\|x\| \leq mr/(1 - r)$, then $\|\phi(x + d')\| = \|M^{-1}(x + d' - d)\| \leq r\|x\| + rm \leq mr^2/(1 - r) + mr = mr/(1 - r)$. Thus, no vector with $\|x\| > mr/(1 - r)$ will ever get into E. □

Lemma 4.2.2. *Let us assume that every vector in E has finite expansion. Let $e \in E$ and $x = \sum_{i=0}^{k} M^i d_{j_i}$ an arbitrary vector with finite expansion. Then $e + x$ has finite expansion as well.*

Proof. The proof goes by induction on k. If $k = 0$, then $x \in D$. Thus, $\phi(e + x) \in E$ by the construction of E, and it has finite expansion by the assumption. Then $x + e$ has finite expansion as well.

If $k > 0$, then put $y = \phi(x)$, so that $x = My + d_{j_0}$. By the construction of E, $e + d_{j_0} = Me' + d'$ for some $e' \in E$ and $d' \in D$, giving

$$\phi(e + x) = \phi(e + My + d_{j_0}) = \phi(My + Me' + d') = y + e' .$$

It suffices to show that $y + e'$ has finite expansion, but that follows from the induction hypothesis. □

The above lemma and proof are best understood if we look at E as the set of possible carries that occur when two points with finite expansion are added. We can now give the generalized algorithm for number system decision. Let the set $B = \{(0, \ldots, 0, \pm 1, 0, \ldots, 0)\}$ (i.e. the n basis vectors and their opposites).

Function SIMPLE-DECIDE(M, D)

```
1 E ← CONSTRUCT-SET-E(M, D) ;
2 forall p ∈ B ∪ E do
3     if p has no finite expansion then
4         return false
5 end
6 return true
```

Theorem 4.2.3. *Algorithm* SIMPLE-DECIDE *returns true if and only if* (M, D) *is a number system.*

Proof. It is clear that a vector with no finite expansion yields rejection. On the other hand, if every vector in E has finite expansion, then, by lemma 4.2.2, vectors with finite expansion form an additive sub-semigroup $S \leq \mathbb{Z}^n$. Take a set that generates the whole lattice as a semigroup, like B. If this set has finite expansion, then S contains it, giving $S = \mathbb{Z}^n$. □

4.2.2 The classification algorithm

For classification, one needs to find *all* periodic points. Of course, if the decision algorithm accepts (M, D), we know that the origin is the only periodic point. If it rejects, we have a point without finite expansion, which is either a point of the semigroup-basis, or a point in set E. Thus, following the orbit of this point, we also find a non-zero periodic point, a witness.

A first idea for classification is to search the set $E \cup B$ for points with no finite expansion. Unfortunately, there can be periodic points that evade this search. We show this by the following simple example. Take the expansive polynomial $3 - 3x + x^2$ with canonical digit set. Then there are two non-zero periodic points, both of period 1: $(-4, 2)$ and $(-2, 1)$. Only the latter one is in the set $E \cup B$. $5 - 4x + x^2$ is another example.

The idea of the classification algorithm is to increase the set E so that it contains all periodic points. This is done by iterating the function CONSTRUCT-SET-E several times, replacing set D by a larger set including already known periodic points.

We now give the algorithm for classification.

Function SIMPLE-CLASSIFY(M, D)

```
1  D ← D ;
2  finished ← false ;
3  while not finished do
4      E ← CONSTRUCT-SET-E(M, D) ;
5      finished ← true ;
6      forall p ∈ E ∪ B do
7          if p does not run eventually into D then
8              put newly found periodic points into D ;
9              finished ← false ;
10     end
11 end
12 return D \ D (the set of non-zero periodic points)
```

Lemma 4.2.4. *The classification algorithm terminates.*

Proof. Note that $\mathcal{D}$ may only contain periodic points and digits, so CONSTRUCT-SET-E is only called a finite number of times. It remains to show that the execution of this call terminates for any $\mathcal{D}$. This follows from the finiteness of $\mathcal{D}$ in the same way as in proposition 4.2.1. □

Lemma 4.2.5. *Assume that every vector in $\mathcal{E}$ eventually runs into $\mathcal{D}$. Let $e \in \mathcal{E}$, and $x = \sum_{i=0}^{k-1} M^i d_{j_i} + M^k d$ an arbitrary vector that runs into $\mathcal{D}$ (i.e. $d \in \mathcal{D}$). Then $e + x$ also runs into $\mathcal{D}$ eventually.*

Proof. The proof goes by induction on k. The case $k = 0$ follows from the construction of $\mathcal{E}$. If $k > 0$, then let $y = \phi(x)$. We know that $e + d_{j_0} = Me' + d'$ for some $e' \in \mathcal{E}$ and $d' \in D$, giving

$$\phi(e + x) = \phi(e + My + d_{j_0}) = \phi(My + Me' + d') = y + e' \quad ,$$

so the lemma holds by the induction hypothesis for e' and y. □

Theorem 4.2.6. *Algorithm* SIMPLE-CLASSIFY *is correct.*

Proof. The only thing to show is that on termination $\mathcal{D}$ contains every periodic point. We show that every vector in $\mathbb{Z}^n$ eventually runs into $\mathcal{D}$. This is true for $\mathcal{E}$ by the previous lemma, so the points eventually running into $\mathcal{D}$ form an additive semigroup. Since this semigroup contains B, the proof is complete. □

4.2.3 Implementation

We give directly programmable pseudocode below that optimizes the decision and classification algorithms.

We will use the notation $\phi(X+Y) = \{\phi(x+y) \mid x \in X, y \in Y\}$. Suppose that the loop in lines 4–6 of function CONSTRUCT-SET-E is executed exactly k times. Let us denote the set E before the ith execution of the loop by E_i, let $E_0 = \varnothing$ and $\Delta E_i = E_i \setminus E_{i-1}$. Then we have

$$D = E_1 \subsetneq E_1 \subsetneq E_2 \ldots \subsetneq E_k \,,$$

and

$$\begin{aligned} E_{i+1} &= \phi(E_i + D) = \phi((E_{i-1} \cup \Delta E_i) + D) \\ &= \phi(E_{i-1} + D) \cup \phi(\Delta E_i + D) = E_i \cup \phi(\Delta E_i + D) \end{aligned}$$

for $i = 0, 1, \ldots k-1$. This shows that in order to obtain E_{i+1}, the calculation of $\phi(e + D)$ for $e \in \Delta E_i$ is sufficient. This observation suggests the following algorithm.

Algorithm DECIDE(M, D)

```
1  forall b ∈ B do
2      if b has no finite expansion then
3          return false
4  end
5  E ← D, oldΔE ← D, ΔE ← ∅ ;
6  while oldΔE ≠ ∅ do
7      forall e ∈ oldΔE, d ∈ D do
8          Orbit ← ∅, p ← φ(e + d) ;
9          while p ∉ Orbit and p ∉ E do
10             Orbit ← Orbit ∪ {p} ;
11             ΔE ← ΔE ∪ {p} ;
12             p ← φ(p) ;
13         end
14         if p ∈ Orbit then
15             return false
16         else
17             E ← E ∪ Orbit ;
18     end
19     oldΔE ← ΔE ;
20     ΔE ← ∅ ;
21 end
22 return true
```

When the while loop in lines 6–21 is entered for the ith time, the variables take the following values: $E = E_i$ and $old\Delta E = E_i \setminus E_{i-1}$. This is easily seen by induction, showing that our algorithm is correct.

Consider the classification algorithm. We reformulate the algorithm in a recursive manner as follows. Let $\mathcal{E}_1 = \mathcal{D}_1 = D$, and

$$\begin{aligned} \mathcal{E}_{i+1} &= \phi(\mathcal{E}_i + \mathcal{D}_i)\,, \\ \mathcal{D}_{i+1} &= \{\text{ periodic points in } \mathcal{E}_i \}\,. \end{aligned}$$

Clearly, these sequences are eventually stable.

Let $\Delta\mathcal{E}_i = \mathcal{E}_i \setminus \mathcal{E}_{i-1}$ and $\Delta\mathcal{D}_i = \mathcal{D}_i \setminus \mathcal{D}_{i-1}$. Then

$$\begin{aligned} \phi(\mathcal{E}_i + \mathcal{D}_i) &= \phi(\mathcal{E}_{i-1} + \mathcal{D}_{i-1}) + \phi(\Delta\mathcal{E}_i + \mathcal{D}_{i-1}) + \phi(\mathcal{E}_i + \Delta\mathcal{D}_i) \\ &= \mathcal{E}_i + \phi(\Delta\mathcal{E}_i + \mathcal{D}_{i-1}) + \phi(\mathcal{E}_i + \Delta\mathcal{D}_i)\,, \end{aligned}$$

and $\mathcal{E}_i$ is already known.

We note that putting one periodic point per cycle into the set $\mathcal{D}$ is sufficient. This is because running into that point or running into the cycle is equivalent.

Algorithm CLASSIFY(M, D)

```
1  D ← D ∪ { periodic points on the orbits of B} ;
2  E ← D ;
3  oldΔE ← E, oldΔD ← D ;
4  PrevD ← ∅ ;
5  while oldΔE ≠ ∅ do
6      ΔE ← ∅, ΔD ← ∅ ;
7      CurrE ← E ;
8      forall (e, d) ∈ CurrE × oldΔD ∪ oldΔE × PrevD do
9          Orbit ← ∅, p ← φ(e + d) ;
10         while p ∉ Orbit and p ∉ E do
11             Orbit ← Orbit ∪ {p} ;
12             ΔE ← ΔE ∪ {p} ;
13             p ← φ(p) ;
14         end
15         if p ∈ Orbit then
16             ΔD ← ΔD ∪ {p} ;
17         E ← E ∪ Orbit ;
18     end
19     oldΔE ← ΔE ;
20     oldΔD ← ΔD ;
21     PrevD ← D ;
22     D ← D ∪ ΔD ;
23 end
24 return D \ D, which contains one point from every non-zero cycle
```

When we enter the main loop for the ith time, $\mathcal{E} = \mathcal{E}_i$, $old\Delta\mathcal{E} = \Delta\mathcal{E}_i$, $\mathcal{D} = \mathcal{D}_i$, $old\Delta\mathcal{D} = \Delta\mathcal{D}_i$ and $Prev\mathcal{D} = \mathcal{D}_{i-1}$. This can be proved by induction, establishing the correctness of the algorithm.

4.3 Comparison of the algorithms

In chapter 3 and the article [12] all binary canonical number system polynomials are determined up to degree 11. The decision algorithms described in the present chapter were used to decide the number system property. We

summarize our experiences below.

- Both algorithms find some of the non-zero periodic points quickly, if such points exist. That is, if (M, D) is not a number system, the decision algorithms are fast.

- Both algorithms become exponentially memory and CPU-expensive in the worst cases as the degree of the polynomials grows. The largest set E we encountered was of size 21 223 091, for the polynomial $2 + 3x + 3x^2 + 3x^3 + 3x^4 + 3x^5 + 3x^6 + 3x^7 + 3x^8 + 2x^9 + x^{10}$. Storing such a large set of 10 dimensional vectors required smart programming tricks. The largest brick size we computed was even larger.

- We cannot say that either algorithm outperforms the other. In many cases, e.g. for polynomials fulfilling the monotonity condition of [27], Brunotte's algorithm is faster. In many cases, e.g. for the polynomial $2 + x^3 + x^7 + x^{10}$, the algorithm using bricks is faster. Two small examples showing the set of fractions together with the set E are seen in Figure 1.

- The decision using covering bricks is easily parallelized, with no communication between processing units. Currently his seems the only way to attack degree 12 for binary CNS polynomials.

4.4 Further directions

We are currently working on both algorithms. We try to give a deterministic version of FIND-BASIS-TRANSFORMATION. In the generalized version of Brunotte's algorithm, we try to address the memory problems we encountered in practice. Making the algorithm more memory-efficient is possible through space-time tradeoff.

Chapter 5

Summary and further directions

The thesis presented algorithms for problems defined in the introduction. All these problems are related to generalized number systems. An algorithm for deciding expansivity was presented in chapter 2, two algorithms for searching expansive polynomials in chapter 3, and two algorithms for GNS decision and GNS classification in chapter 4.

There are many directions for continuing the research results presented in the thesis. The expansive polynomials found by the search algorithm have only been examined for canonical digit sets. Symmetrical digit sets and other generalizations are under computation. Even the binary case is incomplete: in order to completely characterize matrices M that are possible number system bases (with not necessarily canonical digits), we need to determine the similarity classes of matrices with given characteristic polynomials. This was only done in smaller dimensions (cf. [6]).

The decision and classification algorithms in chapter 4 could be implemented in a concurrent/parallel environment. This would broaden the range of pairs (M, D) that are accessible with those algorithms. The algorithm using the set of fractions and random base transformations performs well in practice, but we do not yet understand the reasons. More computational experiments should provide insight and possibly yield a deterministic version.

Another direction of generalization is the research of shift radix systems (SRS). Brunotte's algorithm has been used with success for SRS. The algo-

rithm using the set of fractions may be applicable for SRS. The definition and examination of semi-SRS seems also possible.

Appendix A

Binary CNS polynomials

There are altogether 192 expansive polynomials of degree 9 with constant term 2. The following 12 of them are CNS polynomials:

$2-x+x^9$, $2-x^3+x^9$, $2+x^9$, $2+x^4+x^5+x^9$, $2+x^3+x^6+x^9$, $2+2x^3+2x^6+x^9$, $2+x^2+x^7+x^9$, $2+x+x^8+x^9$, $2+x+x^2+x^3+x^4+x^5+x^6+x^7+x^8+x^9$, $2+2x+2x^2+x^3+x^4+x^5+x^6+x^7+x^8+x^9$, $2+2x+2x^2+2x^3+2x^4+2x^5+2x^6+x^7+x^8+x^9$, $2+2x+2x^2+2x^3+2x^4+2x^5+2x^6+2x^7+2x^8+x^9$.

There are altogether 623 expansive polynomials of degree 10 with constant term 2. The following 42 of them are CNS polynomials: $2-x+x^{10}$, $2-x^2+x^{10}$, $2-x^2+x^4+x^{10}$, $2-x^5+x^{10}$, $2+x^{10}$, $2+x^5+x^{10}$, $2+2x^5+x^{10}$, $2+x^4+x^6+x^{10}$, $2+x^3+x^7+x^{10}$, $2+x^2+x^8+x^{10}$, $2+x^2+x^4-x^5+x^6+x^8+x^{10}$, $2+x^2+x^4+x^6+x^8+x^{10}$, $2+x^2+x^4+x^5+x^6+x^8+x^{10}$, $2+x^2+x^3+2x^5+x^7+x^8+x^{10}$, $2+2x^2+2x^4+2x^6+2x^8+x^{10}$, $2+x+x^9+x^{10}$, $2+x+x^4+2x^5+x^6+x^9+x^{10}$, $2+x+x^2+x^3+x^4+x^6+x^7+x^8+x^9+x^{10}$, $2+x+x^2+x^3+x^4+x^5+x^6+x^7+x^8+x^9+x^{10}$, $2+x+x^2+x^3+x^4+2x^5+x^6+x^7+x^8+x^9+x^{10}$, $2+x+2x^2+x^3+2x^4+x^5+2x^6+x^7+2x^8+x^9+x^{10}$, $2+2x+x^2+x^3+x^4+x^5+x^6+x^7+x^8+x^9+x^{10}$, $2+2x+x^2+2x^3+2x^4+x^5+2x^6+2x^7+x^8+x^9+x^{10}$, $2+2x+2x^2+x^3+x^4+x^5+x^6+x^7+x^8+x^9+x^{10}$, $2+2x+2x^2+2x^3+2x^4+x^5+x^6+x^7+x^8+x^9+x^{10}$, $2+2x+2x^2+x^3+2x^4+2x^5+2x^6+x^7+x^8+x^9+x^{10}$, $2+2x+2x^2+2x^3+x^4+x^5+x^6+x^7+x^8+x^9+x^{10}$, $2+2x+2x^2+2x^3+2x^4+x^5+x^6+x^7+x^8+x^9+x^{10}$, $2+2x+2x^2+2x^3+2x^4+2x^5+2x^6+x^7+x^8+x^9+x^{10}$, $2+2x+2x^2+2x^3+2x^4+2x^5+2x^6+2x^7+x^8+x^9+x^{10}$, $2+2x+2x^2+3x^3+3x^4+2x^5+2x^6+2x^7+x^8+x^9+x^{10}$, $2+2x+2x^2+2x^3+2x^4+2x^5+2x^6+2x^7+2x^8+x^9+x^{10}$, $2+2x+3x^2+2x^3+3x^4+2x^5+3x^6+2x^7+2x^8+x^9+x^{10}$,

$2+2x+3x^2+3x^3+3x^4+3x^5+3x^6+2x^7+2x^8+x^9+x^{10}$, $2+2x+3x^2+3x^3+4x^4+3x^5+3x^6+2x^7+2x^8+x^9+x^{10}$, $2+2x+2x^2+2x^3+2x^4+2x^5+2x^6+2x^7+2x^8+2x^9+x^{10}$, $2+3x+3x^2+3x^3+3x^4+3x^5+3x^6+3x^7+3x^8+2x^9+x^{10}$, $2+3x+4x^2+4x^3+4x^4+4x^5+4x^6+4x^7+3x^8+2x^9+x^{10}$, $2+3x+4x^2+5x^3+5x^4+5x^5+5x^6+4x^7+3x^8+2x^9+x^{10}$, $2+3x+4x^2+5x^3+6x^4+6x^5+5x^6+4x^7+3x^8+2x^9+x^{10}$, $2+4x+5x^2+5x^3+5x^4+5x^5+5x^6+4x^7+3x^8+2x^9+x^{10}$. $2+4x+6x^2+7x^3+7x^4+6x^5+5x^6+4x^7+3x^8+2x^9+x^{10}$.

There are altogether 339 expansive polynomials of degree 11 with constant term 2. The following 11 of them are CNS polynomials: $2-x+x^{11}$, $2+x^{11}$, $2+x^5+x^6+x^{11}$, $2+x^4+x^7+x^{11}$, $2+x^3+x^8+x^{11}$, $2+x^2+x^9+x^{11}$, $2+x+x^{10}+x^{11}$, $2+x+x^2+x^3+x^4+x^5+x^6+x^7+x^8+x^9+x^{10}+x^{11}$, $2+2x+2x^2+2x^3+2x^4+x^5+x^6+x^7+x^8+x^9+x^{10}+x^{11}$, $2+2x+2x^2+2x^3+2x^4+2x^5+2x^6+x^7+x^8+x^9+x^{10}+x^{11}$, $2+2x+2x^2+2x^3+2x^4+2x^5+2x^6+2x^7+2x^8+2x^9+2x^{10}+x^{11}$.

Bibliography

[1] S. Akiyama, T. Borbély, H. Brunotte, A. Pethő, and J. Thuswaldner. On a generalization of the radix representation – a survey. *Fields Inst. Commun.*, 41, High Primes and Misdemeanours: lectures in honour of the 60th birthday of Hugh Cowie Williams:19–27, 2004.

[2] S. Akiyama, T. Borbély, H. Brunotte, A. Pethő, and J. Thuswaldner. Generalized radix representations and dynamical systems I. *Acta Math. Hungar.*, 108:207–238, 2005.

[3] S. Akiyama, H. Brunotte, and A. Pethő. Cubic cns-polynomials, notes on a conjecture of W. J. Gilbert. *J. Math. Anal. Appl.*, 281:402–415, 2003.

[4] S. Akiyama and A. Pethő. On canonical number systems. *Theoret. Comput. Sci.*, 270:921–933, 2002.

[5] S. Akiyama and H. Rao. New criteria for canonical number systems. *Acta Arith.*, 111(1):5–25, 2004.

[6] A. Barbé and F. von Haeseler. Binary number systems for $\mathbb{Z}^k$. *J. Number Theory*, 117(1):14–30, 2006.

[7] D. W. Boyd. Pisot and Salem numbers in intervals of the real line. *Math. Comp.*, 32:1244–1260, 1978.

[8] H. Brunotte. On trinomial bases of radix representations of algebraic integers. *Acta Sci. Math. (Szeged)*, 67:407–413, 2001.

[9] H. Brunotte. Characterization of CNS polynomials. *Acta Sci. Math. (Szeged)*, 68:673–679, 2002.

[10] P. Burcsi and A. Kovács. An algorithm checking a necessary condition of number system constructions. *Ann. Univ. Sci. Budapest. Sect. Comput.*, 25:143–152, 2005.

[11] P. Burcsi and A. Kovács. On the importance of cache tuning in a cache-aware algorithm: a case study. *Comput. Math. Appl.*, 53(6):880–885, 2007.

[12] P. Burcsi and A. Kovács. Exhaustive search methods for CNS polynomials. *Monatsh. Math.*, accepted, to appear, 2008.

[13] P. Burcsi and A. Kovács. Some dynamical properties of generalized radix systems. Conference talk at the Journées Numération, Prague, 2008.

[14] P. Burcsi, A. Kovács, and Zs. Papp-Varga. Decision and classification algorithms for generalized number systems. *Ann. Univ. Sci. Budapest. Sect. Comput.*, 28:141–156, 2008.

[15] Ch. Chamfy. Fonctions méromorphes dans le cercle-unité et leurs séries de Taylor. *Ann. Inst. Fourier (Grenoble)*, 8:211–262, 1958.

[16] J. Dufresnoy and Ch. Pisot. Étude de certaines fonctions méromorphes bornées sur le cercle unité. application à un ensemble fermé d'entiers algèbriques. *Ann. Sci. École Norm. Sup.*, 72:69–92, 1955.

[17] W. J. Gilbert. Radix representation of quadratic fields. *J. Math. Anal. Appl.*, 83:264–274, 1981.

[18] P. Henrici. *Applied and Computational Complex Analysis, Vol. I.* Wiley, 1974.

[19] P. Henrici. *Applied and Computational Complex Analysis, Vol. II.* Wiley, 1977.

[20] I. Kátai. Generalized number systems in Euclidean spaces. *Math. Comput. Modelling*, 38:883–892, 2003.

[21] I. Kátai and B. Kovács. Kanonische Zahlensysteme bei reelen quadratischen Zahlen. *Acta Sci. Math. (Szeged)*, 42:99–107, 1980.

[22] I. Kátai and B. Kovács. Canonical number systems in imaginary quadratic fields. *Acta Math. Hungar.*, 37:159–164, 1981.

[23] A. Kovács. On computation of attractors for invertible expanding linear operators in $\mathbb{Z}^k$. *Publ. Math. Debrecen*, 56(1–2):97–120, 2000.

[24] A. Kovács. Generalized binary number systems. *Ann. Univ. Sci. Budapest. Sect. Comput.*, 20:195–206, 2001.

[25] A. Kovács. Number expansion in lattices. *Math. Comput. Modelling*, 38:909–915, 2003.

[26] A. Kovács, Á. Kornafeld, P. Burcsi, N. Podhorszki, A. Cs. Marosi, G. Vida, and G. Gombás. The power of a supercomputer without a supercomputer – project BinSYS (in Hungarian). In *Networkshop Miskolc*, http://nws.iif.hu/ncd2006, 2006.

[27] B. Kovács. Canonical number systems in algebraic number fields. *Acta Math. Hungar.*, 37:405–407, 1981.

[28] C. G. Latimer and C. C. MacDuffee. A correspondence between classes of ideals and classes of matrices. *Ann. Mathematics*, 34:313–316, 1933.

[29] D. H. Lehmer. A machine method for solving polynomial equations. *J. ACM*, 2:151–162, 1961.

[30] A. Pethő. On a polynomial transformation and its application to the construction of a public key cryptosystem. *Comput. Number Theory*, pages 31–43, 1991.

[31] A. Ralston. *A first course in numerical analysis*. McGraw-Hill, 1965.

[32] K. Scheicher and J. M. Thuswaldner. On the characterization of canonical number systems. *Osaka J. Math.*, 41(2):327–351, 2004.

[33] I. Schur. Über potenzreihen die im inneren des einheitskreises beschrankt sind. *J. Reine Angew. Math.*, 147:205–232, 1917.

[34] I. Schur. Über potenzreihen die im inneren des einheitskreises beschrankt sind. *J. Reine Angew. Math.*, 148:128–145, 1918.

[35] SZTAKI Desktop Grid, http://szdg.lpds.sztaki.hu/szdg/.

[36] O. Taussky. On a theorem of Latimer and MacDuffee. *Canad. J. Math.*, 1:300–302, 1949.

[37] J. Tóth, L. Szili, and A. Zachár. Stability of polynomials. *Mathematica in Education and Research*, 2:5–12, 1998.