ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta jaderná a fyzikálně inženýrská

Katedra matematiky

DIPLOMOVÁ PRÁCE

Konstrukce algoritmů pro paralelní sčítání v nestandardních číselných soustavách

Construction of algorithms for parallel addition in non-standard numeration systems

Vypracoval: Bc. Jan Legerský

Školitel: Ing. Štěpán Starosta, Ph.D.

Akademický rok: 2015/2016

Poděkování

Děkuji Ing. Štěpánu Starostovi, Ph.D., za vedení mé diplomové práce. Dále děkuji Ing. Mileně Svobodové, Ph.D., a prof. Ing. Editě Pelantové, CSc., za podnětné diskuze o implementované metodě. Velký dík patří také mým rodičům za podporu během celého studia.

Jan Legerský

Název práce: **Konstrukce algoritmů pro paralelní sčítání v nestandardních číselných soustavách**

Autor: Jan Legerský

Obor: Matematická informatika

Druh práce: Diplomová práce

Vedoucí práce: Ing. Štěpán Starosta, Ph.D., KAM FIT, ČVUT v Praze

Abstrakt: Práce se zabývá konstrukcí algoritmů pro paralelní sčítání v soustavách se základem $\beta \in \mathbb{Z}[\omega]$, $|\beta| > 1$, a abecedou $\mathcal{A} \subset \mathbb{Z}[\omega]$, kde ω je algebraické celé číslo. Popisujeme takzvanou *extending window method* s přepisovacím pravidlem $x - \beta$, která ve dvou fázích zkouší najít algoritmus paralelního sčítání pro danou číselnou soustavu. Ukazujeme, že pro konvergenci celé metody musí být β expandující, což je zároveň postačující podmínka konvergence první fáze. Také představujeme grafové kritérium, pomocí kterého je možné odhalit nekonvergenci v průběhu druhé fáze. Dále uvádíme různé varianty obou fází. Veškeré navržené algoritmy jsou implementovány v systému SageMath. Porovnáváme uvedené varianty a diskutujeme výsledky testování.

Klíčová slova: Paralelní sčítání, nestandardní numerační systémy, extending window method.

Title: **Construction of algorithms for parallel addition in non-standard numeration systems**

Author: Jan Legerský

Abstract: We focus on construction of algorithms for parallel addition in numeration systems with a base $\beta \in \mathbb{Z}[\omega]$, $|\beta| > 1$, and an alphabet $\mathcal{A} \subset \mathbb{Z}[\omega]$, where ω is an algebraic integer. We describe a so-called *extending window method* with the rewriting rule $x - \beta$. In two phases, the method attempts to construct a parallel addition algorithm for the given numeration system. We prove that β must be expanding for convergence of the method. At the same time, this is a sufficient condition of convergence of Phase 1. We also propose a graph condition which may reveal non-convergence of Phase 2. Several variants of both phases are introduced. We implement all algorithms in SageMath. We compare the proposed variants and we discuss the results of testing.

Key words: Parallel addition, non-standard numeration systems, extending window method.

Contents

List of symbols

Symbol	Description
$\mathbb{N}$	set of nonnegative integers $\{0, 1, 2, 3, \dots\}$
$\mathbb{Z}$	set of integers $\{\dots, -2, -1, 0, 1, 2, \dots\}$
$\mathbb{R}$	set of real numbers
$\mathbb{C}$	set of complex numbers
$\mathbb{Q}$	set of rational numbers
$\mathbb{Q}(\beta)$	the smallest field containing $\mathbb{Q}$ and algebraic number β
$\#S$	number of elements of the finite set S
C^*	complex conjugation and transposition of the complex matrix C
m_β	monic minimal polynomial of the algebraic number β
$\deg \beta$	degree of the algebraic number β (over $\mathbb{Q}$)
$(\beta, \mathcal{A})$	numeration system with the base β and the alphabet $\mathcal{A}$
$(x)_{\beta,\mathcal{A}}$	$(\beta, \mathcal{A})$-representation of the number x
$\mathrm{Fin}_{\mathcal{A}}(\beta)$	set of all complex numbers with a finite $(\beta, \mathcal{A})$-representation
$\mathcal{A}^{\mathbb{Z}}$	set of all bi-infinite sequences of digits in $\mathcal{A}$
$\mathbb{Z}[\omega]$	the smallest ring containing $\mathbb{Z}$ and ω
π	isomorphism from $\mathbb{Z}[\omega]$ to $\mathbb{Z}^d$ ($d = \deg \omega$)
$\mathcal{B}$	alphabet of input digits
q_j	weight coefficient for the j-th position
$\mathcal{Q}$	weight coefficients set
$\mathcal{Q}_{[w_0,\dots,w_{-k}]}$	set of possible weight coefficients for digits $(w_0, \dots, w_{-k}) \in \mathcal{B}^{k+1}$

Introduction

Parallel addition is an important part of algorithms for multiplication and division. While carry propagation in standard algorithms requires to compute digits of the sum of $x_n x_{n-1} \cdots x_1 x_0 \bullet$ and $y_n y_{n-1} \cdots y_1 y_0 \bullet$ one by one, a parallel algorithm determines the j-th digit of the sum just from a fixed number of digits around x_j and y_j. Thus it avoids the carry propagation and all digits digits of the result can be outputted at the same time. Besides theoretical reasons, parallel addition is used for instance in floating point division algorithm [16]. The base used is 4 and the digit set is $\{-2, -1, 0, 1, 2\}$.

A parallel addition algorithm was introduced by A. Avizienis in 1961 [3]. The algorithm works with an integer base $\beta \geq 3$ and an alphabet $\{-a, \ldots, 0, \ldots a\}$ where $a \in \mathbb{N}$ is such that $\beta/2 < a \leq \beta - 1$. Later, C. Y. Chow and J. E. Robertson presented a parallel addition algorithm for base 2 and alphabet $\{-1, 0, 1\}$ [4].

So-called non-standard numeration systems are extensively studied. Non-standard means that a base is not a positive integer. An example of such system is the Penney numeration system with the base $\imath - 1$ and the alphabet is $\{0, 1\}$. Complex numbers can be represented by this system without separating real and imaginary part. As division algorithms are also developed for non-standard numeration systems, we focus on construction of a parallel addition algorithm for these systems.

The numeration systems which allow parallel addition must be redundant, i.e, some numbers have more than one representation. C. Frougny, E. Pelantová and M. Svobodová provided parallel addition algorithms with an integer alphabet for all bases β such that $|\beta| > 1$ and no conjugate of β equals 1 in modulus [6]. Nevertheless, the alphabet is not minimal in general. The parallel addition algorithms for several bases (negative integers, complex numbers $\imath - 1, 2\imath$ and $\sqrt{2}\imath$, quadratic Pisot unit and non-integer rationals) with minimal integer alphabet are presented [7].

Let ω be an algebraic integer. We are interested in construction of parallel addition algorithm for a given base $\beta \in \mathbb{Z}[\omega]$ which is an algebraic integer such that $|\beta| > 1$ and generally non-integer alphabet $\mathcal{A} \subset \mathbb{Z}[\omega]$. Our aim is to implement a method which attempts to construct an algorithm automatically for the numeration system $(\beta, \mathcal{A})$. The implementation is used to find as many numeration systems allowing parallel addition as possible.

The structure of the thesis is the following: we review terminology and known results on numeration systems and parallel addition in Chapter 1. In Chapter 2, a general concept of addition is recalled and so-called extending window method for construction of parallel addition algorithm is outlined. This method, which consists of two phases, was proposed by M. Svobodová [15].

Some useful tools related to the ring $\mathbb{Z}[\omega]$ are developed in Chapter 3, including a norm derived from a companion matrix. Using these results, a necessary condition of

convergence the extending window method is proved in Chapter 4. We show that it is also a sufficient condition of convergence of Phase 1. We establish control of non-convergence of Phase 2 in Theorem 4.7. Moreover, a lower bound on the size of an alphabet which allows parallel addition is stated. We also indicate a way how an alphabet with necessary properties can be generated.

Different variants of both phases of the extending window method are proposed in Chapter 5. In Chapter 6, we present algorithms based on the theorems from Chapter 4 and we describe their implementation in SageMath.

Finally, several examples of results can be found in Chapter 7. We compare various methods from Chapter 5 and discuss successful numeration systems with a quadratic base and integer or non-integer alphabet.

Appendix contains images illustrating the extending window method, interfaces of program and details of tested examples.

Chapter 1

Preliminaries

This chapter summarizes definitions related to numeration systems and parallel addition. We recall some known results on properties of a numeration system which allows parallel addition. At the end of the chapter, we define the ring $\mathbb{Z}[\omega]$, where ω is an algebraic integer, and determine numeration systems within the scope of this thesis.

1.1 Numeration systems

In the binary system, numbers are expressed as a sum of powers of 2 multiplied by 0 or 1. The following definition generalizes this concept.

Definition 1.1. Let $\beta \in \mathbb{C}, |\beta| > 1$ and $\mathcal{A} \subset \mathbb{C}$ be a finite set containing 0. A pair $(\beta, \mathcal{A})$ is called a *positional numeration system* with *base* β and *digit set* $\mathcal{A}$, usually called *alphabet*.

Sometimes, the term *radix* is used instead of base. So-called standard numeration systems have an integer base β and an alphabet $\mathcal{A}$ which is a set of contiguous integers. We restrict ourselves to a base β which is an algebraic integer and possibly non-integer alphabet $\mathcal{A}$.

Definition 1.2. Let $(\beta, \mathcal{A})$ be a positional numeration system. Let x be a complex number and $x_n, x_{n-1}, x_{n-2}, \ldots \in \mathcal{A}, n \geq 0$. We say that ${}^{\omega}0x_nx_{n-1}\cdots x_1x_0 \bullet x_{-1}x_{-2}\cdots$ is a $(\beta, \mathcal{A})$-*representation* of x if $x = \sum_{j=-\infty}^{n} x_j\beta^j$, where ${}^{\omega}0$ denotes the left-infinite sequence of zeros.

We write briefly a *representation* instead of a $(\beta, \mathcal{A})$-representation if the base β and alphabet $\mathcal{A}$ follow from context. The assumption $|\beta| > 1$ implies that the sum for a given representation converges.

Definition 1.3. Let $(\beta, \mathcal{A})$ be a positional numeration system. The set of all complex numbers with a finite $(\beta, \mathcal{A})$-representation is defined by

$$\mathrm{Fin}_{\mathcal{A}}(\beta) := \left\{ \sum_{j=-m}^{n} x_j\beta^j : n, m \in \mathbb{N}, x_j \in \mathcal{A} \right\}.$$

For $x \in \mathrm{Fin}_{\mathcal{A}}(\beta)$, we write

$$(x)_{\beta,\mathcal{A}} = {}^{\omega}0x_nx_{n-1}\cdots x_1x_0 \bullet x_{-1}x_{-2}\cdots x_{-m}0^{\omega},$$

where 0^ω denotes the right-infinite sequence of zeros. From now on, we omit the starting ${}^\omega 0$ and ending 0^ω when we work with numbers in $\mathrm{Fin}_{\mathcal{A}}(\beta)$, but we still consider a representation as a bi-infinite sequence of digits. Note that indices are decreasing from left to right as it is usual to write the most significant digits first.

We remark that the set $\mathrm{Fin}_{\mathcal{A}}(\beta)$ is not necessarily closed under addition. Nevertheless, existence of a parallel addition algorithm in the sense of definitions in the next section implies, as we will see later, that the set $\mathrm{Fin}_{\mathcal{A}}(\beta)$ is closed under addition, i.e.,

$$\mathrm{Fin}_{\mathcal{A}}(\beta) + \mathrm{Fin}_{\mathcal{A}}(\beta) \subset \mathrm{Fin}_{\mathcal{A}}(\beta)\,.$$

Designing an algorithm for parallel addition requires some redundancy in numeration system [11]. According to [13], a numeration system $(\beta, \mathcal{A})$ is called *redundant* if there exists $x \in \mathrm{Fin}_{\mathcal{A}}(\beta)$ which has two different $(\beta, \mathcal{A})$-representations. For instance, the number 1 has $(2, \{-1, 0, 1\})$-representations $1\bullet$ and $1(-1)\bullet$. Redundant numeration system may allow to avoid carry propagation in addition. On the other hand, redundancy brings some disadvantages. For example, comparison is problematic.

1.2 Parallel addition

Informally, parallel algorithm consists of many threads or processes which run at the same time. Usually, the main task is to reduce communication among processes to minimum, since waiting for a result of another process slows down the whole computation. In the scope of parallel addition, parallelism is formalized by a local function, which is also often called a sliding block code.

Definition 1.4. Let $\mathcal{A}$ and $\mathcal{B}$ be alphabets. A function $\varphi : \mathcal{B}^{\mathbb{Z}} \to \mathcal{A}^{\mathbb{Z}}$ is said to be *p-local* if there exist $r, t \in \mathbb{N}$ satisfying $p = r + t + 1$ and a function $\phi : \mathcal{B}^p \to \mathcal{A}$ such that, for any $w = (w_j)_{j\in\mathbb{Z}} \in \mathcal{B}^{\mathbb{Z}}$ and its image $z = \varphi(w) = (z_j)_{j\in\mathbb{Z}} \in \mathcal{A}^{\mathbb{Z}}$, we have $z_j = \phi(w_{j+t}, \cdots, w_{j-r})$ for every $j \in \mathbb{Z}$. The parameter t, resp. r, is called *anticipation*, resp. *memory*.

This means that a sliding window of a length p computes the digit z_j of the image $\varphi(w)$ from digits $(w_{j+t}, \cdots, w_{j-r})$, the digit z_{j+1} from digits $(w_{j+t-1}, \cdots, w_{j-r-1})$ etc.

Since two $(\beta, \mathcal{A})$-representations may be easily summed up digitwise in parallel, the crucial point of parallel addition is conversion of a $(\beta, \mathcal{A} + \mathcal{A})$-representation of the sum to a $(\beta, \mathcal{A})$-representation. The notion of a p-local function is applied to this conversion.

Definition 1.5. Let β be a base and let $\mathcal{A}$ and $\mathcal{B}$ be alphabets containing 0. A function $\varphi : \mathcal{B}^{\mathbb{Z}} \to \mathcal{A}^{\mathbb{Z}}$ such that

i) for any $w = (w_j)_{j\in\mathbb{Z}} \in \mathcal{B}^{\mathbb{Z}}$ with finitely many non-zero digits, $z = \varphi(w) = (z_j)_{j\in\mathbb{Z}} \in \mathcal{A}^{\mathbb{Z}}$ has only finite number of non-zero digits, and

ii) $\sum_{j\in\mathbb{Z}} w_j\beta^j = \sum_{j\in\mathbb{Z}} z_j\beta^j$

is called *digit set conversion* in the base β from $\mathcal{B}$ to $\mathcal{A}$. Such a conversion φ is said to be *computable in parallel* if φ is a p-local function for some $p \in \mathbb{N}$.

Suppose that there is a processing unit on each position of an input $w = (w_j)_{j\in\mathbb{Z}}$ with access to t input digits on the left and r input digits on the right. If w has only

finitely many non-zero digits, then computation of $\varphi(w)$ can be done in a constant time independent on the number of non-zeros in the sequence w.

We recall some results on parallel addition in a numeration system with an integer alphabet. C. Frougny, E. Pelantová and M. Svobodová proved the following sufficient condition of existence of an algorithm for parallel addition in [6].

Theorem 1.1. *Let $\beta \in \mathbb{C}$ be an algebraic number such that $|\beta| > 1$ and all its conjugates in modulus differ from 1. There exists an alphabet $\mathcal{A}$ of contiguous integers containing 0 such that addition on $Fin_{\mathcal{A}}(\beta)$ can be performed in parallel.*

An algorithm for an alphabet of the form $\{-a, -a+1, \ldots, 0, \ldots, a-1, a\}$ is provided in the proof, but in general, a is not minimal.

The same authors showed in [5] that the condition on the conjugates of the base β is also necessary:

Theorem 1.2. *Let the base $\beta \in \mathbb{C}, |\beta| > 1$, be an algebraic number with a conjugate β' such that $|\beta'| = 1$. If $\mathcal{A} \subset \mathbb{Z}$ is an alphabet of contiguous integers containing 0, then addition on $Fin_{\mathcal{A}}(\beta)$ cannot be computable in parallel.*

We will see later that the construction of a parallel addition algorithm by the method from Chapter 2 is also significantly influenced by the absolute value of conjugates of the base.

A lower bound on the size of an integer alphabet is provided in [7].

Theorem 1.3. *Let $\beta \in \mathbb{C}, |\beta| > 1$, be an algebraic integer with the minimal polynomial m_β. Let $\mathcal{A} \subset \mathbb{Z}$ be an alphabet of contiguous integers containing 0 and 1. If addition on $Fin_{\mathcal{A}}(\beta)$ is computable in parallel, then $\#\mathcal{A} \geq |m_\beta(1)|$. Moreover, if β is a real number, $\beta > 1$, then $\#\mathcal{A} \geq |m_\beta(1)| + 2$.*

In Section 4.3, we prove the same bound for a larger class of alphabets. The most general alphabets, which are considered in this thesis, are finite subsets of the set from the next definition. For our purposes, a ring is associative under multiplication and there is a multiplicative identity.

Definition 1.6. Let ω be a complex number. The smallest ring which contains integers $\mathbb{Z}$ and ω is denoted by

$$\mathbb{Z}[\omega] = \left\{ \sum_{i=0}^{n} a_i \omega^i \colon n \in \mathbb{N}, a_i \in \mathbb{Z} \right\}.$$

In other words, the set $\mathbb{Z}[\omega]$ are all polynomials with integer coefficients evaluated in ω. Obviously, it is a subset of the field extension $\mathbb{Q}(\omega)$ and commutative ring.

From now on, let ω be an algebraic integer which generates the set $\mathbb{Z}[\omega]$ and let $\beta \in \mathbb{Z}[\omega]$ be a base, i.e., $|\beta| > 1$. We remark that β is also an algebraic integer as all elements of $\mathbb{Z}[\omega]$ are algebraic integers. Finally, let $\mathcal{A} \subset \mathbb{Z}[\omega]$ be an alphabet. By definition, it means that it is a finite set containing 0 .

A few parallel addition algorithms for such numeration system with a non-integer alphabet were found manually [15]. We introduce the method for construction of a parallel addition algorithm for a given numeration system $(\beta, \mathcal{A})$ in Chapter 2.

Chapter 2

Design of extending window method

In this chapter, the general concept of addition and digit set conversion is recalled. We outline a so-called *extending window method* which is due to M. Svobodová [15]. The method consists of two phases. For a given numeration system $(\beta, \mathcal{A})$, it attempts to construct a digit set conversion algorithm which is computable in parallel. We recall that ω is an algebraic integer, $\beta \in \mathbb{Z}[\omega]$ is a base and $0 \in \mathcal{A} \subset \mathbb{Z}[\omega]$ is an alphabet.

2.1 Addition

The general idea of addition (standard or parallel) in any numeration system $(\beta, \mathcal{A})$ is the following: we sum up two numbers digitwise and then we convert the result with digits in $\mathcal{A} + \mathcal{A}$ into the alphabet $\mathcal{A}$. Obviously, digitwise addition is computable in parallel, thus the problematic part is the digit set conversion of the obtained result. It can be easily done in a standard way but a parallel digit set conversion is non-trivial. A parallel conversion is based on the same formulas as the standard one but the choice of so-called *weight coefficients* differs.

Now, we go step by step more precisely. Let $x, y \in \mathrm{Fin}_{\mathcal{A}}(\beta)$ with $(\beta, \mathcal{A})$-representations $x_{n'} x_{n'-1} \cdots x_1 x_0 \bullet x_{-1} x_{-2} \cdots x_{-m'}$ and $y_{n'} y_{n'-1} \cdots y_1 y_0 \bullet y_{-1} y_{-2} \cdots y_{-m'}$ padded by zeros to have the same length $n' + m' + 1$. We set

$$\begin{aligned} w = x + y &= \sum_{i=-m'}^{n'} x_i \beta^i + \sum_{i=-m'}^{n'} y_i \beta^i = \sum_{i=-m'}^{n'} (x_i + y_i) \beta^i \\ &= \sum_{i=-m'}^{n'} w_i \beta^i \,, \end{aligned}$$

where $w_i = x_i + y_i \in \mathcal{A} + \mathcal{A}$. Thus, $w_{n'} w_{n'-1} \cdots w_1 w_0 \bullet w_{-1} w_{-2} \cdots w_{-m'}$ is a $(\beta, \mathcal{A} + \mathcal{A})$-representation of $w \in \mathrm{Fin}_{\mathcal{A}+\mathcal{A}}(\beta)$.

We also use column notation for digitwise addition in what follows, e.g.,

$$\begin{array}{l} x_{n'}\ x_{n'-1}\cdots x_1\ x_0 \bullet x_{-1}\ x_{-2}\ \cdots x_{-m'} \\ y_{n'}\ y_{n'-1}\cdots y_1\ \ y_0 \bullet y_{-1}\ y_{-2}\ \ \cdots y_{-m'} \\ \hline w_{n'}w_{n'-1}\cdots w_1w_0 \bullet w_{-1}w_{-2}\cdots w_{-m'}\,. \end{array}$$

We search for a $(\beta, \mathcal{A})$-representation of w, i.e., a sequence

$$z_n z_{n-1}\cdots z_1 z_0 z_{-1} z_{-2}\cdots z_{-m}$$

such that $z_j \in \mathcal{A}$ and

$$z_n z_{n-1}\cdots z_1 z_0 \bullet z_{-1} z_{-2}\cdots z_{-m} = (w)_{\beta,\mathcal{A}}\,.$$

Note that the index of the first, resp. last, non-zero digit of the converted representation $z_n z_{n-1}\cdots z_1 z_0 \bullet z_{-1} z_{-2}\cdots z_{-m} = (w)_{\beta,\mathcal{A}}$ may differ from the original representation $w_{n'}w_{n'-1}\cdots w_1w_0 \bullet w_{-1}w_{-2}\cdots w_{-m'}$. We assume that $n \geq n'$ and $m \geq m'$, otherwise we pad the converted representation by zeros.

Multiplication of a representation $w_{n'}w_{n-1}\cdots w_1w_0 \bullet w_{-1}w_{-2}\cdots w_{-m'}$ by a power of β is obvious:

$$\beta^m \cdot w_{n'}w_{n'-1}\cdots w_1w_0 \bullet w_{-1}w_{-2}\cdots w_{-m'} = w_n w_{n'-1}\cdots w_1w_0w_{-1}w_{-2}\cdots w_{-m'}\bullet$$

and after conversion

$$z_n z_{n-1}\cdots z_1 z_0 z_{-1} z_{-2}\cdots z_{-m'} \bullet \cdots z_{-m} = \beta^m \cdot z_n z_{n-1}\cdots z_1 z_0 \bullet z_{-1} z_{-2}\cdots z_{-m'}\,.$$

Hence, without lost of generality, we consider only conversion of so-called β-integers – numbers from $\mathrm{Fin}_{\mathcal{A}+\mathcal{A}}(\beta)$ whose representations have all digits with negative indices equal to zero.

Digits w_j are converted into the alphabet $\mathcal{A}$ by adding a suitable representation of zero digitwise. For our purpose, we use the simplest possible representation which is deduced from the polynomial

$$x - \beta \in (\mathbb{Z}[\omega])\,[x]\,.$$

We remark that any polynomial $R(x) = r_s x^s + r_{s-1}x^{s-1} + \cdots + r_1 x + r_0$ with coefficients $r_i \in \mathbb{Z}[\omega]$ such that $R(\beta) = 0$ gives us a possible representation of zero. The polynomial R is called a *rewriting rule*. One of the coefficients of R which is greatest in modulus (so-called *core coefficient*) is used for the conversion of a digit w_j. Nevertheless, the extending window method is strongly dependent on the rewriting rule, so we focus only on the simplest possible rewriting rule $R(x) = x - \beta$. Usage of an arbitrary rewriting rule R is out of scope of this thesis.

As $0 = \beta^j \cdot R(\beta) = 1 \cdot \beta^{j+1} - \beta \cdot \beta^j$, we have a representation of zero

$$1(-\beta)\underbrace{0\cdots 0}_{j}\bullet = (0)_\beta\,.$$

for all $j \in \mathbb{N}$. We multiply this representation by $q_j \in \mathbb{Z}[\omega]$, which is called a *weight coefficient*, to obtain another representation of zero $q_j(-q_j\beta)0\cdots 0\bullet = (0)_\beta$. This is digitwise added to $w_n w_{n-1}\cdots w_1w_0\bullet$ to convert the digit w_j into the alphabet $\mathcal{A}$. The conversion of j-th digit causes a *carry* q_j on the $(j+1)$-th position.

In standard addition, the digit set conversion runs from the right ($j = 0$) to the left until all non-zero digits and carries are converted into the alphabet $\mathcal{A}$:

$$\begin{array}{ccccccccccccc} w_{n'} & w_{n'-1} & \cdots & w_{j+1} & w_j & w_{j-1} & \cdots & w_1 & w_0 & \bullet & = & (w)_{\beta,\mathcal{A}+\mathcal{A}} \\ & & & & & q_{j-2} & \iddots & & & & & \\ & & & & q_{j-1} & -\beta q_{j-1} & & & & & & \\ & & & \iddots & q_j & -\beta q_j & & & & & & \\ \hline z_n & \cdots z_{n'} & z_{n'-1} & \cdots & z_{j+1} & z_j & z_{j-1} & \cdots & z_1 & z_0 & \bullet & = & (w)_{\beta,\mathcal{A}} \end{array}$$

Hence, the desired formula for conversion on the j-th position is

$$z_j = w_j + q_{j-1} - q_j\beta$$

for $j \in \mathbb{N}$. We set $q_{-1} = 0$ as there is no carry from the right on the 0-th position.

The terms carry and weight coefficient are related to a position: while q_{j-1} is a carry from the right and q_j is a chosen weight coefficient on the j-th position, q_j is a carry from the right on the $(j+1)$-th position etc.

We remark that the conversion with the rewriting rule $x - \beta$ prolongs the part of non-zero digits only to the left as there is no carry from the left. Thus, all digits with negative indices of the converted sequence are zero.

The fact that the conversion preserves the value of w follows from adding a representation of zero:

$$\begin{aligned} \sum_{j\geq 0} z_j\beta^j &= w_0 - \beta q_0 + \sum_{j>0}(w_j + q_{j-1} - q_j\beta)\beta^j \\ &= \sum_{j\geq 0} w_j\beta^j + \sum_{j>0} q_{j-1}\beta^j - \sum_{j\geq 0} q_j \cdot \beta^{j+1} \\ &= \sum_{j\geq 0} w_j\beta^j + \sum_{j>0} q_{j-1}\beta^j - \sum_{j>0} q_{j-1} \cdot \beta^j \\ &= \sum_{j\geq 0} w_j\beta^j = w\,. \end{aligned} \tag{2.1}$$

The weight coefficient q_j must be chosen so that the converted digit is in the alphabet $\mathcal{A}$, i.e.,

$$z_j = w_j + q_{j-1} - q_j\beta \in \mathcal{A}\,. \tag{2.2}$$

The choice of weight coefficients is a crucial part of the construction of addition algorithms which are computable in parallel. The extending window method determining weight coefficients for a given input is described in Section 2.2.

On the other hand, the following example shows that determining weight coefficients is trivial for numeration systems such that an alphabet contains right one representative of each class modulo β.

Example 2.1. Assume now a numeration system $(\beta, \mathcal{A})$, where

$$\beta \in \mathbb{N}\,, \beta \geq 2\,, \mathcal{A} = \{0, 1, 2, \ldots, \beta - 1\}\,.$$

Notice that

$$z_j \equiv w_j + q_{j-1} \mod \beta\,.$$

There is only one representative of each class modulo β in the standard numeration system $(\beta, \mathcal{A})$. Therefore, the digit z_j is uniquely determined for a given digit $w_j \in \mathcal{A}+\mathcal{A}$

and carry q_{j-1} and thus so is the weight coefficient q_j. This means that $q_j = q_j(w_j, q_{j-1})$ for all $j \geq 0$. Generally,

$$q_j = q_j(w_j, q_{j-1}(w_{j-1}, q_{j-2})) = \cdots = q_j(w_j, \ldots, w_1, w_0)$$

and

$$z_j = z_j(w_j, \ldots, w_1, w_0)\,,$$

which implies that addition runs in linear time. For instance, the carry $q_{j-1} = 1$ propagates through the whole result when we sum up $(\beta-1)(\beta-1)\ldots(\beta-1)\bullet$ and $1\bullet$.

We require that the digit set conversion from $\mathcal{A}+\mathcal{A}$ into $\mathcal{A}$ is computable in parallel, i.e., there exist constants $r, t \in \mathbb{N}_0$ such that for all $j \geq 0$ is $z_j = z_j(w_{j+t}, \ldots, w_{j-r})$. Anticipation t equals zero since we use the rewriting rule $x-\beta$. To avoid the dependency on all less significant digits, we need variety in the choice of the weight coefficient q_j. This implies that the used numeration system must be redundant.

We remark that sometimes it is sufficient to find a digit set conversion from a so-called *input alphabet* $\mathcal{B}$, $\mathcal{A} \subsetneq \mathcal{B} \subset \mathcal{A}+\mathcal{A}$, in order to construct a parallel addition algorithm. Consider a numeration system with the alphabet $\mathcal{A} = \{-a, \ldots, 0, \ldots, a\}$ for some positive integer a. Assume that there is a digit set conversion from $\mathcal{B}$ to $\mathcal{A}$ computable in parallel, where $\mathcal{B} = \{-a-1, \ldots, 0, \ldots, a+1\} \subset \mathcal{A}+\mathcal{A}$. Obviously, $\mathcal{A}+\mathcal{A} = \mathcal{B}+\mathcal{A}_1$, where $\mathcal{A}_1 = \{-a+1, \ldots, 0, \ldots, a-1\}$. Hence, we write a $(\beta, \mathcal{A}+\mathcal{A})$-representation of the converted number w as a digitwise sum of a $(\beta, \mathcal{B})$-representation and $(\beta, \mathcal{A}_1)$-representation. After conversion the former one from $\mathcal{B}$ to $\mathcal{A}$ and summing up with the latter one again, we obtain a $(\beta, \mathcal{A}+\mathcal{A}_1)$-representation of w. But $\mathcal{A}+\mathcal{A}_1 = \mathcal{B}+\mathcal{A}_2$, where $\mathcal{A}_2 = \{-a+2, \ldots, 0, \ldots a-2\}$. Proceeding in the same manner, we obtain a $(\beta, \mathcal{A}+\mathcal{A}_a)$-representation of w after a iterations. Since $\mathcal{A}_a = \{0\}$, we have a $(\beta, \mathcal{A})$-representation of w.

Nevertheless, we take $\mathcal{B} = \mathcal{A}+\mathcal{A}$ in this thesis.

2.2 Extending window method

Let $\mathcal{B}$ be an input alphabet such that $\mathcal{A} \subsetneq \mathcal{B} \subset \mathcal{A}+\mathcal{A}$. The extending window method attempts to construct a digit set conversion in the numeration system $(\beta, \mathcal{A})$ from $\mathcal{B}$ to $\mathcal{A}$ which is computable in parallel. As mentioned above, the key problem is to find for every $j \geq 0$ a weight coefficient q_j such that

$$z_j = \underbrace{w_j}_{\in\mathcal{B}} + q_{j-1} - q_j\beta \in \mathcal{A}$$

for any input $w_{n'}w_{n'-1}\ldots w_1w_0\bullet = (w)_{\beta,\mathcal{B}}, w \in \mathrm{Fin}_{\mathcal{B}}(\beta)$. We remark that the weight coefficient q_{j-1} is determined by the input $w_{j-1}\ldots w_1w_0\bullet$. For a digit set conversion with the rewriting rule $x-\beta$ to be computable in parallel, the digit z_j is required to satisfy $z_j = z_j(w_j, \ldots, w_{j-r})$ for a fixed memory r in $\mathbb{N}$.

Note that the digit z_j is given by the input digit w_j and carry q_{j-1} which is determined by input digits $w_{j-1}w_{j-2}\ldots$. Thus, if we find a weight coefficient q_j for all possible combinations of input digits $w_jw_{j-1}w_{j-2}\ldots$, then the position j is not important. Therefore, we may strongly simplify our notation if we omit j in subscripts. From

now on, $w_0 \in \mathcal{B}$ is a converted digit, $w_{-1}w_{-2}\ldots \in \mathcal{B}$ are digits on right, $q_{-1} \in \mathbb{Z}[\omega]$ is a carry from the right and we search for a weight coefficient $q_0 \in \mathbb{Z}[\omega]$ such that

$$z_0 = w_0 + q_{-1} - q_0\beta \in \mathcal{A}\,.$$

We introduce two definitions before we describe the extending window method.

Definition 2.1. Let $\mathcal{B}$ be a set such that $\mathcal{A} \subsetneq \mathcal{B} \subset \mathcal{A} + \mathcal{A}$. Any finite set $\mathcal{Q} \subset \mathbb{Z}[\omega]$ containing 0 such that

$$\mathcal{B} + \mathcal{Q} \subset \mathcal{A} + \beta\mathcal{Q}$$

is called a *weight coefficients set*.

We see that if $\mathcal{Q}$ is a weight coefficients set, then

$$(\forall w_0 \in \mathcal{B})(\forall q_{-1} \in \mathcal{Q})(\exists q_0 \in \mathcal{Q})(\underbrace{w_0 + q_{-1} - q_0\beta}_{z_0} \in \mathcal{A})\,.$$

In other words, there is a weight coefficient $q_0 \in \mathcal{Q}$ for a carry from the right $q_{-1} \in \mathcal{Q}$ and a digit w_0 in the input alphabet $\mathcal{B}$ such that z_0 is in the alphabet $\mathcal{A}$. Notice that the carry from the right for the rightmost non-zero digit of the converted sequence which is 0 is in $\mathcal{Q}$ by the definition.

Definition 2.2. Let r be an integer and $q : \mathcal{B}^r \to \mathcal{Q}$ be a mapping such that

$$w_0 + q(w_{-1}, \ldots, w_{-r}) - \beta q(w_0, \ldots, w_{-(r-1)}) \in \mathcal{A}$$

for all $w_0, w_{-1}, \ldots, w_{-r} \in \mathcal{B}$, and $q(0, 0, \ldots, 0) = 0$. The mapping q is called *weight function* and r is called *length of window*.

Having a weight function q, we define a function $\phi : \mathcal{B}^{r+1} \to \mathcal{A}$ by

$$\phi(w_0, \ldots, w_{-r}) = w_0 + \underbrace{q(w_{-1}, \ldots, w_{-r})}_{=q_{-1}} - \beta\,\underbrace{q(w_0, \ldots, w_{-(r-1)})}_{=q_0} =: z_0\,, \tag{2.3}$$

which verifies that the digit set conversion is indeed a $(r+1)$-local function with anticipation 0 and memory r. The requirement of zero output of the weight function q for the input of r zeros guarantees that $\phi(0, 0, \ldots, 0) = 0$. Thus, the first condition of Definition 1.5 is satisfied. The second one follows from the equation (2.1).

Let us summarize the construction of the digit set conversion by the rewriting rule $x - \beta$. We need to find weight coefficients for all possible combinations of digits of the input alphabet $\mathcal{B}$. The rewriting rules multiplied by the weight coefficients are digitwise added to an input sequence. In fact, it means that the equation (2.2) is applied on each position. If the digit set conversion is computable in parallel, the weight coefficients are determined as the outputs of the weight function q with some fixed length of window r.

We search for a weight function q for a given base β and input alphabet $\mathcal{B}$ by the extending window method. It consists of two phases. First, we find some weight coefficients set $\mathcal{Q}$. We know that it is possible to convert an input sequence by choosing the weight coefficients from the set $\mathcal{Q}$. The set $\mathcal{Q}$ serves as the starting point for the second phase in which we increment the expected length of the window r until the weight function q is uniquely defined for each $(w_0, \ldots, w_{-(r-1)}) \in \mathcal{B}^r$. Then, the local conversion is determined – we use the weight function outputs as weight coefficients in the formula (2.3).

We describe the general concept of the extending window method in this chapter, while various possibilities of construction of sets during both phases are discussed in Chapter 5. Note that convergence of both phases is studied in Chapter 4.

2.3 Phase 1 – Weight coefficients set

The goal of the first phase is to compute a weight coefficients set $\mathcal{Q}$, i.e., to find a set $\mathcal{Q} \ni 0$ such that

$$\mathcal{B} + \mathcal{Q} \subset \mathcal{A} + \beta\mathcal{Q}\,.$$

We build a sequence $\mathcal{Q}_0, \mathcal{Q}_1, \mathcal{Q}_2, \ldots$ iteratively so that we extend $\mathcal{Q}_k$ to $\mathcal{Q}_{k+1}$ in a way to cover all elements of the set $\mathcal{B} + \mathcal{Q}_k$ by elements of the extended set $\mathcal{Q}_{k+1}$, i.e.,

$$\mathcal{B} + \mathcal{Q}_k \subset \mathcal{A} + \beta\mathcal{Q}_{k+1}\,.$$

This procedure is repeated until the extended weight coefficients set $\mathcal{Q}_{k+1}$ is the same as the previous set $\mathcal{Q}_k$. We remark that the expression "a weight coefficient q covers an element x" means that there is a digit $a \in \mathcal{A}$ such that $x = a + \beta q$.

In other words, we start with $\mathcal{Q}_0 = \{0\}$ meaning that we search all weight coefficients q_0 necessary for digit set conversion for the case where there is no carry from the right, i.e., $q_{-1} = 0$. We add them to the weight coefficients set $\mathcal{Q}_0$ to obtain the set $\mathcal{Q}_1$. Assume now that we have a set $\mathcal{Q}_k$ for some $k \geq 1$. The weight coefficients in $\mathcal{Q}_k$ now may appear as a carry q_{-1}. If there are no suitable weight coefficients q_0 in the weight coefficients set $\mathcal{Q}_k$ to cover all sums of coefficients from $\mathcal{Q}_k$ and digits of the input alphabet $\mathcal{B}$, we extend $\mathcal{Q}_k$ to $\mathcal{Q}_{k+1}$ by suitable coefficients. And so on until there is no need to add more elements, i.e., the extended set $\mathcal{Q}_{k+1}$ equals $\mathcal{Q}_k$. Then the weight coefficients set $\mathcal{Q} := \mathcal{Q}_{k+1}$ satisfies Definition 2.1.

For better understanding, see Figures 1–5 in Appendix A which illustrate the construction of the weight coefficients set $\mathcal{Q}$ for the Eisenstein base and a complex alphabet.

The precise description of the algorithm in a pseudocode is in Algorithm 1. Observe that extending $\mathcal{Q}_k$ to $\mathcal{Q}_{k+1}$ is not unique. Various methods of choice are described in Section 5.1 in Algorithm 3.

Section 4.2 discusses the convergence of Phase 1, i.e. whether it happens that $\mathcal{Q}_{k+1} = \mathcal{Q}_k$ for some k.

Algorithm 1 Search for weight coefficients set (Phase 1)

1: $k := 0$
2: $Q_0 := \{0\}$
3: **repeat**
4: Extend $\mathcal{Q}_k$ to $\mathcal{Q}_{k+1}$ (by Algorithm 3) so that

$$\mathcal{B} + \mathcal{Q}_k \subset \mathcal{A} + \beta\mathcal{Q}_{k+1}$$

5: $k := k + 1$
6: **until** $\mathcal{Q}_k = \mathcal{Q}_{k+1}$
7: $\mathcal{Q} := \mathcal{Q}_k$
8: **return** $\mathcal{Q}$

2.4 Phase 2 – Weight function

We want to find a length of the window r and a weight function $q : \mathcal{B}^r \to \mathcal{Q}$. We start with the weight coefficients set $\mathcal{Q}$ obtained in Phase 1. The idea is to reduce necessary

weight coefficients which are used to convert a given input digit up to a single value. This is done by enlarging the number of considered input digits, i.e. incrementing r. If the window is extended to the right, we know more digits that cause a carry form the right. This may decrease the number of possible carries from the right and hence, less weight coefficients to convert the input digit may be necessary.

We introduce notation for sets of possible weight coefficients for given input digits. Let $\mathcal{Q}$ be a weight coefficients set and $w_0 \in \mathcal{B}$. Denote by $\mathcal{Q}_{[w_0]}$ any set such that

$$(\forall q_{-1} \in \mathcal{Q})(\exists q_0 \in \mathcal{Q}_{[w_0]})(w_0 + q_{-1} - q_0\beta \in \mathcal{A})\,.$$

It means that we do not know any input digits on the right, therefore there might be any carry from the set $\mathcal{Q}$. However, we may determine a set $\mathcal{Q}_{[w_0]}$ of weight coefficients which allow the conversion of w_0 to $\mathcal{A}$ since we know the input digit w_0.

By induction with respect to $k \in \mathbb{N}, k \geq 1$, for all $(w_0, \ldots, w_{-k}) \in \mathcal{B}^{k+1}$ denote by $\mathcal{Q}_{[w_0,\ldots,w_{-k}]}$ any subset of $\mathcal{Q}_{[w_0,\ldots,w_{-(k-1)}]}$ such that

$$(\forall q_{-1} \in \mathcal{Q}_{[w_{-1},\ldots,w_{-k}]})(\exists q_0 \in \mathcal{Q}_{[w_0,\ldots,w_{-k}]})(w_0 + q_{-1} - q_0\beta \in \mathcal{A})\,.$$

Sets of possible weight coefficients and a weight function q are constructed by Algorithm 2. The idea is to check all possible right carries $q_{-1} \in \mathcal{Q}$ and determine values $q_0 \in \mathcal{Q}$ such that

$$z_0 = w_0 + q_{-1} - q_0\beta \in \mathcal{A}\,.$$

So we obtain a set $\mathcal{Q}_{[w_0]} \subset \mathcal{Q}$ of weight coefficients which are necessary to convert the digit w_0 with any carry $q_{-1} \in \mathcal{Q}$. Assuming that we know the input digit w_{-1}, the set of possible carries from the right is also reduced to $\mathcal{Q}_{[w_{-1}]}$. Thus we may reduce the set $\mathcal{Q}_{[w_0]}$ to a set $\mathcal{Q}_{[w_0,\ldots,w_{-1}]} \subset \mathcal{Q}_{[w_0]}$ which is necessary to cover all elements of $w_0 + \mathcal{Q}_{[w_{-1}]}$.

In the k-th step, we search for a set $\mathcal{Q}_{[w_0,\ldots,w_{-k}]} \subset \mathcal{Q}_{[w_0,\ldots,w_{-(k-1)}]}$ such that

$$w_0 + \mathcal{Q}_{[w_{-1},\ldots,w_{-k}]} \subset \mathcal{A} + \beta\mathcal{Q}_{[w_0,\ldots,w_{-k}]}\,.$$

The length of window is $k+1$, i.e., we know k digits on the right. To construct the set $\mathcal{Q}_{[w_0,\ldots,w_{-k}]}$, we select from $\mathcal{Q}_{[w_0,\ldots,w_{-(k-1)}]}$ such weight coefficients which are necessary to convert digit w_0 to the alphabet $\mathcal{A}$ with all possible carries from the set $\mathcal{Q}_{[w_{-1},\ldots,w_{-k}]}$.

Proceeding in this manner may lead to a unique weight coefficient q_0 for enough long window. If there is $r \in \mathbb{N}, r \geq 1$ such that

$$\#\mathcal{Q}_{[w_0,\ldots,w_{-(r-1)}]} = 1$$

for all $(w_0, \ldots, w_{-(r-1)}) \in \mathcal{B}^r$, then the output $q(w_0, \ldots, w_{-(r-1)})$ is defined as the element of $\mathcal{Q}_{[w_0,\ldots,w_{-(r-1)}]}$.

Similarly to Phase 1, the choice of $\mathcal{Q}_{[w_0,\ldots,w_{-k}]}$ is not unique. We list different methods of choice in Section 5.2, Algorithm 5.

Figures 6–8 in Appendix B illustrate the construction of the set $\mathcal{Q}_{[\omega,1,2]}$ for the Eisenstein numeration system.

To verify that

$$z_0 = \phi(w_0, \ldots, w_{-r}) = w_0 + \underbrace{q(w_{-1}, \ldots, w_{-r})}_{=q_{-1}} - \beta\,\underbrace{q(w_0, \ldots, w_{-(r-1)})}_{=q_0}$$

is in the alphabet $\mathcal{A}$, consider that $q_0 = q(w_0, \dots, w_{-(r-1)})$ is the only element of $\mathcal{Q}_{[w_0,\dots,w_{-(r-1)}]}$ which was constructed such that

$$w_0 + \mathcal{Q}_{[w_{-1},\dots,w_{-(r-1)}]} \subset \mathcal{A} + \beta\mathcal{Q}_{[w_0,\dots,w_{-(r-1)}]}\,.$$

At the same time, $q_{-1} = q(w_0, \dots, w_{-(r-1)})$ is the only element of $\mathcal{Q}_{[w_{-1},\dots,w_{-r}]}$ which is a subset of $\mathcal{Q}_{[w_{-1},\dots,w_{-(r-1)}]}$.

Algorithm 2 Search for weight function q (Phase 2)

Input: weight coefficients set $\mathcal{Q}$

1: **for all** $w_0 \in \mathcal{B}$ **do**
2: Find set $\mathcal{Q}_{[w_0]} \subset \mathcal{Q}$ (by Algorithm 5) such that

$$w_0 + \mathcal{Q} \subset \mathcal{A} + \beta\mathcal{Q}_{[w_0]}$$

3: **end for**
4: $k := 0$
5: **while** $\max\{\#\mathcal{Q}_{[w_0,\dots,w_{-k}]} \colon (w_0, \dots, w_{-k}) \in \mathcal{B}^{k+1}\} > 1$ **do**
6: $k := k + 1$
7: **for all** $(w_0, \dots, w_{-k}) \in \mathcal{B}^{k+1}$ **do**
8: Find set $\mathcal{Q}_{[w_0,\dots,w_{-k}]} \subset \mathcal{Q}_{[w_0,\dots,w_{-(k-1)}]}$ (by Algorithm 5) such that

$$w_0 + \mathcal{Q}_{[w_{-1},\dots,w_{-k}]} \subset \mathcal{A} + \beta\mathcal{Q}_{[w_0,\dots,w_{-k}]}\,,$$

9: **end for**
10: **end while**
11: $r := k + 1$
12: **for all** $(w_0, \dots, w_{-(r-1)}) \in \mathcal{B}^r$ **do**
13: $q(w_0, \dots, w_{-(r-1)}) \in \mathcal{B}^r :=$ only element of $\mathcal{Q}_{[w_0,\dots,w_{-(r-1)}]}$
14: **end for**
15: **return** q

Unfortunately, finiteness of Phase 2 is not guaranteed. But the non-convergence of Phase 2 with a specific property may be revealed by Algorithm 7 before the run of Phase 2 or by Algorithm 9 during it. These algorithms are based on theorems in Chapter 4. Modified Phase 2 which includes these algorithms can be found in Section 6.1.

Notice that for a given length of window r, the number of calls of Algorithm 5 within Algorithm 2 is

$$\sum_{k=0}^{r-1} \#\mathcal{B}^{k+1} = \#\mathcal{B}\frac{\#\mathcal{B}^r - 1}{\#\mathcal{B} - 1}\,.$$

It implies that the time complexity grows exponentially. The required memory is also exponential because we have to store sets $\mathcal{Q}_{[w_0,\dots,w_{-k}]}$ at least for $k \in \{r-2, r-1\}$ for all $w_0, \dots, w_{-k} \in \mathcal{B}$.

Chapter 3

Properties of $\mathbb{Z}[\omega]$

We compile some properties of the ring $\mathbb{Z}[\omega]$ in this chapter. We first show that $\mathbb{Z}[\omega]$ is isomorphic to $\mathbb{Z}^d$ and we use this fact to check divisibility in $\mathbb{Z}[\omega]$. We review some results from matrix theory to introduce a norm in $\mathbb{Z}[\omega]$. This norm is used in the proof of convergence of Phase 1 in Chapter 4. In the last section of this chapter, we show how the number of congruence classes modulo β in $\mathbb{Z}[\omega]$ is determined. This result is used in the proof of a lower bound on the size of an alphabet which allows parallel addition in Section 4.3.

3.1 Isomorphism of $\mathbb{Z}[\omega]$ and $\mathbb{Z}^d$

In this section, we recall that the ring $\mathbb{Z}[\omega]$ is isomorphic to the set $\mathbb{Z}^d$ equipped with multiplication, where d is the degree of the algebraic integer ω. This structure is useful as it allows to handle elements of $\mathbb{Z}[\omega]$ as vectors. For example, division in $\mathbb{Z}[\omega]$ can be replaced by searching for an integer solution of a linear system (Theorem 3.2) which is used in our implementation of the extending window method.

Before defining multiplication in $\mathbb{Z}^d$, we recall the notion of companion matrix.

Definition 3.1. Let $p(x) = x^d + p_{d-1}x^{d-1} + \cdots + p_1 x + p_0 \in \mathbb{Z}[x]$ be a monic polynomial with integer coefficients, $d \geq 1$. The matrix

$$S := \begin{pmatrix} 0 & 0 & \cdots & 0 & -p_0 \\ 1 & 0 & \cdots & 0 & -p_1 \\ 0 & 1 & \cdots & 0 & -p_2 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & \cdots & 1 & -p_{d-1} \end{pmatrix} \in \mathbb{Z}^{d\times d}$$

is the *companion matrix* of the polynomial p.

Let S be the companion matrix of a polynomial p. It is well known (see for instance [10]) that the characteristic polynomial of the companion matrix S is p. The matrix S is also root of the polynomial p.

We recall that the minimal polynomial of an algebraic integer ω is denoted by m_ω and is monic by definition. Multiplication in $\mathbb{Z}^d$ is defined in the following way.

Definition 3.2. Let ω be an algebraic integer of degree $d \geq 1$ and let S be the companion matrix of m_ω. We define a mapping $\odot_\omega : \mathbb{Z}^d \times \mathbb{Z}^d \to \mathbb{Z}^d$ by

$$u \odot_\omega v := \left(\sum_{i=0}^{d-1} u_i S^i\right) \cdot \begin{pmatrix} v_0 \\ v_1 \\ \vdots \\ v_{d-1} \end{pmatrix} \quad \text{for all } u = \begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ u_{d-1} \end{pmatrix}, v = \begin{pmatrix} v_0 \\ v_1 \\ \vdots \\ v_{d-1} \end{pmatrix} \in \mathbb{Z}^d .$$

The technical proof of Theorem 3.1, which shows that $\mathbb{Z}^d$ with multiplication $\odot_\omega$ is a ring isomorphic to $\mathbb{Z}[\omega]$, can be found in [12].

Theorem 3.1. *If ω is an algebraic integer of degree d, then*

$$\mathbb{Z}[\omega] = \left\{ \sum_{i=0}^{d-1} a_i \omega^i \colon a_i \in \mathbb{Z} \right\},$$

$(\mathbb{Z}^d, +, \odot_\omega)$ is a commutative ring and the mapping $\pi : \mathbb{Z}[\omega] \to \mathbb{Z}^d$ defined by

$$\pi(u) = \begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ u_{d-1} \end{pmatrix} \quad \text{for every } u = \sum_{i=0}^{d-1} u_i \omega^i \in \mathbb{Z}[\omega]$$

is a ring isomorphism.

This theorem provides simple representation of elements of $\mathbb{Z}[\omega]$ in computer – they are represented by integer vectors and multiplication in $\mathbb{Z}[\omega]$ is replaced by multiplying by an appropriate matrix.

Divisibility in $\mathbb{Z}[\omega]$ can be also transformed into $\mathbb{Z}^d$. To check whether an element of $\mathbb{Z}[\omega]$ is divisible by another element, we look for an integer solution of a linear system given by Theorem 3.2. Moreover, this solution provides the result of division in the positive case.

Theorem 3.2. *Let ω be an algebraic integer of degree d and let S be the companion matrix of its minimal polynomial. If $\beta = \sum_{i=0}^{d-1} b_i \omega^i$ is a nonzero element of $\mathbb{Z}[\omega]$, then for every $u \in \mathbb{Z}[\omega]$*

$$u \in \beta\mathbb{Z}[\omega] \iff S_\beta^{-1} \cdot \pi(u) \in \mathbb{Z}^d ,$$

where $S_\beta = \sum_{i=0}^{d-1} b_i S^i$.

The proof can be found in [12].

3.2 β-norm

The goal of this section is to construct a norm in $\mathbb{Z}[\omega]$. We use the isomorphism with $\mathbb{Z}^d$ and some results from matrix theory. This norm is used for the proof of convergence of Phase 1 in Chapter 4.

First, we recall a simple way how a new norm can be constructed from another one.

Lemma 3.3. *Let ν be a norm of the vector space $\mathbb{C}^d$ and P be a nonsingular matrix in $\mathbb{C}^d$. The mapping $\mu : \mathbb{C}^d \to \mathbb{R}_0^+$ defined by $\mu(x) = \nu(Px)$ is also a norm of the vector space $\mathbb{C}^d$.*

Proof. Let x and y be vectors in $\mathbb{C}^d$ and $\alpha \in \mathbb{C}$. We use linearity of matrix multiplication, nonsingularity of matrix P and the fact that ν is a norm to prove the following statements:

1. $\mu(x) = \nu(Px) \geq 0\,,$
2. $\mu(x) = 0 \iff \nu(Px) = 0 \iff Px = 0 \iff x = 0\,,$
3. $\mu(\alpha x) = \nu(P(\alpha x)) = \nu(\alpha Px) = |\alpha|\nu(Px) = |\alpha|\mu(x)\,,$
4. $\mu(x + y) = \nu(P(x + y)) = \nu(Px + Py) \leq \nu(Px) + \nu(Py) = \mu(x) + \mu(y)\,.$

This verifies that μ is a norm. □

Now we use Lemma 3.3 to define a new norm for a given diagonalizable matrix.

Definition 3.3. Let $M \in \mathbb{C}^{n\times n}$ be a diagonalizable matrix and $P \in \mathbb{C}^{n\times n}$ be a nonsingular matrix which diagonalizes M, i.e., $M = P^{-1}DP$ for some diagonal matrix $D \in \mathbb{C}^{n\times n}$. We define a vector norm $\|\cdot\|_M$ by

$$\|x\|_M := \|Px\|_2 \tag{3.1}$$

for all $x \in \mathbb{C}^n$, where $\|\cdot\|_2$ is Euclidean norm. A matrix norm $|||\cdot|||_M$ is induced by the norm $\|\cdot\|_M$, i.e,

$$|||A|||_M = \sup_{\|x\|_M=1} \|Ax\|_M$$

for all $A \in \mathbb{C}^{n\times n}$.

The following theorem is a known result from matrix theory – for a given diagonalizable matrix, there is a matrix norm such that the norm of the matrix equals its spectral radius [10].

Theorem 3.4. *If $M \in \mathbb{C}^{n\times n}$ is a diagonalizable matrix, then*

$$\rho(M) = |||M|||_M\,,$$

where $\rho(M)$ is the spectral radius of the matrix M.

Proof. First, we prove that $|||M|||_M \geq \rho(M)$. For all eigenvalues λ in the spectrum $\sigma(M)$ with a respective eigenvector u such that $\|u\|_M = 1$, we have

$$|||M|||_M = \sup_{\|x\|_M=1} \|Mx\|_M \geq \|Mu\|_M = \|\lambda u\|_M = |\lambda| \cdot \|u\|_M = |\lambda|\,.$$

Secondly, we show that $|||M|||_M \leq \rho(M)$. Following Definition 3.3, let $P \in \mathbb{C}^{n\times n}$ be a nonsingular matrix and $D \in \mathbb{C}^{n\times n}$ a diagonal matrix with the eigenvalues of M on the diagonal such that $PMP^{-1} = D$.

Let y be a vector such that $\|y\|_M = 1$ and set $z = Py$. Notice that

$$\sqrt{z^*z} = \|z\|_2 = \|Py\|_2 = \|y\|_M = 1\,.$$

Consider

$$\begin{aligned}\|My\|_M &= \|PMy\|_2 = \|DPy\|_2 = \|Dz\|_2 = \sqrt{z^*D^*Dz} \\ &\leq \sqrt{\max_{\lambda\in\sigma(M)} |\lambda|^2 z^*z} = \max_{\lambda\in\sigma(M)} |\lambda| = \rho(M)\ .\end{aligned}$$

which implies the statement. □

Before we define a norm in $\mathbb{Z}[\omega]$, we verify that a specific matrix given by an algebraic number $\beta \in \mathbb{Z}[\omega]$ is diagonalizable. Lemma 3.5 summarizes also some other properties of this matrix and the norm which it induces according to Theorem 3.4.

Lemma 3.5. *Let ω be an algebraic integer of degree d and let S be the companion matrix of its minimal polynomial m_ω. Let $\beta = \sum_{i=0}^{d-1} b_i\omega^i$, where $b_i \in \mathbb{Z}$, be a nonzero element of $\mathbb{Z}[\omega]$. Set $S_\beta = \sum_{i=0}^{d-1} b_i S^i$.*

i) The matrix S_β is diagonalizable.

ii) The characteristic polynomial of S_β is m_β^k with $k = d/\deg\beta$.

iii) $|\det S_\beta| = |m_\beta(0)|^k$.

iv) $\|x\|_{S_\beta} = \|x\|_{S_\beta^{-1}}$ *for all* $x \in \mathbb{C}^d$ *and* $|||X|||_{S_\beta} = |||X|||_{S_\beta^{-1}}$ *for all* $X \in \mathbb{C}^{d\times d}$.

v) $|||S_\beta|||_{S_\beta} = \max\{|\beta'|\colon \beta'$ *is conjugate of* $\beta\}$ *and*

$$\left|\left|\left|S_\beta^{-1}\right|\right|\right|_{S_\beta} = \max\left\{\frac{1}{|\beta'|}\colon \beta' \text{ is a conjugate of } \beta\right\}\ .$$

Proof. The characteristic polynomial of the companion matrix S is the same as the minimal polynomial of ω which has no multiple roots. Hence, S is diagonalizable, i.e., $S = P^{-1}DP$ where D is diagonal matrix with the conjugates of ω on the diagonal and P is a nonsingular complex matrix. The matrix S_β is also diagonalized by P:

$$S_\beta = \sum_{i=0}^{d-1} b_i S^i = \sum_{i=0}^{d-1} b_i \left(P^{-1}DP\right)^i = P^{-1}\underbrace{\left(\sum_{i=0}^{d-1} b_i D^i\right)}_{D_\beta} P\,.$$

It is known (see for instance [8]) that if $\sigma : \mathbb{Q}(\omega) \to \mathbb{Q}(\omega')$ is a field isomorphism and $\beta \in \mathbb{Q}(\omega)$, then $\sigma(\beta)$ is a conjugate of β. Moreover, a so-called field polynomial $\Pi_{i=0}^{d-1}(x - \sigma_i(\beta))$, where $\sigma_i : \mathbb{Q}(\omega) \to \mathbb{Q}(\omega_i)$ are all possible isomorphisms, is a power of m_β. In order to prove this, suppose that $\Pi_{i=0}^{d-1}(x - \sigma_i(\beta)) = m_\beta^k g$, where $k \in \mathbb{N}$ and g is a monic rational polynomial which does not divide m_β. If $g \neq 1$, then some $\sigma_i(\beta)$ with the minimal polynomial m_β is a root g. Since any rational polynomial is divisible by the minimal polynomials of its roots, m_β divides g which is a contradiction.

Since $\beta = \sum_{i=0}^{d-1} b_i\omega^i$ and D has conjugates of ω on the diagonal, the diagonal elements of the diagonal matrix D_β are conjugates of β. The characteristic polynomial p_{S_β} of S_β equals $\Pi_{i=0}^{d-1}(\sigma_i(\beta) - x)$. Thus, there exists $k \in \mathbb{N}, k \geq 1$ such that $p_{S_\beta} = \pm m_\beta^k$. The value k follows from the equality $d = \deg(m_\beta^k) = k\deg m_\beta$.

The modulus of the determinant of S_β equals the modulus of the absolute coefficient of the characteristic polynomial p_{S_β} which is $|m_\beta(0)|^k$.

The matrix S_β^{-1} is also diagonalized by P since $S_\beta^{-1} = (P^{-1}D_\beta P)^{-1} = P^{-1}D_\beta^{-1}P$. Thus, the norms $\|\cdot\|_{S_\beta}$ and $\|\cdot\|_{S_\beta^{-1}}$ are the same and so are the induced matrix norms $|||\cdot|||_{S_\beta}$ and $|||\cdot|||_{S_\beta^{-1}}$.

The matrix S_β is diagonalizable and its eigenvalues are the conjugates of β. Theorem 3.4 implies that

$$|||S_\beta|||_{S_\beta} = \rho(S_\beta) = \max\{|\beta'| : \beta' \text{ is conjugate of } \beta\}\,.$$

For the second part of the last statement, we use the part *iv)*, Theorem 3.4 and the fact that the eigenvalues of S_β^{-1} are reciprocal of the conjugates of β. □

Finally, we may define a norm in $\mathbb{Z}[\omega]$.

Definition 3.4. Let π be the isomorphism between $\mathbb{Z}[\omega]$ and $(\mathbb{Z}^d, +, \odot_\omega)$. Using the notation of the previous lemma, we define *β-norm* $\|\cdot\|_\beta : \mathbb{Z}[\omega] \to \mathbb{R}_0^+$ by

$$\|x\|_\beta = \|\pi(x)\|_{S_\beta}$$

for all $x \in \mathbb{Z}[\omega]$.

We can easily verify that β-norm is a norm in $\mathbb{Z}[\omega]$:

1. $\|x\|_\beta = \|\pi(x)\|_{S_\beta} \geq 0\,,$
2. $\|x\|_\beta = 0 \iff \|\pi(x)\|_{S_\beta} = 0 \iff \pi(x) = 0 \iff x = 0\,,$
3. $\|\alpha x\|_\beta = \|\pi(\alpha x)\|_{S_\beta} = |\alpha|\|\pi(x)\|_{S_\beta} = |\alpha|\|x\|_\beta\ ,$
4. $\|x+y\|_\beta = \|\pi(x+y)\|_{S_\beta} = \|\pi(x)+\pi(y)\|_{S_\beta} \leq \|\pi(x)\|_{S_\beta} + \|\pi(y)\|_{S_\beta} = \|x\|_\beta + \|y\|_\beta\ ,$

for all $x, y \in \mathbb{Z}[\omega]$ and $\alpha \in \mathbb{Z}[\omega]$.

The important property of β-norm is that there is only finitely many elements of $\mathbb{Z}[\omega]$ which are bounded in this norm by a given constant. The explanation is following – images of elements of $\mathbb{Z}[\omega]$ under the isomorphism π are integer vectors and there are only finitely many integer vectors in any finite dimensional vector space bounded by any norm. It follows from equivalence of all norms a finite dimensional vector space.

3.3 Number of congruence classes

Congruence classes play important role in the structure of an alphabet which allows parallel addition, as we will see later. We have seen that the isomorphism with $\mathbb{Z}^d$ is an efficient tool for handling elements of $\mathbb{Z}[\omega]$. We use it also for counting number of congruence classes. The definition of congruence in $\mathbb{Z}^d$ is following.

Definition 3.5. Let $M \in \mathbb{Z}^{d\times d}$ be a nonsingular integer matrix. Vectors $x, y \in \mathbb{Z}^d$ are *congruent modulo M in $\mathbb{Z}^d$* if $x - y \in M\mathbb{Z}^d$.

Let $x, y, z \in \mathbb{Z}^d$. We verify that congruence modulo M is an equivalence.

i) reflexivity: $x - x = 0 = M \cdot 0$,

ii) symmetry: if $\exists\, v \in \mathbb{Z}^d$ such that $x - y = M \cdot v$, then $y - x = M \cdot (-v)$,

iii) transitivity: if $\exists\, v, v' \in \mathbb{Z}^d$ such that $x - y = M \cdot v$ and $y - z = M \cdot v'$, then $z - x = (z - y) + (y - x) = M \cdot (v' + v)$.

In Theorem 3.7, we will see that a congruence class modulo β in $\mathbb{Z}[\omega]$ corresponds to a congruence class modulo S_β in $\mathbb{Z}^d$, where we use the notation from the previous section. Therefore, we count number of congruence classes modulo a matrix M in Lemma 3.6.

Lemma 3.6. *Let $M \in \mathbb{Z}^{d\times d}$ be a nonsingular integer matrix. The number of congruence classes modulo M in $\mathbb{Z}^d$ is $|\det M|$.*

Proof. Set $y_i := Me_i$ for $i \in \{0, \dots, d-1\}$ and

$$B_{(\alpha_0,\dots,\alpha_{d-1})} := \left\{ \sum_{i=0}^{d-1} (\alpha_i + t_i) y_i \colon t_i \in [0,1) \right\}$$

for $(\alpha_0, \dots, \alpha_{d-1}) \in \mathbb{Z}^d$. Obviously,

$$\mathbb{R}^d = \bigcup_{(\alpha_0,\dots,\alpha_{d-1})\in\mathbb{Z}^d} B_{(\alpha_0,\dots,\alpha_{d-1})}\,.$$

For fixed $(\alpha_0, \dots, \alpha_{d-1}) \in \mathbb{Z}^d$, the number of points of $\mathbb{Z}^d$ in $B_{(\alpha_0,\dots,\alpha_{d-1})}$ is the volume of $B_{(\alpha_0,\dots,\alpha_{d-1})}$ which is $|\det M|$. Hence, it is enough to prove that there is exactly one representative of each congruence class in $B_{(\alpha_0,\dots,\alpha_{d-1})}$.

To show that there are representatives of all classes, assume an arbitrary vector $x \in \mathbb{Z}^d$. Since $(y_0, \dots, y_{d-1})$ is a basis of $\mathbb{R}^d$, there are scalars $s_0, \dots, s_{d-1} \in \mathbb{R}$ such that $x = \sum_{i=0}^{d-1} s_i y_i$. Set $\gamma_i := \lfloor s_i \rfloor$ and $t_i := s_i - \gamma_i$. Now

$$x = \sum_{i=0}^{d-1} (\gamma_i + t_i) y_i = \sum_{i=0}^{d-1} t_i y_i + \sum_{i=0}^{d-1} (\gamma_i - \alpha_i) y_i + \sum_{i=0}^{d-1} \alpha_i y_i = \underbrace{\sum_{i=0}^{d-1} (\alpha_i + t_i) y_i}_{\in B_{(\alpha_0,\dots,\alpha_{d-1})}} + M \underbrace{(\gamma - \alpha)}_{\in\mathbb{Z}^d},$$

where $\alpha = (\alpha_0, \dots, \alpha_{d-1})^T$ and $\gamma = (\gamma_0, \dots, \gamma_{d-1})^T$. Hence, there is an integer vector $\sum_{i=0}^{d-1} (\alpha_i + t_i) y_i$ in $B_{(\alpha_0,\dots,\alpha_{d-1})}$ which is congruent to x modulo M.

Let $x' = \sum_{i=0}^{d-1} s'_i y_i \in \mathbb{Z}^d$ and $x'' = \sum_{i=0}^{d-1} s''_i y_i \in \mathbb{Z}^d$ be distinct elements of $B_{(\alpha_0,\dots,\alpha_{d-1})}$ which are congruent modulo M, i.e., there exists $z = (z_0, \dots, z_{d-1})^T \in \mathbb{Z}^d$ such that $x' = x'' + Mz$. There is $i_0 \in \{0, \dots, d-1\}$ such that $|z_{i_0}| \geq 1$ as $x' \neq x''$. Thus, $|s'_{i_0} - s''_{i_0}| = |z_{i_0}| \geq 1$ which contradicts that $x', x'' \in B_{(\alpha_0,\dots,\alpha_{d-1})}$. □

Now we compute number of congruence classes modulo β in $\mathbb{Z}[\omega]$ since two elements of $\mathbb{Z}[\omega]$ are congruent modulo β if and only if the corresponding vectors in $\mathbb{Z}^d$ are congruent modulo S_β.

Theorem 3.7. *Let ω be an algebraic integer of degree d and $\beta = \sum_{i=0}^{d} b_i \omega^i$, where $b_i \in \mathbb{Z}$, be such that $\deg \omega = \deg \beta$. The number of congruence classes modulo β in $\mathbb{Z}[\omega]$ is $|m_\beta(0)|$.*

Proof. Let $x, y \in \mathbb{Z}[\omega]$ and let S be the companion matrix of the minimal polynomial m_ω. Set $S_\beta = \sum_{i=0}^{d-1} b_i S^i$. Then

$$\begin{aligned} x \equiv y \mod \beta &\iff \exists z \in \mathbb{Z}[\omega] \colon x - y = \beta z \\ &\iff \exists z \in \mathbb{Z}[\omega] \colon \pi(x - y) = S_\beta \pi(z) \\ &\iff \pi(x) \equiv \pi(y) \mod S_\beta \,. \end{aligned}$$

Thus, the number of congruence classes modulo β is $|\det S_\beta|$ by Lemma 3.6. The statement follows from Lemma 3.5. □

Chapter 4

Convergence

We discuss convergence of the extending window method designed in Chapter 2. We prove a necessary condition of convergence of the whole method and we show that it is also a sufficient condition of convergence of Phase 1, using the tools from the previous chapter. Next, we establish a condition which implies non-convergence of Phase 2. The condition is formulated by existence of an infinite walk in a specific directed graph. We also prove that there is the same lower bound on the size of an alphabet from $\mathbb{Z}[\beta]$ as for integer alphabets when the extending method converges. We propose a way how an alphabet can be generated for a given base.

4.1 Convergence of Phase 1

In this section, we show that if the extending window method converges, then the base β must be expanding, i.e., all its conjugates are greater than one in modulus. Then we prove that it is also a sufficient condition for convergence of Phase 1.

We use the following notation:

Definition 4.1. Let ω be a complex number and $\beta \in \mathbb{Z}[\omega]$ be such that $|\beta| > 1$. Let $\mathcal{A} \subset \mathbb{Z}[\omega]$ be an alphabet. Set

$$\mathcal{A}[\beta] := \left\{ \sum_{i=0}^{N} a_i \beta^i \colon a_i \in \mathcal{A}, N \in \mathbb{N} \right\}.$$

The essential part of the proof that β must be expanding is Theorem 4.1 which is based on the papers of Akiyama, Thuswaldner and Zäimi [1, 2].

Theorem 4.1. *Let ω be a complex number and $\beta \in \mathbb{Z}[\omega]$ be such that $|\beta| > 1$. Let $\mathcal{A} \subset \mathbb{Z}[\omega]$ be an alphabet. If $\mathbb{N} \subset \mathcal{A}[\beta]$, then the number β is expanding.*

Proof. For all $n \in \mathbb{N}$ we may write

$$n = \sum_{i=0}^{N} a_i \beta^i ,$$

where $N \in \mathbb{N}$, $a_i \in \mathcal{A}$ and $a_N \neq 0$.

Set $m := \max\{|a| \colon a \in \mathcal{A}\}$. We take $n \in \mathbb{N}$ such that $n > m$. Since $|a_0| \leq m < n$, we have $N \geq 1$ and there is $i_0 \in \{1, 2, \ldots, N\}$ such that $a_{i_0} \neq 0$. Thus, ω is an algebraic

number as $a_i \in \mathcal{A} \subset \mathbb{Z}[\omega]$ and β can be expressed as an integer combination of powers of ω. Therefore, β is also an algebraic number.

Let β' be an algebraic conjugate of β. Since $\beta \in \mathbb{Z}[\omega] \subset \mathbb{Q}(\omega)$, there is an algebraic conjugate ω' of ω and an isomorphism $\sigma : \mathbb{Q}(\omega) \to \mathbb{Q}(\omega')$ such that $\sigma(\beta) = \beta'$. Now

$$n = \sigma(n) = \sum_{i=0}^{N} \sigma(a_i)(\beta')^i \,.$$

Set $\tilde{m} := \max\{|\sigma(a)| \colon a \in \mathcal{A}\}$. For all $n \in \mathbb{N}$, we have

$$n = |n| \leq \sum_{i=0}^{N} |\sigma(a_i)| \cdot |\beta'|^i \leq \sum_{i=0}^{\infty} |\sigma(a_i)| \cdot |\beta'|^i \leq \tilde{m} \sum_{i=0}^{\infty} |\beta'|^i \,.$$

Hence, the sum on the right side diverges which implies that $|\beta'| \geq 1$. Thus, all conjugates of β are at least one in modulus.

If the degree of β is one, the statement is obvious. Therefore, we may assume that $\deg \beta \geq 2$.

Suppose for contradiction that $|\beta'| = 1$ for an algebraic conjugate β' of β. The complex conjugate $\overline{\beta'}$ is also an algebraic conjugate of β. Take any algebraic conjugate γ of β and the isomorphism $\sigma' : \mathbb{Q}(\beta') \to \mathbb{Q}(\gamma)$ given by $\sigma'(\beta') = \gamma$. Now

$$\frac{1}{\gamma} = \frac{1}{\sigma'(\beta')} = \sigma'\left(\frac{1}{\beta'}\right) = \sigma'\left(\frac{\overline{\beta'}}{\beta'\overline{\beta'}}\right) = \sigma'\left(\frac{\overline{\beta'}}{|\beta'|^2}\right) = \sigma'(\overline{\beta'}) \,.$$

Hence, $\frac{1}{\gamma}$ is also an algebraic conjugate of β. Moreover, $\left|\frac{1}{\gamma}\right| \geq 1$ and $|\gamma| \geq 1$ which implies that $|\gamma| = 1$. We may set $\gamma = \beta$ which contradicts $|\beta| > 1$. Thus all conjugates of β are greater than one in modulus, i.e., β is an expanding algebraic number. □

Now we can easily prove that existence of a parallel addition algorithm with the rewriting rule $x - \beta$ implies that β is expanding.

Theorem 4.2. *Let $\mathcal{A} \subset \mathbb{Z}[\omega]$ be an alphabet such that $1 \in \mathcal{A}[\beta]$. If the extending window method with the rewriting rule $x - \beta$ converges for the numeration system $(\beta, \mathcal{A})$, then the base β is expanding.*

Proof. The existence of an algorithm for addition which is computable in parallel implies that the set $\mathrm{Fin}_{\mathcal{A}}(\beta)$ is closed under addition. Moreover, the set $\mathcal{A}[\beta]$ is closed under addition since there is no carry to the right when the rewriting rule $x - \beta$ is used. For any $n \in \mathbb{N}$, the sum $1 + 1 + \cdots + 1 = n$ is in $\mathcal{A}[\beta]$ by the assumption $1 \in \mathcal{A}[\beta]$. Therefore, $\mathbb{N} \subset \mathcal{A}[\beta]$ and thus the base β is expanding by Theorem 4.1. □

Since we know that it makes sense to consider only expanding base, we may ask if Phase 1 converges for such a base. The answer is positive, with some natural assumption on the alphabet $\mathcal{A}$. The following lemma provides a finite set of weight coefficients $\mathcal{Q}$.

Lemma 4.3. *Let ω be an algebraic integer, $\deg \omega = d$, and β be an expanding algebraic integer in $\mathbb{Z}[\omega]$. Let $\mathcal{A}$ and $\mathcal{B}$ be finite subsets of $\mathbb{Z}[\omega]$ such that $\mathcal{A}$ contains at least one representative of each congruence class modulo β in $\mathbb{Z}[\omega]$. There exists a finite set $\mathcal{Q} \subset \mathbb{Z}[\omega]$ such that $\mathcal{B} + \mathcal{Q} \subset \mathcal{A} + \beta\mathcal{Q}$.*

Proof. We use the isomorphism $\pi : \mathbb{Z}[\omega] \to \mathbb{Z}^d$ and β-norm $\|\cdot\|_\beta$ to give a bound on the elements of $\mathbb{Z}[\omega]$. Let γ be the smallest conjugate of β in modulus. Denote $C := \max\{\|b-a\|_\beta : a \in \mathcal{A}, b \in \mathcal{B}\}$. Consequently, set $R := \frac{C}{|\gamma|-1}$ and $\mathcal{Q} := \{q \in \mathbb{Z}[\omega] : \|q\|_\beta \leq R\}$. By Lemma 3.5, we have

$$\left|\left|\left|S_\beta^{-1}\right|\right|\right|_{S_\beta} = \max\{\frac{1}{|\beta'|} : \beta' \text{ is conjugate of } \beta\} = \frac{1}{|\gamma|}\,.$$

Also, $|\gamma| > 1$ as β is an expanding integer. Since $C > 0$, the set $\mathcal{Q}$ is nonempty. Any element $x = b + q \in \mathbb{Z}[\omega]$ with $b \in \mathcal{B}$ and $q \in \mathcal{Q}$ can be written as $x = a + \beta q'$ for some $a \in \mathcal{A}$ and $q' \in \mathbb{Z}[\omega]$ due to existence of a representative of each congruence class in $\mathcal{A}$. Using the isomorphism π, we may write $\pi(q') = S_\beta^{-1} \cdot \pi(b - a + q)$. We prove that q' is in $\mathcal{Q}$:

$$\begin{aligned}\left\|q'\right\|_\beta = \left\|\pi\left(q'\right)\right\|_{S_\beta} &= \left\|S_\beta^{-1} \cdot \pi\left(b-a+q\right)\right\|_{S_\beta} \leq \left|\left|\left|S_\beta^{-1}\right|\right|\right|_{S_\beta} \|b-a+q\|_\beta \\ &\leq \frac{1}{|\gamma|}\left(\|b-a\|_\beta + \|q\|_\beta\right) = \frac{1}{|\gamma|}\left(C+R\right) = \frac{C}{|\gamma|}\left(1 + \frac{1}{|\gamma|-1}\right) = R\,.\end{aligned}$$

Hence $q' \in \mathcal{Q}$ and thus $x = b + q \in \mathcal{A} + \beta\mathcal{Q}$.

Since there are only finitely many elements of $\mathbb{Z}^d$ bounded by the constant R, the set $\mathcal{Q}$ is finite. □

The way how candidates for the weight coefficients are chosen in Algorithm 4 is the same as in the proof of Lemma 4.3. Therefore, the convergence of Phase 1 is guaranteed by the following theorem.

Theorem 4.4. *Let ω be an algebraic integer and $\beta \in \mathbb{Z}[\omega]$. Let the alphabet $\mathcal{A} \subset \mathbb{Z}[\omega]$ be such that $\mathcal{A}$ contains at least one representative of each congruence class modulo β in $\mathbb{Z}[\omega]$. Let $\mathcal{B} \subset \mathbb{Z}[\omega]$ be the input alphabet.*

If β is expanding, then Phase 1 of the extending window method converges.

Proof. Let R be a constant and $\mathcal{Q}$ be a finite set from Lemma 4.3 for the alphabet $\mathcal{A}$ and input alphabet $\mathcal{B}$. We prove by induction that all intermediate weight coefficient sets $\mathcal{Q}_k$ in Algorithm 1 are subsets of the finite set $\mathcal{Q}$.

We start with $\mathcal{Q}_0 = \{0\}$ whose elements are bounded by any positive constant. Suppose that the intermediate weight coefficients set $\mathcal{Q}_k$ has elements bounded by the constant R. We see from the previous proof that the candidates obtained by Algorithm 4 for the set $\mathcal{Q}_k$ are also bounded by R. Thus, the next intermediate weight coefficients set $\mathcal{Q}_{k+1}$ has elements bounded by the constant R, i.e., $\mathcal{Q}_{k+1} \subset \mathcal{Q}$.

Since $\#\mathcal{Q}$ is finite and $\mathcal{Q}_0 \subset \mathcal{Q}_1 \subset \mathcal{Q}_2 \subset \cdots \subset \mathcal{Q}$, Phase 1 successfully ends. □

4.2 Convergence of Phase 2

We have no simple sufficient or necessary condition for convergence of Phase 2 on properties of a base β or an alphabet $\mathcal{A}$. Nevertheless, the non-convergence can be controled during a run of algorithm. An easy check of non-convergence can be done by finding $\mathcal{Q}_{[b,\ldots,b]}$ for each $b \in \mathcal{B}$ separately. This was already described in [12], but we recall it in this section with a simpler proof. For its purposes, we introduce a notion of stable

Phase 2, which is used also in the main result of this section – the control of non-convergence during Phase 2 is transformed into searching for a cycle in a directed graph.

Firstly, we mention some equivalent conditions of non-convergence of Phase 2.

Lemma 4.5. *The following statements are equivalent:*

i) Phase 2 does not converge,

ii) $\forall\, k \in \mathbb{N}\, \exists\, (w_0, \dots, w_{-k}) \in \mathcal{B}^{k+1}\colon \#\mathcal{Q}_{[w_0,\dots,w_{-k}]} \geq 2$,

iii) $\exists\, (w_{-j})_{j\geq 0}, w_{-j} \in \mathcal{B}\, \exists\, k_0 \forall k \geq k_0\colon \#\mathcal{Q}_{[w_0,\dots,w_{-k}]} = \#\mathcal{Q}_{[w_0,\dots,w_{-(k-1)}]} \geq 2$.

Proof. i) $\iff$ *ii):* The while loop in Algorithm 2 ends if and only if there exist $k \in \mathbb{N}$ such that $\#\mathcal{Q}_{[w_0,\dots,w_{-k}]} = 1$ for all $(w_0, \dots, w_{-k}) \in \mathcal{B}^{k+1}$.

ii) $\iff$ *iii):* There is an infinite sequence $(w_{-k})_{k\geq 0}$ such that $\#\mathcal{Q}_{[w_0,\dots,w_{-k}]} \geq 2$ for all $k \in \mathbb{N}$ since $\mathcal{Q}_{[w_0,\dots,w_{-k}]} \supset \mathcal{Q}_{[w_0,\dots,w_{-(k+1)}]}$. Hence, the sequence of integers $(\#\mathcal{Q}_{[w_0,\dots,w_{-k}]})_{k\geq 0}$ is eventually constant. The opposite implication is trivial. □

We need to ensure that choice of a possible weight coefficient set $\mathcal{Q}_{[w_0,\dots,w_{-k}]} \subset \mathcal{Q}_{[w_0,\dots,w_{-(k-1)}]}$ is determined by an input digit w_0 and a set $\mathcal{Q}_{[w_{-1},\dots,w_{-0}]}$, while the influence of the set $\mathcal{Q}_{[w_0,\dots,w_{-(k-1)}]}$ is limited. It is formalized in the following definition.

Definition 4.2. Let $\mathcal{B}$ be an alphabet of input digits. We say that Phase 2 is *stable* if

$$\mathcal{Q}_{[w_{-1},\dots,w_{-k}]} = \mathcal{Q}_{[w_{-1},\dots,w_{-(k-1)}]} \implies \mathcal{Q}_{[w_0,\dots,w_{-k}]} = \mathcal{Q}_{[w_0,\dots,w_{-(k-1)}]}$$

for all $k \in \mathbb{N}, k \geq 2$ and for all $(w_0, \dots, w_{-(k+1)}) \in \mathcal{B}^{k+1}$.

The definition may seem too restrictive, but note that $\mathcal{Q}_{[w_0,\dots,w_{-k}]}$ if fully determined by $\mathcal{Q}_{[w_{-1},\dots,w_{-k}]}$, $\mathcal{Q}_{[w_0,\dots,w_{-(k-1)}]}$ and w_0 for a fixed deterministic way of choice of possible weight coefficients sets. Therefore, the assumption $\mathcal{Q}_{[w_{-1},\dots,w_{-k}]} = \mathcal{Q}_{[w_{-1},\dots,w_{-(k-1)}]}$, i.e. carries from the right are the same, implies that the only difference in the choice of $\mathcal{Q}_{[w_0,\dots,w_{-k}]}$ and $\mathcal{Q}_{[w_0,\dots,w_{-(k-1)}]}$ is that $\mathcal{Q}_{[w_0,\dots,w_{-k}]}$ is a subset of $\mathcal{Q}_{[w_0,\dots,w_{-(k-1)}]}$, while $\mathcal{Q}_{[w_0,\dots,w_{-(k-1)}]}$ is chosen as a subset of $\mathcal{Q}_{[w_0,\dots,w_{-(k-2)}]}$. At the same time, $\mathcal{Q}_{[w_0,\dots,w_{-(k-1)}]}$ is a subset of $\mathcal{Q}_{[w_0,\dots,w_{-(k-2)}]}$. Thus, the property that Phase 2 is stable means that the same possible weight coefficients set is found even if it is constructed as a subset of smaller set. This is natural way how an algorithm of choice should be constructed – the set $\mathcal{Q}_{[w_0,\dots,w_{-k}]}$ is constructed such that

$$\mathcal{B} + \mathcal{Q}_{[w_{-1},\dots,w_{-k}]} \subset \mathcal{A} + \beta\mathcal{Q}_{[w_0,\dots,w_{-k}]}\,,$$

i.e., there is no reason to choose the set $\mathcal{Q}_{[w_0,\dots,w_{-k}]}$ to be a proper subset of $\mathcal{Q}_{[w_0,\dots,w_{-(k-1)}]}$ as we know that

$$\mathcal{B} + \underbrace{\mathcal{Q}_{[w_{-1},\dots,w_{-(k-1)}]}}_{=\mathcal{Q}_{[w_{-1},\dots,w_{-k}]}} \subset \mathcal{A} + \beta\mathcal{Q}_{[w_0,\dots,w_{-(k-1)}]}\,,$$

and $\mathcal{Q}_{[w_{-1},\dots,w_{-(k-1)}]}$ was chosen as sufficient. In other words, if $\mathcal{Q}_{[w_0,\dots,w_{-k}]}$ is a proper subset of $\mathcal{Q}_{[w_0,\dots,w_{-(k-1)}]}$, the set $\mathcal{Q}_{[w_0,\dots,w_{-(k-1)}]}$ might have been chosen better.

We may guarantee that Phase 2 is stable by wrapping the choice of the set $\mathcal{Q}_{[w_0,\dots,w_{-k}]}$ into a simple while loop, see Algorithm 8 in Chapter 6.

Now we use that finiteness of Phase 2 implies that there exists a length of window m such that the set $\mathcal{Q}^m_{[b]}$ contains only one element for all $b \in \mathcal{B}$, where $\mathcal{Q}^m_{[b]}$ is a shorter notation for

$$\mathcal{Q}_{[\underbrace{b,\ldots,b}_{m}]}\,.$$

The following theorem was proved in [12] with the assumption that Phase 2 is deterministic. Briefly, it says that $\#\mathcal{Q}^m_{[b]}$ must decrease every time we increase m, otherwise Phase 2 does not converge. When we consider only inputs of the form $bb\ldots b$ for some $b \in \mathcal{B}$, determinism implies that Phase 2 is stable. The given proof with Phase 2 being stable is slightly shorter.

Theorem 4.6. *Let $m_0 \in \mathbb{N}$ and $b \in \mathcal{B}$ be such that sets $\mathcal{Q}^{m_0}_{[b]}$ and $\mathcal{Q}^{m_0-1}_{[b]}$ produced by stable Phase 2 have the same size. Then*

$$\#\mathcal{Q}^m_{[b]} = \#\mathcal{Q}^{m_0}_{[b]} \qquad \forall m \geq m_0 - 1\,.$$

Particularly, if $\#\mathcal{Q}^{m_0}_{[b]} \geq 2$, Phase 2 does not converge.

Proof. As $\mathcal{Q}^{m_0}_{[b]} \subset \mathcal{Q}^{m_0-1}_{[b]}$, the assumption of the same size implies

$$\mathcal{Q}^{m_0}_{[b]} = \mathcal{Q}^{m_0-1}_{[b]}\,.$$

By the assumption that Phase 2 is stable, we have

$$\begin{aligned}\mathcal{Q}^{m_0}_{[b]} = \mathcal{Q}^{m_0-1}_{[b]} &\implies \mathcal{Q}^{m_0+1}_{[b]} = \mathcal{Q}^{m_0}_{[b]}\\ &\implies \mathcal{Q}^{m_0+2}_{[b]} = \mathcal{Q}^{m_0+1}_{[b]}\\ &\vdots\end{aligned}$$

This implies the statement.

If $\#\mathcal{Q}^{m_0}_{[b]} \geq 2$, then the statement *iii)* in Lemma 4.5 holds for the sequence $(b)_{k\geq 0}$. □

In general, it happens that $\mathcal{Q}_{[w_0,\ldots,w_{-k}]} = \mathcal{Q}_{[w_0,\ldots,w_{-(k-1)}]}$ for some combination of input digits $(w_0,\ldots,w_{-k}) \in \mathcal{B}^{k+1}$ and Phase 2 still converges. Thus, a condition which signifies non-convergence during Phase 2 is more complicated. It is formulated as searching for an infinite path in a so-called Rauzy graph.

Definition 4.3. Let $\mathcal{B}$ be an alphabet of input digits. Let Phase 2 be stable. Let $k \in \mathbb{N}, k \geq 2$. We set

$$V := \left\{(w_{-1},\ldots,w_{-k}) \in \mathcal{B}^k \colon \#\mathcal{Q}_{[w_{-1},\ldots,w_{-k}]} = \#\mathcal{Q}_{[w_{-1},\ldots,w_{-(k-1)}]}\right\}$$

and

$$E := \left\{(w_{-1},\ldots,w_{-k}) \to (w'_{-1},\ldots,w'_{-k}) \in V\times V \colon (w_{-2},\ldots,w_{-k}) = (w'_{-1},\ldots,w'_{-(k-1)})\right\}.$$

The directed graph $G_k = (V, E)$ is called *Rauzy graph of Phase 2 (for the window of length k).*

This term comes from combinatorics on words. The vertices of our graph are combinations of input digits for which the size of their possible weight coefficients sets did not decrease with an increment of length of the window. Whereas in combinatorics on words, vertices are given as factors of some language. But the directed edges are placed in the same manner – if some combination of digits without the first digit equals another combination without the last digit.

The structure of Rauzy graph G_k signifies whether the non-decreasing combinations are such that they cause non-convergence of Phase 2. Existence of an infinite walk in G_k implies that Phase 2 does not converge:

Theorem 4.7. *Let Phase 2 is stable. If there exists $k_0 \in \mathbb{N}, k_0 \geq 2$, and $(w_0, \ldots, w_{-k_0}) \in \mathcal{B}^{k_0+1}$ such that*

i) $\#\mathcal{Q}_{[w_0,\ldots,w_{-(k_0-1)}]} > 1$ *and*

ii) there exists an infinite walk $((w_{-1}^{(i)}, \ldots, w_{-k_0}^{(i)}))_{i\geq 1}$ in G_{k_0} which starts in the vertex

$$(w_{-1}^{(1)}, \ldots, w_{-k_0}^{(1)}) = (w_{-1}, \ldots, w_{-k_0}),$$

then Phase 2 does not converge.

Proof. Set

$$(w_k)_{k\geq 0} := w_0, w_1^{(1)}, \ldots, w_{k_0-1}^{(1)}, w_{k_0}^{(1)}, w_{k_0}^{(2)}, w_{k_0}^{(3)}, w_{k_0}^{(4)}, \ldots$$

We prove that $\#\mathcal{Q}_{[w_0,\ldots,w_{-k}]} = \#\mathcal{Q}_{[w_0,\ldots,w_{-(k_0-1)}]} > 1$ for all $k \geq k_0 - 1$, i.e., the condition *iii)* in Lemma 4.5 is satisfied.

Let $\ell \in \mathbb{N}$. Since $(w_{-(1+\ell)}, \ldots, w_{-(k_0+\ell)})$ is a vertex of G_{k_0}, the set $\mathcal{Q}_{[w_{-\ell},\ldots,w_{-(k_0+\ell)}]}$ equals $\mathcal{Q}_{[w_{-\ell},\ldots,w_{-(k_0+\ell-1)}]}$. As Phase 2 is stable, we have

$$\begin{aligned}
&\mathcal{Q}_{[w_{-\ell},\ldots,w_{-(k_0+\ell)}]} = \mathcal{Q}_{[w_{-\ell},\ldots,w_{-(k_0+\ell-1)}]} \\
\implies\ &\mathcal{Q}_{[w_{-(\ell-1)},\ldots,w_{-(k_0+\ell)}]} = \mathcal{Q}_{[w_{-(\ell-1)},\ldots,w_{-(k_0+\ell-1)}]} \\
&\vdots \\
\implies\ &\mathcal{Q}_{[w_{-1},\ldots,w_{-(k_0+\ell)}]} = \mathcal{Q}_{[w_{-1},\ldots,w_{-(k_0+\ell-1)}]} \\
\implies\ &\mathcal{Q}_{[w_0,\ldots,w_{-(k_0+\ell)}]} = \mathcal{Q}_{[w_0,\ldots,w_{-(k_0+\ell-1)}]}.
\end{aligned}$$

Hence, $\#\mathcal{Q}_{[w_0,\ldots,w_{-k}]} = \#\mathcal{Q}_{[w_0,\ldots,w_{-(k_0-1)}]} > 1$ for all $k \geq k_0 - 1$. □

We remark that existence of an infinite walk in a finite graph is equivalent to existence of a cycle in the graph. Thus, if there is an infinite walk, we may find another one whose sequence of vertices is eventually periodic. We use this fact in Section 6.1 which describes modified algorithm for Phase 2 which implements the result of Theorem 4.7.

4.3 Minimal alphabet $\mathcal{A}$

Frougny, Pelantová and Svodová [7] proved a lower bound on the size of an alphabet $0 \in \mathcal{A} \subset \mathbb{Z}$ of consecutive integers which enables parallel addition. In this section, we prove the same bound for an arbitrary alphabet $\mathcal{A} \in \mathbb{Z}[\beta]$. We recall their auxiliary

results in Theorem 4.8 an Lemma 4.9, but only for a parallel digit set conversion without anticipation as our rewriting rule $x - \beta$ does not require memory.

We remark that we indicate by the assumption $\mathcal{A} \in \mathbb{Z}[\beta]$ that we work in $\mathbb{Z}[\beta]$ instead of $\mathbb{Z}[\omega]$. Notice that congruence classes modulo β in $\mathbb{Z}[\omega]$ and $\mathbb{Z}[\beta]$ are generally different. It implies that even an integer alphabet behaves differently if $\beta \in \mathbb{Z}[\omega]$ and $\mathbb{Z}[\omega] \neq \mathbb{Z}[\beta]$. On the other hand, if $\beta = \pm\omega + c$, where $c \in \mathbb{Z}$, then $\mathbb{Z}[\omega] = \mathbb{Z}[\beta]$

The following theorem says that all classes modulo β in $\mathbb{Z}[\omega]$ which are contained in $\mathcal{A} + \mathcal{A}$ must have their representatives in $\mathcal{A}$.

Theorem 4.8. *Let ω be an algebraic integer. Let the base $\beta \in \mathbb{Z}[\omega]$ be such that $|\beta| > 1$ and the alphabet $\mathcal{A} \subset \mathbb{Z}[\omega]$ be such that $0 \in \mathcal{A}$. If there exists a p-local digit set conversion defined by the function $\phi\colon (\mathcal{A}+\mathcal{A})^p \to \mathcal{A}$ and $p = r + 1$, then the number $\phi(b, \dots, b) - b$ belongs to the set $(\beta - 1)\mathbb{Z}[\omega]$ for any $b \in \mathcal{A} + \mathcal{A}$.*

Proof. Let $b \in \mathcal{A} + \mathcal{A}$ and $a = \phi(b, \dots, b)$. For $n \in \mathbb{N}, n \geq 1$, we denote S_n the number represented by

$$^{\omega}0\underbrace{b\dots b}_{n}\bullet\underbrace{b\dots b}_{r}0^{\omega}\,.$$

The representation of S_n after the digit set conversion is of the form

$$^{\omega}0\underbrace{w_r\dots w_1}_{\beta^n W}\underbrace{a\dots a}_{n}\bullet\underbrace{\widetilde{w_1}\dots\widetilde{w_r}}_{\beta^{-r}\widetilde{W}}0^{\omega}\,,$$

where

$$W = \sum_{j=1}^{r} w_j\beta^{j-1} \qquad \text{and} \qquad \widetilde{W} = \sum_{j=1}^{r}\widetilde{w_j}\beta^{r-j}\,.$$

Since both representations have same value, we have

$$b\sum_{j=-r}^{n-1}\beta^j = W\beta^n + a\sum_{j=0}^{n-1}\beta^j + \beta^{-r}\widetilde{W}$$

$$b\sum_{j=-r}^{-1}\beta^j + b\frac{\beta^n - 1}{\beta - 1} = W\beta^n + a\frac{\beta^n - 1}{\beta - 1} + \beta^{-r}\widetilde{W}\,, \tag{4.1}$$

for all $n \geq 1$. We subtract this equation for n and $n - 1$ to obtain

$$b\frac{\beta^n - \beta^{n-1}}{\beta - 1} = W(\beta^n - \beta^{n-1}) + a\frac{\beta^n - \beta^{n-1}}{\beta - 1}\,.$$

We simplify it to

$$b = W(\beta - 1) + a\,. \tag{4.2}$$

Hence, $a = \phi(b, \dots, b) \equiv b$ modulo $\beta - 1$. □

If a base β has a real conjugate greater than one, then there are some extra requirements on the alphabet $\mathcal{A}$. For simplicity, we assume that the base β itself is real and greater than one. We show later that this assumption is without loss of generality.

Lemma 4.9. *Let ω be a real algebraic integer and the base $\beta \in \mathbb{Z}[\omega]$ be such that $\beta > 1$. Let the alphabet $\mathcal{A} \subset \mathbb{Z}[\omega]$ be such that $0 \in \mathcal{A}$ and denote $\lambda = \min \mathcal{A}$ and $\Lambda = \max \mathcal{A}$. If there exists a p-local digit set conversion defined by the function $\phi\colon (\mathcal{A}+\mathcal{A})^p \to \mathcal{A}$ and $p = r+1$, then:*

i) $\phi(b,\ldots,b) \neq \lambda$ for all $b \in \mathcal{A}+\mathcal{A}$ such that $b > \lambda$.

ii) $\phi(b,\ldots,b) \neq \Lambda$ for all $b \in \mathcal{A}+\mathcal{A}$ such that $b < \Lambda$.

iii) If $\Lambda \neq 0$, then $\phi(\Lambda,\ldots,\Lambda) \neq \Lambda$.

iv) If $\lambda \neq 0$, then $\phi(\lambda,\ldots,\lambda) \neq \lambda$.

Proof. To prove *i)*, assume in contradiction that $\phi(b,\ldots,b) = \lambda$. We proceed in the same manner as in Theorem 4.8, the equation (4.1) implies

$$b\sum_{j=-r}^{-1}\beta^j + b\frac{\beta^n-1}{\beta-1} = \beta^n W + \lambda\frac{\beta^n-1}{\beta-1} + \beta^{-r}\widetilde{W}\,.$$

We apply also the equation c to obtain

$$b\sum_{j=-r}^{-1}\beta^j + b\frac{\beta^n-1}{\beta-1} = \beta^n\frac{b-\lambda}{\beta-1} + \lambda\frac{\beta^n-1}{\beta-1} + \beta^{-r}\widetilde{W}\,.$$

Now we may simplify and estimate

$$b\sum_{j=-r}^{-1}\beta^j + \frac{-b}{\beta-1} = \frac{-\lambda}{\beta-1} + \beta^{-r}\sum_{j=1}^{r}\widetilde{w_j}\beta^{r-j}$$

$$b\underbrace{\left(\sum_{j=1}^{r}\frac{1}{\beta^j} - \frac{1}{\beta-1}\right)}_{-\sum_{j=r+1}^{\infty}\frac{1}{\beta^j}} = -\lambda\frac{1}{\beta-1} + \sum_{j=1}^{r}\widetilde{w_j}\beta^{-j} \geq \lambda\underbrace{\left(-\frac{1}{\beta-1} + \sum_{j=1}^{r}\frac{1}{\beta^j}\right)}_{-\sum_{j=r+1}^{\infty}\frac{1}{\beta^j}}\,.$$

Hence $b \leq \lambda$ which is a contradiction. The proof of *ii)* can be done in the same way.

For *iii)*, assume that $\phi(\Lambda,\ldots,\Lambda) = \Lambda$. Now consider a number T_q represented by

$$^{\omega}0 \bullet \underbrace{\Lambda\ldots\Lambda}_{r}\underbrace{(2\Lambda)\ldots(2\Lambda)}_{q}0^{\omega}\,.$$

After the digit set conversion, a representation is

$$^{\omega}0\underbrace{w_r\ldots w_1}_{W}\bullet z_1\ldots z_{r+q}0^{\omega}\,.$$

The value T_q preserves, thus,

$$\Lambda\sum_{j=1}^{r}\beta^{-j} + 2\Lambda\sum_{j=r+1}^{r+q}\beta^{-j} = W + \sum_{j=1}^{r+q}z_j\beta^{-j}\,.$$

But $W = 0$ from the equation (4.2). We estimate

$$\Lambda \sum_{j=1}^{r+q} \beta^{-j} + \Lambda \sum_{j=r+1}^{r+q} \beta^{-j} = \sum_{j=1}^{r+q} z_j \beta^{-j} \leq \Lambda \sum_{j=1}^{r+q} \beta^{-j}$$

$$\Lambda \sum_{j=r+1}^{r+q} \beta^{-j} \leq 0\,.$$

This contradicts that Λ is positive as it is a nonzero, maximal element of the alphabet $\mathcal{A}$ which contains 0. The proof of *iv)* is analogous. □

In order to prove the lower bound, we need to show that the alphabet $\mathcal{A}$ must contain all representatives modulo β and $\beta - 1$.

Theorem 4.10. *Let β be an algebraic integer such that $|\beta| > 1$. Let $0 \in \mathcal{A} \subset \mathbb{Z}[\beta]$ be an alphabet such that $1 \in \mathcal{A}[\beta]$. If addition in the numeration system $(\beta, \mathcal{A})$ which uses the rewriting rule $x - \beta$ is computable in parallel, then the alphabet $\mathcal{A}$ contains at least one representative of each congruence class modulo β and $\beta - 1$ in $\mathbb{Z}[\beta]$.*

Proof. The existence of an algorithm for addition with the rewriting rule $x - \beta$ implies that the set $\mathcal{A}[\beta]$ is closed under addition. By the assumption $1 \in \mathcal{A}[\beta]$, the set $\mathbb{N}$ is subset of $\mathcal{A}[\beta]$. Since $0 \in \mathcal{A}$, we have $\beta \cdot \mathcal{A}[\beta] \subset \mathcal{A}[\beta]$. Hence, $\mathbb{N}[\beta] \subset \mathcal{A}[\beta]$.

For any element $x = \sum_{i=0}^{N} x_i \beta^i \in \mathbb{Z}[\beta]$ there is an element $x' = \sum_{i=0}^{N} x'_i \beta^i \in \mathbb{N}[\beta]$ such that $x \equiv_\beta x'$ since $m_\beta(0) \equiv_\beta 0$ and $\beta^i \equiv_\beta 0$. As $x' \in \mathbb{N}[\beta] \subset \mathcal{A}[\beta]$, we have

$$x \equiv_\beta x' = \sum_{i=0}^{n} a_i \beta^i \equiv_\beta a_0\,,$$

where $a_i \in \mathcal{A}$. Hence, for any element $x \in \mathbb{Z}[\omega]$, there is a letter $a_0 \in \mathcal{A}$ such that $x \equiv_\beta a_0$.

In order to prove that there is at least one representative of each congruence class modulo $\beta - 1$ in the alphabet $\mathcal{A}$, we consider again an element $x = \sum_{i=0}^{N} x_i \beta^i \in \mathbb{Z}[\beta]$. Similarly, there is an element $x' = \sum_{i=0}^{N} x'_i \beta^i \in \mathbb{N}[\beta]$ such that $x \equiv_{\beta-1} x'$ since $m_{\beta-1}(0) \equiv_{\beta-1} 0$ and $(\beta - 1)^i \equiv_{\beta-1} 0$.

Since $x' \in \mathbb{N}[\beta] \subset \mathcal{A}[\beta]$,

$$x' = \sum_{i=0}^{n} a_i \beta^i\,,$$

where $a_i \in \mathcal{A}$. We prove by induction with respect to n that $x' \equiv_{\beta-1} a$ for some $a \in \mathcal{A}$. If $n = 0$, $x' = a_0$. Now we use the fact that if there is a parallel addition algorithm, for each letter $b \in \mathcal{A} + \mathcal{A}$, there is $a \in \mathcal{A}$ such that $b \equiv_{\beta-1} a$ (Theorem 4.8). For $n + 1$, we have

$$\begin{aligned} x' &= \sum_{i=0}^{n+1} a_i \beta^i = a_0 + \sum_{i=1}^{n+1} a_i \beta^i \\ &= a_0 + \beta \sum_{i=0}^{n} a_{i+1} \beta^i - \sum_{i=0}^{n} a_{i+1} \beta^i + \sum_{i=0}^{n} a_{i+1} \beta^i \\ &\equiv_{\beta-1} a_0 + (\beta - 1) \sum_{i=0}^{n} a_{i+1} \beta^i + a \equiv_{\beta-1} a_0 + a \equiv_{\beta-1} a' \in \mathcal{A}\,, \end{aligned}$$

where we use the induction assumption

$$\sum_{i=0}^{n} a_{i+1}\beta^i \equiv_{\beta-1} a\,.$$

□

Unfortunately, the claim cannot be generalized to modulo in $\mathbb{Z}[\omega]$ – there are numeration systems with integer alphabets which allow parallel addition, but these alphabets do not contain all representatives modulo $\beta-1$ in $\mathbb{Z}[\omega]$, see Table 7.5 and Examples D.13, D.15, D.17 and D.18. Nevertheless, each class modulo $\beta-1$ in $\mathbb{Z}[\omega]$ which is contained in $\mathcal{A}+\mathcal{A}$ must still have its representative in $\mathcal{A}$ according to Theorem 4.8.

The following lemma summarizes that if we have a parallel addition algorithm for a base β, then we easily obtain an algorithm also for conjugates of β.

Lemma 4.11. *Let ω be an algebraic integer with a conjugate ω'. Let $\beta \in \mathbb{Z}[\omega], |\beta|>1$ and let $\sigma : \mathbb{Q}(\omega) \to \mathbb{Q}(\omega')$ be an isomorphism such that $|\sigma(\beta)|>1$. Let φ be a digit set conversion in the base β from $\mathcal{A}+\mathcal{A}$ to $\mathcal{A}$. There exists is a digit set conversion φ' in the base β' from $\mathcal{A}'+\mathcal{A}'$ to $\mathcal{A}'$ where $\beta'=\sigma(\beta)$ and $\mathcal{A}'=\{\sigma(a)\colon a\in\mathcal{A}\}$.*

Proof. Let $\phi : \mathcal{A}^p \to \mathcal{A}$ be a mapping which defines φ with $p=r+t+1$. We define a mapping $\phi' : \mathcal{A}^p \to \mathcal{A}$ by

$$\phi'(w'_{j+t},\ldots,w'_{j-r}) = \sigma\left(\phi\left(\sigma^{-1}(w'_{j+t}),\ldots,\sigma^{-1}(w'_{j-r})\right)\right)\,.$$

Next, we define a digit set conversion $\varphi' : (\mathcal{A}'+\mathcal{A}') \to \mathcal{A}'$ by $\varphi'(w') = (z'_j)_{j\in\mathbb{Z}}$ where $w' = (w'_j)_{j\in\mathbb{Z}}$ and $z'_j = \phi'(w'_{j+t},\ldots,w'_{j-r})$. Obviously, if w' has only finitely many nonzero entries, then there is only finitely many nonzeros in $(z'_j)_{j\in\mathbb{Z}}$ since

$$\phi'(0,\ldots,0) = \sigma\left(\phi\left(\sigma^{-1}(0),\ldots,\sigma^{-1}(0)\right)\right) = \sigma\left(\phi\left(0,\ldots,0\right)\right) = \sigma\left(0\right) = 0\,.$$

The value of the number represented by w' is also preserved:

$$\begin{aligned}\sum_{j\in\mathbb{Z}} w'_j\beta'^j &= \sum_{j\in\mathbb{Z}} \sigma(w_j)\sigma(\beta)^j = \sigma\left(\sum_{j\in\mathbb{Z}} w_j\beta^j\right)\\ &= \sigma\left(\sum_{j\in\mathbb{Z}} z_j\beta^j\right) = \sigma\left(\sum_{j\in\mathbb{Z}} \phi\left(w_{j+t},\ldots,w_{j-r}\right)\beta^j\right)\\ &= \sum_{j\in\mathbb{Z}} \sigma(\phi\left(w_{j+t},\ldots,w_{j-r}\right))\beta'^j = \sum_{j\in\mathbb{Z}} z'_j\beta'^j\end{aligned}$$

where $w_j=\sigma^{-1}(w'_j)$ for $j\in\mathbb{Z}$ and $\varphi((w_j)_{j\in\mathbb{Z}}) = (z_j)_{j\in\mathbb{Z}}$. □

Finally, we put together that the alphabet $\mathcal{A}$ contains all representative modulo β and $\beta-1$, number of congruence classes and restrictions on the alphabet for a base with a real conjugate greater than one.

Theorem 4.12. *Let β be an algebraic integer such that $|\beta|>1$. Let $0 \in \mathcal{A} \subset \mathbb{Z}[\beta]$ be an alphabet such that $1 \in \mathcal{A}[\beta]$. If addition in the numeration system $(\beta, \mathcal{A})$ which uses the rewriting rule $x - \beta$ is computable in parallel, then*

$$\#\mathcal{A} \geq \max\{|m_\beta(0)|, |m_\beta(1)|\}\,.$$

Moreover, if β is such that it has a real conjugate greater than 1, then

$$\#\mathcal{A} \geq \max\{|m_\beta(0)|, |m_\beta(1)|+2\}\,.$$

Proof. By Theorem 4.10, there are all representatives modulo β and modulo $\beta - 1$ in the alphabet $\mathcal{A}$. The numbers of congruence classes are $|m_\beta(0)|$ and $|m_{\beta-1}(0)|$ by Theorem 3.7. Obviously, $m_{\beta-1}(x) = m_\beta(x+1)$. Thus $m_{\beta-1}(0) = m_\beta(1)$.

Let ϕ be a mapping which defines the parallel addition. According to Lemma 4.11, we may assume that β is real and greater than 1 in the proof of the second part. The assumption $1 \in \mathcal{A}[\beta]$ implies that $\Lambda > 0$. Thus, there are at least three elements in the alphabet $\mathcal{A}$, because $\mathcal{A} \ni \phi(\Lambda, \ldots, \Lambda) \neq \lambda$ and $\mathcal{A} \ni \phi(\Lambda, \ldots, \Lambda) \neq \Lambda$ by Lemma 4.9. It also implies that there are at least two representatives modulo $\beta - 1$ in the alphabet in the class which contains Λ, since $\phi(\Lambda, \ldots, \Lambda) \equiv_{\beta-1} \Lambda$.

If $\lambda \equiv_{\beta-1} \Lambda$, there must be one more element of the alphabet $\mathcal{A}$ in this class, since $\lambda \neq \Lambda$. Therefore, $\#\mathcal{A} \geq |m_\beta(1)|+2$.

The case that $\lambda \not\equiv_{\beta-1} \Lambda$ is divided into two subcases. Suppose now that $\lambda \neq 0$. Then $\phi(\lambda, \ldots, \lambda) \neq \lambda$ and hence there is one more element in the alphabet in the class containing λ. Thus, there are at least two congruence classes which contain at least two elements of the alphabet $\mathcal{A}$. Therefore, $\#\mathcal{A} \geq |m_\beta(1)|+2$.

If $\lambda = 0$, then all elements of $\mathcal{A} + \mathcal{A}$ are nonnegative and $\phi(b, \ldots, b) \neq 0$ for all $b \in (\mathcal{A}+\mathcal{A}) \setminus 0$. Suppose for contradiction, that there is no nonzero element of the alphabet $\mathcal{A}$ congruent to 0. We know that there is at least one representative of each congruence class class modulo $\beta - 1$ in $\mathcal{A}$ and at least two representatives in the congruence class which contains Λ. Let $k \in \mathbb{N}$ denote the number of elements which are in $\mathcal{A}$ extra to one element in each congruence class, i.e., $\#A = |m_\beta(1)|+k$. For $d \in \Lambda + \mathcal{A}$, the value $\phi(d, \ldots, d) \in \mathcal{A}$ is not congruent to 0 as it is nonzero and the class containing zero has only one element by the assumption. Therefore, the values $\phi(d, \ldots, d) \in \mathcal{A}$ for $|m_\beta(1)|+k$ distinct letters $d \in \Lambda + \mathcal{A}$ belong to only $|m_\beta(1)|-1$ congruence classes. Hence, there exists $e_1, \ldots e_k, e_{k+1} \in \mathcal{A}$, pairwise distinct, and $f_1, \ldots f_k, f_{k+1} \in \mathcal{A}$ such that $e_i \neq f_i$ and $\phi(e_i + \Lambda, \ldots, e_i + \Lambda) \equiv_{\beta-1} \phi(f_i + \Lambda, \ldots, f_i + \Lambda)$ for all $i, 1 \leq i \leq k+1$. Since

$$e_i + \Lambda \equiv_{\beta-1} \phi(e_i + \Lambda, \ldots, e_i + \Lambda) \equiv_{\beta-1} \phi(f_i + \Lambda, \ldots, f_i + \Lambda) \equiv_{\beta-1} f_i + \Lambda\,.$$

also $e_i \equiv_{\beta-1} f_i$ for all $i, 1 \leq i \leq k+1$. This is a contradiction since it implies that $\#\mathcal{A} = |m_\beta(1)|+k+1$. Hence, classes containing λ and Λ have both at least one more element of the alphabet $\mathcal{A}$, i.e., $\#\mathcal{A} \geq |m_\beta(1)|+2$. □

Since the proof is based on Theorem 4.10, it cannot be easily extended to alphabets which are subsets of $\mathbb{Z}[\omega]$.

4.4 Alphabet generation

Before we sketch how an alphabet $\mathcal{A}$ that allows parallel addition for a given base β can be generated, we summarize its necessary properties.

The alphabet $\mathcal{A}$ must contain representatives of all congruence classes modulo β in order to guarantee convergence of Phase 1, see assumptions of Theorem 4.4. According to Theorem 4.8, there must be representatives of all congruence classes modulo $\beta-1$ which are contained in $\mathcal{A}+\mathcal{A}$. Moreover, if the base β is real and greater than one, there must be another digit which is congruent modulo $\beta-1$ to the minimal, resp., maximal digit of $\mathcal{A}$ (Lemma 4.9). If the base β has a real conjugate $\sigma(\beta)>1$, then the same condition must hold for the minimal and maximal digit of the alphabet $\mathcal{A}'=\{\sigma(a)\colon a\in\mathcal{A}\}$ modulo $\sigma(\beta)-1$. Otherwise it contradicts Lemma 4.11.

For construction of an integer alphabet, we start with $a=1$ and examine whether $\mathcal{A}=\{-a+1,\ldots,-1,0,1,\ldots,a-1,a\}$ or $\mathcal{A}=\{-a,-a+1,\ldots,-1,0,1,\ldots,a-1,a\}$ is a suitable alphabet. If not, we increment a and check again. It might happen that there are not all representatives modulo β in $\mathbb{Z}[\omega]$ if $\mathbb{Z}[\omega]\neq\mathbb{Z}[\beta]$. Therefore, we stop without success when $a>m_\beta(0)$.

Generation of a non-integer alphabet is slightly more complicated. We start with $\mathcal{A}_0=\{-1,0,1\}$ and $a=1$. We generate the set

$$L_a=\left\{\sum_{i=0}^{d-1}a_i\omega^i\colon |a_i|\leq a\right\}.$$

We increment a until representatives of all classes modulo $\beta-1$ are contained in $\mathcal{A}_0\cup L_a$. The set $\mathcal{A}_0$ is extended to $\mathcal{A}_1$ by adding the smallest element in β-norm of each class of $\mathcal{A}_0\cup L_a$ modulo $\beta-1$. It is repeated in the same manner with $\mathcal{A}_1$ and congruences modulo β to obtain the alphabet $\mathcal{A}$.

If the base β is real and greater than one, check if $\mathcal{A}$ contains another element congruent to the minimal, resp. maximal digit λ, resp. Λ. If not, add $\lambda+\beta-1$, resp. $\Lambda-(\beta-1)$ to the alphabet $\mathcal{A}$. We proceed in the same manner also if the base β has a real conjugate $\sigma(\beta)>1$, taking $\sigma(\beta)$ as a base and the alphabet $\mathcal{A}'$.

We remark that this procedure does not guarantee that $\mathcal{A}$ has minimal size.

Chapter 5

Different methods in Phases 1 and 2

We mentioned in Chapter 2 that there is a lot of variability in both phases of the extending window method. Within this chapter, we propose various methods how an intermediate weight coefficients set $\mathcal{Q}_k$ can be extended to $\mathcal{Q}_{k+1}$ in Phase 1 and how the set $\mathcal{Q}_{[w_0,\ldots,w_{-k}]}$ can be constructed in Phase 2. The presented methods are based on many preliminary tests.

5.1 Different methods in Phase 1

We recall that the ambiguous part of Phase 1 is extending an intermediate weight coefficient set $\mathcal{Q}_k$ to $\mathcal{Q}_{k+1}$ so that

$$\mathcal{B} + \mathcal{Q}_k \subset \mathcal{A} + \beta\mathcal{Q}_{k+1}\,.$$

It means that a weight coefficient q such that $x = a + \beta q$ for some $a \in \mathcal{A}$ must be found for all $x \in \mathcal{B} + \mathcal{Q}_k$. Since the alphabet $\mathcal{A}$ is redundant, there might be more such weight coefficients. Let $C_x \subset \mathbb{Z}[\omega]$ be a set of all these candidates for some $x \in \mathcal{B} + \mathcal{Q}_k$, i.e., $C_x = \{q \in \mathbb{Z}[\omega] \colon x = a + \beta q, a \in \mathcal{A}\}$. Set $C := \{C_x \colon x \in \mathcal{B} + \mathcal{Q}_k\}$. For a given $x \in \mathcal{B} + \mathcal{Q}_k$, the set C_x is constructed so that all digits $a \in \mathcal{A}$ are tested whether $x - a$ is divisible by β according to Theorem 3.2. See Algorithm 3 which constructs the set C.

Now we extend the set $\mathcal{Q}_k$ to $\mathcal{Q}_{k+1}$ so that $C_x \cap \mathcal{Q}_{k+1} \neq \emptyset$ for all $C_x \in C$. We describe five methods how it can be done (1a, 1b, 1c, 1d and 1e).

As $Q_k \subset \mathcal{Q}_{k+1}$, set $\mathcal{Q}_{k+1} := \mathcal{Q}_k$. For methods 1a, 1b and 1d, add the element of all $C_x \in C$ such that $\#C_x = 1$ to $\mathcal{Q}_{k+1}$. Next, select elements from all $C_x \in C$ such that $C_x \cap \mathcal{Q}_{k+1} = \emptyset$ according to a chosen method and add them to $\mathcal{Q}_{k+1}$. Selection for different methods is following:

1a – all elements of C_x,

1b and 1c – all smallest elements in absolute value of C_x,

1d and 1e – all smallest elements in β-norm of C_x,

Note that more elements may be added by method 1e than 1d, resp. 1c than 1b, since adding necessary elements before ($\#C_x = 1$) may cause that $C_{x'} \cap \mathcal{Q}_{k+1} \neq \emptyset$ for some $C_{x'} \in C$.

The procedure is summarized in Algorithm 3. Another methods may decrease the number of added elements for instance by picking only one of all smallest elements.

We can slightly improve performance by substituting the set $\mathcal{B} + \mathcal{Q}_k$ by $(\mathcal{B} + \mathcal{Q}_k) \setminus (\mathcal{B} + \mathcal{Q}_{k-1})$ on the line 2 in Algorithm 4, because

$$\mathcal{B} + \mathcal{Q}_{k-1} \subset \mathcal{A} + \beta\mathcal{Q}_k \,.$$

Since $\mathcal{Q}_{k+1} \supset \mathcal{Q}_k$, $C_x \cap \mathcal{Q}_{k+1} \neq \emptyset$ for $x \in \mathcal{B} + \mathcal{Q}_{k-1}$.

Algorithm 3 Extending intermediate weight coefficients set

Input: previous weight coefficients set $\mathcal{Q}_k$, method number $M \in \{1a, 1b, 1c, 1d, 1e\}$

1: $\widetilde{\mathcal{Q}} := \mathcal{Q}_k$
2: **if** $M \in \{1a,1b,1d\}$ **then**
3: **for all** $C_x \in C$ **do**
4: **if** $\#C_x = 1$ **then**
5: Add the element of C_x to $\widetilde{\mathcal{Q}}$
6: **end if**
7: **end for**
8: **end if**
9: $\mathcal{Q}_{k+1} := \widetilde{\mathcal{Q}}$
10: By Algorithm 4, find set of candidates C
11: **for all** $C_x \in C$ **do**
12: **if** $C_x \cap \widetilde{\mathcal{Q}} = \emptyset$ **then**
13: **if** $M = 1a$ **then**
14: Add all elements of C_x to $\mathcal{Q}_{k+1}$
15: **else if** $M \in \{1b, 1c\}$ **then**
16: Add all smallest elements in absolute value of C_x to $\mathcal{Q}_{k+1}$
17: **else if** $M \in \{1d, 1e\}$ **then**
18: Add all smallest elements in β-norm of C_x to $\mathcal{Q}_{k+1}$
19: **end if**
20: **end if**
21: **end for**
22: **return** $\mathcal{Q}_{k+1}$

5.2 Different methods in Phase 2

We propose some methods which can be used in Phase 2 to construct a set $\mathcal{Q}_{[w_0,\dots,w_{-k}]}$.

The set of possible weight coefficients $\mathcal{Q}_{[w_0,\dots,w_{-k}]}$ must be a subset of the previous set of possible weight coefficients $\mathcal{Q}_{[w_0,\dots,w_{-(k-1)}]}$. It is constructed such that

$$w_0 + \mathcal{Q}_{[w_{-1},\dots,w_{-k}]} \subset \mathcal{A} + \beta\mathcal{Q}_{[w_0,\dots,w_{-k}]} \,,$$

i.e, it is determined also by the input digit w_0 and the set of carries from the right $\mathcal{Q}_{[w_{-1},\dots,w_{-k}]}$. Whereas in Phase 1 the constructed set $\mathcal{Q}_{k+1}$ is extended from $\mathcal{Q}_k$, now we reduce the set $\mathcal{Q}_{[w_0,\dots,w_{-(k-1)}]}$ to obtain $\mathcal{Q}_{[w_0,\dots,w_{-k}]}$. Set

$$D_x := \left\{ q_0 \in \mathcal{Q}_{[w_0,\dots,w_{-(k-1)}]} \colon \exists\, a \in \mathcal{A} : x = a + \beta q_0 \right\} ,$$

Algorithm 4 Search for set of candidates C

Input: the previous weight coefficients set $\mathcal{Q}_k$, alternatively also the set $\mathcal{Q}_{k-1}$

$C := \{\emptyset\}$
for all $x \in \mathcal{B} + \mathcal{Q}_k$ **do** {Alternatively, $x \in (\mathcal{B} + \mathcal{Q}_k) \setminus (\mathcal{B} + \mathcal{Q}_{k-1})$}
    $C_x := \emptyset$
    for all $a \in \mathcal{A}$ **do**
        if $(x - a)$ is divisible by β in $\mathbb{Z}[\omega]$ (using Theorem 3.2) **then**
            Add $\frac{x-a}{\beta}$ to C_x
        end if
    end for
    Add the set C_x to C
end for
return C

where $x \in w_0 + \mathcal{Q}_{[w_{-1},\dots,w_{-k}]}$, and

$$D := \left\{ D_x \colon x \in w_0 + \mathcal{Q}_{[w_{-1},\dots,w_{-k}]} \right\} .$$

The goal is to find $\mathcal{Q}_{[w_0,\dots,w_{-k}]}$ such that $D_x \cap \mathcal{Q}_{[w_0,\dots,w_{-k}]} \neq \emptyset$ for all $x \in w_0 + \mathcal{Q}_{[w_{-1},\dots,w_{-k}]}$. In other words, if at least one element of each covering sets $D_x \in D$ is contained in $\mathcal{Q}_{[w_0,\dots,w_{-k}]}$, then

$$w_0 + \mathcal{Q}_{[w_{-1},\dots,w_{-k}]} \subset \mathcal{A} + \beta \mathcal{Q}_{[w_0,\dots,w_{-k}]} .$$

We follow Algorithm 5 now. Choice of possible weight coefficients set starts with $\mathcal{Q}'_{[w_0,\dots,w_{-k}]} := \emptyset$. We add the elements of all $D_x \in D$ such that $\#D_x = 1$ to $\mathcal{Q}'_{[w_0,\dots,w_{-k}]}$. We remove from covering set D all sets $D_x \in D$ such that $\mathcal{Q}'_{[w_0,\dots,w_{-k}]} \cap D_x \neq \emptyset$.

Next, while D is nonempty, we pick an element q from $\bigcup D$ according to a chosen method, we add q to $\mathcal{Q}'_{[w_0,\dots,w_{-k}]}$ and we remove all sets D_x which contain q from D. Finally, the possible weight coefficient set $\mathcal{Q}_{[w_0,\dots,w_{-k}]} = \mathcal{Q}'_{[w_0,\dots,w_{-k}]}$ is found.

Before we explain different methods to pick an element from $\bigcup D$, we define a linear order on $\mathbb{Z}[\omega]$.

Definition 5.1. Let n be a positive integer. Let $(x_1, \dots, x_n)$ and $(y_1, \dots, y_n)$ be elements of $\mathbb{Z}^n$. We say that $(x_1, \dots, x_n)$ is *lexicographically smaller* than $(y_1, \dots, y_n)$, denoted by $(x_1, \dots, x_n) \preceq (y_1, \dots, y_n)$, if

$$x_1 < y_1 \text{ or } (x_1 = y_1 \wedge (x_2, \dots, x_n) \preceq (y_2, \dots, y_n)) .$$

If x and y are elements of $\mathbb{Z}[\omega]$, then we say that x is *lexicographically smaller* than y if

$$\pi(x) \preceq \pi(y) .$$

Roughly speaking, elements of $\mathbb{Z}[\omega]$ are first ordered according their coefficient of 1, then of ω, then of ω^2, etc.

Algorithm 6 outlines methods 2a, 2b, 2c, 2d and 2d. All methods select a set T of elements of $\bigcup D = \{D_x \colon D_x \in D\}$. If there is more than one element in T, then the lexicographically smallest element is picked.

Method 2a computes the center of gravity g of elements of $\bigcup D$ considered as complex numbers. The set T are elements of $\bigcup D$ which are closest to g in absolute value.

For remaining methods, set

$$D' := \left\{ D_x \in D \colon \#D_x = \min_{D_x \in D} \#D_x \right\}.$$

Method 2b computes the center of gravity g of already picked elements $\mathcal{Q}'_{[w_0,\dots,w_{-k}]}$ considered as complex numbers. The set T are elements of $\bigcup D'$ which are closest to g in absolute value.

Methods 2c, resp. 2d, builds the set T to be the smallest elements of $\bigcup D'$ in absolute value, resp. β-norm.

Method 2e computes the number n_q of $D_x \in D$ such that $q \in D_x$ for each $q \in \bigcup D$. The set T consists of elements q such that $n_q = n$ which are closest to g in absolute value, where g is again the center of gravity of already picked elements $\mathcal{Q}'_{[w_0,\dots,w_{-k}]}$ and $n = \max\{n_q \colon q \in \bigcup D\}$.

The motivation of these methods is to obtain the set $\mathcal{Q}_{[w_0,\dots,w_{-k}]}$ small and "compact" as it should be easier to cover it in the next iteration of Phase 2 (for $k+1$). Moreover, methods 2b, 2c and 2d prefer elements from the smallest covering sets D_x in the hope that the picked elements are also in the bigger covering sets. Method 2e takes first elements which covers the most of the sets D_x.

Other experimental methods can be found in the source code.

Algorithm 5 Search for set $\mathcal{Q}_{[w_0,\dots,w_{-k}]}$

Input: Input digit w_0, set of possible carries $\mathcal{Q}_{[w_{-1},\dots,w_{-k}]}$, previous set of possible weight coefficients $\mathcal{Q}_{[w_0,\dots,w_{-(k-1)}]}$

1: $D := \{\emptyset\}$
2: **for all** $x \in w_0 + \mathcal{Q}_{[w_{-1},\dots,w_{-k}]}$ **do**
3: $\quad D_x := \{q_0 \in \mathcal{Q}_{[w_0,\dots,w_{-(k-1)}]} \colon \exists\, a \in \mathcal{A} : x = a + \beta q_0\}$
4: $\quad$ Add D_x to D
5: **end for**
6: $\mathcal{Q}'_{[w_0,\dots,w_{-k}]} := \emptyset$
7: **for all** $D_x \in D$ **do**
8: $\quad$ **if** $\#D_x = 1$ **then**
9: $\quad\quad$ Add the element $q \in D_x$ to $\mathcal{Q}'_{[w_0,\dots,w_{-k}]}$
10: $\quad\quad$ Remove sets $D_{x'}$ such that $q \in D_{x'}$ from the set D
11: $\quad$ **end if**
12: **end for**
13: **while** $D \neq \emptyset$ **do**
14: $\quad$ By Algorithm 6, pick an element q from $\bigcup D$
15: $\quad$ Add the element q to $\mathcal{Q}'_{[w_0,\dots,w_{-k}]}$
16: $\quad$ Remove sets D_x such that $q \in D_x$ from the set D
17: **end while**
18: $\mathcal{Q}_{[w_0,\dots,w_{-k}]} := \mathcal{Q}'_{[w_0,\dots,w_{-k}]}$
19: **return** $\mathcal{Q}_{[w_0,\dots,w_{-k}]}$

Algorithm 6 Choose one element from the set of covering D

Input: set of coverings D, method number $M \in \{2a, 2b, 2c, 2d, 2e\}$, already added elements $\mathcal{Q}'_{[w_0,\dots,w_{-k}]}$

if M =2a **then**
  $g :=$ center of gravity of elements of $\bigcup D$ as complex numbers
  $T :=$ elements of $\bigcup D$ which are closest to g in absolute value
else if $M \in \{2b, 2c, 2d\}$ **then**
  $m := \min\{\#D_x : D_x \in D\}$
  $D' := \{D_x \in D : \#D_x = m\}$
  if $M = 2b$ **then**
    $g :=$ center of gravity of elements of $\mathcal{Q}'_{[w_0,\dots,w_{-k}]}$ as complex numbers
    $T :=$ elements of $\bigcup D'$ which are closest to g in absolute value
  else if $M = 2c$ **then**
    $T :=$ elements of $\bigcup D'$ which are smallest in absolute value
  else if $M = 2d$ **then**
    $T :=$ elements of $\bigcup D'$ which are smallest in β-norm
  end if
else if $M = 2e$ **then**
  $n_q := \#\{D_x \in D : q \in D_x\}$ for all $q \in \bigcup D$
  $n = \max\{n_q : q \in \bigcup D\}$
  $g :=$ center of gravity of elements of $\mathcal{Q}'_{[w_0,\dots,w_{-k}]}$ as complex numbers
  $T :=$ elements of $\{q \in \bigcup D : n_q = n\}$ which are closest to g in absolute value
end if
return the lexicographically smallest element of T according to Definition 5.1

Chapter 6

Design and implementation

In the first section of this chapter, we propose algorithms which may reveal non-convergence of Phase 2. They are based on the theorems from Chapter 4. We present a simple algorithm which makes Phase 2 stable. We develop Algorithm 11 that executes Phase 2 with all modifications. All designed algorithms are implemented in SageMath. The program is described in Section 6.2.

6.1 Modified Phase 2

In this section, we introduce algorithms based on the theorems proved in Section 4.2. The first one checks convergence of Phase 2 for $bb\dots b$ inputs. Next, we show that it is possible to make Phase 2 stable by wrapping the choice of a set of possible weight coefficients into a simple while loop. Finally, we present an algorithm for Phase 2 which includes all modifications – the mentioned check for $bb\dots b$ inputs and control of non-convergence by searching for an infinite walk in Rauzy graph G_k.

Checking $bb\dots b$ inputs

Algorithm 7 was proposed in [12]. It checks whether Phase 2 stops when it processes input digits $bb\dots b$. Sets $\mathcal{Q}^m_{[b]}$ can be easily constructed separately for each $b \in \mathcal{B}$ for given m. We build the set $\mathcal{Q}^m_{[b]}$ for input digits $bb\dots b$ in the same way as in Phase 2. This means that we first search for $\mathcal{Q}^1_{[b]}$ such that

$$b + \mathcal{Q} \subset \mathcal{A} + \beta\mathcal{Q}^1_{[b]}\,.$$

Until the set $\mathcal{Q}^m_{[b]}$ contains only one element, we increment the length of window m and, using Algorithm 8, we find a subset $\mathcal{Q}^{m+1}_{[b]}$ of the set $\mathcal{Q}^m_{[b]}$ such that

$$b + \mathcal{Q}^m_{[b]} \subset \mathcal{A} + \beta\mathcal{Q}^{m+1}_{[b]}\,.$$

In each iteration, we check whether the set $\mathcal{Q}^{m+1}_{[b]}$ is strictly smaller than the set $\mathcal{Q}^m_{[b]}$. If not, we know by Theorem 4.6 that Phase 2 does not converge because $\#\mathcal{Q}^m_{[b]}$ is eventually a constant greater than 2.

Hence, non-finiteness of Phase 2 can be revealed by running Algorithm 7 for each input digit $b \in \mathcal{B}$.

Algorithm 7 Check the input $bb\dots b$

Input: Weight coefficient set $\mathcal{Q}$, digit $b \in \mathcal{B}$

Ouput: TRUE if there is a unique weight coefficient for input $bb\dots b$, FALSE otherwise

1: Find minimal set $\mathcal{Q}^1_{[b]} \subset \mathcal{Q}$ such that

$$b + \mathcal{Q} \subset \mathcal{A} + \beta\mathcal{Q}^1_{[b]}\,.$$

2: $m := 1$
3: **while** $\#\mathcal{Q}^m_{[b]} > 1$ **do**
4: $\quad m := m + 1$
5: $\quad$ By Algorithm 8, find minimal set $\mathcal{Q}^m_{[b]} \subset \mathcal{Q}^{m-1}_{[b]}$ such that

$$b + \mathcal{Q}^{m-1}_{[b]} \subset \mathcal{A} + \beta\mathcal{Q}^m_{[b]}\,.$$

6: $\quad$ **if** $\#\mathcal{Q}^m_{[b]} = \#\mathcal{Q}^{m-1}_{[b]}$ **then**
7: $\quad\quad$ **return** FALSE
8: $\quad$ **end if**
9: **end while**
10: **return** TRUE

Stable Phase 2

Recall that in order to use Theorem 4.7 to check non-convergence of Phase 2, we require Phase 2 to be stable. An algorithm of choice of possible weight coefficients set can be easily modified to ensure stability of Phase 2, i.e.,

$$\mathcal{Q}_{[w_{-1},\dots,w_{-k}]} = \mathcal{Q}_{[w_{-1},\dots,w_{-(k-1)}]} \implies \mathcal{Q}_{[w_0,\dots,w_{-k}]} = \mathcal{Q}_{[w_0,\dots,w_{-(k-1)}]}$$

for all $(w_{-1},\dots,w_{-k}) \in \mathcal{B}^k$. If the algorithm is deterministic, then the set $\mathcal{Q}_{[w_0,\dots,w_{-k}]}$ is determined by $\mathcal{Q}_{[w_{-1},\dots,w_{-k}]}$, $\mathcal{Q}_{[w_0,\dots,w_{-(k-1)}]}$ and w_0. Similarly, the set $\mathcal{Q}_{[w_0,\dots,w_{-(k-1)}]}$ is determined by the set $\mathcal{Q}_{[w_{-1},\dots,w_{-(k-1)}]}$, $\mathcal{Q}_{[w_0,\dots,w_{-(k-2)}]}$ and w_0.

Suppose that $\mathcal{Q}_{[w_{-1},\dots,w_{-k}]} = \mathcal{Q}_{[w_{-1},\dots,w_{-(k-1)}]}$. Now, the only difference between $\mathcal{Q}_{[w_0,\dots,w_{-k}]}$ and $\mathcal{Q}_{[w_0,\dots,w_{-(k-1)}]}$ is that $\mathcal{Q}_{[w_0,\dots,w_{-k}]}$ is searched as a subset of $\mathcal{Q}_{[w_0,\dots,w_{-(k-1)}]}$, whereas $\mathcal{Q}_{[w_0,\dots,w_{-(k-1)}]}$ as a subset of $\mathcal{Q}_{[w_0,\dots,w_{-(k-2)}]}$. In order to find $\mathcal{Q}_{[w_0,\dots,w_{-(k-1)}]}$, we use Algorithm 5 repeatedly instead of once. In each iteration, the input digit w_0 and the set $\mathcal{Q}_{[w_{-1},\dots,w_{-(k-1)}]}$ remains the same but we search for $\mathcal{Q}_{[w_0,\dots,w_{-(k-1)}]}$ as a subset of $\mathcal{Q}'_w$, where $\mathcal{Q}'_w$ is the output of the previous iteration or $\mathcal{Q}_{[w_0,\dots,w_{-(k-2)}]}$ in the first iteration. The algorithm stops when $\mathcal{Q}_{[w_0,\dots,w_{-(k-1)}]} = \mathcal{Q}'_w$. This loop is described in Algorithm 8.

Now, when the set $\mathcal{Q}_{[w_0,\dots,w_{-k}]}$ is searched as a subset of $\mathcal{Q}_{[w_0,\dots,w_{-(k-1)}]}$ in the same manner, it runs with the same inputs as the last iteration of the search for $\mathcal{Q}_{[w_0,\dots,w_{-(k-1)}]}$. Hence, $\mathcal{Q}_{[w_0,\dots,w_{-k}]} = \mathcal{Q}_{[w_0,\dots,w_{-(k-1)}]}$.

Infinite walk in Rauzy graph G_k

As the Rauzy graph G_k is finite, there exist an infinite walk in G_k if and only if there exists an oriented cycle. Algorithm 9 checks whether some walk starting in $(w_{-1},\dots,w_{-k})$ enters such a cycle eventually. First, it checks if the vertex $(w_{-1},\dots,w_{-k})$ is in the graph G_k. If yes, all vertices which are accessible by appropriately directed edge from

Algorithm 8 Stable search for possible weight coefficient set $\mathcal{Q}_{[w_0,\dots,w_{-k}]}$

Input: Input digit w_0, set of possible carries $\mathcal{Q}_{[w_{-1},\dots,w_{-k}]}$, previous set of possible weight coefficients $\mathcal{Q}_{[w_0,\dots,w_{-(k-1)}]}$

$\mathcal{Q}'_w := \mathcal{Q}_{[w_0,\dots,w_{-(k-1)}]}$
while TRUE **do**
    By Algorithm 5, find the set of possible weight coefficients $\mathcal{Q}_{[w_0,\dots,w_{-k}]}$ as a subset of $\mathcal{Q}'_w$ instead of the previous set of weight coefficients $\mathcal{Q}_{[w_0,\dots,w_{-(k-1)}]}$.
    if $\mathcal{Q}_{[w_0,\dots,w_{-k}]} = \mathcal{Q}'_w$ **then**
        return $\mathcal{Q}_{[w_0,\dots,w_{-k}]}$
    end if
    $\mathcal{Q}'_w := \mathcal{Q}_{[w_0,\dots,w_{-k}]}$
end while

$(w_{-1},\dots,w_{-k})$ are entered by Algorithm 10. It is called recursively to enter all accessible vertices from the given one. The already traversed path is passed in each call and if an vertex is visited second time, then the cycle is found and algorithm ends. If all branches of the recursion ends up without visiting some vertex twice, then there is no cycle and thus no infinite path.

Note that saving of the traversed path requires only the label of the first vertex and last digits of the labels of next visited vertices due to the construction of Rauzy graph.

Algorithm 9 Check if there is in an infinite walk in G_k starting in $(w_{-1},\dots,w_{-k})$

Input: Rauzy graph G_k, combination of input digits $(w_{-1},\dots,w_{-k})$
Ouput: Return TRUE if TRUE is returned in any step of the recursion, that is when a walk $w_1, w_2, \dots$ enters a directed cycle in G_k. Otherwise return FALSE.

if $(w_{-1},\dots,w_{-k}) \in G_k$ **then**
    By Algorithm 10, enter next vertices from $(w_{-1},\dots,w_{-k})$ with the traversed path $(w_{-1},\dots,w_{-k})$.
else
    return FALSE
end if
return FALSE

Phase 2 with control of non-convergence

Algorithm 11 modifies the basic proposal of Phase 2 to reveal its possible non-convergence. First, the necessary condition given by Theorem 4.6 is checked, i.e., the convergence of Phase 2 for inputs $bb\dots b$ is verified by Algorithm 7 for all $b \in \mathcal{B}$.

Then we proceed in the same way as in Algorithm 2 with the following modifications: it is sufficient to process only $(w_0,\dots,w_{-k}) \in \mathcal{B}^{k+1}$ such that $\#\mathcal{Q}_{[w_0,\dots,w_{-(k-1)}]} > 1$ since if $\#\mathcal{Q}_{[w_0,\dots,w_{-(k-1)}]} = 1$, then $\mathcal{Q}_{[w_0,\dots,w_{-k'}]} = \mathcal{Q}_{[w_0,\dots,w_{-(k-1)}]}$ for all $k' > k$.

Moreover, we check possible non-convergence according to Theorem 4.7. If $\mathcal{Q}_{[w_0,\dots,w_{-k}]}$ equals $\mathcal{Q}_{[w_0,\dots,w_{-(k-1)}]}$, then the vertex $\mathcal{Q}_{[w_0,\dots,w_{-k}]}$ is added into the Rauzy graph G_{k+1} and the Rauzy graph G_k is examined by Algorithm 9 whether it contains an infinite walk starting in $(w_{-1},\dots,w_{-k})$. Note that $\mathcal{Q}_{[w_0,\dots,w_{-k}]} = \mathcal{Q}_{[w_0,\dots,w_{-(k-1)}]}$ is a necessary condition for $(w_{-1},\dots,w_{-k})$ be a vertex of G_k. It also implies that $\#\mathcal{Q}_{[w_0,\dots,w_{-k}]} > 1$.

Algorithm 10 Enter vertices from $(w'_{-1}, \ldots, w'_{-k})$

Input: Rauzy graph G_k, vertex $(w'_{-1}, \ldots, w'_{-k})$, traversed path $(w_1, \ldots, w_l)$.

1: **for all** $w'_{k+1} \in \mathcal{B}$ such that $(w'_{-1}, \ldots, w'_{-k}) \to (w'_{-2}, \ldots, w'_{-(k+1)}) \in G_k$ **do**
2: **if** $(w'_{-2}, \ldots, w'_{-(k+1)})$ is in the traversed path $(w_1, \ldots, w_l)$ **then**
3: **return** TRUE
4: **else**
5: By Algorithm 10, enter next vertices from $(w'_{-2}, \ldots, w'_{-(k+1)})$ with the traversed path $(w_1, \ldots, w_l, w'_{(k+1)})$.
6: **end if**
7: **end for**

We remark that non-convergence caused by an input $bb \ldots b$ for some $b \in \mathcal{B}$ would be revealed also as a cycle in a Rauzy graph. Nevertheless, the number of calls of Algorithm 5 during Algorithm 7 is at most $\#Q$ for each $b \in \mathcal{B}$, while it grows exponentially with the length of window in search for a weight function. Thus, we save a lot of computational time in cases which fail already on $bb \ldots b$ inputs. At the same time, the cost of this check is low in other cases.

6.2 Implementation

Our implementation of the design is based on the program attached to [12]. The chosen programming language is SageMath. It is Python-based language with numerous implemented mathematical structures. That is the main reason of our choice – the extending window method requires to handle elements of $\mathbb{Z}[\omega]$ and arithmetic operations in this set. Moreover, SageMath provides various data structures and plotting tools. The code is simpler and more similar to pseudocode than if we implemented in pure C++. Due to it, we may concern ourselves with the algorithmic part of the problem instead of difficult programming. Unfortunately, SageMath is much slower than C++.

The implementation is object-oriented. It consists of five classes. Class *AlgorithmForParallelAddition* contains structures for computations in $\mathbb{Z}[\omega]$. The build-in classes *PolynomialQuotientRing* and *NumberField* are used to represent elements of $\mathbb{Z}[\omega]$ as an algebraic and complex numbers. The class also links together all functions and instances of other classes which are necessary to construct an algorithm for digit set conversion from $\mathcal{B}$ to $\mathcal{A}$ by the extending window method. The input parameters are an algebraic integer ω given by its minimal polynomial m_ω and an approximate complex value, a base $\beta \in \mathbb{Z}[\omega]$, an alphabet $\mathcal{A} \subset \mathbb{Z}[\omega]$ and input alphabet $\mathcal{B} \subset \mathbb{Z}[\omega]$.

We remark that it must be manually verified whether the input alphabet $\mathcal{B}$ such that $\mathcal{B} \neq \mathcal{A} + \mathcal{A}$ is sufficient for construction of a parallel addition algorithm, see remark at the end of Section 2.1.

Phase 1 of the extending window method is implemented in class *WeightCoefficientsSetSearch* and Phase 2 in *WeightFunctionSearch*. Class *WeightFunction* serves to save a function q computed in Phase 2. The last class *ExceptionParAdd* is inherited from build-in class *Exception* to distinguish between errors which are raised by the algorithm of the extending window method and other ones.

We use notation from previous chapters in descriptions of the classes. We list only the most important methods of each class, see commented source code for all of them.

Algorithm 11 Modified search for a weight function (Phase 2)

Input: weight coefficients set $\mathcal{Q}$

for all $b \in \mathcal{B}$ **do**
  if not Check the input $bb\dots b$ by Algorithm 7 **then**
    return Phase 2 does not converge for input $bb\dots b$.
  end if
end for
for all $w_0 \in \mathcal{B}$ **do**
  By Algorithm 8, find set $\mathcal{Q}_{[w_0]} \subset \mathcal{Q}$ such that

$$w_0 + \mathcal{Q} \subset \mathcal{A} + \beta\mathcal{Q}_{[w_0]}$$

end for
$k := 0$
while $\max\{\#\mathcal{Q}_{[w_0,\dots,w_{-k}]} : (w_0,\dots,w_{-k}) \in \mathcal{B}^{k+1}\} > 1$ **do**
  $k := k + 1$
  for all $\{(w_0,\dots,w_{-k}) \in \mathcal{B}^{k+1} : \#\mathcal{Q}_{[w_0,\dots,w_{-(k-1)}]} > 1\}$ **do**
    By Algorithm 8, find set $\mathcal{Q}_{[w_0,\dots,w_{-k}]} \subset \mathcal{Q}_{[w_0,\dots,w_{-(k-1)}]}$ such that

$$w_0 + \mathcal{Q}_{[w_{-1},\dots,w_{-k}]} \subset \mathcal{A} + \beta\mathcal{Q}_{[w_0,\dots,w_{-k}]}\,.$$

    if $\mathcal{Q}_{[w_0,\dots,w_{-k}]} = \mathcal{Q}_{[w_0,\dots,w_{-(k-1)}]}$ **and** $k \geq 2$ **then**
      Add the vertex $(w_0,\dots,w_{-k})$ to the Rauzy graph G_{k+1}.
      if Check infinite walks in G_k starting in $(w_{-1},\dots,w_{-k})$ by Alg. 9 **then**
        return Phase 2 does not converge.
      end if
    end if
  end for
end while
$r := k + 1$
for all $l \in \mathbb{N}, l \leq r$ **do**
  for all $\{(w_0,\dots,w_{-l}) \in \mathcal{B}^{l+1} : \mathcal{Q}_{[w_0,\dots,w_{-l}]}$ was found, $\#\mathcal{Q}_{[w_0,\dots,w_{-l}]} = 1\}$ **do**
    for all $(w_{-(l+1)},\dots,w_{-r}) \in \mathcal{B}^{r-l}$ **do**
      $q(w_0,\dots,w_{-r}) :=$ only element of $\mathcal{Q}_{[w_0,\dots,w_{-l}]}$
    end for
  end for
end for
return q

For basic use, load `AlgorithmForParallelAddition.sage`, create an instance of *AlgorithmForParallelAddition* and call **findWeightFunction()**.

We also provide an interface – a shell script with given parameters is executed. If the network access is enabled and the modul `gspread` is installed (see [9]), then results of computation are automatically saved to Google spreadsheet ParallelAddition_results. The spreadsheet can be also used for loading input parameters. Both interfaces are described Section 6.3. The whole implementation is on the attached DVD or it can be downloaded from `https://github.com/Legersky/ParallelAddition`.

Class AlgorithmForParallelAddition

The necessary structures for computation in $\mathbb{Z}[\omega]$ are constructed when an instance of this class is created. The set $\mathbb{Z}[\omega]$ is represented by *PolynomialQuotientRing* which is obtained by factorization of *PolynomialRing* over integers by polynomial m_ω. We remark that arithmetic operations in $\mathbb{Z}[\omega]$ are independent on the choice of root of the minimal polynomial m_ω. Since comparison of absolute value of numbers in $\mathbb{Z}[\omega]$ is required, we specify ω by its approximate complex value to form a factor ring of rational polynomials by using class *NumberField*. Elements of this class can be coerced to complex numbers and their absolute value is computed.

The constructor of class *AlgorithmForParallelAddition* is
__init__(*minPol_str, embd, alphabet, base, name='NumerationSystem', inputAlphabet='', printLog=True, printLogLatex=False, verbose=0*)

> The method takes *minPol_str* which is a string of symbolic expression in the variable x of an irreducible polynomial m_ω. The closest root of *minPol_str* to *embd* is used as the ring generator ω (see more in the documentation of *NumberField* in SageMath [14]). The structures for $\mathbb{Z}[\omega]$ are constructed as described above.
>
> A base β is given by a symbolic expression *base* where `omega` is used for ω. Setter **setBase**(*base*) checks if the base β is expanding and inspects whether it has a real conjugate greater than one.
>
> The parameter *alphabet* and *inputAlphabet* expect a string with a list of symbolic expressions (use `omega` for ω) to define an alphabet $\mathcal{A}$ and input alphabet $\mathcal{B}$. If the string *alphabet* is empty, then an alphabet $\mathcal{A} \subset \mathbb{Z}[\omega]$ is generated such that it contains all representatives modulo β and $\beta - 1$ in $\mathbb{Z}[\omega]$, see the end of Section 4.3. If *alphabet* is `'integer'`, an algorithm attempts to find an integer alphabet. An exception is raised when alphabet generating fails. If *inputAlphabet* is empty, then $\mathcal{B}$ is set to $\mathcal{A} + \mathcal{A}$.
>
> Messages saved to logfile during existence of an instance are printed (using LaTeX) on standard output depending on *printLog* and *printLogLatex*. The level of messages for a development is set by *verbose*.
>
> Example:
> ```
> alg= AlgorithmForParallelAddition('x^2+x+1',-0.5+0.8*I,
>  '[0,1,-1,omega,-omega,-omega-1,omega+1]','omega-1','Eisenstein')
> ```

Methods for the construction of a weight function for a digit set conversion from $\mathcal{B}$ to $\mathcal{A}$ are the following.

checkAlphabet()

It is verified that the alphabet $\mathcal{A}$ contains all representatives mod β in and that all elements of the input alphabet $\mathcal{B}$ have their representatives mod $\beta - 1$ in the alphabet $\mathcal{A}$ according to Theorem 4.12, including the case when the base β has a real conjugate greater than one.

_findWeightCoefSet(*max_iterations, method_number***)**

An instance of *WeightCoefficientsSetSearch(self,method_number)* is created and its method **findWeightCoefficientsSet**(*max_iterations*) is called to obtain a weight coefficients set $\mathcal{Q}$.

_findWeightFunction(*max_input_length,method_number***)**

An instance of *WeightFunctionSearch(self, $\mathcal{Q}$, method_number, maxInputs)* is created and its methods **check_one_letter_inputs**(*max_input_length*) and **findWeightFunction**(*max_input_length*) are called to obtain a weight function q.

findWeightFunction(*max_iterations=infinity, max_input_length=infinity, method_weightCoefSet=None, method_weightFunSearch=None***)**

The method calls **checkAlphabet**() and returns the weight function q obtained by calling
_findWeightCoefSet(*max_iterations, method_weightCoefSet*) and
_findWeightFunction(*max_input_length, method_weightFunSearch*).

Methods for the addition and the digit set conversion computable in parallel are the following:

addParallel(*a,b***)**

Numbers represented by the lists of digits a and b are summed up digitwise and the result is converted by **parallelConversion**(). If $\mathcal{B} \neq \mathcal{A} + \mathcal{A}$ and a digitwise sum is not in $\mathcal{B}$, then an exception is raised.

parallelConversion(*_w***)**

The method returns $(\beta, \mathcal{A})$-representation of the number represented by the list *_w* of digits from the input alphabet $\mathcal{B}$. According to the equation (2.2), it is computed locally by using the weight function q. An exception is raised when an outputted digit is not in the alphabet $\mathcal{A}$.

The correctness of the implementation of the extending window for a given numeration system can be verified by

sanityCheck_conversion(*num_digits***)**

It is checked whether the values of all possible numbers of the length *num_digits* with digits in the input alphabet $\mathcal{B}$ are the same as their $(\beta, \mathcal{A})$-representation obtained by **parallelConversion**().

Class WeightCoefficientsSetSearch

Class *WeightCoefficientsSetSearch* implements Phase 1 of the extending window method described in Section 2.3 with different methods how an intermediate weight coefficients set $\mathcal{Q}_k$ is extended to $\mathcal{Q}_{k+1}$. Algorithm 3 explains methods 1a, 1b, 1c, 1d and 1e. There are also implemented other experimental methods denoted by numbers. Method 1a corresponds to 14, 1b to 12, 1c to 16, 1d to 13 and 1e to 15.

The constructor of the class is

__init__(*algForParallelAdd, method*)

The ring generator ω, base β, an alphabet $\mathcal{A}$ and input alphabet $\mathcal{B}$ are initialize by values obtained from *algForParallelAdd*. The parameter *method* is a number of an experimental method or `'1a'`, `'1b'`, etc. The chosen method determines how an intermediate weight coefficients set $\mathcal{Q}_k$ is extended to $\mathcal{Q}_{k+1}$. If *None*, then the method 1d from Algorithm 3 is used as default.

Class methods implementing Phase 1 are the following:

_findCandidates(*to_cover*)

The method returns the list of lists *candidates*, which corresponds to a covering set C in Algorithm 4, such that each element in *to_cover* is covered by all values of the appropriate list in *candidates*.

_chooseQk_FromCandidates(*candidates*)

The method takes the previous intermediate weight coefficients set $\mathcal{Q}_k$ and constructs a new intermediate weight coefficients set $\mathcal{Q}_{k+1}$ from *candidates* by Algorithm 3.

_getQk(*to_cover*)

Methods **_chooseQk_FromCandidates**() and **_findCandidates**(*to_cover*) are linked together to return an intermediate weight coefficients set $\mathcal{Q}_{k+1}$.

findWeightCoefficientsSet(*maxIterations*)

According to Algorithm 1, a weight coefficients set $\mathcal{Q}$ is constructed by iterative using **_getQk**(). A computation is aborted if the number of iterations exceeds *maxIterations*.

Class WeightFunctionSearch

This class implements Algorithm 11 of modified Phase 2 of the extending window method from Section 6.1 with different methods of choice of possible weight coefficients sets and control of non-convergence. Methods 2a, 2b, 2c, 2d and 2e from Algorithm 6 correspond to 9, 15, 22, 23 and 14 respectively. A weight function q is returned by method **findWeightFunction**(). The constructor of the class is

__init__(*algForParallelAdd, weightCoefSet, method*)

The ring generator ω, base β, alphabet $\mathcal{A}$ and input alphabet $\mathcal{B}$ are initialized by the values obtained from *algForParallelAdd*. The weight coefficients set $\mathcal{Q}$ is set to *weightCoefSet*. The parameter *method* (the number of an experimental method or `'2a'`, `'2b'`, etc.) determines the way of the choice of a possible weight coefficients set $\mathcal{Q}_{[w_0,\ldots,w_{-k}]}$ from $\mathcal{Q}_{[w_0,\ldots,w_{-(k-1)}]}$, see Algorithm 6. If *method* is *None*, then the default method is 2b. Possible weight coefficients sets are saved in a dictionary *_Qw_w*, which is set to be empty.

The following methods are used for search for a weight function q:

_find_weightCoef_for_comb_B(*combinations***)**

All combinations of input digits $(w_0, \ldots, w_{-(k-1)}) \in \mathcal{B}^k$ in *combinations* such that $\#\mathcal{Q}_{[w_0,\ldots,w_{-(k-1)}]} > 1$ are extended by all letters $w_{-k} \in \mathcal{B}$. A possible weight coefficients set $\mathcal{Q}_{[w_0,\ldots,w_{-k}]}$ is constructed by the method **_findQw**$((w_0, \ldots, w_{-(k-1)}))$. If there is only one element in $\mathcal{Q}_{[w_0,\ldots,w_{-k}]}$, it is saved as a solved input of the weight function q. Otherwise, the set $\mathcal{Q}_{[w_0,\ldots,w_{-k}]}$ is saved in *_Qw_w* as an unsolved combination which requires extending of the window. The unsolved combinations are returned.

_findQw(*w_tuple***)**

The method returns a set $\mathcal{Q}_{[w_0,\ldots,w_{-k}]} = \mathcal{Q}_{[\text{w\_tuple}]}$ by wrapping **_findQw_once** () into a while loop according to Algorithm 8. If $\mathcal{Q}_{[w_0,\ldots,w_{-k}]} = \mathcal{Q}_{[w_0,\ldots,w_{-(k-1)}]}$, then add a vertex $(w_0, \ldots, w_{-k})$ to a Rauzy graph G_{k+1} and call **_checkCycles** (*w_tuple*) in G_k.

_findQw_once(*w_tuple, Qw_prev***)**

The set $\mathcal{Q}'_{[w_0,\ldots,w_{-k}]} = \mathcal{Q}'_{[\text{w\_tuple}]}$ obtained by Algorithm 5 as a subset of *Qw_prev*= $\mathcal{Q}_{[w_0,\ldots,w_{-(k-1)}]}$ is returned. The set of possible carries from the right $\mathcal{Q}_{[w_{-1},\ldots,w_{-k}]}$ is taken from the class attribute *_Qw_w*. The methods of Algorithm 6 are implemented here along with the experimental ones.

_checkCycles(*w_tuple***)**

Using Algorithm 9, it is controlled if there is a cycle in the Rauzy graph G_k starting in the vertex which label equals *w_tuple* without the first digit. If yes, then an exception *RuntimeErrorParAdd* is raised.

findWeightFunction(*max_input_length***)**

The method attempts to construct a weight function q by Algorithm 11. It increments the window length and calls the method **_find_weightCoef_for_comb_B** () until a unique weight coefficient is found for all possible combinations of input digits. If the length of window exceeds *max_input_length*, then an exception *RuntimeErrorParAdd* is raised.

check_one_letter_inputs(*max_input_length***)**

It is checked by Algorithm 7 if there is a unique weight coefficient for inputs $(b, b, \ldots, b) \in \mathcal{B}^r$ for some length of window r. An exception *RuntimeErrorParAdd* is raised in the case of an infinite loop. Otherwise the list of input digits b which require the largest length of window is returned.

Class WeightFunction

This class serves for saving a weight function q. The methods are following:

__init__(*B***)**

Set the input alphabet $\mathcal{B}$ to B and maximum length of window to 1. Initialize an attribute *_mapping* for saving the weight function q to an empty dictionary.

addWeightCoefToInput(*_input, coef*)

Save the weight coefficient *coef* for the combination of digits *_input* to *_mapping*. The digits of *_input* must be in the input alphabet $\mathcal{B}$.

getWeightCoef(w)

The digits of the list w are taken from the left until a weight coefficient in the dictionary *_mapping* is found.

The result of the method **getWeightCoef**() is used to make this class callable, i.e., if $\_q$ is an instance of *WeightFunction*, then $\_q$.**getWeightCoef**(w) is the same as $\_q(w)$.

6.3 User guide

We provide two options of loading inputs for running the implemented extending window method. SageMath must be installed as they are executed as shell scripts.

The first one is launching in a shell by typing `sage ewm_inputs.sage`. The parameters are given in the head of the file `ewm_inputs.sage`, see Appendix C for an example.

We describe four parts of the file. The name of the numeration system, minimal polynomial of generator ω, an approximate value of ω, the base β, alphabet $\mathcal{A}$ and input alphabet $\mathcal{B}$ are set in the part INPUTS. Different methods of choice for Phase 1 and 2 can be set. If there are more methods in the lists, then methods for Phase 1 are compared first. Next, each distinct result is processed with each method for Phase 2.

For verification of output, **sanityCheck_conversion**() is launched according to the boolean value in the part SANITY CHECK.

The boolean values in the part SAVING determines which formats of the outputs are saved. All outputs are saved in the folder `./outputs/<name>/`, where `<name>` is the name of the numeration system. General information about the computation can be saved in the LaTeX format, a computed weight function and local digit set conversion in the CSV file format. A log of the whole computation can be saved as a text file.

Figures of the alphabet, input alphabet or weight coefficients set are saved in the PNG format in the folder `./outputs/<name>/img/` according to the boolean values in the part IMAGES. Images of individual steps of both phases of the extending window method can be also saved. For Phase 2, searching for a weight coefficient is plotted for given input digits.

The program prints out all inputs and then it computes a weight function q by calling **findWeightFunction**(). Intermediate weight coefficients sets in each iteration of Phase 1 and the obtained weight coefficients set $\mathcal{Q}$ are printed out. Non-convergence of Phase 2 for combinations given by repetition $b \in \mathcal{B}$ is verified by **check_one_letter_inputs**(). The processed length of window is showed during computing of Phase 2. At the end, the final length of window, elapsed time and info about saved outputs are printed. Results are also saved in the Google spreadsheet ParallelAddition_results in the worksheet `results` and `comparePhase1` if there are more methods for Phase 1.

The second option of loading input parameters is to execute the script `ewm_gspreadsheet.sage` (see Appendix C). Parameters are loaded from the worksheet `inputs` in the Google spreadsheet ParallelAddition_results. The column A marks whether a row should be tested. The columns B–G, i.e., *Name*, *Alphabet*, *Input alphabet*, *Approximate*

value of ring generator omega, *Minimal polynomial omega* and *Base* must be filled. If the column *Input alphabet* is empty, then the input alphabet $\mathcal{A} + \mathcal{A}$ is used. Methods for Phase 1, resp. 2, are given in the header cell C1, resp. C2.

Program runs in the same way as before, but results are saved only to the Google spreadsheet. Notice that column A and cells with the methods may be changed after executing the script, but other cells or order of rows should not be modified. The reason is that the program reads the methods at the beginning and it remembers position of rows to be tested, but the parameters are loaded on the fly.

Chapter 7

Testing and results

We attempted to find a parallel addition algorithm for more than 5000 different numeration systems. Including various methods in Phase 1 and 2, the implementation of the extending window method was launched over 7000 times.

Only expanding bases were considered as it is a necessary condition for convergence of the extending window method. At the same time, convergence of Phase 1 is guaranteed. We take an input alphabet $\mathcal{B}$ equal to $\mathcal{A} + \mathcal{A}$.

Most of the bases were given by polynomial combinations of ω with coefficients in a limited range, where ω was generated as a root of a monic polynomial with bounded integer coefficients. Mostly, ω were quadratic, but cubic ones were also tested. Alphabets for these bases were generated automatically according to Section 4.4.

Besides that, the extending window method was run for selected numeration systems in order to compare different methods in Phase 1 and 2, see Section 7.1.

Processing such a number of inputs is enabled by the developed criteria of non-convergence of Phase 2. The control of $bb\dots b$ inputs often reveals non-convergence very quickly, while an infinite loop in Phase 2 is avoided due to check of directed cycles in a Rauzy graph. This condition seems to be really strong – so far we have only four examples which were interrupted because of lack of memory before non-convergence was revealed or a weight function found. We discuss them later.

Altogether, a parallel addition algorithm was found for about 140 numeration systems with integer and non-integer alphabets. Some of them are listed in Section 7.2.

In Section 7.3, we describe Google spreadsheet ParallelAddition_results which contains all tested inputs.

7.1 Comparing different choices in Phase 1 and 2

As we mentioned, both phases of the extending window method are significantly dependent on the way of extending $\mathcal{Q}_k$ to $\mathcal{Q}k+1$, respectively choice of $\mathcal{Q}_{[w_0,\dots,w_{-k}]}$ from $\mathcal{Q}_{[w_0,\dots,w_{-(k-1)}]}$. Various methods were designed and implemented. They were all tested on the numeration systems which are listed in Table 7.1. Parallel addition algorithms were found manually [15] for some of them, for instance Eisenstein_1–block_complex, Penney_1–block_complex, Penney_2–block_integer or Quadratic+1+4+5_complex2.

The methods 1a, 1b, 1c, 1d and 1e, resp. 2a, 2b, 2c, 2d and 2e, which are described in Chapter 5, were selected as they represent groups of methods which seems to behave

Name	ω	m_ω	β	m_β	conj.	$\#\mathcal{A}$	min.	$\#\mathcal{Q}$ 1a	1b	1c	1d	1e
Eisenstein_1–block_complex	$\frac{1}{2}i\sqrt{3}-\frac{1}{2}$	t^2+t+1	$\omega-1$	$x^2+3\,x+3$	no	7	yes	19	19	19	19	19
Eisenstein_1–block_integer	$\frac{1}{2}i\sqrt{3}-\frac{1}{2}$	t^2+t+1	$\omega-1$	$x^2+3\,x+3$	no	7	yes	139	57	57	57	57
Eisenstein_2–block_complex	$\frac{1}{2}i\sqrt{3}-\frac{1}{2}$	t^2+t+1	$-3\,\omega$	$x^2-3\,x+9$	no	14	no	17	17	17	17	17
Eisenstein_2–block_integer	$\frac{1}{2}i\sqrt{3}-\frac{1}{2}$	t^2+t+1	$-3\,\omega$	$x^2-3\,x+9$	no	16	no	26	26	26	26	26
Penney_1–block_complex	$i-1$	$t^2+2\,t+2$	ω	$x^2+2\,x+2$	no	5	yes	45	45	45	45	45
Penney_1–block_integer	i	t^2+1	$\omega-1$	$x^2+2\,x+2$	no	5	yes	141	49	49	49	49
Penney_2–block_integer	i	t^2+1	$-2\,\omega$	x^2+4	no	9	no	27	27	27	27	27
Quadratic+1+0–2_integer	$\sqrt{2}$	t^2-2	ω	x^2-2	yes	3	yes	9	9	9	9	9
Quadratic+1+0–21_integer	$-\frac{1}{2}\sqrt{21}+\frac{3}{2}$	$t^2-3\,t-3$	$2\,\omega-3$	x^2-21	yes	22	yes	9	9	9	9	9
Quadratic+1+0–3_integer	$\sqrt{3}-1$	$t^2+2\,t-2$	$-\omega-1$	x^2-3	yes	4	yes	9	9	9	9	9
Quadratic+1+0–5_integer	$\frac{1}{2}\sqrt{5}-\frac{1}{2}$	t^2+t-1	$2\,\omega+1$	x^2-5	yes	8	no	9	9	9	9	9
Quadratic+1+2+3_complex	$i\sqrt{2}-1$	$t^2+2\,t+3$	$-\omega-2$	$x^2+2\,x+3$	no	6	yes	27	27	27	27	27
Quadratic+1+3+4_complex	$\frac{1}{2}i\sqrt{7}-\frac{1}{2}$	t^2+t+2	$\omega-1$	$x^2+3\,x+4$	no	8	yes	21	20	19	20	20
Quadratic+1+3+5_complex1	$\frac{1}{2}i\sqrt{11}-\frac{3}{2}$	$t^2+3\,t+5$	ω	$x^2+3\,x+5$	no	9	yes	19	11	11	17	17
Quadratic+1+3+5_complex2	$\frac{1}{2}i\sqrt{11}-\frac{3}{2}$	$t^2+3\,t+5$	ω	$x^2+3\,x+5$	no	9	yes	43	33	33	39	39
Quadratic+1+4+5_complex1	i	t^2+1	$\omega-2$	$x^2+4\,x+5$	no	10	yes	19	17	17	17	17
Quadratic+1+4+5_complex2	i	t^2+1	$\omega-2$	$x^2+4\,x+5$	no	10	yes	17	17	17	17	17
Cubic+1+0+0+2_integer	$2^{\frac{1}{3}}$	t^3-2	$-\omega$	x^3+2	no	3	yes	27	27	27	27	27
Cubic+1+0+0–2_integer	$2^{\frac{1}{3}}$	t^3-2	ω	x^3-2	yes	3	yes	27	27	27	27	27

Table 7.1: Comparing methods in Phase 1

Name	Methods Phase 1	$\#\mathcal{Q}$	2a			2b			2c			2d			2e			Ex.
			bbb	Ph.2	r	*bbb*	Ph.2	r	*bbb*	Ph.2	r	*bbb*	Ph.2	r	*bbb*	Ph.2	r	
Eisenstein_1–block_complex	1a, 1b, 1c, 1d, 1e	19	✓	✓	3	✓	✓	3	✓	✓	3	✓	✓	3	✓	✓	3	
Eisenstein_1–block_integer	1b, 1c, 1d, 1e	57	✗	-	-	✗	-	-	✗	-	-	✗	-	-	✗	-	-	D.1
	1a	139	✗	-	-	✗	-	-	✗	-	-	✗	-	-	✗	-	-	
Eisenstein_2–block_complex	1a, 1b, 1c, 1d, 1e	17	✗	-	-	✗	-	-	✗	-	-	✗	-	-	✗	-	-	D.2
Eisenstein_2–block_integer	1a, 1b, 1c, 1d, 1e	26	✗	-	-	✗	-	-	✗	-	-	✗	-	-	✗	-	-	D.3
Penney_1–block_complex	1a, 1b, 1c, 1d, 1e	45	✓	✓	6	✓	✓	6	✓	✓	6	✓	✓	6	✓	✓	6	
Penney_1–block_integer	1b, 1c, 1d, 1e	49	✗	-	-	✗	-	-	✗	-	-	✗	-	-	✗	-	-	D.4
	1a	141	✗	-	-	✗	-	-	✗	-	-	✗	-	-	✗	-	-	
Penney_2–block_integer	1a, 1b, 1c, 1d, 1e	27	✓	✓	5	✓	✓	5	✓	✓	5	✓	✓	5	✓	✓	5	
Quadratic+1+0–2_integer	1a, 1b, 1c, 1d, 1e	9	✓	✓	5	✓	✓	5	✓	✓	5	✓	✓	4	✓	✓	5	
Quadratic+1+0–3_integer	1a, 1b, 1c, 1d, 1e	9	✓	✓	4	✓	✓	5	✓	✓	5	✓	✓	5	✓	✓	5	
Quadratic+1+0–5_integer	1a, 1b, 1c, 1d, 1e	9	✗	-	-	✓	✓	3	✓	✓	2	✓	✓	2	✓	✓	3	D.5
Quadratic+1+2+3_complex	1a, 1b, 1c, 1d, 1e	27	✓	✗	-	✓	✓	7	✓	✗	-	✓	✗	-	✓	✓	7	D.6
Quadratic+1+3+4_complex	1b	20	✓	✓	7	✓	✓	7	✓	✗	-	✗	-	-	✓	✓	7	D.7
	1c	19	✓	✗	-	✓	✓	7	✓	✗	-	✗	-	-	✓	✓	7	
	1d, 1e	20	✓	✗	-	✓	✓	7	✓	✗	-	✗	-	-	✓	✓	7	
	1a	21	✓	✓	7	✓	✓	7	✓	✗	-	✗	-	-	✓	✓	7	
Quadratic+1+3+5_complex1	1a	19	✗	-	-	✗	-	-	✗	-	-	✗	-	-	✗	-	-	D.8
	1b, 1c	11	✗	-	-	✓	✗	-	✗	-	-	✗	-	-	✓	✗	-	
	1d, 1e	17	✗	-	-	✗	-	-	✗	-	-	✗	-	-	✗	-	-	
Quadratic+1+3+5_complex2	1b, 1c	33	✗	-	-	✓	✗	-	✗	-	-	✗	-	-	✓	✗	-	D.9
	1d, 1e	39	✗	-	-	✓	✗	-	✓	✗	-	✗	-	-	✓	✗	-	
	1a	43	✗	-	-	✓	✗	-	✓	✗	-	✗	-	-	✓	✗	-	
Quadratic+1+4+5_complex1	Lemma 4.3	69	✓	✗	-	✗	-	-	✗	-	-	✗	-	-	✓	✓	6	D.10
	1a	19	✗	-	-	✗	-	-	✗	-	-	✗	-	-	✗	-	-	
	1b, 1c, 1d, 1e	17	✗	-	-	✗	-	-	✗	-	-	✗	-	-	✗	-	-	
Quadratic+1+4+5_complex2	1a, 1b, 1c, 1d, 1e	17	✓	✓	3	✓	✓	3	✓	✓	3	✓	✓	3	✓	✓	3	
Cubic+1+0+0+2_integer	1a, 1b, 1c, 1d, 1e	27	✓	✗	-	✓	✗	-	✓	✗	-	✓	✓	6	✓	✗	-	D.11
Cubic+1+0+0–2_integer	1a, 1b, 1c, 1d, 1e	27	✓	✗	-	✓	✗	-	✓	✗	-	✓	✓	6	✓	✗	-	D.12

Table 7.2: Comparing methods in Phase 2

similarly from our experience with many preliminary tests. Moreover, if a parallel addition algorithm was found for a numeration system in a preliminary test, then at least one successful method is contained in the selection.

Name	$\mathcal{A}$
Eisenstein_1–block_complex	$\{0, 1, -1, \omega, -\omega, -\omega - 1, \omega + 1\}$
Eisenstein_1–block_integer	$\{-3, -2, -1, 0, 1, 2, 3\}$
Eisenstein_2–block_complex	$\{0, 1, \omega, \omega + 1, 2\,\omega, 2\,\omega - 1, \omega - 1, -1, -2,$ $-\omega, -\omega - 1, -\omega - 2, -2\,\omega, -2\,\omega - 1\}$
Eisenstein_2–block_integer	$\{-\omega + 3, -\omega + 2, -\omega + 1, -\omega, 2, 1, 0, -1, \omega + 1,$ $\omega, \omega - 1, \omega - 2, 2\,\omega, 2\,\omega - 1, 2\,\omega - 2, 2\,\omega - 3\}$
Penney_1–block_complex	$\{0, \omega + 1, -\omega - 1, 1, -1\}$
Penney_1–block_integer	$\{-2, -1, 0, 1, 2\}$
Penney_2–block_integer	$\{0, 1, -1, \omega, -\omega, \omega - 1, -\omega + 1, \omega - 2, -\omega + 2\}$
Quadratic+1+0–2_integer	$\{-1, 0, 1\}$
Quadratic+1+0–21_integer	$\{-10, -9, -8, \ldots, -1, 0, 1, \ldots, 9, 10, 11\}$
Quadratic+1+0–3_integer	$\{-1, 0, 1, 2\}$
Quadratic+1+0–5_integer	$\{-3, -2, -1, 0, 1, 2, 3, 4\}$
Quadratic+1+2+3_complex	$\{0, \omega + 1, -\omega - 1, 1, -1, \omega\}$
Quadratic+1+3+4_complex	$\{0, \omega + 1, -\omega - 1, 1, -1, \omega, -\omega, \omega + 2\}$
Quadratic+1+3+5_complex1	$\{0, 1, -1, \omega + 1, -\omega - 1, \omega + 2, -\omega - 2, \omega + 3, -\omega - 3\}$
Quadratic+1+3+5_complex2	$\{0, 1, -1, \omega + 1, -\omega - 1, \omega + 2, -\omega - 2, 2\omega + 2, -2\omega - 2\}$
Quadratic+1+4+5_complex1	$\{\omega + 2, \omega + 1, \omega, 1, 0, -1, -\omega + 1, -\omega, -\omega - 1, \omega - 1\}$
Quadratic+1+4+5_complex2	$\{\omega + 2, \omega + 1, \omega, 1, 0, -1, -\omega + 1, -\omega, -\omega - 1, 2\}$
Cubic+1+0+0+2_integer	$\{-1, 0, 1\}$
Cubic+1+0+0–2_integer	$\{-1, 0, 1\}$

Table 7.3: Alphabets for numeration systems in Table 7.1 and 7.2

Let us explain Tables 7.1, 7.2 and 7.3. Each row in Table 7.1 represents one numeration system with a base $\beta \in \mathbb{Z}[\omega]$ for a given algebraic integer ω. The column *conj.* signifies whether the base β has a real conjugate greater than 1. The sizes of alphabets are listed and the column *min.* says if the alphabets, which are listed in Table 7.3, are minimal in the sense of the lower bound given by Theorem 4.12. We remark that the size of an alphabet is compared with the bound regardless to working in $\mathbb{Z}[\beta]$ or $\mathbb{Z}[\omega]$. In the last columns, there are the sizes of weight coefficients sets $\mathcal{Q}$ which were found with various methods of Phase 1.

The results of Phase 2 for the selected numeration systems are shown in Table 7.2. More rows for one numeration system correspond to distinct weight coefficients sets from Phase 1. The column $bb\ldots b$ says whether check of $bb\ldots b$ inputs was successful. If a weight function is found, it is denoted by ✓ in the column *Ph.2* and the length of window is in the column r. Symbol ✗ in the column *Ph.2* means that a cycle in a Rauzy graph was found, i.e., Phase 2 does not converge. Reasons for non-convergence (digits b or a cycle in Rauzy graph) can be recognized in the example given by the last column in Appendix D.

We remark that Lemma 4.3 instead of some method of Phase 1 means that the set given by this lemma was used as $\mathcal{Q}$ instead of a computed one. We see that the

ω	β	m_β	conj.	$\#\mathcal{A}$	min.	$\#\mathcal{Q}$	Ph. 2	r
$\frac{1}{2}i\sqrt{11}+\frac{1}{2}$	$-i\sqrt{11}-4$	$x^2+8\,x+27$	no	36	yes	13	✓	7
$\frac{1}{2}i\sqrt{11}-\frac{1}{2}$	$i\sqrt{11}-4$	$x^2+8\,x+27$	no	36	yes	13	✓	5
$\frac{1}{2}i\sqrt{11}-\frac{1}{2}$	$\frac{1}{2}i\sqrt{11}-\frac{7}{2}$	$x^2+7\,x+15$	no	23	yes	13	✓	5
$\frac{1}{2}i\sqrt{7}-\frac{1}{2}$	$\frac{1}{2}i\sqrt{7}-\frac{1}{2}$	x^2+x+2	no	4	yes	29	✓	8
$\frac{1}{2}i\sqrt{7}-\frac{1}{2}$	$i\sqrt{7}-4$	$x^2+8\,x+23$	no	32	yes	10	✓	5
$\frac{1}{2}i\sqrt{3}+\frac{3}{2}$	$-\frac{3}{2}i\sqrt{3}-\frac{15}{2}$	$x^2+15\,x+63$	no	79	yes	13	✓	3
$\frac{1}{2}i\sqrt{3}+\frac{1}{2}$	$-\frac{3}{2}i\sqrt{3}-\frac{9}{2}$	$x^2+9\,x+27$	no	37	yes	13	✓	2
$\frac{1}{2}i\sqrt{3}+\frac{1}{2}$	$-i\sqrt{3}-1$	$x^2+2\,x+4$	no	8	no	23	✓	5
$i\sqrt{2}-1$	$2i\sqrt{2}$	x^2+8	no	11	no	13	✓	5
$i\sqrt{2}-1$	$i\sqrt{2}-3$	$x^2+6\,x+11$	no	18	yes	15	✓	4
$i+1$	$-2i-4$	$x^2+8\,x+20$	no	29	yes	11	✓	2
i	$-3i-3$	$x^2+6\,x+18$	no	25	yes	15	✓	4
$-\frac{1}{2}\sqrt{5}+\frac{3}{2}$	$\frac{3}{2}\sqrt{5}-\frac{15}{2}$	$x^2+15\,x+45$	no	61	yes	15	✓	3
$-\frac{1}{2}\sqrt{5}+\frac{3}{2}$	$\sqrt{5}-5$	$x^2+10\,x+20$	no	31	yes	11	✓	3
$\frac{1}{2}\sqrt{17}-\frac{3}{2}$	$\frac{1}{2}\sqrt{17}-\frac{9}{2}$	$x^2+9\,x+16$	no	26	yes	17	✓	5

Table 7.4: Quadratic bases with a non-integer alphabet (using methods 1d and 2b)

smaller weight coefficients set $\mathcal{Q}$ does not mean automatically better (Quadratic+1+4+5 _complex1 or Quadratic+1+3+4_complex). An observation for many numeration systems is that if the extending window method is successful, then weight coefficients sets produced by different methods are similar. But it is not a rule.

Unfortunately, there is also no best method for Phase 2. For example, method 2e is successful for all selected quadratic bases, but it fails for the cubic ones. Moreover, the length of window r is not always minimal (Quadratic+1+0–5_integer). On the other hand, the method 2d is the only one which finds a weight function for cubic bases, but it fails in many other cases. The method 2b seems to be often successful, but not with the optimal length of window.

An interesting example is Quadratic+1+4+5_complex1. Only one element of the alphabet is different, comparing with Quadratic+1+4+5_complex2. Whereas a weight function is easily found for Quadratic+1+4+5_complex2 by all methods, the only successful combination of methods for Quadratic+1+4+5_complex1 is the weight coefficients set given by Lemma 4.3 and method 2e. We remark that many of elements of such $\mathcal{Q}$ are not used as outputs of the obtained weight function. Notice that there are only few inputs digits b such that Quadratic+1+4+5_complex1 fails in the check of $bb\ldots b$ inputs in Example D.10.

7.2 Examples of results

We divide numeration systems with quadratic base according to the type of alphabet. Numeration systems with non-integer alphabets are in Table 7.4, while numeration systems with integer alphabets are in Table 7.5. The structure of tables is the same as in the previous section. The methods 1d for Phase 1 and 2b for Phase 2 were used for the listed numeration systems.

Firstly, we focus on non-integer alphabets. Since many alphabets are large, Table 7.4 outlines only their sizes. The whole alphabets can be found in the spreadsheet ParallelAddition_results, see Section 7.3.

Note that none of the bases has a real conjugate greater than 1. Observe that the linear and absolute coefficients of the minimal polynomial m_β are positive if the corresponding alphabet attains the lower bound given by Theorem 4.12. A possible explanation is that if a linear coefficient was negative, then $\#\mathcal{A} = m_\beta(0) \geq m_\beta(1)$, i.e., the numeration system would not be redundant enough.

Parallel addition algorithms can be found for quite large alphabets if the final length of window is small or many combinations of input digits are saved for a shorter window than the final one. Examples D.19 and D.20 in Appendix D are the only inputs with quadratic base which were interrupted because of the exponential growth of memory requirement.

ω	β	m_β	conj.	$\#\mathcal{A}$	min.	$\#\mathcal{Q}$	Ph. 2	r	Ex.
$\frac{1}{2}i\sqrt{11}+\frac{1}{2}$	$-i\sqrt{11}$	x^2+11	no	13	no	9	✓	2	
$\frac{1}{2}i\sqrt{11}+\frac{1}{2}$	$-i\sqrt{11}$	x^2+11	no	12	yes	9	✓	4	D.13
$\frac{1}{2}i\sqrt{7}-\frac{1}{2}$	$-i\sqrt{7}$	x^2+7	no	9	no	9	✓	2	
$\frac{1}{2}i\sqrt{7}-\frac{1}{2}$	$-i\sqrt{7}$	x^2+7	no	8	yes	9	✓	4	
$\frac{1}{2}i\sqrt{3}+\frac{1}{2}$	$-\frac{3}{2}i\sqrt{3}+\frac{1}{2}$	x^2-x+7	no	11	no	9	✓	2	D.14
$i\sqrt{3}$	$i\sqrt{3}$	x^2+3	no	4	yes	9	✓	4	
$i\sqrt{2}$	$i\sqrt{2}$	x^2+2	no	3	yes	9	✓	4	
$\sqrt{2}$	$-\sqrt{2}$	x^2-2	yes	3	yes	9	✓	5	
$\sqrt{3}-1$	$-\sqrt{3}$	x^2-3	yes	4	yes	9	✓	5	
$\frac{1}{2}\sqrt{5}+\frac{1}{2}$	$-\sqrt{5}$	x^2-5	yes	6	yes	9	✓	4	D.15
$-\sqrt{5}+1$	$-\sqrt{5}$	x^2-5	yes	6	yes	9	✓	4	D.16
$-\sqrt{6}+1$	$-\sqrt{6}$	x^2-6	yes	7	yes	9	✓	4	
$\sqrt{6}-1$	$\sqrt{6}$	x^2-6	yes	7	yes	9	✓	4	
$-\sqrt{7}+2$	$\sqrt{7}$	x^2-7	yes	8	yes	9	✓	4	
$\frac{1}{2}\sqrt{13}+\frac{1}{2}$	$-\sqrt{13}$	x^2-13	yes	15	no	9	✓	2	D.17
$\frac{1}{2}\sqrt{13}+\frac{1}{2}$	$-\sqrt{13}$	x^2-13	yes	14	yes	9	✓	4	
$-\frac{1}{2}\sqrt{17}+\frac{3}{2}$	$-\sqrt{17}$	x^2-17	yes	18	yes	9	✓	4	
$-\frac{1}{2}\sqrt{21}+\frac{3}{2}$	$-\sqrt{21}$	x^2-21	yes	22	yes	9	✓	4	D.18

Table 7.5: Quadratic bases with an integer alphabet (using methods 1d and 2b)

Table 7.5 shows some examples of numeration systems with integer alphabets for which the extending window method was successful. The alphabets are of the form $\{-a, -a+1, \ldots, -1, 0, 1, \ldots, a-1, a\}$ or $\{-a+1, \ldots, -1, 0, 1, \ldots, a-1, a\}$, depending on the parity of their size, where a is a positive integer.

As we mentioned in Section 4.3, congruences behave differently in $\mathbb{Z}[\beta]$ and $\mathbb{Z}[\omega]$. The alphabets divided into congruence classes modulo β and $\beta - 1$ in $\mathbb{Z}[\omega]$ for some of the numeration systems can be found in Examples D.13 – D.18 in Appendix D. However, we recall that if $\beta = \pm\omega + c$ for some $c \in \mathbb{Z}$, then $\mathbb{Z}[\beta] = \mathbb{Z}[\omega]$.

We made the following observations:

1. The congruence classes modulo $\beta - 1$ in Examples D.15 and D.16 are different,

since we work in $\mathbb{Z}\left[\frac{1}{2}\sqrt{5}+\frac{1}{2}\right]$ for the former and in $\mathbb{Z}\left[-\sqrt{5}+1\right]$ for the latter one.

2. If $\mathbb{Z}[\beta] \neq \mathbb{Z}[\omega]$, then an alphabet may not contain representatives of all congruence classes modulo $\beta - 1$ in $\mathbb{Z}[\omega]$, see Examples D.13, D.15, D.17 and D.18.

3. The division of the alphabet into congruence in Example D.14 is the same modulo β and $\beta - 1$.

4. If an alphabet is minimal, there is only one congruence class modulo β which has two representatives in the alphabet, i.e, it is only a little redundant.

5. If a base has a real conjugate greater than one, then the alphabet contains one more element congruent to the smallest and greatest digit modulo $\beta - 1$ as it is necessary according to Lemma 4.9.

6. Examples D.13 and D.17 show that an alphabet with one more digit than the minimal number of digits has shorter length of window.

7. The weight coefficients set $\mathcal{Q}$ has size 9 for all successful numeration systems with integer alphabets.

We remark that we have no example of a base for which the extending window method converges with integer and also non-integer alphabet. We are aware of the fact that Theorem 1.1 provides a parallel addition algorithm with an integer alphabet for an expanding base, but a different rewriting rule than $x - \beta$ is used.

We tested also about 230 cubic bases, but the only successful inputs are in Table 7.1. There are only two more examples (D.21 and D.22) when the check of $bb\ldots b$ inputs was successful. Unfortunately, the computations were interrupted before non-convergence by a Rauzy graph was revealed or a weight function found. Both were interrupted having the length of window 3 processed, but the final length of window would be at least 5, resp., 6 according to the lenght of window for $bb, \ldots, b$ inputs.

7.3 Google spreadsheet ParallelAddition_results

All results can be found in the spreadsheet ParallelAddition_results. Its structure is the following – the worksheet `results` is used for automatically saved results. Some results were copied to `results_archive` to reduce the number of rows in `results`. The meaning of each column is obvious from the heading line.

The worksheet `inputs` serves for loading inputs by the script `ewm_gspreadsheet.sage`. The numeration systems for which a weight function was successfully found are in the worksheet `successful`. We remark that they are listed with a single combination of methods for Phase 1 and 2. Nevertheless, all tested variants remain in `results` and `results_archive`.

Results for selected numeration systems which were tested with various methods for Phase 1 and 2 are sorted in the worksheet `comparePhase2`. Note that if the result of Phase 1 is the same for more methods, then only one of them is used for testing of methods for Phase 2. This fact can be found in the columns *Groups of methods with the same result* and *Sizes of weight coefficients sets for groups* in the worksheet `comparePhase1`. For instance, the values `[['1a'], ['1b', '1c', '1d', '1e']]` and `[19, 17]` mean that Phase 1 with method 1a produces a weight coefficients set $\mathcal{Q}$ of

size 19, while methods 1b, 1c, 1d and 1e outputs the same weight coefficients set $\mathcal{Q}$ of size 17.

Very useful property of storing data in a worksheet is easy sorting and filtering.

The version of ParallelAddition_results which is on the attached DVD was downloaded on May 4, 2016.

Summary

The main goal of this thesis was to improve the extending window method with the rewriting rule $x - \beta$ which attempts to construct a parallel addition algorithm for a given base β and alphabet $\mathcal{A}$ and discuss its convergence.

We recalled necessary definitions and notation in the scope of numeration systems and parallel addition. The concept of parallel addition and extending window method was explained. We proposed various methods for both phases of the method. Some of them involve so-called β-norm, which was constructed by using the companion matrix of the minimal polynomial m_β of β.

We gave a sufficient condition for convergence of Phase 1: the base β must be expanding, i.e., all its conjugates are greater than one in modulus. We showed that it is also a necessary condition of convergence of the whole extending window method.

The check of convergence of Phase 2 for $bb\ldots b$ inputs was reviewed. Next, we introduced the notion of stable Phase 2 and Rauzy graph. We proved Theorem 4.7 which may reveal non-convergence of Phase 2 by searching for a cycle in a Rauzy graph.

We indicated that there is the same lower bound on the size of an alphabet from $\mathbb{Z}[\beta]$ as the size of an integer one. The necessary conditions on the alphabet which allow parallel addition were summarized and the way of generating such an alphabet was sketched.

Algorithms based on the obtained theoretical results were designed. Algorithm 11 executes Phase 2 with the control of non-convergence. The extending window method was implemented with all proposed algorithms in SageMath.

The shell interfaces are provided for running the implementation. All results are automatically saved to Google spreadsheet which enables easy sorting.

We tested large number of numeration systems with different methods for both phases. The controls of non-convergence reveal failure of Phase 2 very efficiently. Thus, many inputs could be processed. We compared various methods for Phase 1 and 2 for selected numeration systems including those for which a parallel addition algorithm was found manually.

To conclude, there is about 140 numeration systems for which the implemented extending window method successfully found a parallel addition algorithm. The numeration systems have integer or non-integer alphabet, often of the minimal size.

The tasks for future work are the following:

- The question of sufficient condition of convergence of Phase 2 remains open.
- Generalization and implementation of the extending window method with an arbitrary rewriting rule.
- Testing on a computer with higher performance or faster implementation.
- Development of theoretical analysis of Phase 2 in order to improve convergence and speed.

References

[1] S. Akiyama, J.M. Thuswaldner, and T. Zaïmi, *Comments on the height reducing property II*, Indag. Math. **26** (2015), 28–39.

[2] S. Akiyama and T. Zaïmi, *Comments on the height reducing property*, Cent. Eur. J. Math. **11** (2013), 1616–1627.

[3] A. Avizienis, *Signed-digit number representations for fast parallel arithmetic*, IEEE Trans. Comput. **10** (1961), 389–400.

[4] C.Y. Chow and J.E. Robertson, *Logical design of a redundant binary adder*, Proc. IEEE 4th Symp. on Comp. Arith. (1978), 109–115.

[5] C. Frougny, P. Heller, E. Pelantová, and M. Svobodová, *k-block parallel addition versus 1-block parallel addition in non-standard numeration systems*, Theoret. Comput. Sci. **543** (2014), 52–67.

[6] C. Frougny, E. Pelantová, and M. Svobodová, *Parallel addition in non-standard numeration systems*, Theoret. Comput. Sci. **412** (2011), 5714–5727.

[7] C. Frougny, E. Pelantová, and M. Svobodová, *Minimal digit sets for parallel addition in non-standard numeration systems*, J. Integer Seq. **16** (2013), 36.

[8] *Algebraic Number Theory – summary of notes*, `http://empslocal.ex.ac.uk/people/staff/rjchapma/notes/ant2.pdf`, Accessed: 2016-4-29.

[9] *Google Spreadsheets Python API*, `https://github.com/burnash/gspread`, Accessed: 2016-4-24.

[10] R. A. Horn and C. R. Johnson, *Matrix analysis*, Cambridge University Press, 1990.

[11] P. Kornerup, *Necessary and sufficient conditions for parallel, constant time conversion and addition*, Proc. 14th IEEE Symp. on Comp. Arith. (1999), 152–155.

[12] J. Legerský, *Construction of algorithms for parallel addition*, Research thesis, Czech Technical University in Prague, FNSPE, Czech Republic, 2015.

[13] A. M. Nielsen and P. Kornerup, *Redundant radix representations of rings*, IEEE Trans. Comput. **48** (1999), 1153–1165.

[14] *SageMath reference manual*, `http://doc.sagemath.org/html/en/reference/index.html`, Accessed: 2016-4-24.

[15] M. Svobodová, Private communication, 2014–2015.

[16] C. L. Wey and C. P. Wang, *Design of a fast radix-4 SRT divider and its VLSI implementation*, IEE Proc. - Comp. and Digit. Tech. **146** (1999), 205–210.

Appendices

A Illustration of Phase 1

Figures 1 – 5 illustrates first and last iterations of the construction of the weight coefficients set $\mathcal{Q}$ for the Eisenstein base $\beta = -\frac{3}{2} + \frac{\imath\sqrt{3}}{2}$ with the complex alphabet $\mathcal{A} = \{0, 1, -1, \omega, -\omega, -\omega - 1, \omega + 1\}$ and input alphabet $\mathcal{B} = \mathcal{A} + \mathcal{A}$. The second last iteration is skipped.

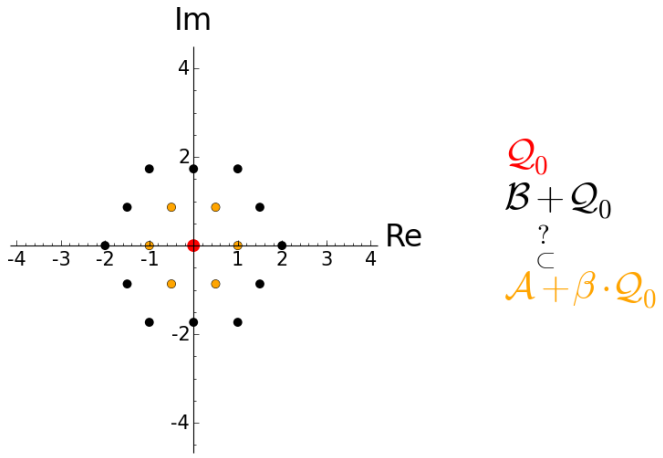


Figure 1: The set $\mathcal{Q}_0$ does not cover the set $\mathcal{B} + \mathcal{Q}_0$, i.e., the set $\mathcal{A}+\beta\cdot\mathcal{Q}_0$ is not superset of $\mathcal{B}+\mathcal{Q}_0$.

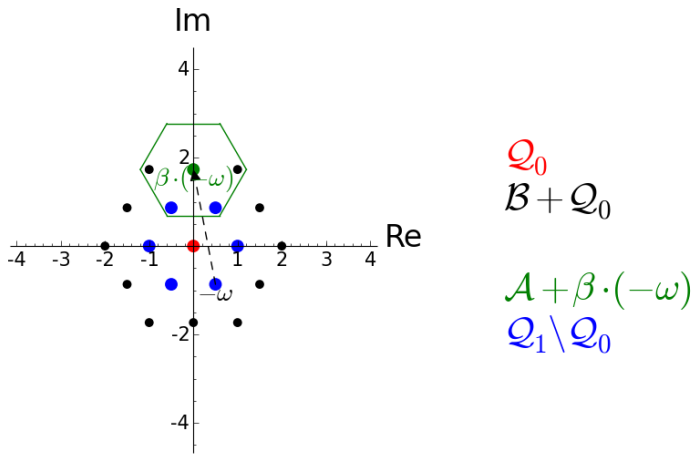
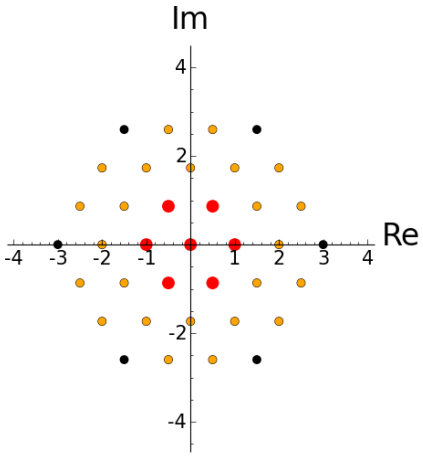
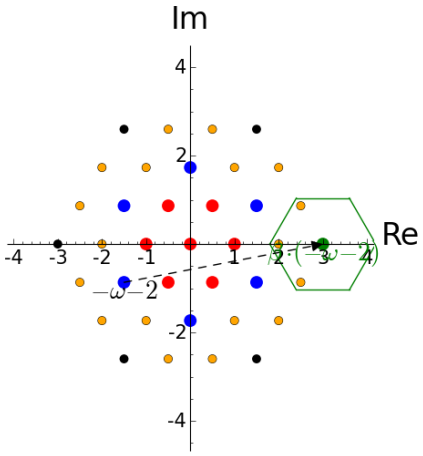


Figure 2: The set $\mathcal{Q}_0$ is extended to $\mathcal{Q}_1$ to cover all elements of $\mathcal{B} + \mathcal{Q}_0$.



$\mathcal{Q}_1$
$\mathcal{B}+\mathcal{Q}_1$
$\overset{?}{\subset}$
$\mathcal{A}+\beta\cdot\mathcal{Q}_1$

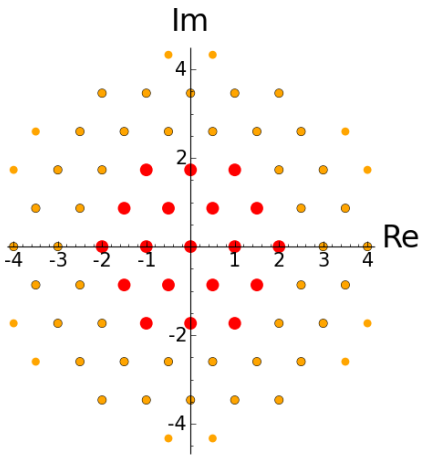
Figure 3: The set $\mathcal{Q}_1$ does not cover the set $\mathcal{B}+\mathcal{Q}_1$, i.e., the set $\mathcal{A}+\beta\cdot\mathcal{Q}_1$ is not superset of $\mathcal{B}+\mathcal{Q}_1$.



$\mathcal{Q}_1$
$\mathcal{B}+\mathcal{Q}_1$

$\mathcal{A}+\beta\cdot(-\omega-2)$
$\mathcal{Q}_2\backslash\mathcal{Q}_1$

Figure 4: The set $\mathcal{Q}_1$ is extended to $\mathcal{Q}_2$ to cover all elements of $\mathcal{B}+\mathcal{Q}_1$.

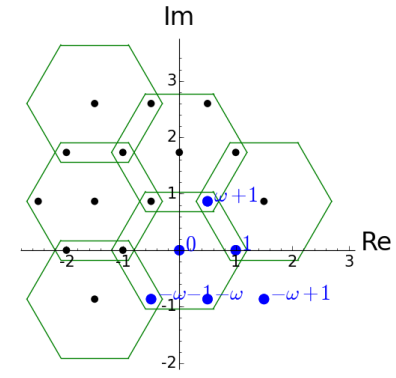


$\mathcal{Q}_3$
$\mathcal{B}+\mathcal{Q}_3$
$\overset{?}{\subset}$
$\mathcal{A}+\beta\cdot\mathcal{Q}_3$

Figure 5: In the last iteration, the set $\mathcal{Q}_3$ covers the set $\mathcal{B}+\mathcal{Q}_3$, i.e., the set $\mathcal{A}+\beta\cdot\mathcal{Q}_3$ is superset of $\mathcal{B}+\mathcal{Q}_2$. The weight coefficients set $\mathcal{Q}$ equals $\mathcal{Q}_3$.

B Illustration of Phase 2

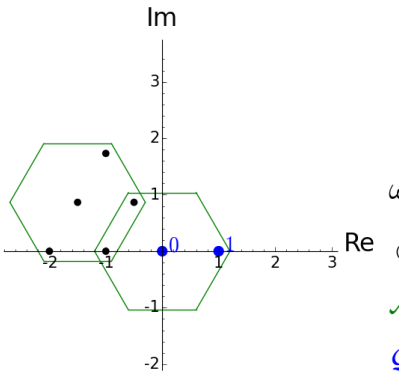
The construction of set $\mathcal{Q}_{[\omega,1,2]}$ for the Eisenstein base $\beta = -\frac{3}{2} + \frac{\imath\sqrt{3}}{2}$ with the complex alphabet $\mathcal{A} = \{0, 1, -1, \omega, -\omega, -\omega-1, \omega+1\}$ and input alphabet $\mathcal{B} = \mathcal{A}+\mathcal{A}$ is illustrated on Figures 6 – 8.



$$\omega + \mathcal{Q} \subset \mathcal{A} + \beta \cdot \mathcal{Q}_{[\omega]}$$

$\mathcal{Q}_{[\omega]}$

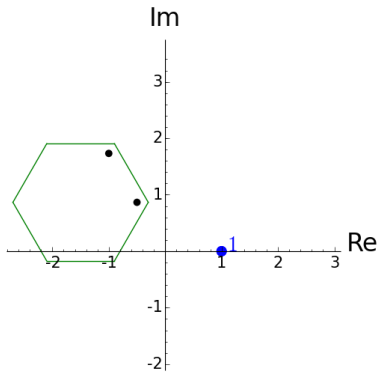
Figure 6: The elements of $\omega + \mathcal{Q}$ are covered by the set $\mathcal{Q}_{[\omega]} \subset \mathcal{Q}$.



$$\omega + \mathcal{Q}_{[1]} \subset \mathcal{A} + \beta \cdot \mathcal{Q}_{[\omega,1]}$$

$\mathcal{Q}_{[\omega,1]}$

Figure 7: The elements of $\omega + \mathcal{Q}_{[1]}$ are covered by the set $\mathcal{Q}_{[\omega,1]} \subset \mathcal{Q}_{[\omega]}$.



$$\omega + \mathcal{Q}_{[1,2]} \subset \mathcal{A} + \beta \cdot \mathcal{Q}_{[\omega,1,2]}$$

$\mathcal{Q}_{[\omega,1,2]}$

Figure 8: The elements of $\omega + \mathcal{Q}_{[1]}$ are covered by the set $\mathcal{Q}_{[\omega,1,2]} \subset \mathcal{Q}_{[\omega,1]}$ which has only one element. This element is the output of the weight function $q(\omega, 1, 2)$.

C Interfaces

File `ewm_inputs.sage`:

```
#------------INPUTS---------------------
#Name of the numeration system
name = 'Eisenstein_1-block_complex'
#Minimal polynomial of ring generator (use variable x)
minPol ='x^2 + x + 1'
#Embedding (the closest root of the minimal polynomial to
   this value is taken as the ring generator)
omegaCC= -0.5 + 0.8*I
#Alphabet (use 'omega' as ring generator)
alphabet = '[0, 1, -1, omega, -omega, -omega - 1, omega + 1]
   '
#Input alphabet (if empty, A + A is used)
inputAlphabet = ''
#Base (use 'omega' as ring generator)
base ='omega - 1'

#------------EWM SETTING----------------
methods_phase1=['1a']     #methods in the list are used. If
   empty, default method is used.
methods_phase2=['2c']     #methods in the list are used. If
   empty, default method is used.
#Cartesian product of lists methods_phase1 and
   methods_phase2 is computed

#------------SANITY CHECK---------------
sanityCheck=False         #run sanity check

#------------SAVING---------------------
info=True                 #save general info to .tex file
WFcsv=False               #save weight function to .csv file
localConversionCsv=False #save local conversion to .csv file
saveLog=True              #save log file

#------------IMAGES---------------------
alphabet_img=False       #save image of alphabet and input
   alphabet
phase1_images=False      #save step-by-step images of Phase 1
weightCoefSet_img=False #save image of the weight coeff. set
phase2_images=False      #save step-by-step images of Phase 2
    for input:
phase2_input='(omega,1,omega,1,omega,1,omega,1)'

#---RUN EXTENDING WINDOW METHOD--------
load('ewm.sage')
```

File `ewm_gspreadsheet.sage`:

```
# This script loads inputs from the list 'inputs' of google
   spreadsheet https://docs.google.com/spreadsheets/d/1
   TnhrHdefHfHa0WSeVs4q6XVj3epjPlPlnoekE0E1xeM/edit#gid
   =209657865
# Methods for Phase 1, resp. 2, are given by a list in the
   cell C1, resp. C2
# Values in the columns 'Name', 'Alphabet', 'Input alphabet
   ', 'Ring generator', 'Minimal polynomial of generator
   omega' and 'Base' must be filled for the tested rows

compareWith=['this', 'that'] # if some of these values is
   found in column A, the corresponding row will be tested
general_note='my own note'
onlyComparePhase1=False      #if True, then only methods for
    Phase 1 are compared.

load('run_gspreadsheet.sage')
# !Do not change order of rows in the  list 'inputs' when
   processing!
```

D Tested examples

Unsuccessful examples comparing different methods

The reasons of failure of Phase 2 for numeration systems in Section 7.1 can be found here. See Tables 7.1, 7.2 and 7.3 for parameters of the numeration systems.

Example D.1. Eisenstein_1–block_integer

Phase 1 (methods 1b,1c,1d,1e):

2a – Check of $b, b, \ldots, b$ inputs fails for $b \in \{2, 3, 5, 6, -5, -4, -3\}$.

2b – Check of $b, b, \ldots, b$ inputs fails for $b \in \{2, 3, 6, -6, -4, -3\}$.

2c – Check of $b, b, \ldots, b$ inputs fails for $b \in \{0, 1, 3, 4, 6, -6, -4, -3, -1\}$.

2d – Check of $b, b, \ldots, b$ inputs fails for $b \in \{3, 4, 6, -6, -4, -3\}$.

2e – Check of $b, b, \ldots, b$ inputs fails for $b \in \{2, 3, 6, -2, -6, -4, -3\}$.

Phase 1 (methods 1a):

2a – Check of $b, b, \ldots, b$ inputs fails for $b \in \{0, 2, 4, 5, -2, -5, -4\}$.

2b – Check of $b, b, \ldots, b$ inputs fails for $b \in \{0, 2, 4, 5, -2, -5, -4\}$.

2c – Check of $b, b, \ldots, b$ inputs fails for $b \in \{0, 2, 3, 5, 6, -2, -6, -5, -3\}$.

2d – Check of $b, b, \ldots, b$ inputs fails for $b \in \{0, 3, 4, 5, 6, -6, -5, -4, -3\}$.

2e – Check of $b, b, \ldots, b$ inputs fails for $b \in \{0, 2, 4, 5, -2, -5, -4\}$.

Example D.2. Eisenstein_2–block_complex
Phase 1 (methods 1a,1b,1c,1d,1e):

2a – Check of $b, b, \ldots, b$ inputs fails for $b \in \{2\omega - 1, \omega + 1, -2\omega, -4, -\omega - 2\}$.

2b – Check of $b, b, \ldots, b$ inputs fails for $b \in \{2\omega - 1, \omega + 1, -2\omega, -4, -\omega - 2\}$.

2c – Check of $b, b, \ldots, b$ inputs fails for $b \in \{2\omega - 1, \omega + 1, -2\omega, -4, -\omega - 2\}$.

2d – Check of $b, b, \ldots, b$ inputs fails for $b \in \{2\omega - 1, \omega + 1, -2\omega, -4, -\omega - 2\}$.

2e – Check of $b, b, \ldots, b$ inputs fails for $b \in \{2\omega - 1, \omega + 1, -2\omega, -4, -\omega - 2\}$.

Example D.3. Eisenstein_2–block_integer
Phase 1 (methods 1a,1b,1c,1d,1e):

2a – Check of $b, b, \ldots, b$ inputs fails for $b \in \{0, 1, 2, 2\omega - 4, \omega - 2, 4\omega, 3\omega - 5, \omega - 1, -\omega + 3, -2\omega + 5, 2\omega - 3\}$.

2b – Check of $b, b, \ldots, b$ inputs fails for $b \in \{0, 1, 2, 2\omega - 4, 2\omega - 2, \omega - 2, \omega, 4\omega, 3\omega - 5, \omega - 1, -\omega + 3, -2\omega + 5, 2\omega - 3\}$.

2c – Check of $b, b, \ldots, b$ inputs fails for $b \in \{0, 1, 2, 2\omega - 4, \omega - 2, \omega, 4\omega, 3\omega - 5, \omega - 1, -\omega + 3, -2\omega + 5, 2\omega - 3\}$.

2d – Check of $b, b, \ldots, b$ inputs fails for $b \in \{1, 2, 2\omega - 4, \omega - 2, 4\omega, 3\omega - 5, \omega - 1, -\omega + 3, -2\omega + 5, 2\omega - 3\}$.

2e – Check of $b, b, \ldots, b$ inputs fails for $b \in \{0, 1, 2, 2\omega - 4, 2\omega - 2, \omega - 2, \omega, 4\omega, 3\omega - 5, \omega - 1, -\omega + 3, -2\omega + 5, 2\omega - 3\}$.

Example D.4. Penney_1–block_integer
Phase 1 (methods 1b,1c,1d,1e):

2a – Check of $b, b, \ldots, b$ inputs fails for $b \in \{0, 3, 4, -4, -3\}$.

2b – Check of $b, b, \ldots, b$ inputs fails for $b \in \{0, 3, -3\}$.

2c – Check of $b, b, \ldots, b$ inputs fails for $b \in \{0, 3, 4, -4, -3, -2\}$.

2d – Check of $b, b, \ldots, b$ inputs fails for $b \in \{0, 3, 4, -4, -3, -2\}$.

2e – Check of $b, b, \ldots, b$ inputs fails for $b \in \{0, 3, -3\}$.

Phase 1 (methods 1a):

2a – Check of $b, b, \ldots, b$ inputs fails for $b \in \{0, 1, 2, -1, -2\}$.

2b – Check of $b, b, \ldots, b$ inputs fails for $b \in \{0, 1, 2, -1, -2\}$.

2c – Check of $b, b, \ldots, b$ inputs fails for $b \in \{0, 1, 2, 3, 4, -4, -3, -2\}$.

2d – Check of $b, b, \ldots, b$ inputs fails for $b \in \{0, 1, 2, 3, 4, -4, -3, -2\}$.

2e – Check of $b, b, \ldots, b$ inputs fails for $b \in \{2, -2\}$.

Example D.5. Quadratic+1+0–5_integer
Phase 1 (methods 1a,1b,1c,1d,1e):

2a – Check of $b, b, \ldots, b$ inputs fails for $b \in \{-2\}$.

Example D.6. Quadratic+1+2+3_complex
Phase 1 (methods 1a,1b,1c,1d,1e):

2a – Phase 2 fails because the sequence $(1, -2\omega - 2, -\omega - 2, 2\omega + 2, -2\omega - 2, -\omega - 2, 2\omega + 2, -2\omega - 2, \ldots, -2\omega - 2, -\omega - 2, 2\omega + 2, -2\omega - 2, \ldots)$ leads to an infinite loop.

2c – Phase 2 fails because the sequence $(1, -2\omega - 2, -\omega - 2, 2\omega + 2, -2\omega - 2, -\omega - 2, 2\omega + 2, -2\omega - 2, \ldots, -2\omega - 2, -\omega - 2, 2\omega + 2, -2\omega - 2, \ldots)$ leads to an infinite loop.

2d – Phase 2 fails because the sequence $(0, \omega + 2, -1, -1, 2\omega + 1, -1, -1, 2\omega + 1, -1, \ldots, -1, -1, 2\omega + 1, -1, \ldots)$ leads to an infinite loop.

Example D.7. Quadratic+1+3+4_complex
Phase 1 (methods 1b):

2c – Phase 2 fails because the sequence $(2, 2\omega + 1, -\omega - 2, -2, 2\omega, 2\omega + 1, -\omega - 2, -2, 2\omega, \ldots, 2\omega + 1, -\omega - 2, -2, 2\omega, \ldots)$ leads to an infinite loop.

2d – Check of $b, b, \ldots, b$ inputs fails for $b \in \{\omega + 3\}$.

Phase 1 (methods 1c):

2a – Phase 2 fails because the sequence $(2, 2\omega + 3, -2\omega - 2, 2, 2\omega + 3, -2\omega - 2, 2, \ldots, 2, 2\omega + 3, -2\omega - 2, 2, \ldots)$ leads to an infinite loop.

2c – Phase 2 fails because the sequence $(2, 2\omega + 2, -\omega + 1, 2, 2\omega + 2, -\omega + 1, 2, \ldots, 2, 2\omega + 2, -\omega + 1, 2, \ldots)$ leads to an infinite loop.

2d – Check of $b, b, \ldots, b$ inputs fails for $b \in \{\omega + 3\}$.

Phase 1 (methods 1d,1e):

2a – Phase 2 fails because the sequence $(2, 2\omega + 3, -2\omega - 2, 2, 2\omega + 3, -2\omega - 2, 2, \ldots, 2, 2\omega + 3, -2\omega - 2, 2, \ldots)$ leads to an infinite loop.

2c – Phase 2 fails because the sequence $(2, 2\omega + 2, -\omega + 1, 2, 2\omega + 2, -\omega + 1, 2, \ldots, 2, 2\omega + 2, -\omega + 1, 2, \ldots)$ leads to an infinite loop.

2d – Check of $b, b, \ldots, b$ inputs fails for $b \in \{\omega + 3\}$.

Phase 1 (methods 1a):

2c – Phase 2 fails because the sequence $(2, 2\omega + 2, -\omega + 1, 2, 2\omega + 2, -\omega + 1, 2, \ldots, 2, 2\omega + 2, -\omega + 1, 2, \ldots)$ leads to an infinite loop.

2d – Check of $b, b, \ldots, b$ inputs fails for $b \in \{\omega + 3\}$.

Example D.8. Quadratic+1+3+5_complex1
Phase 1 (methods 1a):

2a – Check of $b, b, \ldots, b$ inputs fails for $b \in \{2\omega + 2, \omega + 4, -\omega - 4, -2\omega - 2\}$.

2b – Check of $b, b, \ldots, b$ inputs fails for $b \in \{2\omega + 2, -2\omega - 2\}$.

2c – Check of $b, b, \ldots, b$ inputs fails for $b \in \{2\omega + 2, \omega + 4, -\omega - 4, -2\omega - 2\}$.

2d – Check of $b, b, \ldots, b$ inputs fails for $b \in \{2\omega + 2, 2\omega + 4, \omega + 4, -\omega - 4, -2\omega - 4, -2\omega - 2\}$.

2e – Check of $b, b, \ldots, b$ inputs fails for $b \in \{2\omega + 2, -2\omega - 2\}$.

Phase 1 (methods 1b,1c):

2a – Check of $b, b, \ldots, b$ inputs fails for $b \in \{-2\omega - 2\}$.

2b – Phase 2 fails because the sequence $(2, 2\omega+2, 2\omega+2, 2, 2\omega+2, 2\omega+2, \ldots, 2, 2\omega+2, 2\omega + 2, \ldots)$ leads to an infinite loop.

2c – Check of $b, b, \ldots, b$ inputs fails for $b \in \{2\omega + 2, -2\omega - 2\}$.

2d – Check of $b, b, \ldots, b$ inputs fails for $b \in \{2\omega + 2, -2\omega - 2\}$.

2e – Phase 2 fails because the sequence $(2, 2\omega+2, 2\omega+2, 2, 2\omega+2, 2\omega+2, \ldots, 2, 2\omega+2, 2\omega + 2, \ldots)$ leads to an infinite loop.

Phase 1 (methods 1d,1e):

2a – Check of $b, b, \ldots, b$ inputs fails for $b \in \{2\omega + 2, -2\omega - 2\}$.

2b – Check of $b, b, \ldots, b$ inputs fails for $b \in \{2\omega + 2, -2\omega - 2\}$.

2c – Check of $b, b, \ldots, b$ inputs fails for $b \in \{2\omega + 2, -2\omega - 2\}$.

2d – Check of $b, b, \ldots, b$ inputs fails for $b \in \{2\omega + 2, 2\omega + 4, -2\omega - 4, -2\omega - 2\}$.

2e – Check of $b, b, \ldots, b$ inputs fails for $b \in \{2\omega + 2, -2\omega - 2\}$.

Example D.9. Quadratic+1+3+5_complex2

Phase 1 (methods 1b,1c):

2a – Check of $b, b, \ldots, b$ inputs fails for $b \in \{-3\omega - 4, 2\omega + 2, 2\omega + 3, \omega + 3, -2\omega - 4, -2\omega - 3, -2\omega - 2, -\omega - 3\}$.

2b – Phase 2 fails because the sequence $(0, 1, 2\omega + 1, -4\omega - 4, 4\omega + 4, 0, 1, 4\omega + 4, 0, 1, 4\omega + 4, \ldots, 4\omega + 4, 0, 1, 4\omega + 4, \ldots)$ leads to an infinite loop.

2c – Check of $b, b, \ldots, b$ inputs fails for $b \in \{-3\omega - 3, 3\omega + 3\}$.

2d – Check of $b, b, \ldots, b$ inputs fails for $b \in \{-3\omega - 4, 2\omega + 3, 2\omega + 4, \omega + 3, -2\omega - 4, -2\omega - 3, -\omega - 3, 3\omega + 4\}$.

2e – Phase 2 fails because the sequence $(0, 1, 2\omega + 1, -4\omega - 4, 4\omega + 4, 0, 1, 4\omega + 4, 0, 1, 4\omega + 4, \ldots, 4\omega + 4, 0, 1, 4\omega + 4, \ldots)$ leads to an infinite loop.

Phase 1 (methods 1d,1e):

2a – Check of $b, b, \ldots, b$ inputs fails for $b \in \{2\omega+2, -2\omega-4, -2\omega-3, -2\omega-2, -\omega-3\}$.

2b – Phase 2 fails because the sequence $(0, 0, -4\omega-4, 3\omega+4, 0, 4\omega+4, 0, 4\omega+4, 0, 4\omega+4, \ldots, 0, 4\omega+4, 0, 4\omega+4, \ldots)$ leads to an infinite loop.

2c – Phase 2 fails because the sequence $(0, 0, -2\omega-1, 3\omega+3, -4\omega-4, \omega, -2\omega-3, 4\omega+4, -\omega, 2\omega+3, -3\omega-3, 4\omega+4, 2\omega+2, 4\omega+4, 2\omega+2, 4\omega+4, 2\omega+2, \ldots, 4\omega+4, 2\omega+2, 4\omega+4, 2\omega+2, \ldots)$ leads to an infinite loop.

2d – Check of $b, b, \ldots, b$ inputs fails for $b \in \{-3\omega-4, 2\omega+3, 2\omega+4, \omega+3, -2\omega-4, -2\omega-3, -\omega-3, 3\omega+4\}$.

2e – Phase 2 fails because the sequence $(0, -3\omega-4, 3\omega+3, -3\omega-4, -4\omega-4, 4\omega+4, 0, 0, 4\omega+4, -3\omega-4, -1, -3\omega-4, -1, -3\omega-4, -1, \ldots, -3\omega-4, -1, -3\omega-4, -1, \ldots)$ leads to an infinite loop.

Phase 1 (methods 1a):

2a – Check of $b, b, \ldots, b$ inputs fails for $b \in \{-3\omega-3, 2\omega+2, -2\omega-3, -\omega-3, 3\omega+3\}$.

2b – Phase 2 fails because the sequence $(0, -1, -3\omega-3, \omega, -3\omega-4, -3\omega-4, -3\omega-3, 4\omega+4, -3\omega-4, 2\omega+1, 0, -1, -3\omega-3, \omega, \ldots, 0, -1, -3\omega-3, \omega, \ldots)$ leads to an infinite loop.

2c – Phase 2 fails because the sequence $(0, 0, -4\omega-4, 2\omega+1, -\omega-2, -3\omega-4, 2\omega+2, -\omega-1, -\omega, 3\omega+3, -2\omega-4, 2\omega+1, -\omega-2, -3\omega-4, 2\omega+2, \ldots, 2\omega+1, -\omega-2, -3\omega-4, 2\omega+2, \ldots)$ leads to an infinite loop.

2d – Check of $b, b, \ldots, b$ inputs fails for $b \in \{2\omega+3, \omega+3, -2\omega-3, -\omega-3\}$.

2e – Phase 2 fails because the sequence $(0, -3\omega-4, 3\omega+3, -\omega, 2\omega+1, -2\omega-1, 1, 2\omega+1, 0, -1, -3\omega-3, \omega, -3\omega-4, -3\omega-4, -3\omega-3, 4\omega+4, -3\omega-4, 2\omega+1, 0, -1, -3\omega-3, \ldots, 2\omega+1, 0, -1, -3\omega-3, \ldots)$ leads to an infinite loop.

Example D.10. Quadratic+1+4+5_complex1

Phase 1 (Lemma 4.3):

2a – Phase 2 fails because the sequence $(2, 2\omega-1, -2\omega+1, 2\omega-2, -\omega+2, 2, 2\omega-2, -\omega+2, 2, 2\omega-2, \ldots, 2\omega-2, -\omega+2, 2, 2\omega-2, \ldots)$ leads to an infinite loop.

2b – Check of $b, b, \ldots, b$ inputs fails for $b \in \{-\omega-2, -2\omega+1\}$.

2c – Check of $b, b, \ldots, b$ inputs fails for $b \in \{-\omega-2\}$.

2d – Check of $b, b, \ldots, b$ inputs fails for $b \in \{-\omega-2\}$.

Phase 1 (methods 1a):

2a – Check of $b, b, \ldots, b$ inputs fails for $b \in \{2, -\omega-2, -2\omega\}$.

2b – Check of $b, b, \ldots, b$ inputs fails for $b \in \{-\omega-2\}$.

2c – Check of $b, b, \ldots, b$ inputs fails for $b \in \{2\omega-2, 2\omega+2, \omega+3, -\omega-2, -2\omega+2\}$.

2d – Check of $b, b, \ldots, b$ inputs fails for $b \in \{2\omega-2, 2\omega+2, \omega+3, -\omega-2, -2\omega+2\}$.

2e – Check of $b, b, \ldots, b$ inputs fails for $b \in \{-\omega - 2\}$.

Phase 1 (methods 1b,1c,1d,1e):

2a – Check of $b, b, \ldots, b$ inputs fails for $b \in \{2, -\omega - 2, -2\omega\}$.

2b – Check of $b, b, \ldots, b$ inputs fails for $b \in \{-\omega - 2\}$.

2c – Check of $b, b, \ldots, b$ inputs fails for $b \in \{2\omega + 2, -\omega - 2\}$.

2d – Check of $b, b, \ldots, b$ inputs fails for $b \in \{2\omega + 2, -\omega - 2\}$.

2e – Check of $b, b, \ldots, b$ inputs fails for $b \in \{-\omega - 2\}$.

Example D.11. Cubic+1+0+0+2_integer

Phase 1 (methods 1a,1b,1c,1d,1e):

2a – Phase 2 fails because the sequence $(0, 1, 0, 1, 0, 1, 0, \ldots, 0, 1, 0, 1, 0, \ldots)$ leads to an infinite loop.

2b – Phase 2 fails because the sequence $(0, 1, 0, -1, 0, 1, 0, -1, 0, \ldots, 0, 1, 0, -1, 0, \ldots)$ leads to an infinite loop.

2c – Phase 2 fails because the sequence $(0, 1, 0, -1, 0, 1, 0, -1, 0, \ldots, 0, 1, 0, -1, 0, \ldots)$ leads to an infinite loop.

2e – Phase 2 fails because the sequence $(0, 1, 0, -1, 0, 1, 0, -1, 0, \ldots, 0, 1, 0, -1, 0, \ldots)$ leads to an infinite loop.

Example D.12. Cubic+1+0+0–2_integer

Phase 1 (methods 1a,1b,1c,1d,1e):

2a – Phase 2 fails because the sequence $(0, 1, 0, 1, 0, 1, 0, \ldots, 0, 1, 0, 1, 0, \ldots)$ leads to an infinite loop.

2b – Phase 2 fails because the sequence $(0, 1, 0, -1, 0, 1, 0, -1, 0, \ldots, 0, 1, 0, -1, 0, \ldots)$ leads to an infinite loop.

2c – Phase 2 fails because the sequence $(0, 1, 0, -1, 0, 1, 0, -1, 0, \ldots, 0, 1, 0, -1, 0, \ldots)$ leads to an infinite loop.

2e – Phase 2 fails because the sequence $(0, 1, 0, -1, 0, 1, 0, -1, 0, \ldots, 0, 1, 0, -1, 0, \ldots)$ leads to an infinite loop.

Quadratic bases with integer alphabet

The following examples show alphabets divided into congruence classes modulo β and $\beta - 1$ for some numeration systems in Table 7.5.

Example D.13. An alphabet $\mathcal{A}$ divided into congruence classes modulo β:

$$\{\{-5,6\},\{-4\},\{-3\},\{-2\},\{-1\},\{0\},\{1\},\{2\},\{3\},\{4\},\{5\}\},$$

and modulo $\beta - 1$:

$$\{\{-5,1\},\{-4,2\},\{-3,3\},\{-2,4\},\{-1,5\},\{0,6\}\}.$$

Example D.14. An alphabet $\mathcal{A}$ divided into congruence classes modulo β:

$$\{\{-5,2\},\{-4,3\},\{-3,4\},\{-2,5\},\{-1\},\{0\},\{1\}\},$$

and modulo $\beta - 1$:

$$\{\{-5,2\},\{-4,3\},\{-3,4\},\{-2,5\},\{-1\},\{0\},\{1\}\}.$$

Example D.15. An alphabet $\mathcal{A}$ divided into congruence classes modulo β:

$$\{\{-2,3\},\{-1\},\{0\},\{1\},\{2\}\},$$

and modulo $\beta - 1$:

$$\{\{-2,0,2\},\{-1,1,3\}\}.$$

Example D.16. An alphabet $\mathcal{A}$ divided into congruence classes modulo β:

$$\{\{-2,3\},\{-1\},\{0\},\{1\},\{2\}\},$$

and modulo $\beta - 1$:

$$\{\{-2,2\},\{-1,3\},\{0\},\{1\}\}.$$

Example D.17. An alphabet $\mathcal{A}$ divided into congruence classes modulo β:

$$\{\{-7,6\},\{-6,7\},\{-5\},\{-4\},\{-3\},\{-2\},\{-1\},\{0\},\{1\},\{2\},\{3\},\{4\},\{5\}\},$$

and modulo $\beta - 1$:

$$\{\{-7,-1,5\},\{-6,0,6\},\{-5,1,7\},\{-4,2\},\{-3,3\},\{-2,4\}\}.$$

Example D.18. An alphabet $\mathcal{A}$ divided into congruence classes modulo β:

$$\{\{-10,11\},\{-9\},\{-8\},\{-7\},\{-6\},\{-5\},\{-4\},\{-3\},\{-2\},\\ \{-1\},\{0\},\{1\},\{2\},\{3\},\{4\},\{5\},\{6\},\{7\},\{8\},\{9\},\{10\}\},$$

and modulo $\beta - 1$:

$$\{\{-10,0,10\},\{-9,1,11\},\{-8,2\},\{-7,3\},\{-6,4\},\\ \{-5,5\},\{-4,6\},\{-3,7\},\{-2,8\},\{-1,9\}\}.$$

Interrupted examples

The computation of a weight function for the following numeration systems was interrupted because of memory limits.

Example D.19.

$\omega = -\frac{1}{2}\sqrt{37} + \frac{5}{2}$	$\beta = -\omega - 3 = \frac{1}{2}\sqrt{37} - \frac{11}{2}$
$m_\omega(t) = t^2 - 5\,t - 3$	$m_\beta(x) = x^2 + 11\,x + 21$
Real conjugate of β greater than 1:	no
$\#\mathcal{A} = 33$	$\mathcal{A}$ is minimal.

$$\mathcal{A} = \{0, 1, -1, \omega+1, -\omega-1, -\omega+1, \omega-1, \omega, -\omega, 2\omega+2, -2\,\omega-2, \omega+2, -\omega-2, -2\,\omega+2, 2\,\omega-2, 2\,\omega+1, -2\,\omega-1, -2\,\omega+1, 2\,\omega-1, 2\,\omega, -2\,\omega, -2\,\omega+3, 2\,\omega-3, -3\,\omega+3, 3\,\omega-3, -3\,\omega+2, 3\,\omega-2, -3\,\omega+1, 3\,\omega-1, 3\,\omega, -3\,\omega, -3\,\omega+4, 3\,\omega-4\}$$

Phase 1 (method 9):	✓, $\#\mathcal{Q} = 17$
$b, b, \ldots, b$ inputs (method 2c):	✓, maximal length of window: 3

Computation of Phase 2 (method 2c) was interrupted when the length of window 5 was being processed. Numbers of saved combinations for each finished length are: (0, 12399, 682670, 2721482)

Example D.20.

$\omega = -\frac{1}{2}\sqrt{29} + \frac{3}{2}$	$\beta = 3\,\omega + 1 = -\frac{3}{2}\sqrt{29} + \frac{11}{2}$
$m_\omega(t) = t^2 - 3\,t - 5$	$m_\beta(x) = x^2 - 11\,x - 35$
Real conjugate of β greater than 1:	yes
$\#\mathcal{A} = 49$	$\mathcal{A}$ is not minimal.

$$\mathcal{A} = \{0, 1, -1, \omega+1, -\omega-1, -\omega+1, \omega-1, \omega, -\omega, 2\,\omega+2, -2\,\omega-2, \omega+2, -\omega-2, 2, -2, 3\,\omega+3, -3\,\omega-3, 2\,\omega+3, -2\,\omega-3, \omega+3, -\omega-3, 4\,\omega+4, -4\,\omega-4, 3\,\omega+4, -3\,\omega-4, 2\,\omega+4, -2\,\omega-4, 5\,\omega+5, -5\,\omega-5, 4\,\omega+5, -4\,\omega-5, 3\,\omega+5, -3\,\omega-5, 6\,\omega+6, -6\,\omega-6, 5\,\omega+6, -5\,\omega-6, 4\,\omega+6, -4\,\omega-6, 7\,\omega+7, -7\,\omega-7, 6\,\omega+7, -6\,\omega-7, 5\,\omega+7, -5\,\omega-7, -2\,\omega+5, 2\,\omega-5, 4\,\omega+7, -4\,\omega-7\}$$

Phase 1 (method 9):	✓, $\#\mathcal{Q} = 46$
$b, b, \ldots, b$ inputs (method 21):	✓, maximal length of window: 5

Computation of Phase 2 (method 21) was interrupted.

Example D.21.

$\omega = \left(\frac{1}{9}\sqrt{19}\sqrt{3}+1\right)^{\frac{1}{3}} + \frac{2}{3\left(\frac{1}{9}\sqrt{19}\sqrt{3}+1\right)^{\frac{1}{3}}}$

$\beta = -2\,\omega^2+\omega+2 = \left(\sqrt{57}-\frac{197}{27}\right)^{\frac{1}{3}} - \frac{14}{9\left(\sqrt{57}-\frac{197}{27}\right)^{\frac{1}{3}}} - \frac{2}{3}$

$m_\omega(t) = t^3-2\,t-2$ $\quad$ $m_\beta(x) = x^3+2\,x^2+6\,x+18$

Real conjugate of β greater than 1: no

$\#\mathcal{A} = 31$ $\quad$ $\mathcal{A}$ is not minimal.

$$\mathcal{A} = \{0,1,-1,\omega^2+\omega+1,-\omega^2-\omega-1,\omega+1,-\omega-1,-\omega^2+\omega+1,\omega^2-\omega-1,\omega^2+1,$$
$$-\omega^2-1,-\omega^2+1,\omega^2-1,\omega^2-\omega+1,-\omega^2+\omega-1,-\omega+1,\omega-1,-\omega^2-\omega+1,\omega^2$$
$$+\omega-1,\omega^2+\omega,-\omega^2-\omega,\omega,-\omega,\omega^2+2,-\omega^2-2,2,-2,-\omega^2+\omega,\omega^2-\omega,\omega^2,-\omega^2\}$$

Phase 1 (method 9): ✓, $\#\mathcal{Q} = 83$

$b, b, \ldots, b$ inputs (method 21): ✓, maximal length of window: 5

Computation of Phase 2 (method 21) was interrupted when the length of window 4 was being processed. Numbers of saved combinations for each finished length are: (0, 71, 1887261)

Example D.22.

$\omega = \left(\frac{1}{9}\sqrt{29}\sqrt{3}+\frac{28}{27}\right)^{\frac{1}{3}} + \frac{1}{9\left(\frac{1}{9}\sqrt{29}\sqrt{3}+\frac{28}{27}\right)^{\frac{1}{3}}} + \frac{1}{3}$

$\beta = -\omega^2+\omega-1 = \left(\frac{2}{9}\sqrt{29}\sqrt{3}-2\right)^{\frac{1}{3}} - \frac{2}{3\left(\frac{2}{9}\sqrt{29}\sqrt{3}-2\right)^{\frac{1}{3}}} - 1$

$m_\omega(t) = t^3-t^2-2$ $\quad$ $m_\beta(x) = x^3+3\,x^2+5\,x+7$

Real conjugate of β greater than 1: no

$\#\mathcal{A} = 16$ $\quad$ $\mathcal{A}$ is minimal.

$$\mathcal{A} = \{0,1,-1,\omega^2+\omega+1,-\omega^2-\omega-1,\omega$$
$$+1,\omega^2+1,-\omega^2-1,-\omega^2+1,\omega^2+\omega,\omega,-\omega,-\omega^2+\omega,\omega^2-\omega,\omega^2,-\omega^2\}$$

Phase 1 (method 9): ✓, $\#\mathcal{Q} = 99$

$b, b, \ldots, b$ inputs (method 21): ✓, maximal length of window: 6

Computation of Phase 2 (method 21) was interrupted when the length of window 4 was being processed. Numbers of saved combinations for each finished length are: (73, 5329, 315494)