Czech Technical University in Prague
Faculty of Nuclear Sciences and Physical Engineering

Representation of complex numbers in redundant numeration systems

Reprezentace komplexních čísel v redundantních číselných systémech

Bachelor's Degree Project

Author: **Adam Blažek**
Supervisor: **prof. Ing. Edita Pelantová, CSc.**
Consultant: **Ing. Milena Svobodová, Ph.D.**

Academic year: 2023/2024

ČVUT
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Blažek** Jméno: **Adam** Osobní číslo: **509216**

Fakulta/ústav: **Fakulta jaderná a fyzikálně inženýrská**

Zadávající katedra/ústav: **Katedra matematiky**

Studijní program: **Matematické inženýrství**

Specializace: **Matematická informatika**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Reprezentace komplexních čísel v redundantních číselných systémech

Název bakalářské práce anglicky:

Represetation of complex numbers in redundant numeration systems

Pokyny pro vypracování:

1) Seznamte se s vlastnostmi gaussovských a eisensteinovských celých čísel, s definicí w-NAF reprezentací v okruzích a známými výsledky o nich.
2) Pro množinu cifer uzavřených na násobení a příslušných k jednomu ze zmíněných okruhů detekujte vhodné báze vzhledem k možnosti w-NAF reprezentace.
3) Pro zvolený numerační systém se pokuste o vytvoření formule pro určení počtu optimálních reprezentací k prvku okruhu zadanému ve w-NAF reprezentaci.
4) Zkoumejte horní odhad na počet optimálních reprezentací při pevné Hammingově váze w-NAF reprezentace.

Seznam doporučené literatury:

[1] Z. Masáková, E. Pelantová, M. Svobodová: Algebraické metody v teoretické informatice, studijní text FJFI ČVUT, 2023. Dostupné online z: https://people.fjfi.cvut.cz/pelanedi/ALTI_k_vystaveni.pdf
[2] J. Tůma, J. Vábek, On the number of binary signed digit representations of a given weight. Commentationes Mathematicae Universitatis Carolinae 56(3), 2015, 287-306.
[3] C. Heuberger, D. Krenn, Optimality of the Width-w Non-adjacent Form: General Characterisation and the Case of Imaginary Quadratic Bases. Journal de théorie des nombres de Bordeaux 25(2), 2013, 353-386.
[4] C. Heuberger, D. Krenn, Analysis of width-w non-adjacent forms to imaginary quadratic bases. Journal of Number Theory 133(5), 2013, 1752-808.
[5] W. Penney, A "binary" system for complex numbers. J.A.C.M. 12, 1965, 247-248.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

prof. Ing. Edita Pelantová, CSc. katedra matematiky FJFI

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Ing. Milena Svobodová, Ph.D. katedra matematiky FJFI

Datum zadání bakalářské práce: **31.10.2023** Termín odevzdání bakalářské práce: **05.08.2024**

Platnost zadání bakalářské práce: **30.09.2025**

Edita Pelantová	Masáková	[illegible]
prof. Ing. Edita Pelantová, CSc. podpis vedoucí(ho) práce	prof. Ing. Zuzana Masáková, Ph.D. podpis vedoucí(ho) ústavu/katedry	doc. Ing. Václav Čuba, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací.
Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

28.11.2023	[illegible]
Datum převzetí zadání	Podpis studenta

Acknowledgment:
I would like to thank my supervisor, prof. Edita Pelantová, for providing me this research topic, as well expertly guiding me to turn it into a bachelor's degree project and participate in various conferences and contests with it. I also thank my consultant Ing. Milena Svobodová for collaborating on the project and my family for supporting me throughout my studies.

Author's declaration:
I declare that this Bachelor's Degree Project is entirely my own work and I have listed all the used sources in the bibliography.

Prague, August 2, 2024 Adam Blažek

Název práce:
Reprezentace komplexních čísel v redundantních číselných systémech

Autor: Adam Blažek

Obor: Matematické inženýrství

Zaměření: Matematická informatika

Druh práce: Bakalářská práce

Vedoucí práce: prof. Ing. Edita Pelantová, CSc.

Konzultant: Ing. Milena Svobodová, Ph.D.

Abstrakt: *Penneyova číselná soustava* je poziční číselná soustava se základem $\mathrm{i} - 1$ a číslicemi $\{0, 1\}$; lze v ní reprezentovat všechna Gaussova celá čísla. Rozšířením na množinu číslic $\{0, \pm 1, \pm \mathrm{i}\}$ získáme redundantní číslenou soustavu. Každé číslo má jedinečnou reprezentaci, v níž mezi každými dvěma nenulovými číslicemi jsou alespoň dvě nuly; toto je zobecnění NAF reprezentací u dvojkové soustavy se znaménky. Tyto reprezentace jsou *optimální* ve smyslu Hammingovy váhy, což je důležité pro zvýšení rychlosti některých algoritmů v kryptografii na eliptických křivkách. V této práci ukážeme některé známé výsledky přímočařejším způsobem, poté pomocí konečného stavového automatu odvodíme nové výsledky. Konkrétně nalezneme čísla s nejvyšším počtem optimálních reprezentací, vypočteme asymptoticky průměrný počet optimálních reprezentací všech čísel a poskytneme algoritmus pro generaci náhodné optimální reprezentace daného čísla.

Klíčová slova: Gaussova celá čísla, NAF reprezentace, poziční číselné soustavy, redundantní číselné soustavy

Title:
Representation of complex numbers in redundant numeration systems

Author: Adam Blažek

Abstract: The *Penney numeration system* is a positional numeration system with base $\mathrm{i} - 1$ and digits $\{0, 1\}$, capable of representing all Gaussian integers. Extending it to the digit set $\{0, \pm 1, \pm \mathrm{i}\}$ produces a redundant numeration system. Each number has a unique representation such that every pair of non-zero digits has at least two zeros between them; this is a generalization of NAF representations in the signed binary system. These representations are *optimal* in terms of Hamming weight, which is important for improving the performance of certain algorithms in elliptic curve cryptography. In this work, we demonstrate some known properties of this system in a more straightforward way, then use a finite transducer to derive some new results. Namely, we identify numbers which have the most optimal representations, calculate the asymptotic average number of optimal representations of all numbers and provide an algorithm for generating a uniformly random representation of a given number.

Key words: Gaussian integers, NAF representations, positional numeration systems, redundant numeration systems

Contents

Introduction

Positional numeration systems have a long history. Over centuries, civilizations have slowly accepted them as the most practical way to denote numbers. Several different numbers have been used as the base, including twenty, sixty, twelve and even six, but we have mostly settled on ten — the number of fingers on both our hands.

Only recently, with the advent of computers, the matters have changed yet again. While we still use the decimal system, pretty much all modern electronic devices use base two — the binary system. There is a good reason for this: it is much easier to design electronic circuits that handle only two states — off and on, representing the digits 0 and 1 — than ten different states. Despite that, decimal computers have been tried, but they are now long lost in history.

All the aforementioned positional numeration systems had one thing in common: they used the non-negative integers strictly lower than the base as the digits. However, this is far from being the only possible choice for the set of digits. A few computers built in the Soviet Union [12] [13] used *balanced ternary*: a system that uses base three, but the digits are 0, 1 and -1. This has a number of advantages, the biggest one being that negative numbers can be represented without any special treatment and it is trivial to compute the negation of a number.

It is also possible to use the digits 0, 1 and -1 for representing numbers in base two, which is called *signed binary*. In this system, numbers have multiple representations, which allows a certain degree of choice. As shown in [6], the performance of addition and multiplication is optimal if we minimize the number of non-zero digits — the *Hamming weight*. This is achieved by representations in the *non-adjacent form (NAF)*, where each pair of adjacent digits contains at most one non-zero digit. Such a representation exists and is unique for every number.

In [1], this idea is generalized to all numeration systems, with emphasis on the case of generalized integers in the complex plane. The NAF condition is generalized to the w-NAF condition, which requires every adjacent w digits to contain at most one non-zero. It is shown that if a digit set satisfies a certain condition (*minimal norm representatives digit set*), then each number has a unique w-NAF representation, and if the digit set is also *w-subadditive*, then these representations are optimal (have the minimal Hamming weight).

Optimal representations in complex bases are useful for algorithms in elliptic curve cryptography [1]. For this application, due to redundancy being a key element in cryptography, it is important to know how many optimal representations (with the minimal Hamming weight) each number has and, in particular, what the upper bound is for a given weight. In [3] and [4], it is shown that for numbers in the signed binary system, the number of optimal representations for a number whose representation has a given length or Hamming weight is bounded by the Fibonacci sequence.

The signed binary system also has the useful property that there exists a paralellizable algorithm for adding numbers [9]. Since the set of digits $\{0, 1, -1\}$ is closed under multiplication, this also allows the standard long multiplication algorithm to be partially parallelized, running in linear time rather than quadratic. In [10], it is discussed which complex bases and digit sets allow parallel addition.

In this work, we focus on 3-NAF representations of the Gaussian integers in base $\mathrm{i}-1$, where the minimal norm representatives digit set $\{0, \pm1, \pm\mathrm{i}\}$ is closed under multiplication, as well as being the minimal set of digits to allow parallel addition. This is an extension of a "complex binary" system introduced by W. Penney in [5]. In Chapter 2, we demonstrate some properties of this system which have already been proven before, but in a more general and complicated way, including the optimality of 3-NAF representations, the average density of their non-zero digits and a recursive formula for finding the number of representations of a given Gaussian integer.

After that, we show some new results about this system. In Chapter 3, a bound for the number of equivalent optimal representations of a given Hamming weight, analogous to the result in [4]. In Chapter 4, the average number of optimal representations across all numbers of a given length, analogous to a result in [3]. Finally, in Chapter 5, an algorithm for generating a uniformly random optimal representation of a given number; the motivation for such an algorithm is shortly discussed in [3]. All of these new results use a transducer that converts any optimal representation of a number to its 3-NAF representation or the other way around, which is inspired by [4].

Chapter 1

Basics

1.1 Positional numeration systems

A *positional numeration system* is the most common way of representing numbers. It consists of a number β, the *base*, and a set of numbers $\mathscr{D}$, the *digits*. Usually the base and digits are positive integers, but the definition can be generalized to any abelian group, with multiplication by a base being replaced with a group endomorphism [1]. However, for the purposes of this project, it will suffice to generalize the notion to complex numbers.

Definition 1.1. Let $\mathscr{A}$ be a finite set. A *word* over the *alphabet* $\mathscr{A}$ is a sequence of elements from $\mathscr{A}$, denoted via juxtaposition: $a_1 \cdots a_n$. The set of all finite words over $\mathscr{A}$, including the *empty word* ϵ, is denoted $\mathscr{A}^*$. For $w \in \mathscr{A}^*$ and $n \in \mathbb{N}$, w^n denotes the finite word consisting of n repetitions of w and w^ω denotes the infinite word consisting of infinitely many repetitions of w.

Definition 1.2. A *positional numeration system* is an ordered pair $(\beta, \mathscr{D})$, where $\beta \in \mathbb{C}$ and $\mathscr{D} \subset \mathbb{C}$ is a finite set.

Definition 1.3. Let $(\beta, \mathscr{D})$ be a positional numeration system and $x_n \cdots x_0 \in \mathscr{D}^*$. Then the number *represented* by the word $x_n \cdots x_0$ *in base* β is

$$x = [x_n \cdots x_0]_\beta := \sum_{k=0}^{n} x_k \beta^k.$$

The word $x_n \cdots x_0$ is called a $(\beta, \mathscr{D})$*-representation* of x.

Example. Let $\beta = 4$, $\mathscr{D} = \{0, 1, 2, 3\}$. Then

$$[133220]_4 = 1 \cdot 4^5 + 3 \cdot 4^4 + 3 \cdot 4^3 + 2 \cdot 4^2 + 2 \cdot 4^1 + 0 \cdot 4^0 = 2024.$$

Example. Let $\beta = \varphi = \frac{1+\sqrt{5}}{2}$, $\mathscr{D} = \{\pm 1\}$. Then

$$[(-1)11]_\varphi = -1 \cdot \varphi^2 + 1 \cdot \varphi^1 + 1 \cdot \varphi^0 = 0.$$

Example. Let $\beta = \mathrm{i}$, $\mathscr{D} = \{0, 1\}$. Then

$$[11001100110011001100]_\mathrm{i} = \sum_{k=0}^{4} \left(\mathrm{i}^{4k+2} + \mathrm{i}^{4k+3} \right) = -5 - 5\mathrm{i}.$$

Note. A positional representation of a number is not necessarily unique. For example, with $\beta = 2$, $\mathscr{D} = \{0, 1, 2\}$,

$$[1000]_2 = [200]_2 = [120]_2 = [112]_2 = 8.$$

Definition 1.4. A positional representation $x_n \cdots x_0$ is called *reduced* if $x_n \neq 0$. As a special case, the empty representation (representing the number 0) is reduced.

Since adding leading zeros to a representation does not change its value, we will usually consider representations that differ only in leading zeros to be the same, with the reduced representation being their canonical form. In some cases, it is more useful to consider all representations as implicitly having an infinite sequence of zeros to their left.

It is a well-known fact that if β is an integer greater than 1 and $\mathscr{D} = \{0, \dots, \beta - 1\}$, as in our usual decimal systems, then every non-negative integer has a unique reduced representation. However, the basic concept of the proof is going to be useful later for proving similar facts about more unusual systems.

Definition 1.5. Let $\beta \in \mathbb{Z}$, $\beta \geq 2$, $\mathscr{D} = \{0, \dots, \beta - 1\}$. Then $(\beta, \mathscr{D})$ is the *standard positional numeration system with base β.*

Theorem 1.6. Let $(\beta, \mathscr{D})$ be a standard positional numeration system. Then every $x \in \mathbb{N}_0$ has exactly one reduced $(\beta, \mathscr{D})$-representation $x_n \cdots x_0$.

Proof. It follows immediately from Definition 1.3 that if $x = [x_n \cdots x_0]_\beta$, then $x \equiv x_0 \pmod{\beta}$. Since every digit from $\mathscr{D}$ belongs to a different congruence class modulo β and all classes are covered, this uniquely determines x_0. Since $[x_n \cdots x_0]_\beta = x_0 + \beta \cdot [x_n \cdots x_1]_\beta$ (which also follows from the definition), $x_n \cdots x_1$ has to be a $(\beta, \mathscr{D})$-representation of $\frac{x - x_0}{\beta}$ (which is obviously a non-negative integer). Therefore, the same argument can be used to uniquely find the value of x_1. Since $\frac{x - x_0}{\beta} < x$ for all $x \neq 0$, repeating this process will eventually result in trying to find a representation of 0. Since 0 obviously has exactly one represention that does not start with 0 — the empty string, this gives an algorithm for finding the $(\beta, \mathscr{D})$-representation of x along with a proof of its uniqueness. ■

Example. Let us find the base-17 representation of 2024 (with digits $\{0, 1, \dots, 16\}$). We start by dividing 2024 by 17 with remainder:

$$2024 = 119 \cdot 17 + 1.$$

From this, we know that the representation ends in 1, which is preceded by the representation of 119. We perform another division:

$$119 = 7 \cdot 17 + 0.$$

This tells us that the representation of 119 ends in 0, preceded by the representation of 7. Since $7 < 17$, its representation is just the single digit 7; if we were to continue dividing, we would get an infinite sequence of leading zeros. Therefore, 2024 is represented as $[701]_{17}$ in the standard base-17 system.

1.2 NAF binary representations

Consider the positional numeration system with $\beta = 2$ and $\mathscr{D} = \{0, \pm 1\}$. For the sake of compactness, we will write -1 as $\overline{1}$ when used as a digit (this notation is not to be confused with the notation for complex conjugation). This system can represent any $x \in \mathbb{Z}$, including negative integers: simply take the standard binary representation of $|x|$ and multiply all digits by $\operatorname{sgn} x$. In this system, many integers have multiple representations, for example $[101]_2 = \left[11\overline{1}\right]_2 = \left[10\overline{1}\overline{1}\right]_2 = \left[1\overline{1}01\right]_2 = 5$. However, it turns out that if we add a simple restriction to which representations are eligible, reduced representations will once again be unique.

Definition 1.7. A positional representation $x_n \cdots x_0$ is in *non-adjacent form (NAF)* if there are no two non-zero digits next to each other, that is,

$$x_k = 0 \text{ or } x_{k+1} = 0 \text{ for all } k \in \mathbb{N},$$

using the convention of representations having infinitely many leading zeros.

Note. The NAF property was first introduced in [6], where it was called "property M".

Note. If all non-zero digits have an absolute value of 1 (as is the case with $\mathscr{D} = \{0, \pm 1\}$), Definition 1.7 can be equivalently written in a more compact, but also more obfuscated way:

$$|x_k| + |x_{k+1}| \leq 1 \text{ for all } k \in \mathbb{N}.$$

Example. Out of the aforementioned representations of 5: $[101]_2$, $\left[11\bar{1}\right]_2$, $\left[10\overline{11}\right]_2$, $\left[1\bar{1}01\right]_2$, precisely one is in non-adjacent form, namely $[101]_2$. It turns out that this is the case for all integers, as shown in the following theorem.

Theorem 1.8. Let $\mathscr{D} = \{0, \pm 1\}$. Then every $x \in \mathbb{Z}$ has exactly one reduced $(2, \mathscr{D})$-representation $x_n \cdots x_0$ with the NAF property.

Note. First proven in [6].

Proof. Analogously as in the proof of Theorem 1.6, we need to have $x_0 \equiv x \pmod 2$. If x is even, this uniquely determines x_0 to be 0; we can then recursively find a represention of $\frac{x}{2}$. If x is odd, we have two options: either choose $x_0 = 1$ and find a representation of $\frac{x-1}{2}$, or choose $x_0 = -1$ and find a representation of $\frac{x+1}{2}$. Either way, x_0 will be a non-zero digit, so in order to fulfill the NAF constraint, we need $x_1 = 0$, which means that $[x_n \cdots x_1]_2$ has to be even. Since exactly one of $\frac{x \pm 1}{2}$ is even, we have precisely one choice. Repeating this process, we will eventually reach 0, since for all non-zero x we have $\left|\frac{x - x_0}{2}\right| < |x|$. ■

Example. Let us find the NAF binary representation of 119. Since 119 is odd, its representation has to end in either 1 or $\bar{1}$. If we were to choose 1, we would have to find a representation of $\frac{119-1}{2} = 59$ and then append the 1 to it. However, since 69 is also odd, the resulting representation would end in two non-zero digits, making it not NAF. So instead, we need to use $\bar{1}$ as the last digit and prepend the representation of $\frac{119+1}{2} = 60$. 60 is even, so its representation ends in a zero, exactly as we need. After dividing by 2, we get the even number 30, giving us another zero. After that, we have to find a representation of 15. Again, it is necessary to choose $\bar{1}$ because choosing 1 would violate the NAF condition. After a few more divisions by 2, we get to 1, which must be represented as 1 because any other of its infinitely many representations is not NAF:

$$1 = [1]_2 = \left[1\bar{1}\right]_2 = \left[1\overline{11}\right]_2 = \left[1\overline{111}\right]_2 = \cdots.$$

Putting all this together, we obtain the desired NAF representation

$$119 = \left[1000\bar{1}00\bar{1}\right]_2.$$

The system with $\beta = 2$, $\mathscr{D} = \{0, \pm 1\}$ also has the useful property that it is possible to tell the sign of a number just from the first non-zero digit of its representation, no matter whether the representation is NAF. We will need this property later, so here is a simple proof.

Theorem 1.9. Let $\mathscr{D} = \{0, \pm 1\}$ and $x_n \cdots x_0$ be a reduced $(2, \mathscr{D})$ representation of a non-zero integer x. Then $\operatorname{sgn} x = x_n$.

Proof. We shall prove the case $x_n = 1$, the proof for $x_n = -1$ is analogous. From Definition 1.3:

$$x = \sum_{k=0}^{n} x_k 2^k = 2^n + \sum_{k=0}^{n-1} x_k 2^k \geq 2^n + \sum_{k=0}^{n-1} -2^k = 2^n - \frac{2^n - 1}{2 - 1} = 1.$$

■

NAF binary representations have the interesting property that they contain the fewest non-zero digits out of all $(2, \mathscr{D})$-representations of the same number. As mentioned in the introduction, this has positive implications for parallel multiplication algorithms. In order to prove this fact, we first need some definitions and a lemma.

Definition 1.10. The *Hamming weight* of a $(\beta, \mathscr{D})$-representation $x_n \cdots x_0$ is its count of non-zero digits:

$$W(x_n \cdots x_0) := |\{k \in \{0, \ldots, n\} \mid x_k \neq 0\}|.$$

Note. If all non-zero digits in $\mathscr{D}$ have an absolute value of 1, then Definition 1.10 can be equivalently expressed as

$$W(x_n \cdots x_0) := \sum_{k=0}^{n} |x_k|.$$

Definition 1.11. A $(\beta, \mathscr{D})$-representation is *optimal* if its Hamming weight is minimal among all $(\beta, \mathscr{D})$-representations of the same number.

Lemma 1.12. Let $\mathscr{D} = \{0, \pm 1\}$. Then any $(2, \mathscr{D})$-representation $x_n \cdots x_0$ can be converted to a NAF $(2, \mathscr{D})$-representation of the same number by repeatedly applying the following substitutions:

- $1\bar{1} \mapsto 01$,
- $\bar{1}1 \mapsto 0\bar{1}$,
- $011 \mapsto 10\bar{1}$,
- $0\bar{1}\bar{1} \mapsto \bar{1}01$.

Proof. It is obvious from Definition 1.3 that the substitutions preserve the value of the representation. Also, if this procedure terminates, then the resulting representation has to be NAF, because if there was a pair of non-zero digits next to each other, one of the substitutions could be applied to the first such pair. Therefore, we only need to prove that this procedure terminates. Suppose that $x_n = 1$; the proof is analogous for $x_n = -1$. Clearly, the only way the representation can increase in length is by applying the third substitution at the beginning:

$$11y_{n-2} \cdots y_0 \to 10\bar{1}y_{n-2} \cdots y_0.$$

In order for the representation to increase in length again, a series of substitutions would have to change $0\bar{1}y_{n-2} \cdots y_0$ into $1z_{n-1}z_{n-2} \cdots z_0$ for some digits z_i. However, by Theorem 1.9, the former represents a negative number, whereas the latter represents a positive number. Since the substitutions preserve the value, this is impossible. Therefore, the length of the initial representation can increase at most once.

In other words, the length of all intermediate representations is at most $n+2$. For a given intermediate representation $y_{n+1} \cdots y_0$, we define

$$s(y_{n+1} \cdots y_0) := \sum_{k=0}^{n+1} (n+2-k)|y_k|.$$

It can be easily verified that every substitution strictly decreases s. Since s is a non-negative integer, the procedure always terminates. ■

Example. Consider the number 119 with its standard binary representation $[1110111]_2$. By applying these substitutions from right to left, we obtain

$$\begin{aligned} 119 &= [1110111]_2 \\ &= \left[11110\bar{1}1\right]_2 \\ &= \left[111100\bar{1}\right]_2 \\ &= \left[10\bar{1}11100\bar{1}\right]_2 \\ &= \left[100\bar{1}100\bar{1}\right]_2 \\ &= \left[1000\bar{1}00\bar{1}\right]_2. \end{aligned}$$

This gave us the same result as the modular algorithm, which was to be expected because binary NAF representations are unique according to Theorem 1.8. Notice that several of the substitutions decreased the Hamming weight of the representation, whereas none of them increased it. This is the key to proving that binary NAF representations are optimal, as formalized by the following theorem.

Theorem 1.13. Binary NAF representations are optimal: Let $\mathscr{D} = \{0, \pm 1\}$. Let $x_n \cdots x_0,\ y_m \cdots y_0$ be two $(2, \mathscr{D})$-representations of the same integer, with $x_n \cdots x_0$ also being NAF. Then $W(x_n \cdots x_0) \le W(y_n \cdots y_0)$.

Note. First proven in [6].

Proof. By Lemma 1.12, we can convert $y_m \cdots y_0$ into a NAF representation $z_l \cdots z_0$ of the same number. None of the possible substitutions increase the Hamming weight, therefore $W(z_l \cdots z_0) \le W(y_m \cdots y_0)$. According to Theorem 1.8, binary NAF representations of the same number are equal, so $W(z_l \cdots z_0) = W(x_n \cdots x_0)$, which concludes the proof. ■

Not only are binary NAF representations optimal, they have asymptotically fewer non-zero digits on average than standard binary representations. To be specific: If we list all possible standard binary representations of some fixed length (not necessarily reduced), obviously exactly half of the digits will be 0 and the other half will be 1. However, if we do the same with NAF representations in signed binary, the total proportion of non-zero digits will only be approximately $\frac{1}{3}$, as shown in the following theorem.

Theorem 1.14. Let $\mathscr{D} = \{0, \pm 1\}$. Let a_n be the number of all (not necessarily reduced) NAF $(2, \mathscr{D})$-representations of length n and b_n the sum of the Hamming weights of these representations. Then $\lim\limits_{n \to \infty} \frac{b_n}{n a_n} = \frac{1}{3}$. That is, for an average sufficiently long NAF $(2, \mathscr{D})$-representation, about $\frac{1}{3}$ of its digits are non-zeros.

Note. Shown in [8].

Proof. Let us find an explicit formula for a_n and b_n. We can think of NAF $(2, \mathscr{D})$-representations as consisting of three kinds of "blocks": $\boxed{0}$, $\boxed{01}$ and $\boxed{0\bar{1}}$. Note that these blocks can only be used to construct representations starting with a zero, so we are actually expressing a_{n-1} and b_{n-1}, but this does not matter when $n \to \infty$. Take one representation with n digits and let k be its Hamming weight, which corresponds to the number of $\boxed{01}$ and $\boxed{0\bar{1}}$ blocks. These blocks take up $2k$ digits, so the number of $\boxed{0}$ blocks is $n - 2k$, for a total of $n - k$ blocks. There are $\binom{n-k}{k}$ ways to choose which of these blocks contain a non-zero digit and 2^k ways to choose whether the digit will be 1 or $\bar{1}$. Therefore, the total number of these representations is

$$a_n = \sum_{k=0}^{\infty} 2^k \binom{n-k}{k}$$

and the sum of Hamming weights is

$$b_n = \sum_{k=0}^{\infty} k 2^k \binom{n-k}{k},$$

where we define $\binom{n}{k} := 0$ for $k > n$ so that we do not have to worry about the upper limit. If we also define $\binom{n}{k} := 0$ for $k < 0$, the formula $\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$ will work for all k, which we can use in finding a recursive expression for a_n:

$$\begin{aligned}
a_n &= \sum_{k=0}^{\infty} 2^k \binom{n-k}{k} \\
&= \sum_{k=0}^{\infty} 2^k \binom{n-k-1}{k} + \sum_{k=0}^{\infty} 2^k \binom{n-k-1}{k-1} \\
&= \sum_{k=0}^{\infty} 2^k \binom{n-k-1}{k} + \sum_{k=-1}^{\infty} 2^{k+1} \binom{n-k-2}{k} \\
&= \sum_{k=0}^{\infty} 2^k \binom{n-k-1}{k} + 2 \sum_{k=0}^{\infty} 2^k \binom{n-k-2}{k} \\
&= a_{n-1} + 2a_{n-2}.
\end{aligned}$$

Solving this recurrence using its characteristic polynomial $\lambda^2 - \lambda - 2 = (\lambda - 2)(\lambda + 1)$, we get

$$a_n = \alpha 2^n + \beta(-1)^n$$

for some $\alpha, \beta \in \mathbb{R}$. Clearly $\alpha \neq 0$, otherwise some terms would be negative. For b_n, we analogously have

$$\begin{aligned}
b_n &= \sum_{k=0}^{\infty} k2^k \binom{n-k}{k} \\
&= \sum_{k=0}^{\infty} k2^k \binom{n-k-1}{k} + \sum_{k=0}^{\infty} k2^k \binom{n-k-1}{k-1} \\
&= \sum_{k=0}^{\infty} k2^k \binom{n-k-1}{k} + \sum_{k=-1}^{\infty} (k+1)2^{k+1} \binom{n-k-2}{k} \\
&= \sum_{k=0}^{\infty} k2^k \binom{n-k-1}{k} + 2\sum_{k=0}^{\infty} k2^k \binom{n-k-2}{k} + 2\sum_{k=0}^{\infty} 2^k \binom{n-k-2}{k} \\
&= b_{n-1} + 2b_{n-2} + 2a_{n-2} \\
&= b_{n-1} + 2b_{n-2} + 2\alpha 2^{n-2} + 2\beta(-1)^{n-2}.
\end{aligned}$$

This is a non-homogenous linear recurrence. The homogenous solution is the same as for a_n, except with possibly different coefficients $\tilde{\alpha}, \tilde{\beta} \in \mathbb{R}$:

$$b_n^{\text{hom}} = \tilde{\alpha}2^n + \tilde{\beta}(-1)^n.$$

The particular solution is of the form

$$b_n^{\text{par}} = \mu n 2^n + \nu n(-1)^n$$

for some $\mu, \nu \in \mathbb{R}$. We can find those by substituting in the recurrence:

$$\mu n2^n + \nu n(-1)^n - \mu(n-1)2^{n-1} - \nu(n-1)(-1)^{n-1} - 2\mu(n-2)2^{n-2} - 2\nu(n-2)(-1)^{n-2} = 2\alpha 2^{n-2} + 2\beta(-1)^{n-2}.$$

Collecting terms with 2^n and cancelling, we obtain

$$\mu\left(n - \frac{n-1}{2} - \frac{n-2}{2}\right) = \frac{\alpha}{2},$$

$$\mu = \frac{\alpha}{3} \neq 0.$$

This suffices to evaluate the desired limit, since only the fastest-growing terms matter:

$$\lim_{n\to\infty} \frac{b_n}{na_n} = \lim_{n\to\infty} \frac{\tilde{\alpha}2^n + \tilde{\beta}(-1)^n + \frac{\alpha}{3}n2^n + \nu n(-1)^n}{\alpha n 2^n + \beta n(-1)^n} = \frac{1}{3}.$$

■

In Theorem 1.14, we have shown that NAF binary representations of a fixed length have an asymptotically smaller Hamming weight than standard binary representations. One might object that if, on average, the NAF representation of a given number is significantly longer than the standard binary representation, this might not result in such an improvement for fixed numbers. However, as shown in the following theorem, a NAF binary representation of a given integer is at most one digit longer than its standard binary representation. This is in fact already apparent from the proof of Lemma 1.12, but can easily be proven separately.

Theorem 1.15. Let $\mathcal{D} = \{0, \pm 1\}$ and $x \in \mathbb{Z},\ x \neq 0$. Then the reduced NAF $(2, \mathcal{D})$-representation of x has at most $\lfloor \log_2 |x| \rfloor + 2$ digits.

Note. First shown in [6].

Proof. Let h_n be the smallest positive integer whose binary NAF representation consists of exactly n digits. From Theorem 1.9, we know that the representation starts with 1. To minimize the value while respecting the NAF condition, it is easy to see that this 1 is followed by alternating 0 and $\bar{1}$, so for all $m \in \mathbb{N}_0$,

$$h_{2m+1} = \left[1\left(0\bar{1}\right)^m\right]_2,\ h_{2m+2} = \left[1\left(0\bar{1}\right)^m 0\right]_2.$$

Consider first the odd case $n = 2m + 1$. By Definition 1.3,

$$h_n = h_{2m+1} = 2^{2m} - \sum_{k=0}^{m-1} 2^{2k} = 2^{2m} - \sum_{k=0}^{m-1} 4^k = 2^{2m} - \frac{4^m - 1}{3} = \frac{2^n + 1}{3},$$

$$n = \log_2(3h_n - 1) < \log_2(4h_n) = \log_2 h_n + 2.$$

Since h_n has the minimal absolute value out of all integers with an n-digit NAF $(2, \mathcal{D})$-representation, this concludes the proof for odd n. For even n, $h_n = 2h_{n-1}$, so by the already proven odd case,

$$n = (n-1) + 1 < \log_2 h_{n-1} + 2 + 1 = \log_2 h_n + 2.$$

■

1.3 Complex integers

There are two different natural ways to extend the concept of an integer to complex numbers: either using a square lattice, or using a triangular lattice. These generalizations are called *Gaussian integers* and *Eisenstein integers* respectively.

Definition 1.16. The *Gaussian integers* are the numbers

$$\mathbb{Z}[\mathrm{i}] := \{a + b\mathrm{i} \mid a, b \in \mathbb{Z}\}.$$

Definition 1.17. Let $\omega := \exp\left(\frac{2}{3}\pi\mathrm{i}\right) = -\frac{1}{2} + \frac{\sqrt{3}}{2}\mathrm{i}$. The *Eisenstein integers* are the numbers

$$\mathbb{Z}[\omega] := \{a + b\omega \mid a, b \in \mathbb{Z}\}.$$

In Appendix A, there are programs implementing basic arithmetic on Gaussian and Eisenstein integers.

It is obvious how to naturally extend the notion of divisibility and modular arithmetic to integers. A more interesting question is how to extend the notion of parity. We could, just like in the ordinary integers, define a complex integer to be even if it is divisible by 2. However, this way we would lose the useful property that if x is even, then $x \pm 1$ is odd and vice-versa. Instead, we will use the following definition for the case of Gaussian integers, which preserves this property (additionally ensuring that $x \pm \mathrm{i}$ also has the opposite parity from x):

Definition 1.18. Let $x \in \mathbb{Z}[\mathrm{i}]$. If $x \equiv 0 \pmod{\mathrm{i} - 1}$, then x is *even*, otherwise x is *odd.* (Note that divisibility by $\mathrm{i} - 1$ is the same as divisibility by $\mathrm{i} + 1$, we are choosing the former for reasons that will be apparent soon.)

However, it is not possible to do this with Eisenstein integers, the reason being that the triangular grid is not a bipartite graph. Note that if there were names for different congruence classes modulo 3, they could be generalized to Eisenstein integers by considering congruence classes modulo $\omega - 1$.

The most obvious way to represent complex integers using a positional numeration system is to simply choose a system for representing regular integers and apply it component-wise. However, with the right choice of base and digits, we can naturally represent both kinds of complex integers directly.

Theorem 1.19. Let $\beta = \mathrm{i} - 1$, $\mathscr{D} = \{0, 1\}$. Then every Gaussian integer x has a unique reduced $(\beta, \mathscr{D})$-representation $x_n \cdots x_0$.

Note. This system is known as the *Penney numeration system* due to *Walter F. Penney* being one of the first mathematicians to work with it [5].

Proof. Analogously as in the proof of Theorem 1.6, if x is even (by Definition 1.18), then $x_0 = 0$, otherwise $x_0 = 1$. We then recursively find a representation of $\frac{x - x_0}{\mathrm{i}-1}$. It remains to show that this procedure terminates. We can estimate

$$\left|\frac{x - x_0}{\mathrm{i} - 1}\right| = \frac{|x - x_0|}{\sqrt{2}} \le \frac{|x| + |x_0|}{\sqrt{2}} \le \frac{|x| + 1}{\sqrt{2}}.$$

This implies that if $|x| > \sqrt{2} + 1$, then $\left|\frac{x-x_0}{\mathrm{i}-1}\right| < |x|$. Since $|x|$ is always the square root of an integer, after finitely many steps we will reach an x such that $|x| \le \sqrt{2} + 1$. Therefore, we just have to manually check that every such x has a representation. There are 21 such numbers:

$$\begin{array}{rrrr}
0 = [\epsilon]_{\mathrm{i}-1}\,, & & & \\
1 = [1]_{\mathrm{i}-1}\,, & \mathrm{i} = [11]_{\mathrm{i}-1}\,, & -1 = [11101]_{\mathrm{i}-1}\,, & -\mathrm{i} = [111]_{\mathrm{i}-1}\,, \\
1 + \mathrm{i} = [1110]_{\mathrm{i}-1}\,, & -1 + \mathrm{i} = [10]_{\mathrm{i}-1}\,, & -1 - \mathrm{i} = [110]_{\mathrm{i}-1}\,, & 1 - \mathrm{i} = [111010]_{\mathrm{i}-1}\,, \\
2 = [1100]_{\mathrm{i}-1}\,, & 2\mathrm{i} = [1110100]_{\mathrm{i}-1}\,, & -2 = [11100]_{\mathrm{i}-1}\,, & -2\mathrm{i} = [100]_{\mathrm{i}-1}\,, \\
2 + \mathrm{i} = [1111]_{\mathrm{i}-1}\,, & -1 + 2\mathrm{i} = [11001]_{\mathrm{i}-1}\,, & -2 - \mathrm{i} = [11101011]_{\mathrm{i}-1}\,, & 1 - 2\mathrm{i} = [101]_{\mathrm{i}-1}\,, \\
1 + 2\mathrm{i} = [1110101]_{\mathrm{i}-1}\,, & -2 + \mathrm{i} = [11111]_{\mathrm{i}-1}\,, & -1 - 2\mathrm{i} = [11101001]_{\mathrm{i}-1}\,, & 2 - \mathrm{i} = [111011]_{\mathrm{i}-1}\,.
\end{array}$$

■

Note. It is apparent from the proofs of Theorem 1.6 and Theorem 1.19 that they can be generalized to a much larger class of positional numeration systems, with the only two requirements being that the digits uniquely cover all congruence classes modulo β and that all numbers with an absolute value within some bound have a representation. The general statement is outside the scope of this project.

Note. It is *not* possible to represent every Gaussian integer with $\beta = \mathrm{i} + 1$ and $\mathscr{D} = \{0, 1\}$. If we tried to find such a representation for some numbers, such as -1, using the modular algorithm, we would get into an infinite loop. This shows that verifying the representability of certain small numbers is an important part of the proof of Theorem 1.19 that cannot be left out.

Theorem 1.20. Let $\beta = \omega - 1$, $\mathscr{D} = \{0, 1, \omega + 1\}$. Then every Eisenstein integer x has a unique reduced $(\beta, \mathscr{D})$-representation $x_n \cdots x_0$.

Proof. Analogous to the proof of Theorem 1.19. Left as an exercise for the reader. ■

1.4 Exercises

Exercise 1.21. Let $\beta \in \mathbb{Z}$, $\beta \geq 2$. Theorem 1.6 shows the *modular algorithm* for finding the standard base-β representation of a non-negative integer x. There also exists a *greedy algorithm* for doing the same, with the difference that it produces digits starting from the beginning of the word, rather than the end. The basic idea is that we find the maximal n such that $\beta^n \leq x$, then set $x_n := \left\lfloor \frac{x}{\beta^n} \right\rfloor$ and fill in the remaining digits by finding the representation of $x - x_n\beta^n$. Prove that this algorithm works and use it to find the standard base-17 representation of 2024.

Exercise 1.22. Let $\beta \in \mathbb{Z}$, $\beta \geq 1$, $\mathscr{D} = \{1, \dots, \beta\}$. The system $(\beta, \mathscr{D})$ is called the *bijective positional numeration system with base β*. Prove that every non-negative integer has a unique representation in this system (hence the name "bijective"). What do these representations look like when $\beta = 1$?

Exercise 1.23. Prove Theorem 1.20.

Chapter 2

w-NAF representations of complex integers

In Chapter 1, we extended the standard binary system by adding the digit -1, creating a system where a number can have multiple representations, then restricted it to NAF representations, which once again guaranteed uniqueness while also decreasing the average ratio of non-zero digits in a representation. In this chapter, we will show analogous systems for the Gaussian and Eisenstein integers.

Note that the various facts about the systems shown in this chapter are special cases of more general results known before, but here they are formulated in a simpler way that does not require a lot of abstract machinery.

2.1 Gaussian integers

We shall start by adding the digit -1 to the Penney system; that is, we use $\beta = \mathrm{i} - 1$, $\mathscr{D} = \{0, \pm 1\}$. The natural question is whether every Gaussian integer has a NAF representation in this system. It is not hard to find that the answer is negative. For example, the number i is odd, so its representation would have to end in either 1 or -1, but after subtracting either digit and dividing by $\mathrm{i} - 1$, we are left with another odd number, forcing us to add another non-zero digit.

However, we can add two more digits without violating the useful property that the set of digits is closed under multiplication: i and $-\mathrm{i}$. As with -1, we will denote $-\mathrm{i}$ as $\bar{\mathrm{i}}$ when used as a digit. The resulting system is the main subject of this project, so we shall give it a name.

Definition 2.1. The *extended Penney system* is the positional numeration system with $\beta = \mathrm{i} - 1$ and $\mathscr{D} = \{0, \pm 1, \pm \mathrm{i}\}$.

Obviously, most Gaussian integers can be represented in multiple ways in this system. However, they can even have multiple NAF representations, for example $[1]_{\mathrm{i}-1} = \left[\mathrm{i}0\bar{1}\right]_{\mathrm{i}-1} = 1$. This suggests that we could restrict the possible representations even further in order to possibly obtain even better results in terms of average weight. It turns out that a simple generalization of the NAF condition is sufficient.

Definition 2.2. Let $w \in \mathbb{N}^+$. A positional representation $x_n \cdots x_0$ is in *w-non-adjacent form (w-NAF)* if every contiguous subsequence of length at most w contains at most one non-zero digit, that is,

$$W(x_{k+w-1} \cdots x_k) \leq 1 \text{ for all } k \in \mathbb{N}.$$

Note. If all non-zero digits have an absolute value of 1, Definition 2.2 can be equivalently written in a more compact, but also more obfuscated way:

$$\sum_{j=k}^{k+w-1} |x_j| \leq 1 \text{ for all } k \in \mathbb{N}.$$

Note. Trivially, every representation is 1-NAF and the 2-NAF condition is equivalent to the original NAF condition.

Lemma 2.3. Let x be an odd Gaussian integer. Then there exists exactly one $x_0 \in \{\pm 1, \pm \mathrm{i}\}$ such that $x \equiv x_0 \pmod{2 + 2\mathrm{i}}$.

Proof. Notice that $(2 + 2\mathrm{i}) \mid 4$. Let $x = a + b\mathrm{i}$, $a, b \in \mathbb{Z}$. Consider the possible congruence classes of a, b modulo 4, which cannot have the same parity since x is odd:

- If $a \equiv 0 \pmod 4$ and $b \equiv 1 \pmod 4$, then $x \equiv \mathrm{i} \pmod 4$, so $x \equiv \mathrm{i} \pmod{2 + 2\mathrm{i}}$,
- If $a \equiv 0 \pmod 4$ and $b \equiv -1 \pmod 4$, then $x \equiv -\mathrm{i} \pmod 4$, so $x \equiv -\mathrm{i} \pmod{2 + 2\mathrm{i}}$,
- If $a \equiv 2 \pmod 4$ and $b \equiv 1 \pmod 4$, then $x \equiv 2 + \mathrm{i} \pmod 4$, so $x \equiv -\mathrm{i} \pmod{2 + 2\mathrm{i}}$,
- If $a \equiv 2 \pmod 4$ and $b \equiv -1 \pmod 4$, then $x \equiv 2 - \mathrm{i} \pmod 4$, so $x \equiv \mathrm{i} \pmod{2 + 2\mathrm{i}}$,
- If $a \equiv 1 \pmod 4$ and $b \equiv 0 \pmod 4$, then $x \equiv 1 \pmod 4$, so $x \equiv 1 \pmod{2 + 2\mathrm{i}}$,
- If $a \equiv -1 \pmod 4$ and $b \equiv 0 \pmod 4$, then $x \equiv -1 \pmod 4$, so $x \equiv -1 \pmod{2 + 2\mathrm{i}}$,
- If $a \equiv 1 \pmod 4$ and $b \equiv 2 \pmod 4$, then $x \equiv 1 + 2\mathrm{i} \pmod 4$, so $x \equiv -1 \pmod{2 + 2\mathrm{i}}$,
- If $a \equiv -1 \pmod 4$ and $b \equiv 2 \pmod 4$, then $x \equiv -1 + 2\mathrm{i} \pmod 4$, so $x \equiv 1 \pmod{2 + 2\mathrm{i}}$.

This shows that x_0 exists. It is also unique because none of $\{\pm 1, \pm \mathrm{i}\}$ are congruent modulo $2 + 2\mathrm{i}$. ∎

Theorem 2.4. Every $x \in \mathbb{Z}[\mathrm{i}]$ has exactly one reduced 3-NAF representation $x_n \cdots x_0$ in the extended Penney system.

Note. Follows from a general result in [1].

Proof. Once again, we will use a modular algorithm to find the representation. If x is even, then the last digit has to be 0. If x is odd, we have a choice between four digits: ± 1 and $\pm \mathrm{i}$, but in order to fulfill the 3-NAF constraint, we need the following two digits to be zeros, meaning that $x - x_0$ has to be divisible by $(\mathrm{i} - 1)^3 = 2 + 2\mathrm{i}$. Lemma 2.3 ensures that there is precisely one choice. Either way, we continue by finding the representation of $\frac{x - x_0}{\mathrm{i} - 1}$. It remains to be proven that this procedure terminates. This can be done using the exact same argument as in the proof of Theorem 1.19, except with different representations for small numbers:

$$\begin{array}{rrrr}
0 = [\epsilon]_{\mathrm{i}-1}\,, & & & \\
1 = [1]_{\mathrm{i}-1}\,, & \mathrm{i} = [\mathrm{i}]_{\mathrm{i}-1}\,, & -1 = \left[\bar{1}\right]_{\mathrm{i}-1}\,, & -\mathrm{i} = \left[\bar{\mathrm{i}}\right]_{\mathrm{i}-1}\,, \\
1 + \mathrm{i} = \left[\bar{\mathrm{i}}0\right]_{\mathrm{i}-1}\,, & -1 + \mathrm{i} = [10]_{\mathrm{i}-1}\,, & -1 - \mathrm{i} = [\mathrm{i}0]_{\mathrm{i}-1}\,, & 1 - \mathrm{i} = \left[\bar{1}0\right]_{\mathrm{i}-1}\,, \\
2 = [\mathrm{i}00]_{\mathrm{i}-1}\,, & 2\mathrm{i} = \left[\bar{1}00\right]_{\mathrm{i}-1}\,, & -2 = \left[\bar{\mathrm{i}}00\right]_{\mathrm{i}-1}\,, & -2\mathrm{i} = [100]_{\mathrm{i}-1}\,, \\
2 + \mathrm{i} = \left[100\bar{\mathrm{i}}\right]_{\mathrm{i}-1}\,, & -1 + 2\mathrm{i} = [\mathrm{i}001]_{\mathrm{i}-1}\,, & -2 - \mathrm{i} = \left[\bar{1}00\mathrm{i}\right]_{\mathrm{i}-1}\,, & 1 - 2\mathrm{i} = \left[\bar{\mathrm{i}}00\bar{1}\right]_{\mathrm{i}-1}\,, \\
1 + 2\mathrm{i} = \left[100\bar{1}\right]_{\mathrm{i}-1}\,, & -2 + \mathrm{i} = \left[\mathrm{i}00\bar{\mathrm{i}}\right]_{\mathrm{i}-1}\,, & -1 - 2\mathrm{i} = \left[\bar{1}001\right]_{\mathrm{i}-1}\,, & 2 - \mathrm{i} = \left[\bar{\mathrm{i}}00\mathrm{i}\right]_{\mathrm{i}-1}\,.
\end{array}$$

∎

The algorithm for finding 3-NAF representations of Gaussian integers in the extended Penney system is implemented in a program in Appendix A.

Example. Let us find the 3-NAF representation of $-3 + 11\mathrm{i}$ in the extended Penney system. Because the number is even, we start by writing a 0 and dividing by $\mathrm{i} - 1$:

$$\frac{-3 + 11\mathrm{i}}{\mathrm{i} - 1} = 7 - 4\mathrm{i}.$$

This number is odd, so we have to find an $x_1 \in \{\pm 1, \pm\mathrm{i}\}$ such that $x_1 \equiv 7 - 4\mathrm{i} \pmod{2 + 2\mathrm{i}}$. We choose $x_1 = -1$ because $(7 - 4\mathrm{i}) - (-1) = 8 - 4\mathrm{i}$ is divisible by $2 + 2\mathrm{i}$. Now, we could divide $8 - 4\mathrm{i}$ by $\mathrm{i} - 1$ and continue. However, we know that the next two digits will be zeros, so we might as well write them immediately and divide by $2 + 2\mathrm{i}$ directly:

$$\frac{8 - 4\mathrm{i}}{2 + 2\mathrm{i}} = 1 - 3\mathrm{i}.$$

We see that this is an even number, so we write another 0 and divide by $i - 1$, yielding $-2 + \mathrm{i}$. This number is odd, so we need to find a digit which, when subtracted, gives a number divisible by $2 + 2\mathrm{i}$. Such a digit is $-\mathrm{i}$, giving $-2 + 2\mathrm{i}$. Again, we can write two zeros and directly divide by $2 + 2\mathrm{i}$, getting i. This is just a single digit, so we write it down and terminate, getting the result

$$-3 + 11\mathrm{i} = \left[\mathrm{i}00\bar{\mathrm{i}}000\bar{1}0\right]_{\mathrm{i}-1}.$$

Now that we have proven that the 3-NAF extended Penney system is usable for uniquely (up to leading zeros) representing Gaussian integers, the next step is to find out how it performs in terms of the Hamming weight. Like in Lemma 1.12, it can be proven that every possible extended-Penney representation can be converted into an equivalent 3-NAF representation using a set of substitutions that never increase the Hamming weight. However, this set consists of a total of 312 substitutions and requires temporarily switching to a larger set of digits. Therefore, we are going to avoid this method of proving the optimality of the representations and leave it for the next chapter, which is going to introduce a transducer for converting any representation to 3-NAF. What we can prove right now, however, is a result analogous to Theorem 1.14: the asymptotic behavior of the ratio of non-zero digits to all digits.

Theorem 2.5. Let a_n be the number of all (not necessarily reduced) 3-NAF extended Penney system representations of length n and b_n the sum of the Hamming weights of these representations. Then $\lim\limits_{n\to\infty} \frac{b_n}{na_n} = \frac{1}{4}$. That is, for an average sufficiently long 3-NAF extended Penney system representation, about $\frac{1}{4}$ of its digits are non-zeros.

Note. This result can be obtained as a special case of a much more general theorem shown in [2] (Theorem 5.1).

Proof. Let us find an explicit formula for a_n and b_n. We can think of 3-NAF extended Penney system representations as consisting of five kinds of "blocks": $\boxed{0}$, $\boxed{00\mathrm{i}}$, $\boxed{001}$, $\boxed{00\bar{1}}$ and $\boxed{0\bar{\mathrm{i}}}$. Note that these blocks can only be used to construct representations starting with two zeros, so we are actually expressing a_{n-2} and b_{n-2}, but this does not matter when $n \to \infty$. Analougously as in the proof of Theorem 1.14, we derive

$$a_n = \sum_{k=0}^{\infty} 4^k \binom{n-2k}{k},$$

$$b_n = \sum_{k=0}^{\infty} k4^k \binom{n-2k}{k}$$

and find a recurrence relation for a_n:

$$\begin{aligned}
a_n &= \sum_{k=0}^{\infty} 4^k \binom{n-2k}{k} \\
&= \sum_{k=0}^{\infty} 4^k \binom{n-2k-1}{k} + \sum_{k=0}^{\infty} 4^k \binom{n-2k-1}{k-1} \\
&= \sum_{k=0}^{\infty} 4^k \binom{n-2k-1}{k} + \sum_{k=-1}^{\infty} 4^{k+1} \binom{n-2k-3}{k} \\
&= \sum_{k=0}^{\infty} 4^k \binom{n-2k-1}{k} + 4\sum_{k=0}^{\infty} 4^k \binom{n-2k-3}{k} \\
&= a_{n-1} + 4a_{n-3}.
\end{aligned}$$

Solving this recurrence using its characteristic polynomial

$$\lambda^3 - \lambda^2 - 4 = (\lambda-2)\big(\lambda^2+\lambda+2\big) = (\lambda-2)(\lambda-\kappa)(\lambda-\bar{\kappa}),$$

where $\kappa = \frac{-1+\sqrt{7}\mathrm{i}}{2}$, we get

$$a_n = \alpha 2^n + \beta\kappa^n + \gamma\bar{\kappa}^n$$

for some $\alpha, \beta, \gamma \in \mathbb{C}$. We can verify that $\alpha \neq 0$ by manually computing the first three terms of the sequences and solving a set of linear equations. Finding a recurrence for b_n:

$$\begin{aligned}
b_n &= \sum_{k=0}^{\infty} k4^k \binom{n-2k}{k} \\
&= \sum_{k=0}^{\infty} k4^k \binom{n-2k-1}{k} + \sum_{k=0}^{\infty} k4^k \binom{n-2k-1}{k-1} \\
&= \sum_{k=0}^{\infty} k4^k \binom{n-2k-1}{k} + \sum_{k=-1}^{\infty} (k+1)4^{k+1} \binom{n-2k-3}{k} \\
&= \sum_{k=0}^{\infty} k4^k \binom{n-2k-1}{k} + 4\sum_{k=0}^{\infty} k4^k \binom{n-2k-3}{k} + 4\sum_{k=0}^{\infty} 4^k \binom{n-2k-3}{k} \\
&= b_{n-1} + 4b_{n-3} + 4a_{n-3} \\
&= b_{n-1} + 4b_{n-3} + 4\alpha 2^{n-3} + 4\beta\kappa^{n-3} + 4\gamma\bar{\kappa}^{n-3}
\end{aligned}$$

Homogenous solution:

$$b_n^{\mathrm{hom}} = \tilde{\alpha}2^n + \tilde{\beta}\kappa^n + \tilde{\gamma}\bar{\kappa}^n.$$

Particular solution:

$$b_n^{\mathrm{par}} = \mu n 2^n + \nu n\kappa^n + \omega n\bar{\kappa}^n.$$

It is evident from the proof of Theorem 1.14 that we only need to find the value of μ, since $|\kappa| = \sqrt{2} < 2$. Therefore, when substituting in the recurrence, we only consider terms with 2^n:

$$\mu n 2^n - \mu(n-1)2^{n-1} - 4\mu(n-3)2^{n-3} = 4\alpha 2^{n-3},$$

$$\mu\left(n - \frac{n-1}{2} - \frac{n-3}{2}\right) = \frac{\alpha}{2},$$
$$\mu = \frac{\alpha}{4} \neq 0.$$

Substituting into the limit:

$$\lim_{n\to\infty} \frac{b_n}{na_n} = \lim_{n\to\infty} \frac{\tilde{\alpha}2^n + \tilde{\beta}\kappa^n + \tilde{\gamma}\bar{\kappa}^n + \frac{\alpha}{4}n2^n + \nu n\kappa^n + \omega n\bar{\kappa}^n}{\alpha n2^n + \beta n\kappa^n + \gamma n\bar{\kappa}^n} = \frac{1}{4}.$$

■

Notice that this is more efficient in terms of the non-zero digits ratio than if we were to represent Gaussian integers by representing the real and imaginary parts separately using the binary NAF system, which would yield a ratio of $\frac{1}{3}$ as shown in Theorem 1.14.

It can be useful to know how many representations of a given Hamming weight a given number has. There exists a straightforward formula for this that can be generalized to other positional numeration systems, as shown in the following theorem.

Theorem 2.6. A Gaussian integer $x \in \mathbb{Z}[\mathrm{i}]$ has exactly $R(x,k)$ representations with a given Hamming weight $k \in \mathbb{N}_0$, where $R(x,k)$ is given by the following recursive formula:

$$R(x,k) = \begin{cases} 1, & \text{if } x = 0 \text{ and } k = 0, \\ 0, & \text{if } x = 0 \text{ and } k \neq 0, \\ 0, & \text{if } x \neq 0 \text{ and } k = 0, \\ R\left(\frac{x}{\mathrm{i}-1}, k\right), & \text{if } x \neq 0, k \neq 0 \text{ and } (\mathrm{i}-1) \mid x, \\ \sum_{d\in\{\pm1,\pm\mathrm{i}\}} R\left(\frac{x-d}{\mathrm{i}-1}, k-1\right), & \text{if } x \neq 0, k \neq 0 \text{ and } (\mathrm{i}-1) \nmid x. \end{cases}$$

Proof. The first three cases are trivial. If x is divisible by $\mathrm{i}-1$, any of its representations has to end in a zero, meaning that it is simply a representation of $\frac{x}{\mathrm{i}-1}$ with a zero appended. Otherwise, a representation has to end in one of the non-zero digits. Removing this digit d would result in a representation of $\frac{x-d}{\mathrm{i}-1}$, but with one fewer non-zero digit. Since all the possibilities are independent, we take the sum over all four non-zero digits. ■

2.2 Eisenstein integers

Just like we extended the Penney system to allow for NAF representations of Gaussian integers, we can extend the system for representing Eisenstein integers with $\beta = \omega - 1$, $\mathscr{D} = \{0, 1, \omega + 1\}$. By adding the digits $\omega, -1, \bar{\omega}, \bar{\omega} + 1$, we obtain a digit set that is not only closed under multiplication, but admits a unique 2-NAF representation for every Gaussian integer. The resulting system has many nice properties as well. In some ways it is better than the extended Penney system, despite only having 2-NAF and not 3-NAF representations for every number. However, these properties are outside the scope of this project and we are not going to discuss them any further, focusing only on the extended Penney system for Gaussian integers.

Chapter 3

Counting maximum optimal representations

We have seen in Theorem 1.13 that with $\beta = 2$, $\mathscr{D} = \{0, \pm 1\}$, the 2-NAF representation of a given number is always optimal. However, it might not be strictly optimal – that is, there might exist other representations of the same number with equal Hamming weight. For certain applications, it is important to know the number of such optimal representations. In [3], it is shown that a binary NAF representation of an integer x has at most $F_{\lfloor \log_4 |x| \rfloor + 3}$ reduced optimal representations, where $(F_n)_{n=0}^{\infty}$ denotes the Fibonacci sequence defined by the recurrence $F_0 = 0$, $F_1 = 1$, $F_{n+2} = F_n + F_{n+1}$. [4] gives a similar result, except in terms of the optimal weight rather than the number itself: a binary NAF representation $x_n \cdots x_0$ has at most $F_{W(x_n \cdots x_0)+1}$ equivalent reduced optimal representations (including itself). The latter proof makes use of a *transducer* that converts any $(2, \{0, \pm 1\})$-representation into the equivalent NAF representation.

In this chapter, we are going to prove a similar statement about 3-NAF extended Penney representations of Gaussian integers by constructing an analogous transducer. However, due to the larger complexity of the transducer, the proof is more complicated and uses a different method based on solving a non-trivial optimization problem with adjacency matrices. At the start, used a computer program (shown in Appendix A) to formulate a hypothesis about what the numbers with the most optimal representaions look like, then we formally confirmed this hypothesis.

We are going to work with the extended Penney system, so we shall use the symbol $\beta_{\mathrm{P}} := \mathrm{i} - 1$ for brevity.

3.1 Transducer

Definition 3.1. A *transducer* is a function $\delta : Q \times \mathscr{D} \to Q \times \mathscr{D}^*$, where Q is its set of *states* and $\mathscr{D}$ is a set of digits.

Given a transducer, a sequence $x_n \cdots x_0 \in \mathscr{D}^*$ and an *initial state* $q_0 \in Q$, we can use the transducer to transform the sequence as follows:

$$\begin{aligned}
(q_1, \eta_0) &:= \delta(q_0, x_0), \\
(q_2, \eta_1) &:= \delta(q_1, x_1), \\
&\vdots \\
(q_{n+1}, \eta_n) &:= \delta(q_n, x_n).
\end{aligned}$$

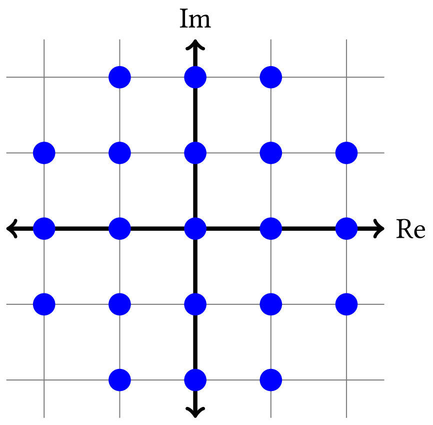

Figure 3.1: The set Q of all states of the transducer for converting any extended Penney representation into the equivalent 3-NAF representation.

Intuitively, we can think of the transducer as a machine that "consumes" the input from the right, one digit at a time, and based on the digit and its internal state, it outputs some digits and changes its internal state. In each step, we obtain a word η_k which forms a part of the output. By concatenating them, we get the result of the transformation: $\eta := \eta_n \cdots \eta_0$. We can define an extended version of the transducer, $\delta^* : Q \times \mathscr{D}^* \to Q \times \mathscr{D}^*$, which performs all the steps at once: $\delta^*(q_0, x_n \cdots x_0) := (q_{n+1}, \eta)$.

The state of our transducer is going to represent the difference between the numbers represented by the digits that have been read so far and the digits that have been written so far. The set of possible states, shown in Figure 3.1, shall be

$$Q = \{0,\ \pm 1,\ \pm\mathrm{i},\ \pm 1 \pm \mathrm{i},\ \pm 2,\ \pm 2\mathrm{i},\ \pm 1 \pm 2\mathrm{i},\ \pm 2 \pm \mathrm{i}\}.$$

We define the transducer itself as follows:

$$\begin{aligned} \delta(q, x) &:= (q', 0), && \text{if } \beta_{\mathrm{P}} \mid (x+q),\ q' = \frac{x+q}{\beta_{\mathrm{P}}}, \\ \delta(q, zyx) &:= (q', 00r), && \text{if } \beta_{\mathrm{P}} \nmid (x+q),\ [zyx]_{\beta_{\mathrm{P}}} + q = q'\beta_{\mathrm{P}}^3 + r,\ r \in \{\pm 1, \pm\mathrm{i}\}. \end{aligned}$$

Note that in the second case, the transducer consumes three digits at once, so it does not strictly satisfy Definition 3.1, but it can still be used to unambiguously define an extended transducer δ^*. We just need to verify that the definition is correct in terms of always producing a valid new state and valid output. The fact that we can always find q', r in the second case follows immediately from Lemma 2.3. It remains to show that always $q' \in Q$, which can be done by manually checking all finitely many inputs (which is made easier by the symmetry present).

Now we need to prove that the transducer δ_0 actually turns any representation into the equivalent 3-NAF representation, starting with $q_0 = 0$.

Lemma 3.2. Let $x_n \cdots x_0$ be a representation in the extended Penney system. Assume that the transducer δ^* described above reads all digits of the representation. Then its output $y_n \cdots y_0$ and the final state q_{n+1} will satisfy

$$[x_n \cdots x_0]_{\beta_{\mathrm{P}}} = q_{n+1}\beta_{\mathrm{P}}^{n+1} + [y_n \cdots y_0]_{\beta_{\mathrm{P}}}.$$

Proof. We shall prove the statement by induction. Given the empty representation, the transducer produces an empty representation and stays in the state $q_0 = 0$, which clearly satisfies the equation.

Now assume that the statement is true for representations of length n. As the induction step, we will show that if the transducer consumes n digits and then performs one more step, the statement will still hold. Naturally, we are going to distiguish two cases based on the definition of δ:

- If $\beta_P \mid (x_{n+1} + q_{n+1})$, the transducer is going to write $y_{n+1} = 0$ and set its state to $q_{n+2} = \frac{x_{n+1}+q_{n+1}}{\beta_P}$. We then have

$$\begin{aligned}[x_{n+1} \cdots x_0]_{\beta_P} &= x_{n+1}\beta_P^{n+1} + [x_n \cdots x_0]_{\beta_P} \\ &= x_{n+1}\beta_P^{n+1} + q_{n+1}\beta_P^{n+1} + [y_n \cdots y_0]_{\beta_P} \\ &= q_{n+2}\beta_P^{n+2} + 0\beta_P^{n+1} + [y_n \cdots y_0]_{\beta_P} \\ &= q_{n+2}\beta_P^{n+2} + [y_{n+1} \cdots y_0]_{\beta_P}.\end{aligned}$$

- If $\beta_P \nmid (x_{n+1} + q_{n+1})$, the transducer is going to read two more digits x_{n+2} and x_{n+3}, find $q' \in Q,\ r \in \{\pm 1, \pm \mathrm{i}\}$ such that $[x_{n+3}x_{n+2}x_{n+1}]_{\beta_P} + q_{n+1} = q'\beta_P^3 + r$, then output three digits $y_{n+1} = r,\ y_{n+2} = y_{n+3} = 0$ and set its state to $q_{n+4} = q'$. We then have

$$\begin{aligned}[x_{n+3} \cdots x_0]_{\beta_P} &= x_{n+3}\beta_P^{n+3} + x_{n+2}\beta_P^{n+2} + x_{n+1}\beta_P^{n+1} + [x_n \cdots x_0]_{\beta_P} \\ &= x_{n+3}\beta_P^{n+3} + x_{n+2}\beta_P^{n+2} + x_{n+1}\beta_P^{n+1} + q_{n+1}\beta_P^{n+1} + [y_n \cdots y_0]_{\beta_P} \\ &= (q'\beta_P^3 + r)\beta_P^{n+1} + [y_n \cdots y_0]_{\beta_P} \\ &= q_{n+4}\beta_P^{n+4} + [y_{n+3} \cdots y_0]_{\beta_P}.\end{aligned}$$

■

Note. It is possible that the transducer will fail to read the whole representation, being left with one or two digits that fall into the second case, which requires three digits. In this case, we can simply prepend up to two zeros to finish the transformation.

Theorem 3.3. Let $x_n \cdots x_0$ be a representation in the extended Penney system. Then it is possible to prepend finitely many zeros to the representation so that the transducer δ^* described above reads all digits, ends up with a final state $q_{l+1} = 0$ and outputs the equivalent 3-NAF representation.

Proof. By Lemma 3.2 and the note below it, we can prepend 0 to 2 zeros so that the transducer reads all digits, ends up in a state q_{m+1} and outputs a representation $y_m \cdots y_0$ satisfying

$$[x_m \cdots x_0]_{\beta_P} = q_{m+1}\beta_P^{m+1} + [y_m \cdots y_0]_{\beta_P}.$$

Notice that when the input to the transducer consists entirely of zeros, it is identical to the algorithm described in Theorem 2.4, finding the 3-NAF representation of q. Therefore, after consuming the original input, we can input a few more zeros into it to get it into the zero state, causing it to output the 3-NAF representation of q_{m+1} as $y_l \cdots y_{m+1}$. We can substitute this into the above equation:

$$[x_m \cdots x_0]_{\beta_P} = [y_l \cdots y_{m+1}]_{\beta_P}\beta_P^{m+1} + [y_m \cdots y_0]_{\beta_P} = [y_l \cdots y_0]_{\beta_P}.$$

Therefore, $y_l \cdots y_0$ is an equivalent representation. Since the transducer always outputs non-zero digits with two zeros before them, this representation is also 3-NAF. ■

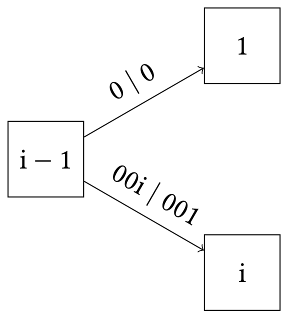

Figure 3.2: Example of edges and their labels in the graph G representing the transducer converting extended Penney representations into 3-NAF

3.2 Transducer as an oriented graph

We can naturally represent the transducer δ as an oriented graph G, where vertices represent the possible states Q and edges represent transitions. An edge corresponding to reading a digit x and outputting a zero will be labelled $x \mid 0$, one that reads three digits zyx and outputs three digits $00r$ will be labelled $zyx \mid 00r$. Since the graph has many edges, a picture of it would be unreadable, but for illustration, Figure 3.2 shows three nodes and two edges from this graph.

Definition 3.4. Let G be a graph with vertices Q and edges E. An *oriented walk* in G of *length l* is a sequence $(q_0, e_1, q_1, e_2, \dots, e_l, q_l)$, where $q_0 \cdots q_l \in Q$ and $e_k = (q_{k-1}, q_k) \in E$ for all $k \in \{1, \dots, l\}$.

Naturally, a transformation using the extended transducer δ^* corresponds to an oriented walk in the graph G of the original transducer, where $q_0 \cdots q_l$ are the visited states and $e_1 \cdots e_l$ are the performed transformations. Notice that since we are reading and writing the representations from right to left, this walk will be written in the “reverse” order in contrast with its input and output.

We are now ready to use the graph for analyzing the optimality of representations. For this, we need to introduce the notion of the weight of an edge, which shall indicate how many non-zero digits the transduces removes when performing the corresponding transition.

Definition 3.5. Let e be an edge of our graph G with label $x_m \cdots x_0 \mid y_m \cdots y_0$. Then we define its *weight* as $W(e) := W(x_m \cdots x_0) - W(y_m \cdots y_0)$.

Definition 3.6. Let $P = (q_0, e_1, \dots, e_l, q_l)$ be an oriented walk in G. Then its *weight* is defined as $W(P) := \sum_{k=1}^{l} W(e_k)$.

Note that some edges in G have a negative weight, so we cannot straight up say that no edge increases the Hamming weight and therefore the 3-NAF representation is optimal. However, it is the case that if we make a complete walk starting and ending in the state 0, then the sum of all edges on the walk is non-negative, as can be easily proven:

Lemma 3.7. Let P be a walk in our graph G starting and ending in 0. Then $W(P) \geq 0$.

Proof. We can use the Bellman-Ford algorithm [14] to find the minimum-weight walk from 0 to itself. The algorithm indicates that the shortest walk has length 0. ■

Theorem 3.8. Every 3-NAF representation $y_n \cdots y_0$ in the extended Penney system is optimal.

Note. This result is proven much more generally with completely different machinery in [1].

Proof. Let $x_m \cdots x_0$ be another representation of the same number and $(q_0, e_1 \cdots e_l, q_l)$ the walk taken in G when transforming $x_m \cdots x_0$ into its 3-NAF representation using δ^*. From Theorem 2.4, we know that the output is equal up to leading zeros to $y_n \cdots y_0$. Using this and Lemma 3.7, we have

$$W(x_n \cdots x_0) - W(y_m \cdots y_0) = \sum_{k=1}^{l} W(e_k) \geq 0.$$

Therefore, an arbitrary equivalent representation has a greater or equal weight than the 3-NAF representation, which was to be proven. ∎

Lemma 3.7 and the proof of Theorem 3.8 motivate the following definition and trivial lemma:

Definition 3.9. A walk P in G is *optimal* if $W(P) = 0$.

Lemma 3.10. An extended Penney representation $x_n \cdots x_0$ is optimal if and only if the walk in G produced by using the transducer δ^* on $x_n \cdots x_0$ is optimal.

3.3 Simplifying the graph

Although the graph G has many edges, we can exploit the symmetries present in the problem in order to simplify it.

Lemma 3.11. Let $e = (q, q') \in E$ be an edge in the graph G of the transducer δ described above and $d \in \{\pm 1, \pm \mathrm{i}\}$. Then

- if the label of e is $x \mid 0$, then G also contains the edges $de : dq \xrightarrow{(d \cdot x) | 0} dq'$ and $\overline{e} : \overline{q} \xrightarrow{\overline{x} | 0} \mathrm{i}\overline{q'}$,
- if the label of e is $zyx \mid 00r$, then G also contains the edges $de : dq \xrightarrow{(d \cdot z)(d \cdot y)(d \cdot x) | 00(d \cdot r)} dq'$ and $\overline{e} : \overline{q} \xrightarrow{(-\overline{z})(\mathrm{i} \cdot \overline{y})\overline{x} | 00\overline{r}} -\mathrm{i}\overline{q'}$.

Proof. Notice that $\overline{\beta_{\mathrm{P}}} = \mathrm{i}\beta_{\mathrm{P}}$, $\overline{\beta_{\mathrm{P}}^2} = -\beta_{\mathrm{P}}^2$ and $\overline{\beta_{\mathrm{P}}^3} = -\mathrm{i}\beta_{\mathrm{P}}^3$.

- If e is labelled $x \mid 0$, it means that $x + q = q' \beta_{\mathrm{P}}$. Then also $dx + dq = dq' \beta_{\mathrm{P}}$ and $\overline{x} + \overline{q} = \overline{q'}\,\overline{\beta_{\mathrm{P}}} = \mathrm{i}\overline{q'} \beta_{\mathrm{P}}$.
- If e is labelled $zyx \mid 00r$, it means that

  $$z\beta_{\mathrm{P}}^2 + y\beta_{\mathrm{P}} + x + q = q' \beta_{\mathrm{P}}^3 + r.$$

  Then also

  $$dz\beta_{\mathrm{P}}^2 + dy\beta_{\mathrm{P}} + dx + dq = dq' \beta_{\mathrm{P}}^3 + dr$$

  and

  $$\begin{aligned} \overline{z}\,\overline{\beta_{\mathrm{P}}^2} + \overline{y}\,\overline{\beta_{\mathrm{P}}} + \overline{x} + \overline{q} &= \overline{q'}\,\overline{\beta_{\mathrm{P}}^3} + \overline{r} \\ -\overline{z}\beta_{\mathrm{P}}^2 + \mathrm{i}\overline{y}\beta_{\mathrm{P}} + \overline{x} + \overline{q} &= -\mathrm{i}\overline{q'} \beta_{\mathrm{P}}^3 + \overline{r}. \end{aligned}$$

∎

The symmetry demonstrated in Lemma 3.11 allows us to introduce an equivalence relation $\sim$ on the set of states Q where $q_1 \sim q_2$ if $q_1 = dq_2$ or $q_1 = d\overline{q_2}$ for some $d \in \{\pm 1, \pm \mathrm{i}\}$. We can then group the states into equivalence groups:

$$\begin{aligned}
[0] &= \{0\},\\
[1] &= \{\pm 1, \pm \mathrm{i}\},\\
[\mathrm{i}+1] &= \{\pm 1 \pm \mathrm{i}\},\\
[2] &= \{\pm 2, \pm 2\mathrm{i}\},\\
[\mathrm{i}+2] &= \{\pm 2 \pm \mathrm{i}, \pm 1 \pm 2\mathrm{i}\}.
\end{aligned}$$

Lemma 3.12. Let P be a walk in our graph G starting in q_0 and ending in q_l. Let $q_0' \in Q$, $q_0' \sim q_0$. Then there exists a walk P' of the same length and weight as P which starts in q_0' and ends in some $q_l' \in Q$, $q_l' \sim q_l$.

Proof. By definition of $\sim$, we can transform q_0 into q_0' by multiplication with a number from $\{\pm 1, \pm \mathrm{i}\}$ and possibly complex conjugation. By Lemma 3.11, there is an edge from q_0' to q_1', where $q_1' \sim q_1$. We can repeat this argument for all edges on P. It is also easy to check that the equivalent edges have the same weights. ■

Lemma 3.13. Let e be an edge from q to q' that is contained in some optimal walk P. Then e has the minimum weight out of all edges from q to q'.

Proof. Assume for the sake of contradiction that there exists an edge e' from q to q' with $W(e') < W(e)$. Then we could replace e with e' in P and obtain a walk from 0 to 0 with a negative weight. However, according to Lemma 3.7, this is impossible. ■

These lemmas allow us to construct a much simpler graph Γ that can still be used to analyze the optimality of representations. Its vertices will be the equivalence classes of Q. By Lemma 3.11, we can group edges in G that differ only in symmetry into one edge in Γ. By Lemma 3.12, the weights of edges and walks are still going to be unambiguously defined, so we can label the edges with the common weight of all corresponding original edges, though we lose the information about the specific inputs and outputs. Lemma 3.13 also allows us to discard edges that cannot be used in an optimal walk. The result is shown in Figure 3.3.

If we use the Bellman-Ford algorithm on this new graph to calculate the minimum-weight walk between each pair of vertices, we will notice that some edges are not the minimum-weight walk between their start and end. Such edges cannot lie on any optimal walk, because if they did, we could replace them with the shorter walk, similarly to the proof of Lemma 3.13. Additionally, the vertices 2 and $\mathrm{i}+2$ have the property that the minimum-weight walk from 0 to either of them and then back to 0 has a positive weight. Therefore, no optimal walk can go through these vertices. If we remove the problematic vertices and edges, we get the graph $\tilde{\Gamma}$ depicted in Figure 3.4, which has only 3 vertices and 7 edges.

By collapsing the vertices of G into equivalence classes, we have lost information about the outputs of the edges, since they are not identical for edges from a given equivelence class to a given equivalence class. We are going to need this information, so we shall introduce yet another graph $\tilde{G}$, a subgraph of G consisting only of the edges that are represented in $\tilde{\Gamma}$. This graph, shown in Figure 3.5, is still quite small, with 9 vertices and 61 edges.

Lemma 3.14. All walks in $\tilde{\Gamma}$, as well as $\tilde{G}$, are optimal.

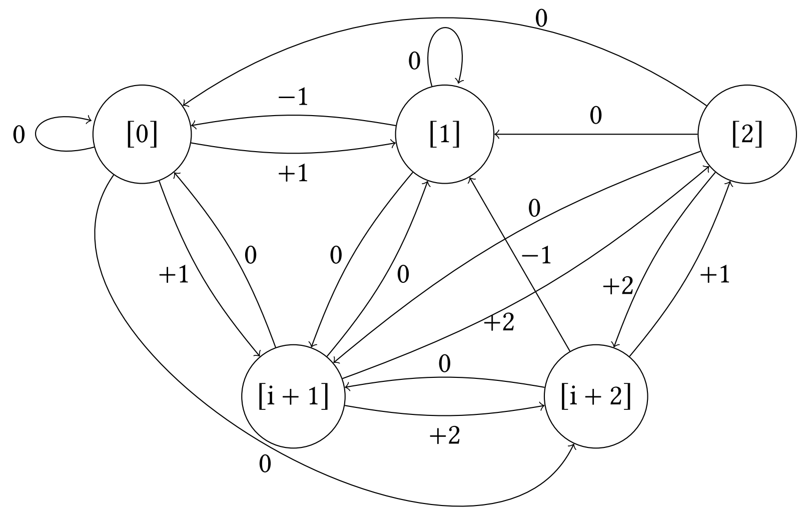

Figure 3.3: The graph Γ.

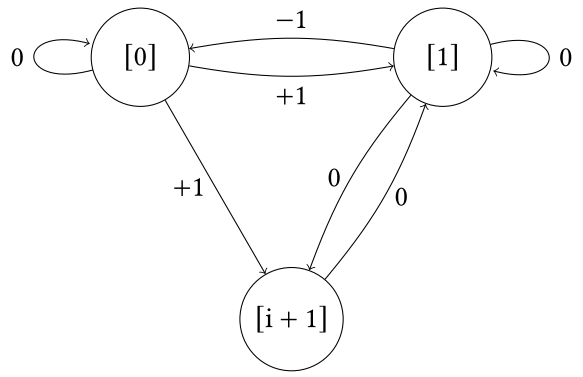

Figure 3.4: The graph $\tilde{\Gamma}$, a subgraph of Γ limited to vertices and edges that can appear in an optimal walk, with weights of its edges.

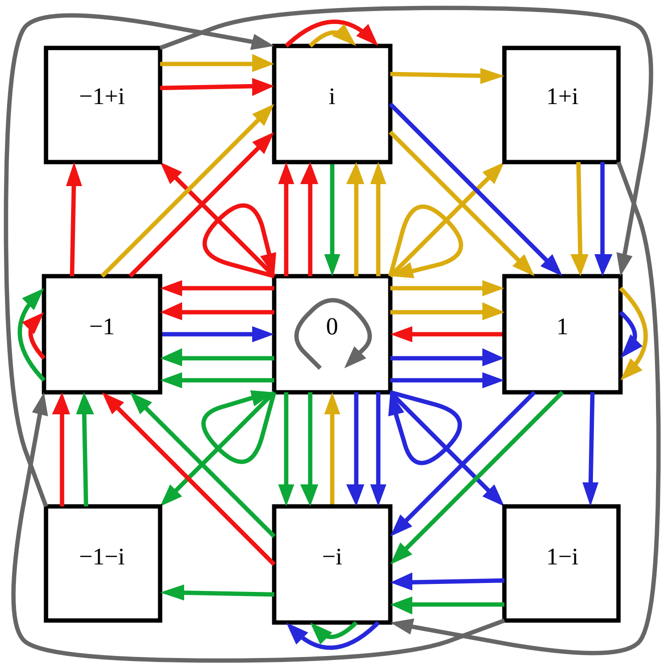

Figure 3.5: The graph $\tilde{G}$, a subgraph of G limited to edges that can appear in an optimal walk. The colors of edges correspond to their output labels: gray $\mapsto$ 0, red $\mapsto$ 001, green $\mapsto$ 00i, blue $\mapsto$ $00\bar{1}$, yellow $\mapsto$ $00\bar{\mathrm{i}}$. To reduce clutter, input labels are not shown.

Proof. Consider first $\tilde{\Gamma}$. The only edges that have a non-zero weight are the ones between [0] and the other two vertices, with those from [0] having +1 and the one to [0] having −1. Any walk that starts and ends in [0] uses an equal number of edges to and from [0], so the sum of weights is 0. Since the edges in $\tilde{G}$ have the same weights as the corresponding edges in $\tilde{\Gamma}$, this argument also applies to $\tilde{G}$. ■

Theorem 3.15. Let $x \in Z[\mathrm{i}]$. Then the number of optimal reduced representations of x is equal to the number of walks in $\tilde{G}$ whose output is the 3-NAF representation of x.

Proof. A direct consequence of Lemma 3.10, Lemma 3.14 and the fact that $\tilde{G}$ is a subgraph of G containing all optimal walks. ■

3.4 Converting to a matrix problem

In Theorem 3.15, we have proven that the graph $\tilde{G}$ is a good tool for counting optimal extended Penney representations. We still need a way to count all the possible walks in $\tilde{G}$. The standard graph-theoretic way to count walks is using adjacency matrices. However, we do not actually want to count all walks, just the ones with a specific output. We can achieve this by defining a separate adjacency matrix for each subgraph consisting only of edges with the same output.

First, we need to put the vertices in a specific order, represented by a tuple of the vertex labels:

$$V := (0, \quad 1, \mathrm{i}, -1, -\mathrm{i}, \quad 1+\mathrm{i}, -1+\mathrm{i}, -1-\mathrm{i}, 1-\mathrm{i}).$$

We also assign each possible output of an edge to a single digit in the natural way:

$$O_0 := 0,\ O_d := 00d,\ d \in \{\pm 1, \pm \mathrm{i}\}.$$

Then, for each $d \in \{0, \pm 1, \pm \mathrm{i}\}$, we define the adjacency matrix $A_d \in \mathbb{N}_0^{9\times 9}$ like so:

$$(A_d)_{i,j} := \text{the number of edges in } \tilde{G} \text{ from } V_i \text{ to } V_j \text{ whose output is } O_d.$$

The matrices A_0 and A_1 look like this:

$$A_0 = \left(\begin{array}{c|cccc|cccc}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ \hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}\right), \qquad
A_1 = \left(\begin{array}{c|cccc|cccc}
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\
2 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}\right).$$

The remaining three matrices $A_\mathrm{i}, A_{-1}, A_{-\mathrm{i}}$ can be expressed in terms of A_1 using the symmetries described in Lemma 3.11. To be specific, we define matrices $R, C \in \mathbb{N}_0^{9\times 9}$ as follows:

$$R := \left(\begin{array}{c|cccc|cccc}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0
\end{array}\right), \qquad
C := \left(\begin{array}{c|cccc|cccc}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0
\end{array}\right).$$

These matrices have the following properties, which can be verified by direct computation:

$$C = C^\mathrm{T}, \qquad R^\mathrm{T} = R^{-1}, \qquad C = C^{-1}, \qquad RC = CR^\mathrm{T},$$

Due to these properties, the matrices form a multiplicative group with 8 elements:

$$\mathscr{P} := \langle R, C\rangle = \{I, R, R^2, R^3, C, CR, CR^2, CR^3\}.$$

The properties in the following lemma establish relationships between the adjacency matrices:

Lemma 3.16.

$$\begin{aligned}
A_{\mathrm{i}\cdot d} &= R^{-1} A_d R \text{ for all } d \in \{0, \pm 1, \pm \mathrm{i}\}, \\
A_{\overline{d}} &= RCA_dC \text{ for all } d \in \{\pm 1, \pm \mathrm{i}\}, \\
A_0 &= CRA_0C.
\end{aligned}$$

Proof. The relationships follow from Lemma 3.11 and the way the graph and matrices are constructed, or they can be easily verified by direct computation. ■

Lemma 3.17. Let $d \in \{\pm 1, \pm \mathrm{i}\}$ and $P \in \mathscr{P}$. Then there exist $h \in \{\pm 1, \pm \mathrm{i}\}$ and $S, T \in \mathscr{P}$ such that $PA_d = A_h S$ and $PA_0 = A_0 T$.

Proof. If we express $d =: \mathrm{i}^n,\ n \in \{0,1,2,3\}$, then by Lemma 3.16, $A_d = R^{-n}A_1R^n$.

- If $P = R^p,\ p \in \{0,1,2,3\}$, we choose an $m \in \{0,1,2,3\}$ such that $m \equiv n - p \pmod 4$ and $h := \mathrm{i}^m,\ S := R^p,\ T := R^p$. Then

$$PA_d = R^pR^{-n}A_1R^n = R^{-m}A_1R^mR^p = A_hS,$$

$$PA_0 = R^pR^{-p}A_0R^p = A_0T.$$

- If $P = CR^p,\ p \in \{0,1,2,3\}$, we choose an $m \in \{0,1,2,3\}$ such that $n - p - m \equiv 1 \pmod 4$ and $h := \mathrm{i}^{-m},\ S := CR^{p+1},\ T := CR^{p-1}$. Then, using all the formulas in Lemma 3.16,

$$PA_d = CR^pR^{-n}A_1R^n = CR^pR^{-n}R^mA_{\overline{h}}R^{-m}R^n = R^{n-p-m}CA_{\overline{h}}CCR^{n-m} = A_hS,$$

$$PA_0 = CR^pR^{1-p}A_0CCR^{p-1} = A_0T.$$

■

Lemma 3.18. Let $d_1, \dots, d_l$ be digits. Then the number of walks $(q_0, e_1, \dots, e_l, q_l)$ in $\tilde{G}$ starting at V_i and ending at V_j such that the output label of each e_k is O_{d_k} is $\left(A_{d_1} \cdots A_{d_l}\right)_{i,j}$.

Note. This is a special case of a well-known general result from graph theory.

Proof. We shall prove the statement by induction. The case $l = 1$ follows directly from the definition of the adjacency matrices. Assume that the statement is true for l and let V_i, V_j be some vertices. Then for each vertex V_k, there are $\left(A_{d_1} \cdots A_{d_l}\right)_{i,k}$ walks from V_i to V_k with l edges and $\left(A_{d_{l+1}}\right)_{k,j}$ edges from V_k to V_j. Each walk from V_i to V_j with $l+1$ consists of one said walk and one said edge, where V_k can be arbitrary. Therefore, the total number of such walks is

$$\sum_{k=1}^{9} \left(A_{d_1} \cdots A_{d_l}\right)_{i,k}\left(A_{d_{l+1}}\right)_{k,j} = \left(A_{d_1} \cdots A_{d_{l+1}}\right)_{i,j}.$$

■

Theorem 3.19. Let $O_{d_1} \cdots O_{d_n}$ be the 3-NAF extended Penney representation of $x \in \mathbb{Z}[\mathrm{i}]$. Then the number of optimal reduced extended Penney representations of x is equal to $\left(A_{d_1} \cdots A_{d_n}\right)_{1,1} = eA_{d_1} \cdots A_{d_n}e^{\mathrm{T}}$, where $e = I_{1,\cdot}$ is the first row unit vector.

Proof. Follows directly from Theorem 3.15 and Lemma 3.18. ■

At last, we have converted our problem to a matrix problem, making it easier to reason about. Our ultimate question is: Given a number $N \in \mathbb{N}$, which 3-NAF representation with Hamming weight N has the most equivalent representations? And how many such 3-NAF representations exist? We can formulate the first question using matrices as follows:

$$M(N) := \max\left\{eA_{d_1} \cdots A_{d_n}e^{\mathrm{T}} \,\middle|\, d_k \in \{0, \pm 1, \pm \mathrm{i}\},\ \sum_{k=1}^{n} |d_k| = N\right\} = ?$$

3.5 Solving the matrix problem

Definition 3.20. Let $u, v \in \mathbb{Z}^9$ be row vectors. We define the relation $\leq$ as

$$u \leq v \iff u_i \leq v_i \text{ for all } i \in \{1, \dots, 9\}.$$

Definition 3.21. Let $u, v \in \mathbb{Z}^9$ be row vectors. We define the relation $\preceq$ as

$$u \preceq v \iff uP \leq v \text{ for some } P \in \mathscr{P}.$$

We say that u *is majorized by* v. If also $v \preceq u$, we denote $u \sim v$.

Definition 3.22. Let $u, v \in \mathbb{Z}^9$ be row vectors. We define the relation $\prec$ as

$$u \prec v \iff u \preceq v \text{ and } u_1 < v_1.$$

We say that u *is strictly majorized by* v. If also $v \preceq u$, we denote $u \sim v$.

Lemma 3.23. $\sim$ is an equivalence relation on $\mathbb{Z}^9$ and $\preceq$ is a partial ordering on $\mathbb{Z}^9/\sim$.

Proof. We will show that $\preceq$ is transitive; all other properties trivially follow from the definitions. Let u, v, w be vectors such that $u \preceq v$ and $v \preceq w$. Then there exist matrices $P, Q \in \mathscr{P}$ such that $uP \leq v$ and $vQ \leq w$. Then also $uPQ \leq w$, with $PQ \in \mathscr{P}$ since $\mathscr{P}$ is closed under multiplication. ∎

Lemma 3.24. Let $d_1 \cdots d_l, f_1 \cdots f_m \in \{0, \pm 1, \pm \mathrm{i}\}^*$ be words such that $\sum_{k=1}^{l} |d_k| = \sum_{k=1}^{m} |f_k|$ and $eA_{d_1} \cdots A_{d_l} \prec eA_{f_1} \cdots A_{f_m}$. Then for any $d_{l+1} \cdots d_n \in \{0, \pm 1, \pm \mathrm{i}\}$ we have

$$eA_{d_1} \cdots A_{d_n} e^{\mathrm{T}} < M(N), \quad N := \sum_{k=1}^{n} |d_k|.$$

Proof. Let $u := eA_{d_1} \cdots A_{d_l}$, $v := eA_{f_1} \cdots A_{f_m}$ and $w := A_{d_{l+1}} \cdots A_{d_n} e^{\mathrm{T}}$. Since $(A_d)_{1,1} = 1$ for all d, the first component of all the vectors is positive: $u_1 > 0,\ v_1 > 0,\ w_1 > 0$. Let $P \in \mathscr{P}$ be a matrix such that $uP \leq v$. Then

$$eA_{d_1} \cdots A_{d_n} e^{\mathrm{T}} = uw = uPP^{-1}w < vP^{-1}w.$$

Now it remains to show that $vP^{-1}w \leq M(N)$. By repeated application of Lemma 3.17, there exist $h_{l+1} \cdots h_n \in \{\pm 1, \pm \mathrm{i}\}$, $S \in \mathscr{P}$ such that $P^{-1}w = A_{h_{l+1}} \cdots A_{h_n} S e^{\mathrm{T}}$, with each $|h_k| = |d_k|$. Also, $Se^{\mathrm{T}} = e^{\mathrm{T}}$ because all matrices in $\mathscr{P}$ have e^{T} as the first column. Therefore, by definition of $M(N)$,

$$vP^{-1}w = eA_{f_1} \cdots A_{f_m} A_{h_{l+1}} \cdots A_{h_n} e^{\mathrm{T}} \leq M(N).$$

∎

Lemma 3.25. Let $d_1 \cdots d_n$ be digits such that $N := \sum_{k=1}^{n} |d_k| \in \{2, 3, 4\}$. Let $u := eA_{d_1} \cdots A_{d_n}$. Denote

$$\begin{aligned}
B_2 &:= A_1 A_{-1}, & v_2 &:= eB_2 = (3, 1, 1, 1, 0, 1, 0, 0, 0), \\
B_3 &:= A_1 A_{-1} A_{-\mathrm{i}}, & v_3 &:= eB_3 = (8, 1, 3, 1, 3, 1, 1, 0, 0), \\
B_4 &:= A_1 A_{-1} A_{-\mathrm{i}} A_{-\mathrm{i}}, & v_4 &:= eB_4 = (17, 1, 5, 3, 8, 1, 3, 0, 0).
\end{aligned}$$

Then either $u \prec v_N$ or the following three statements hold:

- $u \sim v_N$,

- $n = N$,
- $A_{d_1} \cdots A_{d_n} = S^{\mathrm{T}} B_N P$ for some $P, S \in \mathscr{P}$.

Proof. Notice that $B_0^2 = e^{\mathrm{T}} e$, so $wB_0^2 \leq wB_0$ for any vector w, and also $eA_0 = e$. Therefore, we only need to check vectors u that do not contain B_0^2 or start with A_0. In other words,

$$u = eA_{f_1} A_0^{l_1} A_{f_2} \cdots A_{f_N} A_0^{l_N}, \; f_k \in \{\pm 1, \pm \mathrm{i}\}, \; l_k \in \{0, 1\}.$$

There are only finitely many such vectors for $N \in \{2, 3, 4\}$, so we can verify the theorem by direct computation. ■

Definition 3.26. We define the recurrent sequence of integers

$$r_{-1} := 3, \; r_0 := 8, \; r_1 := 17, \; r_{N+3} := r_{N+2} + 2r_{N+1} + 2r_N$$

and the recurrent sequence of vectors

$$\begin{aligned} t_0 &:= eA_1 A_{-1} A_{-\mathrm{i}} A_{-\mathrm{i}} R^3 = (17, 5, 3, 8, 1, 3, 0, 0, 1), \\ t_{N+1} &:= t_N A_1 R^2. \end{aligned}$$

Note. Obviously, r_N is a strictly increasing sequence.

Lemma 3.27. For each $N \in \mathbb{N}^+$,

$$t_N = (r_{N+1}, \; r_{N-2} + r_{N-1}, \; r_{N-1}, \; r_N, \; r_{N-2}, \; r_{N-1}, \; 0, \; 0, \; r_{N-2}).$$

Proof. Straightforward proof by induction. ■

Lemma 3.28. Let $N, l \in \mathbb{N}^+$, $d \in \{\pm 1, \pm \mathrm{i}\}$. Then

1. $t_N A_1 \sim t_{N+1}$,
2. $t_N A_{-1} \prec t_{N+1}$ and $t_N A_{\mathrm{i}} \prec t_{N+1}$,
3. $t_N A_{-\mathrm{i}} A_d \prec t_{N+2}$,
4. $t_N A_0^l A_d \prec t_{N+1}$.

Proof.

1. $t_N A_1 \sim t_N A_1 R^2 = t_{N+1}$.
2. By expressing each component in terms of the sequence r_N (using Lemma 3.27), we obtain $t_N A_{-1} C = t_N A_{\mathrm{i}} RC \leq t_{N+1}$, with the inequality being strict in the first component.
3. In a similar way, we can verify $t_N A_{-\mathrm{i}} A_1 C R^2 \leq t_N A_{-\mathrm{i}} A_{-\mathrm{i}} R^3$ and $t_N A_{-\mathrm{i}} A_{\mathrm{i}} R \leq t_N A_{-\mathrm{i}} A_{-1} C$. Therefore, we just need to show that $t_N A_{-\mathrm{i}} A_{-\mathrm{i}} R^3 \leq t_{N+2}$ and $t_N A_{-\mathrm{i}} A_{-1} C \leq t_{N+2}$. We shall show this by induction. For $N \in \{1, 2, 3\}$, the inequalities can be verified manually. Now assume that either of the inequalities is true for N, $N+1$ and $N+2$. By taking these three inequalities, multiplying the first two by 2 and adding them together with the third one, we get the inequality for $N+3$, which completes the induction step.

4. If $l \geq 2$, then $t_N A_0^l A_d = r_{N+1}e \leq t_{N+2}$ (due to the fact that $B_0^2 = e^{\mathrm{T}}e$). If $l = 1$, then we can again express each component in terms of the sequence r_N and verify $t_n A_0 A_d \prec t_{N+1}$ for each d individually.

■

Lemma 3.29. For each $N \in \mathbb{N}^+$, $M(N+4) = t_N e^{\mathrm{T}} = r_{N+1}$.

Proof. From Lemma 3.24 and Lemma 3.25, it follows that we only need to consider products of the form $t_0 A_{d_1} A_{d_2} \cdots A_{d_n} e^{\mathrm{T}} =: t_0 \Pi e^{\mathrm{T}}$, since swapping out t_0 for anything else with the same weight would not increase the result. We consider $\sum_{k=1}^{n} |d_k| = N$ because t_0 already contains 4 non-zero digits. By Lemma 3.16, we can rewrite Π as a product of matrices from the set A_0, A_1, R. Denote

$$\begin{aligned} \mathscr{B} &:= \{B_1 \cdots B_m \,|\, m \in \mathbb{N},\ B_k \in \{A_0, A_1, R\}\}, \\ \mathscr{B}_0 &:= \{B_1 \cdots B_m \,|\, m \in \mathbb{N},\ B_k \in \{A_0, R\}\}. \end{aligned}$$

Let l be the maximum index such that $\Pi = (A_1 R)^l B,\ B \in \mathscr{B}$. That is, $t_0 \Pi = t_l B$. Clearly, B contains exactly $N - l$ A_1 matrices. If $l = N$, then $B \in \mathscr{B}_0$, therefore $t_0 \Pi e^{\mathrm{T}} = t_N B e^{\mathrm{T}} = t_N e^{\mathrm{T}} = r_{N+1}$ (by Lemma 3.27). This shows a lower bound $M(N+4) \geq r_{N+1}$. It remains to show that if $l < N$, then this bound is not exceeded, that is, $t_0 \Pi e^{\mathrm{T}} < r_{N+1}$. Due to how we chose l, B cannot be of the form $A_1 \tilde{B},\ \tilde{B} \in \mathscr{B}$, because then we could have chosen a higher l. We shall consider several different cases, one of which has to happen:

$B = R^j A_1 \tilde{B},\ j \in \{2,3\},\ \tilde{B} \in \mathscr{B}$
: Then $t_l R^j A_1 \sim t_l A_{\bar{1}^{-j}} \prec t_{l+1}$ (by Lemma 3.16 and Lemma 3.28). Therefore, the maximum cannot be reached by Lemma 3.24.

$B = R A_1 R^j A_1 \tilde{B},\ j \in \{0,1,2,3\},\ \tilde{B} \in \mathscr{B}$
: Then $t_l R A_1 R^j A_1 = t_l A_{-\bar{1}} R^{j+1} A_1 \sim t_l A_{-\bar{1}} A_{\bar{1}^{-j-1}} \prec t_{l+2}$ (by Lemma 3.16 and Lemma 3.28). Therefore, the maximum cannot be reached by Lemma 3.24.

$B = R A_1 R^j A_0^k A_1 \tilde{B},\ k \in \mathbb{N}^+,\ j \in \{0,1,2,3\},\ \tilde{B} \in \mathscr{B}$
: It can be verified that $A_0 A_1$ is component-wise less-or-equal to A_1, therefore $t_l R A_1 R^j A_0^k A_1 < t_l R A_1 R^j A_1$, so this is reduced to the previous case.

$B = R A_1 \tilde{B},\ \tilde{B} \in \mathscr{B}_0$
: This implies that $l = N - 1$ and $e\Pi e^{\mathrm{T}} = t_l R A_1 \tilde{B} e^{\mathrm{T}} = t_l R A_1 e^{\mathrm{T}}$. For $N \leq 3$, we can manually check that $e\Pi e^{\mathrm{T}} = t_{N-1} R A_1 e^{\mathrm{T}} < r_{N+1}$. For $N \geq 4$, we can express t_{N-1} in terms of r_N using Lemma 3.27, then compute that $e\Pi e^{\mathrm{T}} = t_{N-1} R A_1 e^{\mathrm{T}} = r_N + 5r_{N-2} + 2r_{N-3}$. It remains to prove that this is less than r_{N+1}.

$$\begin{aligned} r_{N+1} &= r_N + 2r_{N-1} + 2r_{N-2} \\ &= r_N + 4r_{N-2} + 4r_{N-3} + 4r_{N-4} \\ &= r_N + 5r_{N-2} + 3r_{N-3} + 2r_{N-4} - 2r_{N-5} \\ &> r_N + 5r_{N-2} + 2r_{N-3}. \end{aligned}$$

$B = R A_0 \tilde{B},\ \tilde{B} \in \mathscr{B}$
: We assumed that $l < N$, so B contains at least one A_1 matrix. That is, there exists a $k \in \mathbb{N}^+$ and $j \in \{0,1,2,3\}$ such that $B = A_0^k R^m A_1 \tilde{B},\ \tilde{B} \in \mathscr{B}$ (making use of Lemma 3.16 to separate the A_0 matrices and R matrices). From Lemma 3.28, $t_l A_0^k R^m A_1 \prec t_{l+1}$ and therefore, by Lemma 3.24, it cannot start a product reaching the maximum. ■

Theorem 3.30. Let $N \in \mathbb{N}$, $N \geq 2$. Then each 3-NAF representation in the extended Penney system with exactly N non-zero digits has at most r_{N-3} equivalent representations, and for $N \geq 4$, there are exactly 16 such 3-NAF representations which have exactly r_{N-3} equivalent representations and do not end in 0, namely the first $3N-2$ digits of the infinite words

$$\begin{gathered}
d \cdot \Big(100\bar{\mathrm{i}}00\bar{\mathrm{i}}\big(00100\bar{\mathrm{i}}00\bar{1}00\mathrm{i}\big)^{\omega}\Big), \\
d \cdot \Big(100\bar{1}00\bar{\mathrm{i}}\big(00100\bar{\mathrm{i}}00\bar{1}00\mathrm{i}\big)^{\omega}\Big), \\
d \cdot \Big(100\bar{\mathrm{i}}00\bar{\mathrm{i}}\big(00\bar{\mathrm{i}}00\mathrm{i}\big)^{\omega}\Big), \\
d \cdot \Big(100\bar{1}00\bar{\mathrm{i}}\big(00\bar{\mathrm{i}}00\mathrm{i}\big)^{\omega}\Big),
\end{gathered}$$

for all $d \in \{\pm 1, \pm \mathrm{i}\}$.

Proof. For $N \leq 4$, the second statement follows from Lemma 3.25: The only vectors corresponding to 3-NAF representations with the maximum number of equivalent representations are $v_N P$ with $P \in \mathscr{P}$, giving a total of 8 vectors, which are all distinct. For $n = 4$, it can be verified that each of the vectors can be expressed in exactly two different ways as $eA_{d_1}A_{d_2}A_{d_3}A_{d_4}$ for $d_1, d_2, d_3, d_4 \in \{\pm 1, \pm \mathrm{i}\}$. The maximum number of equivalent representations can then be calculated using Theorem 2.6. For $N > 4$, Lemma 3.29 immediately gives the first statement. Its proof also shows that the vector of any representation with exactly r_{N-3} equivalent representations is of the form $eA_{d_1} \cdots A_{d_n} \sim t_N B,\ B \in \mathscr{N}_0$. Since we are only counting representations that do not end in 0, it follows that B consists only of R matrices, therefore $eA_{d_1} \cdots A_{d_n} \sim t_N$. By definition, there are 8 such vectors, which are distinct due to Lemma 3.27. Each of them corresponds to precisely two representations, because after choosing the first four non-zero digits, the rest are uniquely determined. It can be easily verified that said representations are those listed in the statement of the theorem. ■

Chapter 4

Counting average optimal representations

In Chapter 3, we calculated the maximum number of optimal extended Penney system representations a Gaussian integer can have if its 3-NAF representation contains a given number of non-zero digits. In this chapter, we shall give an estimate for the average number of optimal representations across all Gaussian integers with a 3-NAF representation of a given length, using the machinery developed in Chapter 3.

An analogous result for the signed binary system was shown in [3], using a measure of the interval $[-1, 1]$ which encodes optimal representations. However, it gives a more precise result. Specifically, it determines the average number of optimal representations across all natural numbers up to an arbitrary given limit for the number itself, whereas our result can only average across all numbers with a given number of digits. Since the complex numbers are not ordered, it is not clear what such a more precise estimate could look like; this is a possible subject for future research. We also use a different approach, which is based on the transducer shown in Chapter 3 rather than measure theory.

Definition 4.1. For a given $x \in \mathbb{Z}[\mathrm{i}]$, let $A(x)$ denote the matrix representing the paths in the graph $\tilde{G}$ corresponding to x, as seen in Chapter 3. That is,

$$A(x) \coloneqq A_{d_1} \cdots A_{d_m},$$

where the 3-NAF representation of x is $O_{d_1} \cdots O_{d_k}$.

Definition 4.2. Let $\mathscr{M}_{=N}$ denote the set of Gaussian integers whose reduced 3-NAF representation consists of exactly N digits and $\mathscr{M}_{\leq N}$ the set of Gaussian integers whose 3-NAF representation consists of at most N digits. Further we shall denote

$$p_0 \coloneqq e^{\mathrm{T}},\ p_k \coloneqq \left(\sum_{x \in \mathscr{M}_{=k}} A(x) \right) e^{\mathrm{T}}.$$

Note. With the notation established in Definition 4.2, it follows from Theorem 3.19 that the average number of optimal representations across all Gaussian integers with a 3-NAF representation of at most N digits can be expressed as

$$\frac{1}{|\mathscr{M}_{\leq N}|} \sum_{x \in \mathscr{M}_{\leq N}} eA(x)e^{\mathrm{T}} = \frac{1}{|\mathscr{M}_{\leq N}|} \sum_{k=0}^{N} \sum_{x \in \mathscr{M}_{=k}} eA(x)e^{\mathrm{T}} = \frac{1}{|\mathscr{M}_{\leq N}|} \sum_{k=0}^{N} ep_k.$$

Theorem 4.3. The sequence p_k from Definition 4.2 satisfies the following recurrence relation:

$$p_n = p_{n-1} + Sp_{n-3} + Tp_{n-4} + Up_{n-5},$$

where

$$S := \sum_{d\in\{\pm1,\pm\mathrm{i}\}} A_d,\ T := S(A_0 - I),\ U := SA_0(A_0 - I),$$

with initial conditions

$$p_0 = e^{\mathrm{T}},\ p_1 = p_2 = p_3 = Se^{\mathrm{T}},\ p_4 = \left(S + S^2\right)e^{\mathrm{T}}.$$

Proof. The initial conditions can be verified by direct computation. Consider $n \geq 5$ and $x \in \mathcal{M}_{=n}$. If x has only one non-zero digit (at position $n-1$), then $A(x) = A_{d_1}A_0^{n-1}$. Otherwise, let k be the index of the second non-zero digit of x and $y \in \mathcal{M}_{=k}$ be the number represented by truncating the representation of x starting at this digit, then $A(x) = A_{d_1}A_0^{n-k-3}A(y)$. Notice that although there are $n-k-1$ zeros between the first two non-zero digits, two of them belong to the block of the second non-zero digit, therefore the number of zero blocks is only $n-k-3$. By this reasoning, we can derive a recursive formula for the sum of the matrices of all $x \in \mathcal{M}_{=n}$:

$$\begin{aligned}\sum_{x\in\mathcal{M}_{=n}} A(x) &= \sum_{d_1\in\{\pm1,\pm\mathrm{i}\}} \left(A_{d_1}A_0^{n-1} + A_{d_1}\sum_{k=0}^{n-3}\sum_{y\in\mathcal{M}_{=k}} A_0^{n-k-3}A(y)\right)\\ &= S\left(A_0^{n-1} + \sum_{k=0}^{n-3} A_0^{n-k-3}\sum_{y\in\mathcal{M}_{=k}} A(y)\right).\end{aligned}$$

Multiplying this equality by e^{T} and applying the definition of p_k, we get

$$p_n = S\left(A_0^{n-1}e^{\mathrm{T}} + \sum_{k=0}^{n-3} A_0^{n-k-3}p_k\right).$$

Noticing that $A_0e = e = p_0$ and $A_0^3 = A_0^2$, we can further simplify:

$$p_n = S\left(p_0 + \sum_{k=0}^{n-3} A_0^{n-k-3}p_k\right) = S\left(p_{n-3} + A_0p_{n-4} + A_0^2\sum_{k=0}^{n-5} p_k\right).$$

By subtracting two adjacent terms, we can transform this into a linear recurrence:

$$p_n - p_{n-1} = S\left(p_{n-3} - p_{n-4} + A_0p_{n-4} - A_0p_{n-5} + A_0^2p_{n-5}\right) = Sp_{n-3} + Tp_{n-4} + Up_{n-5}.$$

■

We can easily transform the order-5 recurrence relation on 9-dimensional vectors shown in Theorem 4.3 into an order-1 recurrence relation on 45-dimensional vectors, $q_n = Bq_{n-1}$, by denoting

$$q_n := \begin{pmatrix} p_n \\ p_{n-1} \\ p_{n-2} \\ p_{n-3} \\ p_{n-4}\end{pmatrix},\ B := \begin{pmatrix} I & 0 & S & T & U \\ I & 0 & 0 & 0 & 0 \\ 0 & I & 0 & 0 & 0 \\ 0 & 0 & I & 0 & 0 \\ 0 & 0 & 0 & I & 0\end{pmatrix},$$

where I is the 9×9 identity matrix and 0 is the 9×9 zero matrix. It trivially follows that $q_n = B^{n-4}q_4$. In order to get a better idea of how this sequence grows, we need to find the Jordan canonical form of B, which involves calculating its characteristic polynomial and eigenvalues. If λ is an eigenvalue of B and v the corresponding eigenvector, we have $Bv = \lambda v$. If we denote by x the last 9 components of v, the equality implies that

$$v = \begin{pmatrix} \lambda^4 x \\ \lambda^3 x \\ \lambda^2 x \\ \lambda x \\ x \end{pmatrix}$$

and the first row of the block matrix gives the equation

$$\lambda^4 x + S\lambda^2 x + T\lambda x + Ux = \lambda\lambda^4 x,$$

equivalently written as

$$\left((\lambda^4 - \lambda^5)I + \lambda^2 S + \lambda T + U\right)x = 0.$$

Since x is a non-zero vector (otherwise v would be zero, therefore by definition not an eigenvector), we have

$$\det\left((\lambda^4 - \lambda^5)I + \lambda^2 S + \lambda T + U\right) = 0.$$

It remains to calculate the determinant of $\left(\lambda^4 - \lambda^5\right)I + \lambda^2 S + \lambda T + U$, giving us the characteristic polynomial of B. Although this can be done manually, the process is tedious and error-prone, so it is better to use a computer program involving a symbolic computation library such as SymPy. The program is listed in Appendix A. The resulting polynomial is

$$\lambda^{12}(\lambda-1)^8(\lambda^6-\lambda^2+2)(\lambda^7-\lambda^6-8\lambda^4+3\lambda^3+\lambda^2-2\lambda+2)(\lambda^{12}-4\lambda^9+2\lambda^8+8\lambda^6-4\lambda^5+\lambda^4+8\lambda^3+4).$$

The spectral radius is $\Lambda \approx 2.27348$, a root of the polynomial $\lambda^7 - \lambda^6 - 8\lambda^4 + 3\lambda^3 + \lambda^2 - 2\lambda + 2$. Its algebraic multiplicity is 1, so B can be expressed in Jordan canonical form:

$$B = R\begin{pmatrix} \Lambda & \\ & J \end{pmatrix}R^{-1}, \ R \in \mathbb{C}^{45\times 45}, \ J \in \mathbb{C}^{44\times 44}, \ \rho(J) < \Lambda.$$

By transposing both sides, we also get the Jordan canonical form of B^{T}:

$$B^{\mathrm{T}} = \left(R^{-1}\right)^{\mathrm{T}} \begin{pmatrix} \Lambda & \\ & J^{\mathrm{T}} \end{pmatrix} R^{\mathrm{T}}$$

Theorem 4.4. Let F_N be the average number of optimal representations across all Gaussian integers with a 3-NAF representation of length at most N. Then

$$F_N = (d + o(1)) \cdot \left(\frac{\Lambda}{2}\right)^N,$$

where $\Lambda \approx 2.27348$ as shown above and $d \approx 0.76257$.

Proof. As previously noted, we can express

$$F_N = \frac{\sum_{k=0}^{N} ep_k}{|\mathscr{M}_{\leq N}|}.$$

Let us first examine the last term of the sum:

$$ep_N = eq_N = eR\begin{pmatrix}\Lambda^{N-4} & \\ & J^{N-4}\end{pmatrix}R^{-1}q_4.$$

Since the spectral radius of J is less than Λ, we have $\frac{J^{N-4}}{\Lambda^{N-4}} \to 0$ as $N \to \infty$, therefore

$$\lim_{N\to\infty}\frac{ep_N}{\Lambda^{N-4}} = eRe^{\mathrm{T}}eR^{-1}q_4 = R_{1,1}R^{-1}_{1,\circ}q_4.$$

$R_{1,1}$ is simply the first component of an eigenvector of B corresponding to Λ and, as can be seen from the Jordan canonical from of B^{T}, $R^{-1}_{1,\circ} = \left(\left(R^{-1}\right)^{\mathrm{T}}_{\circ,1}\right)^{\mathrm{T}}$ is the transpose of an eigenvector of B^{T} corresponding to Λ. However, although eigenvectors are determined up to a multiplicative constant, we cannot choose them arbitrarily. Specifically, they have to satisfy $R^{-1}_{\circ,1}R_{1,\circ} = eRR^{-1}e^{\mathrm{T}} = 1$. Given an arbitrary eigenvector v of B and u of B^{T}, both corresponding to Λ, we can normalize them to get

$$\lim_{N\to\infty}\frac{ep_N}{\Lambda^{N-4}} = \frac{v_1u^{\mathrm{T}}}{u^{\mathrm{T}}v}q_4.$$

Using the Stolz–Cesàro theorem, we can create an asymptotic estimate for the numerator in the expression for F_N:

$$\lim_{N\to\infty}\frac{\sum_{k=0}^{N}ep_k}{\Lambda^N} = \lim_{N\to\infty}\frac{ep_N}{\Lambda^N - \Lambda^{N-1}} = \frac{v_1u^{\mathrm{T}}q_4}{\left(\Lambda^4-\Lambda^3\right)u^{\mathrm{T}}v}.$$

Now let us consider the denominator. In the proof of Theorem 2.5, we have determined that

$$\left|\mathscr{M}_{\le N}\right| = a_N = \alpha 2^N + \beta\kappa^N + \gamma\bar{\kappa}^N,$$

where $|\kappa| = \left|\frac{-1+\sqrt{7}\mathrm{i}}{2}\right| < 2$. Therefore,

$$\lim_{N\to\infty}\frac{\left|\mathscr{M}_{\le N}\right|}{2^N} = \alpha.$$

We can determine from the initial conditions of a_N that $\alpha = \frac{31}{15}$. Combining the two results, we get

$$\lim_{N\to\infty}\left(\frac{\Lambda}{2}\right)^{-N}F_N = \frac{15}{31}\cdot\frac{v_1u^{\mathrm{T}}q_4}{\left(\Lambda^4-\Lambda^3\right)u^{\mathrm{T}}v} =: d,$$

which was to be proven. Calculating the eigenvectors and a numeric approximation of d is an exercise for a computer program, which is listed in Appendix A. ■

Chapter 5

Generating optimal representations

In Chapters 3 and 4, we have demonstrated that sufficiently large Gaussian integers have many optimal representations in the extended Penney system, which makes them suitable for ensuring redundancy in cryptographical applications. To minimize predictability, it is necessary to have a way to choose a uniformly random optimal representation of a given integer. The transducer introduced in Chapter 3 provides a simple and efficient algorithm for accomplishing this, which will be described in this chapter.

The idea of generating random representations for cryptographic purposes was studied in [7] for the case of the signed binary system, although the algorithm developed in this paper generates all representations of a given number, not only the optimal ones. The idea of generating only optimal representations (still for the signed binary system) was suggested in [3].

Theorem 5.1. Let $x \in \mathbb{Z}[\mathrm{i}]$ with 3-NAF extended Penney system representation $O_{d_1} \cdots O_{d_n}$. Then the following algorithm generates a uniformly random optimal representation of x:

```
 1: $q \leftarrow 0$
 2: for $i := 1, \dots, n$ do
 3:     $e_1, \dots, e_k :=$ edges of $\tilde{G}$ going to vertex $q$ with output label $O_{d_i}$
 4:     for $j := 1, \dots, k$ do
 5:         $q_j :=$ starting vertex of $e_j$
 6:         $w_j := \left(A_{d_{i+1}} \cdots A_{d_n}\right)_{r,1}$, where $r$ is the index of vertex $q_j$ in the matrices
 7:     end for
 8:     $j :=$ element of $\{1, \dots, k\}$ chosen randomly with weights $w_1, \dots, w_k$
 9:     Write the input label of $e_j$ to the program output
10:     $q \leftarrow q_j$
11: end for
```

Proof. The algorithm starts in the vertex 0 and makes a backward walk with the correct edge colors, deciding randomly at each step where to go and ending back in 0. By Lemma 3.10, Lemma 3.14 and the construction of $\tilde{G}$, such walks correspond to all optimal representations of x, so we just need to prove that the walk is uniformly randomly chosen. Consider the i-th step. Each edge $e_1, \dots, e_k$ presents one option for where to go next. If we choose a particular edge e_j, we will have to finish the walk from the vertex q_j, using the remaining output labels $O_{d_{i+1}}, \dots, O_{d_n}$. By Lemma 3.18, there are exactly w_j such walks, where w_j is defined as in the algorithm. The probability of using the edge e_j is $\frac{w_j}{\sum_{l=1}^{k} w_l}$. If we inductively assume that the algorithm will work correctly from the next step onward, then each of the walks after using e_j has a $\frac{1}{w_j}$ probability to be chosen (conditioned on e_j being chosen in the current step). Therefore, the probability of using e_j and then choosing any particular continuation is $\frac{1}{\sum_{l=1}^{k} w_l}$. Since this does not depend on j or the particular walk chosen, the distribution is uniform. (Note that if $w_j = 0$, the edge e_j will never be used, so the division by zero in the conditional probability is not a problem.) ■

Note. It can be seen that if the products $A_{d_{i+1}} \cdots A_{d_n}$ for all i are computed at once, the time complexity of the algorithm is $\mathcal{O}(n)$. Notice that after computing each product, only the first column needs to be stored.

Example. Consider the number $x := 2 + \mathrm{i}$. It has three optimal representations: $x = \left[100\bar{\mathrm{i}}\right]_{\beta_\mathrm{P}} = [\mathrm{i}0\mathrm{i}]_{\beta_\mathrm{P}} = \left[\bar{\mathrm{i}}1\right]_{\beta_\mathrm{P}}$, with the first one being the 3-NAF representation, consisting of blocks $O_1 \cdots O_{-\mathrm{i}}$. The algorithm starts at $q \leftarrow 0$ and looks at all edges with label O_1 going into 0. There are $k = 2$ such edges: $e_1 : 0 \xrightarrow{001|001} 0$ and $e_2 : 1 \xrightarrow{000|001} 0$. Therefore, we label $q_1 := 0,\ q_2 := 1$ and calculate the weights using the product of the matrices of the remaining blocks, which in our case is only $O_{-\mathrm{i}}$:

$$\begin{aligned} w_1 &= (A_{-\mathrm{i}})_{1,1} = 1 && \text{(because the first row of matrices corresponds to the vertex 0)} \\ w_2 &= (A_{-\mathrm{i}})_{2,1} = 2 && \text{(because the second row of matrices corresponds to the vertex 1)} \end{aligned}$$

This means that we have two options:

1. $\frac{w_1}{w_1+w_2} = \frac{1}{3}$ probability to set $q \leftarrow q_1 = 0$ and output 001. Then we look at edges going into 0 with output label $O_{-\mathrm{i}}$, which are $e_1 : 0 \xrightarrow{00\bar{\mathrm{i}}|00\bar{\mathrm{i}}} 0$ and $e_2 : -\mathrm{i} \xrightarrow{000|00\bar{\mathrm{i}}} 0$, so $q_1 = 0,\ q_2 = -\mathrm{i}$. There are no remaining blocks, so weights are going to be calculated using the identity matrix, which naturally corresponds to the requirement to end the backward walk in the vertex 0:

   $$\begin{aligned} w_1 &= I_{1,1} = 1 && \text{(because the first row of matrices corresponds to the vertex 0)} \\ w_2 &= I_{5,1} = 0 && \text{(because the fifth row of matrices corresponds to the vertex } -\mathrm{i}\text{)} \end{aligned}$$

   Since only the first edge has a non-zero weight, the algorithm is guaranteed at this point to output $00\bar{\mathrm{i}}$, which gets us the representation $\left(100\bar{\mathrm{i}}\right)_{\beta_\mathrm{P}}$.

2. $\frac{w_2}{w_1+w_2} = \frac{2}{3}$ probability to set $q \leftarrow q_2 = 1$ and output 000. Then we consider edges going into 1 with output label $O_{-\mathrm{i}}$. There are $k = 5$ such edges:

   $$e_1 : 0 \xrightarrow{\mathrm{i}0\mathrm{i}|00\bar{\mathrm{i}}} 1$$

   $$e_2 : 0 \xrightarrow{0\bar{\mathrm{i}}1|00\bar{\mathrm{i}}} 1$$

   $$e_3 : 1 \xrightarrow{0\bar{\mathrm{i}}0|00\bar{\mathrm{i}}} 1$$

   $$e_4 : \mathrm{i} \xrightarrow{\mathrm{i}00|00\bar{\mathrm{i}}} 1$$

   $$e_5 : 1 + \mathrm{i} \xrightarrow{\mathrm{i}0\mathrm{i}|00\bar{\mathrm{i}}} 1$$

   By the same logic as above, $w_1 = w_2 = 1$ and $w_3 = w_4 = w_5 = 0$, so the algorithm is going to $1 : 1$ randomly choose between e_1 and e_2, outputting either i0i or $0\bar{\mathrm{i}}1$, which form the two remaining optimal representations.

From the above two cases, it is clear that each of the three optimal representations has a $\frac{1}{3}$ probability of being output, which is what we wanted to achieve.

Conclusion

We have investigated properties of the positional numeration system with $\beta = \mathrm{i}-1$, $\mathcal{D} = \{0, \pm 1, \pm \mathrm{i}\}$. We provided simple proofs of previously known results: every Gaussian integer has a unique 3-NAF representation, which is always optimal and an average digit of such a representation has only a $\frac{1}{4}$ probability to be non-zero.

As a new result, in Chapter 3, we developed a transducer for converting any representation in the extended Penney system to the equivalent 3-NAF representation. We reduced this transducer to only permit optimal representations as the input, meaning that there is a bijection between the possible paths in the graph and all optimal representations of a given number. We used this to calculate the maximum number of optimal representations a Gaussian integer can have if its 3-NAF representation has a given length, as well as which exact Gaussian integers achieve this maximum. The result, stated in Theorem 3.30, is similar to that in [4], except with a different recurrent sequence.

In Chapter 4, we used the transducer from Chapter 3 to give an asymptotic estimate for the average number of optimal representations across all numbers whose 3-NAF representation does not exceed a given length. This estimate is stated in Theorem 4.4. It is analogous to a result about signed binary representations in [4], except less precise; the details are discussed at the beginning of the chapter.

Finally, in Chapter 5, we showed a simple algorithm for generating a uniformly random extended Penney system representation of a given Gaussian integer, which uses a "reversed" form of the reduced transducer shown in Chapter 3. This algorithm, shown in Theorem 5.1, can potentially be generalized to other numeration systems, as long as a similar transducer is constructed.

As a potential application, these results show that for cryptographical algorithms based on complex integers, the extended Penney system is a suitable candidate for achieving high performance if only optimal representations are used, as well as high redundancy due to the exponentially growing number of optimal representations.

Appendix A

Programs

This chapter lists computer programs which implement algorithms or perform computations mentioned in the project.

All programs are written in Python 3. The SymPy symbolic computation library is used for symbolic calculations.

The programs are only for demonstration purposes and not optimized for performance. Some of them use sub-optimal algorithms for the sake of simplicity.

gaussian_integer.py: Implementing basic arithmetic and divisibility testing on Gaussian integers, along with constants necessary for working with the extended Penney system.

```python
from typing import NamedTuple


class GaussianInteger(NamedTuple):
    re: int
    im: int

    def __add__(x, y):
        return GaussianInteger(x.re + y.re, x.im + y.im)
    def __sub__(x, y):
        return GaussianInteger(x.re - y.re, x.im - y.im)

    def __mul__(x, y):
        return GaussianInteger(x.re * y.re - x.im * y.im,
                               x.re * y.im + x.im * y.re)
    def __truediv__(x, y):
        # Note: Returns the current result only if x is divisible by y.
        n = y.re**2 + y.im**2
        return GaussianInteger((x.re * y.re + x.im * y.im) // n,
                               (x.im * y.re - x.re * y.im) // n)
    def divisible_by(x, y):
        return x == (x / y) * y

    def __pow__(x, n):
        result = GaussianInteger(1, 0)
        for _ in range(n):
            result *= x
        return result

    def __repr__(x):
        if x.re == 0 and x.im == 0: return "0"
        re = "" if x.re == 0 else str(x.re)
        sign = "+" if x.re != 0 and x.im > 0 else "-" if x.im == -1 else ""
        im = "" if abs(x.im) <= 1 else str(x.im)
        i = "" if x.im == 0 else "i"
        return re + sign + im + i

zero = GaussianInteger(0, 0)
β = GaussianInteger(-1, 1)
β_cube = GaussianInteger(2, 2)
nonzero_digits = [GaussianInteger(1, 0), GaussianInteger(0, 1),
                  GaussianInteger(-1, 0), GaussianInteger(0, -1)]
all_digits = [zero] + nonzero_digits
```

`eisenstein_integer.py`: Implementing basic arithmetic and divisibility testing on Eisenstein integers.

```
from typing import NamedTuple


class EisensteinInteger(NamedTuple):
    re: int
    om: int

    def __add__(x, y):
        return EisensteinInteger(x.re + y.re, x.om + y.om)
    def __sub__(x, y):
        return EisensteinInteger(x.re - y.re, x.om - y.om)

    def __mul__(x, y):
        return EisensteinInteger(x.re * y.re - x.om * y.om,
                                 x.re * y.om + x.om * y.re - x.om * y.om)
    def __truediv__(x, y):
        # Note: Returns the correct result only if x is divisible by y.
        n = y.re**2 - y.re * y.om + y.om**2
        return EisensteinInteger((x.re * y.re - x.re * y.om + x.om * y.om) // n,
                                 (x.om * y.re - x.re * y.om) // n)
    def divisible_by(x, y):
        return x == (x / y) * y

    def __pow__(x, n):
        result = EisensteinInteger(1, 0)
        for _ in range(n):
            result *= x
        return result

    def __repr__(x):
        if x.re == 0 and x.om == 0: return "0"
        re = "" if x.re == 0 else str(x.re)
        sign = "+" if x.re ≠ 0 and x.om > 0 else "-" if x.om == -1 else ""
        om = "" if abs(x.om) ≤ 1 else str(x.om)
        ω = "" if x.om == 0 else "ω"
        return re + sign + om + ω
```

`penney_3naf.py`: Finding the 3-NAF representation of a given Gaussian integer in the extended Penney system.

```python
from gaussian_integer import GaussianInteger, zero, β, β_cube, nonzero_digits


def to_3naf(x: GaussianInteger) -> list[GaussianInteger]:
    digits = []
    while x ≠ zero:
        if x.divisible_by(β):
            digits += [zero]
            x /= β
        else:
            for d in nonzero_digits:
                if (x - d).divisible_by(β_cube):
                    digits += [d, zero, zero]
                    x = (x - d) / β_cube
                    break
    # Reverse the representation to start with the most significant digit
    digits.reverse()
    # Remove the two leading zeros
    return digits[2:]


def to_3naf_blocks(x: GaussianInteger):
    # The same algorithm, but returns blocks instead of digits.
    blocks = []
    while x ≠ zero:
        if x.divisible_by(β):
            blocks.append((zero,))
            x /= β
        else:
            for d in nonzero_digits:
                if (x - d).divisible_by(β_cube):
                    blocks.append((zero, zero, d))
                    x = (x - d) / β_cube
                    break
    # Reverse the representation to start with the most significant digit
    blocks.reverse()
    return blocks
```

penney_optimal_reprs.py: Calculating the number of optimal representations of a given Gaussian integer.

```python
from gaussian_integer import GaussianInteger, zero, β, β_cube, nonzero_digits
from functools import cache


def min_weight(x: GaussianInteger) -> int:
    # Finds the Hamming weight of the 3-NAF representation of x,
    # which is known to be minimal among all representations.
    nonzeros = 0
    while x != zero:
        if x.divisible_by(β):
            x /= β
        else:
            for d in nonzero_digits:
                if (x - d).divisible_by(β_cube):
                    nonzeros += 1
                    x = (x - d) / β_cube
                    break
    return nonzeros


@cache
def num_representations(x: GaussianInteger, weight: int) -> int:
    # Finds the number of representations of x with the given Hamming weight.
    if x == zero:
        return int(weight == 0)
    elif weight == 0:
        return 0
    elif x.divisible_by(β):
        return num_representations(x / β, weight)
    else:
        return sum(num_representations((x - d) / β, weight - 1)
                   for d in nonzero_digits)


def num_optimal_representations(x: GaussianInteger) -> int:
    return num_representations(x, min_weight(x))
```

`penney_max_optimal_reprs.py`: Calculating the maximal number of optimal representations for a Gaussian integer whose 3-NAF representation has a given length. Note that this is subtly different from the task of finding such an integer whose 3-NAF representation has a given weight. However, it turns out that since such integers always have exactly two zeros between every pair of non-zero digits in their 3-NAF representation, these two tasks are easily convertible to each other.

```python
from gaussian_integer import GaussianInteger, zero, β, nonzero_digits
from penney_3naf import to_3naf
from penney_optimal_reprs import num_optimal_representations
from typing import Iterator


def integers_with_3naf_len(k: int) -> Iterator[GaussianInteger]:
    # Iterates over all integers whose 3-NAF representation has a length of k.
    if k ≤ 0:
        yield zero
    else:
        for d in nonzero_digits:
            p = d * β ** (k - 1)
            for x in integers_with_3naf_len_max(k - 3):
                yield p + x


def integers_with_3naf_len_max(n: int) -> Iterator[GaussianInteger]:
    # Iterates over all integers whose 3-NAF representation has a length of
    # at most k.
    for k in range(max(n + 1, 1)):
        yield from integers_with_3naf_len(k)


def most_optimal_representations(
    n: int,
) -> tuple[int, list[GaussianInteger], list[list[GaussianInteger]]]:
    # Returns the maximum number of optimal representations
    # for an integer whose 3-NAF representation has length exactly n,
    # the list of integers for which it is reached
    # and the list of their 3-NAF representations.
    maximum = 0
    xs = []
    for x in integers_with_3naf_len(n):
        optimal_reprs = num_optimal_representations(x)
        if optimal_reprs > maximum:
            maximum = optimal_reprs
            xs = [x]
        elif optimal_reprs == maximum:
            xs.append(x)
    return (maximum, xs, [to_3naf(x) for x in xs])
```

`G_tilde.py`: Constructing the graph $\tilde{G}$ that can be used for reasoning about optimal representations in the extended Penney system.

```
from gaussian_integer import GaussianInteger, zero, β, β_cube, all_digits, \
                             nonzero_digits
from typing import NamedTuple
from sympy import Matrix


class Edge(NamedTuple):
    start: GaussianInteger
    end: GaussianInteger
    input: list[GaussianInteger]
    output: list[GaussianInteger]


vertices = [zero,
            GaussianInteger(1, 0), GaussianInteger(0, 1),
            GaussianInteger(-1, 0), GaussianInteger(0, -1),
            GaussianInteger(1, 1), GaussianInteger(-1, 1),
            GaussianInteger(-1, -1), GaussianInteger(1, -1)]
edges = []
for q0 in vertices:
    for q1 in vertices:
        if q1 * β == q0:
            edges.append(Edge(q0, q1, (zero,), (zero,)))
        for r in all_digits:
            for x in all_digits:
                for y in all_digits:
                    for z in all_digits:
                        weight = (
                            (x ≠ zero) + (y ≠ zero) + (z ≠ zero) - (r ≠ zero)
                        )
                        if (
                            weight ≤ (q0 == zero) - (q1 == zero)
                            and z * β * β + y * β + x + q0 == q1 * β_cube + r
                        ):
                            edges.append(Edge(q0, q1, (z, y, x), (zero, zero, r)))

blocks = [(zero,)] + [(zero, zero, r) for r in nonzero_digits]
A = {}
for block in blocks:
    matrix = [[0] * 9 for _ in range(9)]
    for e in edges:
        if e.output == block:
            matrix[vertices.index(e.end)][vertices.index(e.start)] += 1
    A[block] = Matrix(matrix)
```

`penney_average_reprs_characteristic_polynomial.py`: Calculating the determinant of the matrix $(\lambda^4 - \lambda^5)I + \lambda^2 S + \lambda T + U$ as a part of determining the average number of equivalent representations in the extended Penney system.

```
from sympy import *

A_0 = Matrix([
    [1, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 1, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 1, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 1],
    [0, 0, 0, 0, 0, 1, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0],
])
S = Matrix([
    [4, 1, 1, 1, 1, 0, 0, 0, 0],
    [4, 2, 2, 0, 0, 2, 0, 0, 0],
    [4, 0, 2, 2, 0, 0, 2, 0, 0],
    [4, 0, 0, 2, 2, 0, 0, 2, 0],
    [4, 2, 0, 0, 2, 0, 0, 0, 2],
    [1, 0, 1, 0, 0, 0, 0, 0, 0],
    [1, 0, 0, 1, 0, 0, 0, 0, 0],
    [1, 0, 0, 0, 1, 0, 0, 0, 0],
    [1, 1, 0, 0, 0, 0, 0, 0, 0],
])
I = eye(9)
T = S * (A_0 - I)
U = S * A_0 * (A_0 - I)
λ = Symbol("λ")
p = factor(((λ**4 - λ**5) * I + λ**2 * S + λ * T + U).det())
pretty_print(p)
```

`penney_average_reprs_constant.py`: Calculating the multiplicative constant d in Theorem 4.4.

```
from sympy import *

A_0 = Matrix([
    [1, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 1, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 1, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 1],
    [0, 0, 0, 0, 0, 1, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0],
    [0, 0, 0, 0, 0, 0, 0, 0, 0],
])
S = Matrix([
    [4, 1, 1, 1, 1, 0, 0, 0, 0],
    [4, 2, 2, 0, 0, 2, 0, 0, 0],
    [4, 0, 2, 2, 0, 0, 2, 0, 0],
    [4, 0, 0, 2, 2, 0, 0, 2, 0],
    [4, 2, 0, 0, 2, 0, 0, 0, 2],
    [1, 0, 1, 0, 0, 0, 0, 0, 0],
    [1, 0, 0, 1, 0, 0, 0, 0, 0],
    [1, 0, 0, 0, 1, 0, 0, 0, 0],
    [1, 1, 0, 0, 0, 0, 0, 0, 0],
])
I = eye(9)
T = S * (A_0 - I)
U = S * A_0 * (A_0 - I)
O = zeros(9, 9)
λ = Symbol("λ")
Λ = CRootOf(λ**7 - λ**6 - 8 * λ**4 + 3 * λ**3 + λ**2 - 2 * λ + 2, 2)
x = Matrix([4] + [4 + 2 / Λ**3] * 4 + [1] * 4)
y = Matrix([[4 * Λ**4] + [(Λ - 1) * Λ**3] * 4 + [(Λ**2 + 2) * (Λ - 1)] * 4])
v = BlockMatrix([[Λ**4 * x], [Λ**3 * x], [Λ**2 * x], [Λ * x], [x]]).as_explicit()
u = BlockMatrix([[
    y,
    y * (Λ**2 * S + Λ * T + U) / Λ**4,
    y * (Λ**2 * S + Λ * T + U) / Λ**3,
    y * (Λ * T + U) / Λ**2,
    y * U / Λ,
]]).as_explicit()
e = Matrix([1, 0, 0, 0, 0, 0, 0, 0, 0])
q_4 = BlockMatrix([[(S + S**2) * e], [S * e], [S * e], [S * e], [e]]).as_explicit()
d = v[0, 0] / u.dot(v) * u.dot(q_4) / (Λ**4 - Λ**3) * 15 / 31
print(N(d))
```

penney_average_reprs.py: Calculating the average number of optimal representations of all Gaussian integers whose 3-NAF representation is up to a certain length, in order to experimentally confirm the calculation of the constant *d* in Theorem 4.4.

```
from gaussian_integer import GaussianInteger, zero, β, nonzero_digits
from penney_3naf import to_3naf
from penney_optimal_reprs import num_optimal_representations
from typing import Iterator
from sympy import *


def integers_with_3naf_len(k: int) -> Iterator[GaussianInteger]:
    # Iterates over all integers whose 3-NAF representation has a length of k.
    if k ≤ 0:
        yield zero
    else:
        for d in nonzero_digits:
            p = d * β ** (k - 1)
            for x in integers_with_3naf_len_max(k - 3):
                yield p + x


def integers_with_3naf_len_max(n: int) -> Iterator[GaussianInteger]:
    # Iterates over all integers whose 3-NAF representation has a length of
    # at most k.
    for k in range(max(n + 1, 1)):
        yield from integers_with_3naf_len(k)


def average_optimal_reprs(n: int) -> float:
    # Returns the average number of optimal representations
    # across all integers whose 3-NAF representation has length at most n.
    optimal_reprs = 0
    integers = 0
    for x in integers_with_3naf_len_max(n):
        optimal_reprs += num_optimal_representations(x)
        integers += 1
    return optimal_reprs / integers


λ = Symbol("λ")
Λ = N(CRootOf(λ**7 - λ**6 - 8 * λ**4 + 3 * λ**3 + λ**2 - 2 * λ + 2, 2))


for n in range(1, 20):
    # Calculates an approximation for the constant d for small values of n.
    print(n, average_optimal_reprs(n) / (Λ / 2) ** n)
```

`penney_random_optimal_reprs.py`: Implementing the algorithm from Theorem 5.1 for generating a uniformly random optimal extended Penney system representation of a Gaussian integer.

```python
from gaussian_integer import GaussianInteger, zero, all_digits
from G_tilde import vertices, edges, A
from penney_3naf import to_3naf_blocks
from sympy import *
import random


def random_optimal_repr(x: GaussianInteger) -> list[GaussianInteger]:
    result = []
    q = zero
    O = to_3naf_blocks(x)
    n = len(O)
    P = [None] * n
    P[n - 1] = eye(9)
    for i in range(n - 1, 0, -1):
        P[i - 1] = A[O[i]] * P[i]
    for i in range(n):
        e = [edge for edge in edges if edge.end == q and edge.output == O[i]]
        k = len(e)
        w = [P[i][vertices.index(edge.start), 0] for edge in e]
        [edge] = random.choices(e, w)
        result += edge.input
        q = edge.start
    # Remove leading zeros
    i = 0
    while i < len(result) and result[i] == zero:
        i += 1
    return result[i:]
```

Bibliography

[1] C. Heuberger, D. Krenn: *Optimality of the Width-w Non-adjacent Form: General Characterisation and the Case of Imaginary Quadratic Bases*, arXiv:1110.0966v1 (2011), https://doi.org/10.48550/arXiv.1110.0966; Journal de théorie des nombres de Bordeaux, Volume 25 no. 2 (2013), pp. 353–386, https://doi.org/10.5802/jtnb.840.

[2] C. Heuberger, D. Krenn: *Analysis of width-w non-adjacent forms to imaginary quadratic bases*, Journal of Number Theory, Volume 133, Issue 5 (2013), https://dx.doi.org/10.1016/j.jnt.2012.08.029.

[3] P. J. Grabner, C. Heuberger: *On the Number of Optimal Base 2 Representations of Integers*, Des Codes Crypt, Volume 40 (2006), pp.25–39, https://doi.org/10.1007/s10623-005-6158-y.

[4] J. Tůma, J. Vábek: *On the number of binary signed digit representations of a given weight*, Commentationes Mathematicae Universitatis Carolinae, Volume 56, Issue 3 (2015), pp. 287–306, https://dml.cz/manakin/handle/10338.dmlcz/144345.

[5] W. Penney: *A "Binary" System for Complex Numbers*, Journal of the ACM, Volume 12, Issue 2 (1965), pp. 247–248, https://doi.org/10.1145/321264.321274.

[6] G. W. Reitwiesner: *Binary arithmetic*, Advances in Computers, Volume 1 (1960), pp. 231–308, https://doi.org/10.1016/S0065-2458(08)60610-5, **not open-access**.

[7] N. Ebeid, M. A. Hasan: *On Randomizing Private Keys to Counteract DPA Attacks*, Selected Areas in Cryptography (2003), Lecture Notes in Computer Science, Volume 3006, https://doi.org/10.1007/978-3-540-24654-1_5.

[8] U. Güntzer, M. Paul: *Jump interpolation search trees and symmetric binary numbers*, Information Processing Letters, Volume 26, Issue 4 (1987), pp. 193–204, https://doi.org/10.1016/0020-0190(87)90005-6, **not open-access**.

[9] A. Avizienis: *Signed-Digit Numbe Representations for Fast Parallel Arithmetic*, IRE Transactions on Electronic Computers, Volume EC-10, Issue 3 (1961), pp. 389-400, **not open-access**.

[10] Ch. Frougny, E. Pelantová, M. Svobodová: *Minimal digit sets for parallel addition in non-standard numeration systems.* Journal of Integer Sequences, Volume 16 (2013).

[11] Z. Masáková, E. Pelantová, M. Svobodová: *Algebraické metody v teoretické informatice*, Department of Mathematics FNSPE CTU in Prague (2023), pp. 1–7, https://people.fjfi.cvut.cz/pelanedi/ALTI_k_vystaveni.pdf.

[12] N. P. Brousentsov, J. Ramil Alvarez: *Ternary Computers: The Setun and the Setun 70*, Advances in Information and Communication Technology, Volume 357 (2011).

[13] N. P. Brousentsov, S. P. Maslov, J. Ramil Alvarez, E. A. Zhogolev: *Development of ternary computers at Moscow State University*, `https://www.computer-museum.ru/english/setun.htm`.

[14] Jin Y. Yen: *An algorithm for finding shortest routes from all source nodes to a given destination in general networks*, Quarterly of Applied Mathematics, Volume 27 (1970), pp. 526–530, `https://doi.org/10.1090/qam/253822`.