rijksuniversiteit groningen

faculteit Wiskunde en Natuurwetenschappen

# Approximations of π

Bacheloronderzoek Wiskunde

Juli 2010

Student: Henk-Jaap Stelwagen

Begeleider: Prof. dr. Jaap Top

# Approximations of $\pi$

Henk Jaap Stelwagen

July 7, 2010

# Contents

# 1 Introduction

Around 1700 A.D. , mathematicians began using arctangent series to find digits of $\pi$. One of the first to start finding digits this way was John Machin (1680-1752), but also renowned mathematicians like Euler and Newton used arctangent series. Machin used trigonometric relations to find

$$\frac{\pi}{4} = 4\arctan\left(\frac{1}{5}\right) - \arctan\left(\frac{1}{239}\right) \tag{1}$$

This equation worked quite well for calculating $\pi$, as the second term on the right converges very quickly when using the series for the arctangent. Using (1), Machin calculated $\pi$ to 100 decimal places in 1706. A similar result was calculated by Störmer in 1896:

$$\pi = 24\arctan\left(\frac{1}{8}\right) + 8\arctan\left(\frac{1}{57}\right) + 4\arctan\left(\frac{1}{239}\right) \tag{2}$$

but this time yielding no less than 100,265 decimals.

Making use of (1) and (2) there are many ways to derive different expressions. For example, one could use

$$\arctan\left(\frac{1}{p}\right) = \arctan\left(\frac{1}{p+q}\right) + \arctan\left(\frac{q}{p^2+pq+1}\right) \tag{3}$$

to create a new formula for $\pi$. Similar to (3) there are of course many other identities. Formulae such as (1) and (2) is what I will be looking for to find approximations of $\pi$. But rather than using identities like (3), I will make use of Gaussian Integers and Eisenstein integers to create an algorithm that finds these identities with Gaussian Elimination. This is based on an idea which, for the case of the Gaussian Integers, appears in a paper [2] by Jack S. Calcut. How all this works will be explained in section 3: The Method.

In section 4, Accuracy, I look at the solutions found to test how well they approximate $\pi$, and moreover, how fast they converge. But first I will give a short summary of the history of $\pi$ in the next section.

## 2 A brief note on the history of $\pi$

The history of $\pi$ is an old story that goes back for thousands of years. It involves many old cultures and societies that are known to us in the present day. From the old Babylonians to the Egyptians and from the Far East to the Roman Empire, all contributed to the development of mathematics and thus to $\pi$. For $\pi$ is a number that was 'discovered' quite early on in mathematics. As calculations were often expressed geometrically, and geometry was one of the first disciplines of mathematics to book substantial progress, one can imagine that it wasn't long before people started to ponder on calculating the area of a circle. Exactly when and how man came to define the number $\pi$ and, moreover, to approximate it, is unknown. Because the first real documentation (rather than circumstantial evidence from earlier times) was found around 2000 B.C., we can only guess what happened before that time. However, it is not unlikely that the origin of $\pi$ lies in defining proportions. When you assume that it was known at some point that the ratio between proportional quantities is constant, it must have been sooner rather than later that the ratio between a circle's diameter and circumference was discovered. It was then that this strange entity, now known as $\pi$ , came into life. Even though it wasn't labeled with the symbol $\pi$ until the 18th century, for the sake of simplicity, I will denote it as $\pi$. Nonetheless, it is a fact that good approximations were made by the Egyptians as well as the Babylonians. With simple tools the Babylonians calculated

$$\pi \text{ equals } 3\frac{1}{8} = 3.125 \tag{4}$$

and the Egyptians arrived at

$$\pi \text{ equals } 4\left(\frac{8}{9}\right)^2 \tag{5}$$

which gives aprox. 3.16.

Also in different parts of the world approximations of $\pi$ were made. In some of the first available documents of Hindu mathematics, the Siddhantas, the value

$$\pi = 3\frac{177}{1250} \approx 3.1416 \tag{6}$$

is used. The Chinese used, according to documentation from 718 A.D.,

$$\pi = \frac{92}{29} \approx 3.1724 \tag{7}$$

Earlier, Liu Hui found

$$3.141024 < \pi < 3.142704 \tag{8}$$

with a variation of inscribed polygons and at the top are Tsu Chung-Chih and his son Tsu Keng-Chih, who found

$$3.1415926 < \pi < 3.1415927 \tag{9}$$

This is an accuracy that wasn't equaled in Europe until the 1500's.

In the times of the early Greeks, Plutarch seems to be the first to mention the problem of 'The squaring of the circle', a problem which was not completely solved until 1882. Along with the 'Doubling of the cube' and 'Trisection of an angle' it was one of the greatest puzzles of antiquity (and some time after). It is also particularly related to $\pi$, since the problem deals with the construction of a square with equal area as a given circle. Associated with the 'Squaring of the circle' is the Greek philosopher Antiphon, who formulated the "principle of exhaustion". This principle is an algorithm which doubles the sides of an inscribed polygon until the circle is 'exhausted' (i.e. the polygon coincides with the circle). So one starts with a inscribed square moving to an octagon, etc. Since any polygon can be squared, it was Antiphon's belief that a circle could be squared this way. The problem is of course that this cannot be attained in a finite number of steps, therefore not being a 'correct' solution. It is however a method that can approximate $\pi$ quite closely by doubling the sides of the polygon a great number of times, and in this light it was of great influence on mathematicians searching for $\pi$'s decimals.

A name that one could also expect is that of Euclid. For finding decimals of $\pi$ he may not have been as important as for mathematics in general. With Euclid's Elements came the mathematical rigor, which has proven priceless for mathematics.

And then, during the Roman Empire, there was Archimedes. Using a method similar to the one Antiphon used, he got the approximation

$$3.14084 < \pi < 3.142858 \tag{10}$$

A big difference with earlier greek mathematics, as can clearly be seen, is that Archimedes used not only a lower bound, but also an upper bound. Instead of thinking in terms of 'exhaustion' he derived the upper bound by constructing not only an inscribed, but also a circumscribed polygon. And so he was the first one to describe a method for calculating $\pi$ to any degree of accuracy.

After Archimedes mathematical developement went quiet for quite some time. As the dark ages came, along with the power of the Catholic church, science experienced a low. Concerning the accuracy of $\pi$ , it wasn't until the 16th century that progress was made. Still based on Archimedes' method, the Frenchman François Viète (1540-1603) sharpened the bounds to:

$$3.1415926535 < \pi < 3.1415926537 \tag{11}$$

Basically, this was nothing new. However, Viète did do something completely new: he found the first infinite analytical expression for $\pi$ .

It was in the same time that people started calculating $\pi$ more and more accurately: the era of the digit hunters had begun. As the invention of decimal fractions and logarithms greatly improved numerical calculations, it was then that more and more decimals were quickly found. To mention a couple of records: Adriaan Anthoniszoon (1527-1607) found the value

$$\pi = \frac{355}{113} \tag{12}$$

which is correct to 6 decimal places. This record was broken by François Viète in 1593 (see above), but already in the same year his record fell to Adriaen van Roomen, who

calculated 15 decimals. Three years later his record was broken by another Dutchman, Ludolph van Ceulen (1539-1610), who eventually found $\pi$ to 35 digits. As mathematics developed, more methods were found to accurately calculate $\pi$ . The astronomer Abraham Sharp (1651-1742) used an arcsine series to obtain 72 decimal digits, and shortly afterwards, in 1706, John Machin (1680-1752) used the difference between two arctangents to find 100 decimal places. Ofcourse, this went on for centuries until $\pi$ was eventually calculated to millions of decimals and the 'art' of finding decimals was no longer an intelligent job, but merely a line in a computer program.

Also noteworthy is the transcendance of $\pi$ . In Eulers time suspicions were raised that $\pi$ was transcendental (i.e. not a solution to an algebraic equation with integer coefficients). When this was eventually proved by Lindemann in 1882 it also appeared to be critical in the solution to the ancient problem of 'Squaring the circle' and it was then clear that it was impossible to square the circle.

# 3 The method

## 3.1 Intro

As pointed out in the introduction, I will try to find arctangent formulae of the form

$$\pi = p_1 \arctan\left(\frac{m_1}{N_1}\right) + p_2 \arctan\left(\frac{m_2}{N_2}\right) + p_3 \arctan\left(\frac{m_3}{N_3}\right) + \dots \tag{13}$$

using the Gaussian Integers $\mathbb{Z}[I]$ (with $I^2$= -1).

To make the connection with Gaussian Integers explicit, consider the expression

$$n(1+I) = \prod_{i=1}^{k} (N_i + m_i I)^{p_i} \tag{14}$$

where the $N_i$ are integers greater than 0 and $p_i \in \mathbb{Z}$. Taking the arguments of both sides of this equation we get

$$\frac{\pi}{4} = \sum_{i=1}^{k} p_i \arg(N_i + m_i I) = \sum_{i=1}^{k} p_i \arctan\left(\frac{m_i}{N_i}\right) \tag{15}$$

and it is now obvious that finding an expression of the form (14) results in a formula for $\pi$ as in (13). This is true mudolu $2\pi$, so it is possible that a multiple of $2\pi$ mus be added, but this will not affect the theory. At this point a few remarks can be made:

- Both Machin's formula and Störmer's formula are special cases of (13) with $m_i = 1$ for all $i$. In general, one doesn't have to be restricted by this choice, and formulae with different $m_i$ are therefore also possible.
- Even though $m_i$ does not have to be 1 we don't want it to be chosen too freely. Since the purpose of a formula like (13) is to quickly approximate $\pi$, it is desirable to have $\frac{m_i}{N_i}$ small. This way the arctangent will converge more quickly.
- As all the $m_i$'s in Machin's formula are equal to 1. If it is demanded that $\gcd(N_i, m_i) = 1$, this is also the only case for which all $m_i$ can be equal. In other words: if $m_i = x$ and $\gcd(N_i, m_i) = 1$ for all $i$, then this implies $x = 1$. This will be proven in the next theorem.

**Theorem 1.** *Assume that a produkt like* (14) *is given in which all* $m_i$ *are equal. Now demand that* $\gcd(N_i, m_i) = 1$ *for all* $i$. *Then* $m_i = 1 \,\forall\, i$.

To prove this theorem I will assume $m_i > 1$. Since we know by Machin's formula (and others) that products like (14) exist for $m_i = 1$, it suffices to prove that such products do not exist for $m_i > 1$. To prove this we need the following Lemma:

**Lemma 1.** *Take any finite product of the form* $P = \prod (N_i + xI)^{p_i}$ *with positive integers* $p_i$. *Also assume* $\gcd(N_i, x) = 1$. *Then* $\Re(P)$ *is not divisible by* $x$ *and* $\Im(P)$ *is divisible by* $x$.

*Proof.* The proof is attained by induction, so we start with two terms:

$$(N_1 + xI)(N_2 + xI) = N_1N_2 - x^2 + x(N_1 + N_2)I$$

Obviously, the imaginary part is divisible by $x$ . The real part is not: $N_1N_2$ is not divisible by $x$, since $N_1, N_2$ are not divisible by $x$, and after substracting $x^2$ the conclusion follows. The lemma is proven for two terms. Now assume that the lemma holds for $n$ terms. Then the product can then be written as $a + xbI$ where $a$ is not divisible by $x$. Repeating the first step of the induction shows that $(a + xbI)(N_j + xI)$ also satisfies the lemma. This completes the proof. □

This is however only half of the proof of Theorem 1. As we only looked at positive $p_i$, the negatives must still be checked. As a matter of fact, it will turn out that in most of the 'good' solutions there is at least 1 factor that has a negative power.

**Lemma 2.** *Take any finite product of the form* $P = \prod(\overline{N_i + xI})^{p_i}$ *with positive integers* $p_i$. *Again, assume that* $\gcd(N_i, x) = 1$. *Then* $\Re(P)$ *is not divisible by* $x$ *and* $\Im(P)$ *is divisible by* $x$

*Proof.* The proof is very similar to the proof of lemma 1. The only difference is that instead of $(N_i + xI)$ you use $(N_i - xI)$. □

Now the proof of theorem 1 is quite straightforward:

*Proof.* The first step is to rearrange the terms of (14) by grouping positive and negative powers:

$$n(1 + I) = \prod_{i=1}^{l}(N_i + m_iI)^{q_i} \prod_{j=l+1}^{m} (N_j + m_jI)^{r_j}$$

where all $q_i$ are postive, and $r_j$ are negative. Looking at a negative exponent $r_j$, it holds that $(N_j+m_jI)^{r_j} = \frac{(N_j-m_jI)^{-r_j}}{|(N_j-m_jI)^{-r_j}|}$. Now $-r_j$ is a positive integer, and $|(N_j-m_jI)^{r_j}| = K_j$ is also a positive integer. Rewriting we have

$$n(1 + I) = \frac{1}{K}\prod_{i=1}^{l}(N_i + m_iI)^{q_i} \prod_{j=l+1}^{m} (N_j - m_jI)^{|r_j|}$$

with $K = \prod_{j=1}^{m} \frac{1}{K_j}$. Using the lemma's 1 and 2 we have

$$n(1 + I) = \frac{1}{K}(a + bxI)(c + dxI)$$

where $\gcd(x, a) = \gcd(x, c) = 1$. Writing out this expression gives us

$$n(1 + I) = (e + fxI)$$

Again, $e$ is not divisible by $x$. On the l.h.s the real and imaginary parts are equal. But on the r.h.s. the imaginary part contains a divisor, $x$, which the real parts does not contain. Therefore, the real and imaginary parts cannot be equal and this proves the theorem. □

To find products like (14) I will make use of the fact that $\mathbb{Z}[I]$ is a unique factorization domain, implying that every $x \in \mathbb{Z}[I]$ can be written as a product of units and irreducible factors in a unique way. This means that $n(1+I)$ can be factored in $\mathbb{Z}[I]$, and that it is possible to construct terms of the desired form. ($N_i + m_i I$ with $\frac{m_i}{N_i}$ small) To do this let's take a closer look at the factorization in $\mathbb{Z}[I]$.

### 3.2 Factorization in $\mathbb{Z}[I]$

As mentioned before, analogous to the situation in $\mathbb{Z}$, it is possible to factor any $x = a + bI \in \mathbb{Z}[I]$ uniquely as a product of irreducible factors and units. Let

$$n = p_1^{n_1} p_2^{n_2} \dots p_t^{n_t}$$

be the unique factorization of $n$ in $\mathbb{Z}$. If we know how to factor each of the $p_i$ into irreducible factors in $\mathbb{Z}[I]$ then this will give us a way of writing $n$ as a product of irreducible factors in $\mathbb{Z}[I]$. This is the prime factorization of $n$ in $\mathbb{Z}[I]$, and the irreducible factors are primes. Some primes in $\mathbb{Z}$ are irreducible in $\mathbb{Z}[I]$ and some are not. The relevant theorem is found in [2] and will be stated here without proof:

- The units of $\mathbb{Z}[I]$ are given by $\{1, -1, I, -I\}$
- We have $2 = (-I)(1+I)^2$, with $-I$ a unit and $(1+I)$ irreducible in $\mathbb{Z}[I]$
- If $q$ is a prime in $\mathbb{Z}$ and $q \equiv 3 \bmod 4$ , then $q$ is irreducible in $\mathbb{Z}[I]$
- If $p$ is a prime in $\mathbb{Z}$ and $p \equiv 1 \bmod 4$ , then

  $$p = \pi \cdot \overline{\pi} \qquad \text{and } \pi \neq u\overline{\pi}$$

  for every unit $u \in \mathbb{Z}[I]^*$. Both $\pi$ and its complex conjugate $\overline{\pi}$ are irreducible in $\mathbb{Z}[I]$.

When we take a look at

$$n(1+I) = \prod_{i=1}^{k} (N_i + m_i I)^{p_i}$$

we have on the l.h.s. $n(1+I)$. Since $(1+I)$ is a irreducible factor, $n$ can only be factored into primes equal to 1 mod 4 in $\mathbb{Z}$. This is true since I demand $\gcd(N)i, m_i) = 1$. The only exception is 2, as 2 can be factored into $(1+I)(1-I)$. This is not hard to see: if $n$ contains a prime $p$ equal to 3 mod 4, then one of the terms on the r.h.s. must contain $p$ because $p$ is irreducible in $\mathbb{Z}[I]$. But we already demanded that $\gcd(N_i, m_i) = 1 \,\forall\, i$, so this is not possible.

### 3.3 Prime factors

In this part I will explain what the general idea is to find products of the form (14). To do this I introduce the following function, which will be called the 'norm':

$$N : \mathbb{Z}[I] \longrightarrow \mathbb{Z}, \qquad N(a+bI) = a^2 + b^2 \tag{16}$$

One can easily verify that the following statements hold: $N(\alpha) = \alpha\overline{\alpha}$, and $N(\alpha\beta) = N(\alpha)N(\beta)\ \forall\ \alpha, \beta \in \mathbb{Z}[I], N(0) = 0$ and $N(1) = 1$.

Consider now the term $n(1+I)$. It is already made clear that there is a unique prime factorization in $\mathbb{Z}[I]$ and since all primes in the factorization of $n$ in $\mathbb{Z}$ are congruent to 1 mod 4, the prime factorization of $n$ in $\mathbb{Z}[I]$ is given by

$$n(1+I) = (1+I)\pi_1^{n_1}\bar{\pi_1}^{n_1}\pi_2^{n_2}\bar{\pi_2}^{n_2}\ldots\pi_t^{n_t}\bar{\pi_t}^{n_t} \tag{17}$$

It can thus be concluded that

$$\prod_{i=1}^{k}(N_i + m_iI)^{p_i} = (1+I)\pi_1^{n_1}\bar{\pi_1}^{n_1}\pi_2^{n_2}\bar{\pi_2}^{n_2}\ldots\pi_t^{n_t}\bar{\pi_t}^{n_t} \tag{18}$$

and this means that every term $(N_i + m_iI)$ can be written as a product of $\pi_i$'s and $\bar{\pi_j}$'s:

$$N_i + m_iI = \prod \pi_j^{v_j}\bar{\pi_k}^{w_k} \tag{19}$$

with some integers $v_j, w_k$. It is important to notice :

**Theorem 2.** *Any term $(N_i + m_iI)$ cannot contain the primefactors $\pi_j$ and $\bar{\pi_j}$ at the same time.*

*Proof.* If $(N_i + m_iI)$ contains both $\pi_j$ and $\bar{\pi_j}$, then we have $(N_i + m_iI) = (a+bI)\bar{\pi_j}\pi_j$ for some $a, b$. Since $\bar{\pi_j}\pi_j = p$, where $p$ is some prime congruent to 1 mod 4, $(N_i + m_iI) = (ap + bpI)$ . But we demanded that $\gcd(N_i, m_i) = 1$, so the proof is complete □

**Corollary 1.** *Suppose the prime factorization of $(N_i + m_iI)$ contains $\pi_j$ to the power $v_j$. Then the prime factor $\bar{\pi_j}$ must occur at least $v_j$ times in other $(N_i + m_iI)$'s of the product $\prod_{i=1}^{k}(N_i + m_iI)^{p_i}$ .*

*Proof.* See (17). If $\pi_j$ occurs $m$ times in the factorization of the product, $\bar{\pi_j}$ also occurs $m$ times in the factorization of the product. □

Now the strategy that I choose becomes more clear. If we would write out the product $\prod_{i=1}^{k}(N_i + m_iI)^{p_i}$ in its prime factors, all imaginary parts cancel (except $1+I$), and the result is the integer $n$. I will approach the problem by considering the factors $(N_i + m_iI)$. In Maple it is easy to find its factorization, and once you know its factors you can look for terms that contain conjugate factors.

**Example 1.** *Consider $18 + I$. Using Maple you can find its primefactorization in $\mathbb{Z}[I]$: $18+I = I(1+2I)^2(-3+2I)$. Also look at the factorization of $32+I$ in $\mathbb{Z}[I]$: $32+I = I(1-2I)^2(5+4I)$. Calculating the product gives us :$(18+I)(32+I) = -1(-3+2I)(5+4I)5^2$.*

In this example it can be seen that the imaginary parts of the factors $1+2I$ and $1-2I$ cancel and give us $5^2$. The factors $5+4I$ and $-3+2I$ remain. To find a set of terms such that all imaginary parts cancel, I will look for $(N_i + m_i I)$ with prime factors that aren't 'too' big. In the section 'The Program' I will discuss this in more detail.

### 3.4 Finding solutions

Now the idea is to find a product $\prod_{i=1}^{k}(N_i + m_i I)^{p_i}$, in such a way that all imaginary parts cancel. When such a product is found, it will be called a solution. To do this I pick a finite set of prime factors, denoted $\mathbb{P}$. Suppose we have a factor $\alpha = (N_i + m_i I)$. Then the norm is given by $N(\alpha) = N_i^2 + m_i^2$. Once a set of primes is chosen, I find as much as possible $(N_i + m_i I)$ such that all of the primes in the factorization of $N(\alpha)$ are contained in $\mathbb{P}$. Then how to find solutions?

Actually, this is a simple problem of linear algebra. Since each prime factor $p \in \mathbb{P}$ is congruent to 1 mod 4 , $p$ factors in $\mathbb{Z}[I]$ as $\pi \cdot \overline{\pi}$ for some $\pi, \overline{\pi} \in \mathbb{Z}[I]$. This means that either $\pi|\alpha$ or $\overline{\pi}|\alpha$. We will construct a matrix in the following way:

**Step 1.** *Let $\alpha = (N_i + m_i I)$ and $N(\alpha) = N_i^2 + m_i^2$. Also let*

$$N(\alpha) = p_1^{n_1} p_2^{n_2} \dots p_t^{n_t} \tag{20}$$

*be the factorization of $N(\alpha)$ in $\mathbb{Z}$. Now we construct a vector $v_i$ with length equal to the number of primes we use. Each coordinate corresponds with one of the primes $p_j \in \mathbb{P}$. This coordinate will store the power of the corresponding prime factor in $N(\alpha)$. In short: the coordinate of the vector $v_i$ corresponding to $p_j$ is equal to $n_j$. (thus $v_i[j] = n_j$)*

**Step 2.** *Each of the integers $n_j$ will be assigned a $+$ or $-$: if $\pi_j|\alpha$, then $n_j$ will be assigned a $+$ and if $\bar{\pi_j}|\alpha$ then $n_j$ will be assigned a $-$. Here we can say without loss of generality that all $\pi_j$ have positive imaginary parts, and the $\bar{\pi_j}$ have negative imaginary parts.*

**Step 3.** *Construct a matrix with rows equal to the vectors constructed in step 1 and 2 for all $(N_i + m_i I)$.*

Here is an example to clarify this proces:

**Example 2.** *First we must choose as set of primes, and every prime must be congruent to* 1 mod 4. *In this example we will use the first 9 primes. Then we have:*

$$\mathbb{P} = \{5, 13, 17, 29, 37, 41, 53, 61, 73\}$$

*A few terms for which $N(\alpha)$ has no factors greater than 73 are*

$$\alpha = \{(5667 + I), (6118 + I), (8368 + I), (9466 + I)\}$$

*With Maple it is easy to calculate its prime factorization in $\mathbb{Z}[I]$. Looking at $(5667 + I)$ we have:*

$$(5667 + I) = (-I)(1 + I)(1 - 2I)(5 - 2I)(1 - 6I)(5 - 4I)(-3 - 8I)$$

*with $-I$ a unit. The factor $1+I$ will be ignored for now. This means the vector $v_i$ looks like*

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

*Doing the same for the other three terms we get the matrix*

$$\begin{bmatrix} -1 & 0 & 0 & -1 & -1 & -1 & 0 & 0 & -1 \\ 2 & -1 & 0 & 0 & 0 & -1 & 2 & 0 & 0 \\ 2 & 0 & -1 & 0 & -1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & -1 & 3 & 0 & 0 & -1 & -3 \end{bmatrix}$$

*and doing this for many more $N_i + m_i I$ will give us a large matrix $A$.*

Now that it is clear that the rows represent the powers of the prime factors in $\mathbb{Z}[I]$ we can state a condition for a solution.

**Theorem 3.** *Let $v_i$ be the $i$-th row of the matrix $A$, representing the factorization of $(N_i + m_i I)$, and let $\prod_{i=1}^{k}(N_i + m_i I)^{p_i}$ be a solution. then*

$$\sum_{i=1}^{k} p_i v_i = 0$$

*where 0 is the row vector with 9 zeros .*

*Proof.* Suppose $\prod_{i=1}^{k}(N_i + m_i I)^{p_i} = n(1+I)$ and $\sum_{i=1}^{k} p_i v_i = w \neq 0$. This means that the $m$-th coordinate of $w$ is nonzero for some $m$. The $m$-th coordinate of a vector $v_i$ respresents the power of the $m$-th prime of $(N_i + m_i I)$. From the product we know that $\pi_m$ and $\bar{\pi}_m$ occur equally often. Since the powers in the matrix have a minus for $\bar{\pi}_m$ and a plus for $\pi_m$ this means that $\sum_{i=1}^{k} p_i v_i = 0$. □

**Corollary 2.** *The solution must be a linear combination of elements in the Null Space of $A^T$*

*Proof.* Trivial □

We have now found a criterion for finding solutions, since Maple can easily calculate Null Spaces. So far we have considered all primes, except for $1+I$. It may well be that by calculating the Null Space not only all primes represented by the matrix cancel, but also the factors $1+I$. And we want to keep exactly one term $1+I$. For example, consider

$$(6118 + I)(4747 + I)(2673 + I)^{-1} = 10865$$

Obviously, this product satisfies the conditions, but there is no factor $1+I$ on the r.h.s. To solve this problem we must first know which factors are divisible in the ring $\mathbb{Z}[I]$ by $(1+I)$.

**Example 3.** *A term $(N_j + I)$ is divisible by $1+I$ if and only if $N_j$ is odd.*

*Proof.* $\Rightarrow$ Assume $(N_j + I)$ is divisible by $1 + I$. This means $(1 + I)(a + bI) = (N_j + I)$ for some integers $a, b$. Working out the product we have : $a - b + (a + b)I = N_j + I$. So $a + b = 1$ and therefore $a - b = N_j$ must be odd.
$\Leftarrow$ Assume $N_j$ is odd. Then $N_j + I = 2k + 1 + I$. Dividing by $1 + I$ we get

$$\frac{2k + 1 + I}{1 + I} = \frac{(2k + 1 + I)(1 - I)}{(1 - I)(1 + I)} = k + 1 - kI$$

and this is a number in the ring $\mathbb{Z}[I]$. So $2k + 1 + I$ is divisible by $1 + I$ and this proves the example. □

For a more general term $N_i + m_i I$ we also have a theorem on divisibility by $1 + I$.

**Theorem 4.** *A term $N_i + m_i I$ is divisible by $1 + I$ if and only if its norm $N(N_i + m_i I)$ is divisible by 2. Moreover, the power of 2 in the prime factorization of $N(N_i + m_i I)$ in $\mathbb{Z}$ is also the power of $1 + I$ in the factorization of $N_i + m_i I$ in $\mathbb{Z}[I]$*

*Proof.* Trivial □

**Theorem 5.** *If a product satisfies $\prod_{i=1}^{k}(N_i + m_i I)^{p_i} = n(1 + I)$ the total power of $1 + I$ in the product must be odd .*

*Proof.* If the total power of $1 + I$ was even, the product would result in an purely imaginary or a real number. □

Conversely, if the power of $1+I$ is odd, the product cannot be real or purely imaginary. Now the method for finding products of the form $\prod_{i=1}^{k}(N_i+m_i I)^{p_i} = n(1+I)$ is complete. To summarize it:

- As a base for the algorithm, we choose a set of primes congruent to 1 mod 4 which we call $\mathbb{P}$. Each prime factorizes in $\mathbb{Z}[I]$ as $\pi\bar{\pi}$.
- Next, we find many numbers of the form $N_i + m_i I$ with $\frac{m_i}{N_i}$ small that satisfy: All primes in the factorization of the norm $N(N_i + m_i I)$ in $\mathbb{Z}$ are contained in $\mathbb{P}$.
- Now that all $N_i + m_i I$ are known, we construct a matrix $A$ with the powers of all their primefactors. The powers are assigned a $-$ if the imaginary part of the prime is negative, otherwise a $+$.
- We find the nullspace of the transpose of the matrix $A$, so that we find a product that cancels all imaginary parts.
- From the products found in the previous step we select those that have in total $(1 + I)^x$ where $x$ is an odd number.

If we have an odd power of $(1 + I)$, it doesn't have to be a first power. Its always possible to raise to a power so that we have a product equal to $n(1 + I)$. However, for the algorithm this doesn't really matter because the arctangent of any power of $1 + I$ is a multiple of $\pi$ and this is what we wanted to accomplish.

### 3.5 The ring $\mathbb{Z}[\omega]$

In the same way as explained in the previous section there is another way to make approximations of $\pi$. We used a product equal to $n(1+I)$, because the arctangent of $n(1+I)$ is equal to $\frac{\pi}{4}$ and so one has an approximation of $\pi$. But since we also have

$$\tan(\frac{\pi}{6}) = \frac{1}{3}\sqrt{3} \tag{21}$$

this provides another possibility for the approximation. In this subsection we make use of the Eisenstein integers.
The Eisenstein integers are given by

$$a + b\omega \qquad \textbf{with} \qquad \omega = \frac{1}{2}(-1 + I\sqrt{3})$$

and choosing $a = 4, b = 2$ gives us in combination with (21)

$$\frac{\pi}{6} = \arg(3 + \sqrt{3}I) = 4 + 2\omega.$$

This gives us the idea to find a product of the form

$$\prod_{i=1}^{k}(a_i + b_i\omega)^{p_i} = n(2 + \omega)$$

where the $a_i + b_i\omega$ are Eisenstein integers. A few remarks must be made here:

- The Eisenstein integers form an Euclidian domain. This implies that these integers have a unique prime factorization.
- The norm is given by

$$N : \mathbb{Z}[\omega] \longrightarrow \mathbb{Z}, \qquad N(a + b\omega) = a^2 - ab + b^2$$

- The units are given by $\{1, -1, \omega, -\omega, \omega^2, -\omega^2\}$. All units have norm 1.
- If $q$ is prime in $\mathbb{Z}$ and $q \equiv 2 \bmod 3$, then $q$ is irreducible in $\mathbb{Z}[\omega]$
- If $p$ is prime in $\mathbb{Z}$ and $q \equiv 1 \bmod 3$, then

$$p = \pi\bar{\pi} \qquad \textbf{and} \qquad \pi \neq u\bar{\pi}$$

  for every unit $u \in \mathbb{Z}[\omega]^*$. Both $\pi$ and $\bar{\pi}$ are irreducible in $\mathbb{Z}[\omega]$
- If $q = 3$, then $q = -1(\sqrt{-3})^2 = -1(1 + 2\omega)^2$.

With these facts in mind, all the steps of the algorithm can be repeated for the Eisenstein integers.

# 4 Accuracy

In this section we will look at the accuracy of the approximation of $\pi$ assuming we found a product

$$\prod_{i=1}^{k}(N_i + m_i I)^{p_i} = n(1+I)$$

As mentioned before, we consider

$$\frac{\pi}{4} = p_1 \arctan\left(\frac{m_1}{N_1}\right) + p_2 \arctan\left(\frac{m_2}{N_2}\right) + p_3 \arctan\left(\frac{m_3}{N_3}\right) + \dots \tag{22}$$

where we develop the Taylor series for the arctangent, using a finite number of terms. The Taylor series of the arctangent is given by

$$\arctan(x) = \sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1} x^{2n+1}$$

So if we take a finite numbers of terms

$$\arctan(x) \approx x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} \cdots + \frac{(-1)^n}{2n+1} x^{2n+1}$$

we can estimate the remainder $R$ with the aid of the geometric series as

$$R \leq \frac{x^{2N+1}}{(2N+1)!(1-x^2)}.$$

The series is truncated at the term $x^{2N+1}$, and all terms are taken positive to obtain the above expression. What I try to do in this algorithm is finding products so that as few as possible terms in the arctangent series are needed to reach a prescribed degree of accuray. For example a hundred digits. Now a few remarks can be made:

- If we choose $x$ to be close to 0, for example $\frac{1}{10}$, $1-x^2$ comes very close to 1. ($\frac{99}{100}$) It follows that the term that determines the outcome of $R$ is $x^{2N+1}$
- Looking at (22) we see that both the number of terms and the factors $p_i$ influence the estimation of the total remainder of our approximation. However, they are both significantly less influentual than the "closeness" of $\frac{m_i}{N_i}$ to zero. For example, if you find a $\frac{m_i}{N_i}$ which is a factor 10 closer to zero, and you use 7 terms in the arctangent series, the remainder is about $10^{2N+1} = 10^{15}$ times as small. This compensates for the slightly bigger $p_i$ or the extra terms you might encounter. The most important thing here is that I want to keep the biggest $\frac{m_i}{N_i}$ as small as possible. This is how I obtain the accurate results.

It is for these reasons that it is impoprtant that the smallest $\frac{m_i}{N_i}$ is also very close to zero, because if you have a lot of very small terms and one big term, the accuracy of your

approximation still isn't optimal. Or you need a lot of terms in the arctangent series. Both are things we don't want. In the algorithm it is possible to specify the search area for solutions so that I can demand $\frac{m_i}{N_i}$ to be as small as I want. Now one can also easily calculate the maximum number of terms needed in the arctangent series to reach the desired degree of accuracy.

# 5 The Program

In this section I will shortly explain the basic steps of the program. The code of the program will be included in the section program code! The following input parameters are required:

- The "b" is the maximum value that the imaginary part of each term $N_i + m_i I$ can have. The algorithm searches all possibilities for positive values of $m_i$ equal or less than "b".
- The "r" is the number of primes in $\mathbb{Z}$ we use.
- The "z"and "y" give the range of the real part of $N_i + m_i I$ in which we search for solutions.

It can now also be seen that the proportion $\frac{m_i}{N_i}$ can be chosen when specifying the parameters $z, b$. The program consists of 5 main sections:

- The section "Priemen" makes a list of primes in $\mathbb{Z}$ that will be used as a base for the algorithm.
- The section "MaakMatrix" finds all suitable terms $N_i + m_i I$ and constructs a matrix with all their powers of the prime factors in $\mathbb{Z}[I]$ or $\mathbb{Z}[\sqrt{-3}]$ as discussed in the section "Finding Solutions".
- The section "SmithProcedure" solves the linear problem of finding the nullspace of $A^T$, where $A$ is the matrix constructed in the previous step. The Smith-procedure is used here because it finds integer solutions and the nullspace is represented in such a way that it is easy to see which solutions are the best.
- The sections "Oplossingen Spelled out" and "Oplossingen in matrix" handle the layout of the solutions: Oplossingen Spelled out gives a matrix where all solutions are explicitly denoted.
- The section "Nauwkeurigheid" calculates how many decimals of $\pi$ are correct when using a specified number of terms in the arctangent series.

## 6 The Results

In this section some results will be shown. Since the code of the program is included, one can pursue better solutions if needed. Here are some of the findings when looking at $\mathbb{Z}[I]$:

$$n(1+I) = (278+I)^{37}(268+I)^{9}(255+I)^{19}(191+I)^{-14}(157+I)^{23}(117+I)^{7}(50+I)^{19}(32+I)$$

Using 31 terms of the arctangent will give an accuracy of $10^{-97}$. Or, if one let's the computer calculate a bit longer:

$$\begin{aligned} n(1+I) = (114669+I)^{-1484}&(85353+I)^{-2097}(72662+I)^{-581}(48737+I)^{-2805}(44179+I)^{1592} \\ &(34208+I)^{-1042}(17557+I)^{-4194}(14773+I)^{-5029}(12943+I)^{-1950}(9466+I)^{398} \end{aligned}$$

which reaches an accuracy of $10^{-98}$ with only 12 terms of the arctangent series!

The slightly adapted version of the algorithm does the same in the ring of Eisenstein integers:

$$\begin{aligned} n(\frac{1}{2}+\frac{1}{2}\sqrt{3}I) = (\frac{5639}{2}+\frac{1}{2}\sqrt{3}I)^{-18}&(\frac{5259}{2}+\frac{1}{2}\sqrt{3}I)^{10}(\frac{4545}{2}+\frac{1}{2}\sqrt{3}I)^{5}(\frac{3633}{2}+\frac{1}{2}\sqrt{3}I)^{-48} \\ &(\frac{1437}{2}+\frac{1}{2}\sqrt{3}I)^{10}(\frac{1125}{2}+\frac{1}{2}\sqrt{3}I)^{-20}(\frac{163}{2}+\frac{1}{2}\sqrt{3}I)^{-48}(\frac{61}{2}+\frac{1}{2}\sqrt{3}I) \end{aligned}$$

where the accuracy is $10^{-96}$ when using 30 terms of the arctangent. Letting the computer calculate more, we have

$$\begin{aligned} &n(\frac{1}{2}+\frac{1}{2}\sqrt{3}I) = \\ &(\frac{534333}{2}+\frac{1}{2}\sqrt{3}I)^{1680}(\frac{390371}{2}+\frac{1}{2}\sqrt{3}I)^{-432}(\frac{357975}{2}+\frac{1}{2}\sqrt{3}I)^{680}(\frac{354505}{2}+\frac{1}{2}\sqrt{3}I)^{-5332} \\ &(\frac{121635}{2}+\frac{1}{2}\sqrt{3}I)^{3225}(\frac{100693}{2}+\frac{1}{2}\sqrt{3}I)^{-3156}(\frac{65839}{2}+\frac{1}{2}\sqrt{3}I)^{-1636}(\frac{50727}{2}+\frac{1}{2}\sqrt{3}I)^{980} \\ &(\frac{43491}{2}+\frac{1}{2}\sqrt{3}I)^{-242}(\frac{42233}{2}+\frac{1}{2}\sqrt{3}I)^{-6312}(\frac{34431}{2}+\frac{1}{2}\sqrt{3}I)^{-4332} \end{aligned}$$

which needs only 11 terms to reach $10^{-97}$.

# 7 Maple Code

## 7.1 Gaussian Integers

```
#Pakketten

restart;
with(GaussInt);
with(numtheory);
with(LinearAlgebra);



#Priemen

LengteLijst := proc (a::set)::integer;
                    local i, j;
                    description "Bepaalt de lengte van
                    een lijst; het laatste getal komt verder
                    niet in de lijst voor";
                    j := 1;
                        for i while a[i] <> a[-1] do j := i+1 end do; j
                end proc;

LijstPriemen := proc (a::integer)::set;
                      local i, listprime;
                      description "Maakt lijst met geschikte priemen ";
                      listprime := {5};
                            for i from 6 to infinity while
                            LengteLijst(listprime) < a
                            do if `mod`(nextprime(i), 4) = 1
                                then listprime := `union`(listprime, {nextprime(i)})
                                end if
                            end do; listprime
                  end proc;

LijstIPriemen := proc (a::set)::Array;
                       local i, listiprime, b;
                       description "Maakt lijst priemen in C ";
                       b := LengteLijst(a); listiprime := Array(1 .. b);
                                  for i to b do
                                  listiprime[i] := GIfacset(a[i])[1] end do;
                       listiprime
                  end proc;
```

```
#MaakMatrix

Lijst := proc (a::integer, b::integer, d::integer, e::integer)::set;
             local i, sum, f, g; global Priemen;
             description "Maakt een lijst van getallen in [a,b] waarvoor
             het kwadraat +1 geen priemen groter 73 bevat";
             Priemen := LijstPriemen(e); sum := {}; f := Priemen[-1]+1;
                for i from a to b do if gcd(i, d) < 3
                        then if factorset(i^2+d^2)[-1] < f
                                then sum := `union`(sum, {i})
                                else sum := sum end if
                        end if
                end do; sum
         end proc;

LengteArr := proc (a::integer, b::integer)::integer;
                   local i, j, k;
                   description "Bepaalt de lengte van een
                   Array, gegeven door GIfactors";
                   j := 1; k := GIfactors(a+I*b);
                          for i while k[2, i, 1] <> k[2, -1, 1]
                          do j := i+1 end do;
                   if k[2, 1, 1] = 1+I
                          then j := j-1
                          else j := j end if; j
             end proc;

Machten := proc (a::integer, b::integer)::array;
               local i, j, k, l, n, f; global Priemen;
               description "Maakt een Array met daarin de machten
               van de priemen waarin a+I zich ontbindt";
               f := LengteLijst(Priemen); n := Array(1 .. f);
               j := LengteArr(a, b); k := GIfactors(a+I*b);
                   for i to j do if k[2, 1, 1] <> 1+I
                          then for l to f do if abs(k[2, i, 1])^2 = Priemen[l]
                               then n[l] := k[2, i, 2] end if
                          end do
                               else for l to f do if abs(k[2, i+1, 1])^2 = Priemen[l]
                               then n[l] := k[2, i+1, 2] end if
```

```
                        end do end if
                end do; n
        end proc;

Ontbinding := proc (a::set, b::integer, d::integer)::array;
                local i, j, l, m, n, o, p, q, f; global Priemen;
                description "Geeft de macht een - als Im<0
                en een + als Im>0 voor het i-de element
                in de lijst";
                f := LengteLijst(Priemen);
                i := GIfacset(a[b]+I*d);
                l := LengteLijst(i); o := Machten(a[b], d);
                p := LengteLijst(a); q := Array(1 .. p, 1 .. f);
                     for j to l do m := Im(i[j])/abs(Im(i[j]));
                          for n to f do if abs(i[j])^2 = Priemen[n]
                                   then q[b, n] := m*o[n] end if
                              end do
                         end do; q
           end proc;

Ontbindt := proc (a::set, b::integer)::array;
                local i, r, p, d, f; global Priemen;
                description "Maakt een Array van de priemontbinding";
                f := LengteLijst(Priemen); d := LengteLijst(c);
                p := LengteLijst(a); r := Array(1 .. p, 1 .. f);
                        for i to d do r := r+Ontbinding(a, i, b) end do; r
         end proc;

MaakMatrix := proc (a::integer, b::integer, d::integer, g::integer)::Matrix;
                local A, B, r, e, i, f; global c, Priemen;
                description "Maakt de Array een Matrix een spiegelt de rijen";
                c := Lijst(a, b, d, g); f := LengteLijst(Priemen);
                e := Ontbindt(c, d); A := convert(e, Matrix);
                r := LengteLijst(c); B := Matrix(r, f);
                         for i to r do B[i] := A[r+1-i] end do; B
           end proc;


#Smithprocedure

SmithHulp := proc (a::set)::Matrix;
                local i, b, A, f; global Priemen;
                description "Maakt de Smith-hulpmatrix";
```

```
                f := LengteLijst(Priemen); b := LengteLijst(a);
                A := Matrix(b, b);
                     for i to b-f do A[i+f, i] := 1 end do; A
            end proc;

Spiegel := proc (a::set)::Matrix;
               local i, b, A;
               description "Maakt een Matrix met enen op de tegengestelde
               diagonaal";
               b := LengteLijst(a); A := Matrix(b, b);
                       for i to b do A[i, b+1-i] := 1 end do; A
          end proc;

SmithProcedure := proc (A::Matrix)::Matrix;
                      local B, X, Xveeg, C, E, K; global c, V;
                      description "Maakt oplossingen met Smith
                             en veegt de Matrixl";
                      B := Transpose(A);
                      X := MatrixMatrixMultiply(V, SmithHulp(c));
                      Xveeg := Transpose(X);
                      C := MatrixMatrixMultiply(Xveeg, Spiegel(c));
                      E := GaussianElimination(C, 'method' = 'FractionFree');
                      K := MatrixMatrixMultiply(E, Spiegel(c)); K
                end proc;


#Oplossingen 'In Matrix'

Richting := proc (a::set, d::integer)::Matrix;
                 local i, b, A;
                 description "Bepaalt of de ontbinding van een element 1+I bevat";
                 b := LengteLijst(a); A := Vector[column](b);
                         for i to b do if GIfactors(a[i]+I*d)[2, 1, 1] = 1+I
                                       then A[b+1-i] := GIfactors(a[i]+I*d)[2, 1, 2]
                                       end if;
                         end do; A
           end proc;

RichtingOplossing := proc (a::Matrix, b::set, e::integer)::Matrix;
                         local i, d, A, Y; global c;
                         description "Bepaalt of een oplossing 'netto' een
                                                  factor 1+I bevat";
                         d := LengteLijst(c); A := Richting(b, e);
```

```
                        Y := MatrixVectorMultiply(a, A);
                                for i to d do if ‘mod‘(Y[i], 2) = 0
                                          then Y[i] := 0
                                           else Y[i] := ‘mod‘(Y[i], 8) end if
                                end do; Y
                    end proc;

OplossingMatrix := proc (a::Vector, b::Matrix)::Matrix;
                        local j, k, Z, m, r, s; global c;
                        description "Zet oplossingen met 1+I in een Matrix";
                        k := 0; s := LengteLijst(c);
                               for j to s do if a[j] = 0
                                         then k := k
                                         else k := k+1 end if
                               end do;
                        m := 1; Z := Matrix(k, s);
                               for r to s do if ‘mod‘(a[r], 2) = 1
                                         then Z[m] := b[r]*(‘mod‘(a[r], 2));
                                         m := m+1 end if
                               end do; Z
                  end proc;

Oplossingen := proc (a::Matrix, b::integer)::Matrix;
                      local OM, Y; global c;
                      description "Voert de bovenstaande procedures uit";
                      Y := RichtingOplossing(a, c, b); OM := OplossingMatrix(Y, a); OM
               end proc;


#Oplossingen 'Spelled Out'

Factoren := proc (a::set, e)::set;
                 local i, b, d;
                 description "Maakt de vectoren n+I, met n een element uit de lijst";
                 b := LengteLijst(a); d := {};
                            for i to b do d := ‘union‘(d, {c[i]+I*e}) end do; d
            end proc;

MatrixMaala := proc (a::set)::Matrix;
                      local i, b, A, l;
                      description "Maakt Matrix met x-en op de diagonaal";
                      b := LengteLijst(a); A := Matrix(b, b);
                                  for i to b do A[i, i] := x end do;  A
```

```
              end proc;

Opl := proc (a::set, b::set, o::Vector)::Matrix;
             local r, gj, n;
             description "Maakt een specifieke oplossing";
             n := LengteLijst(a); gj := 1;
                    for r to n do gj := gj*b[n+1-r]^o[r] end do; gj
       end proc;

OplTot := proc (o::Matrix, b::integer)::Matrix;
                local v, yz, w, fc, p, m; global c;
                description "Zet alle oplossingen in een vector ";
                p := MatrixMaala(c); fc := Factoren(c, b);
                m := MatrixMatrixMultiply(o, p);
                v := RowDimension(o); yz := Vector[column](1 .. v);
                          for w to v do yz[w] := Opl(c, fc, m[w]) end do; yz
          end proc;


#Nauwkeurigheid

Nauwkeurigheid := proc (o::Matrix, e::integer, l::integer)::Matrix;
                      local check, hal, rty, g, h; global c, b;
                      description "Berekent nauwkeurigheid benadering ";
                      g := taylor(arctan(x), x = 0, 2*l); hal := 0;
                      h := LengteLijst(c);
                              for rty to h do
                              hal := hal+OIM[e, rty]*(eval(g, x = b/c[h+1-rty]))
                              end do;
                      evalf[150](4*hal+Pi)
                   end proc;
```

## 7.2 Eisenstein Integers

```
#Priemen

LengteLijst := proc (a::set)::integer;
                  local i, j;
                  description "Bepaalt de lengte van een lijst;
                  het laatste getal komt verder niet in de lijst voor";
                  j := 1;
                        for i while a[i] <> a[-1] do j := i+1 end do; j
              end proc;
```

```
LijstPriemen := proc (a::integer)::set;
                    local i, listprime;
                    description "Maakt lijst met geschikte priemen ";
                    listprime := {7};
                        for i from 8 to infinity
                        while LengteLijst(listprime) < a
                        do if `mod`(nextprime(i), 3) = 1
                                then listprime :=
                              `union`(listprime, {nextprime(i)}) end if
                        end do; listprime
                end proc;

LijstIPriemen := proc (a::set)::Array;
                    local i, b, z, f, g; global listiprime;
                    description "Maakt lijst priemen in C ";
                    z := f+(1/2)*g+((1/2)*I)*sqrt(3)*g;
                    b := LengteLijst(a); listiprime := Array(1 .. b);
                         for i to b do listiprime[i] :=
                         subs(isolve(f^2+f*g+g^2 = a[i])[1], z)
                         end do; listiprime
                 end proc;


#MaakMatrix

Lijst := proc (a::integer, b::integer, d::integer, e::integer)::set;
              local i, sum, f, g; global Priemen;
              description "Maakt een lijst van getallen in [a,b]
              waarvoor het kwadraat +1 geen priemen groter 73 bevat";
              Priemen := LijstPriemen(e); sum := {};
              f := Priemen[-1]+1;
                     for i from a to b do if gcd(i, d) < 3
                           then if factorset(i^2+i*d+d^2)[-1] < f
                                  then sum := `union`(sum, {i})
                                  else sum := sum end if
                           else  end if
                     end do; sum
          end proc;

LengteArr := proc (a::integer, b::integer)::integer;
                 local i, j, k;
                 description "Bepaalt de lengte van een Array,
```

```
                gegeven door GIfactors";
                k := factorset(a^2+a*b+b^2);
                j := LengteLijst(k); j
        end proc;

Machten := proc (a::integer, b::integer)::array;
               local i, j, k, l, n, f; global Priemen, listiprime;
               description "Maakt een Array met daarin de machten
               van de priemen waarin a+I zich ontbindt";
               f := LengteLijst(Priemen); n := Array(1 .. f);
               j := LengteArr(a, b); k := ifactors(a^2+a*b+b^2);
                      for i to j do
                          for l to f do if k[2, i, 1] = Priemen[l]
                                then if type(2*Re(expand((a+1/2+((1/2)*I)*
                                sqrt(3)*b)*listiprime[l]/abs(listiprime[l])^2)),
                                integer) = true
                                      then n[l] := -k[2, i, 2]
                                      else n[l] := k[2, i, 2] end if end if
                           end do
                       end do; n
           end proc;

Ontbinding := proc (a::set, b::integer, d::integer)::array;
                 local i, j, l, m, n, o, p, q, f; global Priemen;
                 description "Geeft de macht een - als Im<0 en
                 een + als Im>0 voor het i-de element in de lijst";
                 f := LengteLijst(Priemen); i := LengteArr(a[b], d);
                 o := Machten(a[b], d); p := LengteLijst(a);
                 q := Array(1 .. p, 1 .. f);
                           for n to f do q[b, n] := o[n]
                           end do; q
            end proc;

Ontbindt := proc (a::set, b::integer)::array;
                local i, r, p, d, f; global Priemen;
                description "Maakt een Array van de priemontbinding";
                f := LengteLijst(Priemen); d := LengteLijst(c);
                p := LengteLijst(a); r := Array(1 .. p, 1 .. f);
                       for i to d do r := r+Ontbinding(a, i, b)
                       end do; r
           end proc;

MaakMatrix := proc (a::integer, b::integer, d::integer, g::integer)::Matrix;
```

```
                local A, B, r, e, i, f; global c, Priemen, listiprime;
                description "Maakt de Array een Matrix een spiegelt de rijen";
                c := Lijst(a, b, d, g); listiprime := LijstIPriemen(LijstPriemen(g));
                f := LengteLijst(Priemen); e := Ontbindt(c, d);
                A := convert(e, Matrix); r := LengteLijst(c);
                B := Matrix(r, f);
                        for i to r do B[i] := A[r+1-i]
                        end do; B
            end proc;


#Smithprocedure

SmithHulp := proc (a::set)::Matrix;
                local i, b, A, f; global Priemen;
                description "Maakt de Smith-hulpmatrix";
                f := LengteLijst(Priemen); b := LengteLijst(a);
                A := Matrix(b, b);
                       for i to b-f do A[i+f, i] := 1
                       end do; A
             end proc;

Spiegel := proc (a::set)::Matrix;
               local i, b, A;
               description "Maakt een Matrix met enen op de
               tegengestelde diagonaal";
               b := LengteLijst(a); A := Matrix(b, b);
                        for i to b do A[i, b+1-i] := 1
                        end do; A
           end proc;

SmithProcedure := proc (A::Matrix)::Matrix;
                    local B, X, Xveeg, C, E, K; global c, V;
                    description "Maakt oplossingen met Smith en
                    veegt de Matrixl";
                    B := Transpose(A);
                    X := MatrixMatrixMultiply(V, SmithHulp(c));
                    Xveeg := Transpose(X);
                    C := MatrixMatrixMultiply(Xveeg, Spiegel(c));
                    E := GaussianElimination(C, 'method' = 'FractionFree');
                    K := MatrixMatrixMultiply(E, Spiegel(c)); K
                end proc;
```

```
#Oplossingen 'In Matrix'

HulpR := proc (a::set, d::integer, e::integer)::integer;
              local b, f, B, prod, i, j, k, p; global listiprime, Priemen;
              b := LengteLijst(a); f := LengteLijst(Priemen);
              B := Machten(a[d], e);
              prod := 1; p := ifactors(a[d]^2+a[d]*e+e^2);
                     for i to f do if B[i] < 0
                                 then prod := prod*conjugate(listiprime[i])^(-B[i])
                                 else prod := prod*listiprime[i]^B[i] end if
                     end do;
                     if p[2, 1, 1] = 3
                            then prod := prod*(3/2+((1/2)*I)*sqrt(3))^p[2, 1, 2]
                            else prod := prod end if;
               j := 0;
                     for k to 7 do if expand(prod*(1/2+((1/2)*I)*sqrt(3))^k) =
                     a[d]+1/2+((1/2)*I)*e*sqrt(3)
                           then j := k end if
                     end do; `mod`(j, 6)
        end proc;

HulpRichting := proc (a::set, d::integer)::Matrix;
                   local i, b, A; global listiprime, Priemen;
                   description "Bepaalt of de ontbinding van een
                   element 1+I bevat";
                   b := LengteLijst(a); A := Vector[column](b);
                      for i to b do A[b+1-i] := HulpR(a, i, d)
                      end do; A
                end proc;

Richting := proc (a::set, d::integer)::Matrix;
                  local i, b, A; global listiprime, Priemen;
                  description "Bepaalt of de ontbinding van een
                  element 1+I bevat";
                  b := LengteLijst(a); A := Vector[column](b);
                      for i to b do if ifactors(a[i]^2+a[i]*d+d^2) [2, 1, 1] = 3
                            then A[b+1-i] := ifactors(a[i]^2+a[i]*d+d^2)[2, 1, 2 ]
                            end if
                      end do; A
             end proc;
```

```
RichtingOplossing := proc (a::Matrix, b::set, e::integer)::Matrix;
                          local i, d, A, Y; global c;
                          description "Bepaalt of een oplossing 'netto'
                          een factor 1+I bevat";
                          d := LengteLijst(c); A := Richting(b, e);
                          Y := MatrixVectorMultiply(a, A);
                               for i to d do if `mod`(Y[i], 6) = 0
                                      then Y[i] := 0
                                      else Y[i] := `mod`(Y[i], 6) end if
                               end do; Y
                     end proc;

HulpRichtingOplossing := proc (a::Matrix, b::set, e::integer)::Matrix;
                            local i, d, A, Y; global c;
                            description "Bepaalt of een oplossing 'netto' een
                            factor 1+I bevat";
                            d := LengteLijst(c); A := HulpRichting(b, e);
                            Y := MatrixVectorMultiply(a, A);
                                     for i to d do if `mod`(Y[i], 6) = 0
                                           then Y[i] := 0
                                           else Y[i] := `mod`(Y[i], 6) end if
                                     end do; Y
                       end proc;

OplossingMatrix := proc (a::Vector, d::Vector, b::Matrix)::Matrix;
                           local j, k, Z, m, r, s; global c;
                           description "Zet oplossingen met 1+I in een Matrix";
                           k := 0; s := LengteLijst(c);
                                 for j to s do if `mod`(a[j]+2*d[j], 3) = 0
                                          then k := k
                                          else k := k+1 end if
                                 end do;
                           m := 1; Z := Matrix(k, s);
                                 for r to s do if `mod`(a[r]+2*d[r], 3) <> 0
                                          then Z[m] := b[r]; m := m+1 end if
                                 end do; Z
                     end proc;

Oplossingen := proc (a::Matrix, b::integer)::Matrix;
                      local OM, Y, X; global c;
                      description "Voert de bovenstaande procedures uit";
                      Y := RichtingOplossing(a, c, b);
                      X := HulpRichtingOplossing(a, c, b);
```

```
                    OM := OplossingMatrix(Y, X, a); OM
                end proc;



#Oplossingen 'Spelled Out'

 Factoren := proc (a::set, e)::set;
                  local i, b, d;
                  description "Maakt de vectoren n+I, met n een element
                  uit de lijst";
                  b := LengteLijst(a); d := {};
                       for i to b do d :=
                     `union`(d, {c[i]+(1/2)*e+((1/2)*I)*sqrt(3)*e})
                       end do; d
             end proc;

MatrixMaala := proc (a::set)::Matrix;
                  local i, b, A, l;
                  description "Maakt Matrix met x-en op de diagonaal";
                  b := LengteLijst(a); A := Matrix(b, b);
                     for i to b do A[i, i] := x
                     end do; A
               end proc;

Opl := proc (a::set, b::set, o::Vector)::Matrix;
                local r, gj, n;
                description "Maakt een specifieke oplossing";
                n := LengteLijst(a); gj := 1;
                       for r to n do gj := gj*b[n+1-r]^o[r]
                       end do; gj
       end proc;

OplTot := proc (o::Matrix, b::integer)::Matrix;
                local v, yz, w, fc, p, m; global c;
                description "Zet alle oplossingen in een vector ";
                p := MatrixMaala(c); fc := Factoren(c, b);
                m := MatrixMatrixMultiply(o, p); v := RowDimension(o);
                yz := Vector[column](1 .. v);
                   for w to v do yz[w] := Opl(c, fc, m[w])
                   end do; yz
         end proc;
```

```
#Nauwkeurigheid

Nauwkeurigheid := proc (o::Matrix, e::integer, l::integer)::Matrix;
                    local check, hal, rty, g, h; global c, b;
                    description "Berekent nauwkeurigheid benadering ";
                    g := taylor(arctan(x), x = 0, 2*l); hal := 0;
                    h := LengteLijst(c);
                         for rty to h do hal := hal+
                         OIM[e, rty]*(eval(g, x = b*sqrt(3)/(2*c[h+1-rty]+b)))
                         end do; evalf[150](3*hal+(1/2)*Pi)
              end proc;
```

## References


[1] P. Beckmann, *A History of* $\pi$, (1971), Boulder, Co; Golem Press, ISBN 0-911762-12-4.

[2] Jack S. Calcut, *Gaussian Integers and Arctangent Identities for* $\pi$, Amer. Math. Monthly, **116** (2009), 515–530.

[3] B. van Geemen, H.W. Lenstra & F. Oort, *Algebra, Ringen en Lichamen*, Rijksuniversiteit Groningen (1997)