HAL
open science

On the sign of a trigonometric expression

Pierre-Vincent Koseleff, Fabrice Rouillier, Cuong Tran

To cite this version:

Pierre-Vincent Koseleff, Fabrice Rouillier, Cuong Tran. On the sign of a trigonometric expression. 2015. hal-01108656

HAL Id: hal-01108656
https://inria.hal.science/hal-01108656

Preprint submitted on 28 Jan 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

On the sign of a trigonometric expression

Pierre-Vincent Koseleff
INRIA, Paris-Rocquencourt, Ouragan
UPMC-Sorbonne Universités
CNRS, UMR 7586, IMJ-PRG
pierre-vincent.koseleff@imj-prg.fr

Fabrice Rouillier
INRIA, Paris-Rocquencourt, Ouragan
UPMC-Sorbonne Universités
CNRS, UMR 7586, IMJ-PRG
fabrice.rouillier@inria.fr

Cuong Tran
UPMC-Sorbonne Universités
CNRS, UMR 7586, IMJ-PRG
cuong.tran@imj-prg.fr

ABSTRACT

We propose a set of simple and fast algorithms for evaluating and using trigonometric expressions in the form $F = \sum_{k=0}^{d} f_k \cos k\frac{\pi}{n}$, $f_k \in \mathbf{Q}$, $d < n$ for a fixed $n \in \mathbf{Z}_{>0}$: computing the sign of such an expression, evaluating it numerically and computing its minimal polynomial in $\mathbf{Q}[x]$. As critical byproducts, we propose simple and efficient algorithms for performing arithmetic operations (multiplication, division, gcd) on polynomials expressed in a Chebyshev basis (with the same bit-complexity than in the monomial basis) and for computing the minimal polynomial of $2\cos\frac{\pi}{n}$ in $\widetilde{\mathcal{O}}(n_0^2)$ bit operations with $n_0 < n$ is the odd squarefree part of n. Within such a framework, we can decide if $F = 0$ in $\widetilde{\mathcal{O}}(d(\tau+d))$ bit operations , compute the sign of F in $\widetilde{\mathcal{O}}(d^2\tau)$ bit operations and compute the minimal polynomial of F in $\widetilde{\mathcal{O}}(n^3\tau)$ bit operations, where τ denotes the maximum bitsize of the f_k's.

Keywords

Chebyshev polynomial, minimal polynomial, cyclotomic polynomial, root isolation

1. INTRODUCTION

The present contribution was initially motivated by the computation of Chebyshev knots with the objective of decreasing the complexity bound of the algorithm proposed in [12]. The bottleneck for this algorithm was the computation of the sign of a multivariate polynomial at the zeroes of a zero-dimensional system. It has been shown later ([11]) that this problem finally reduces to the computation of the sign of a polynomial with rational coefficients at some particular real points in the form $\cos\frac{\pi}{n}$, $n \in \mathbf{Z}_{>0}$, which can easily be turn into the computation of the sign of an expression in the form $F = \sum_{k=0}^{d} f_k \cos k\frac{\pi}{n}$, $f_k \in \mathbf{Z}$ for a fixed n in $\mathbf{Z}_{>0}$.

Evaluating numerically such an expression can be done straightforwardly, but getting its sign (in particular deciding if it is null or not) or computing its minimal polynomial to follow using it for further exact computations need some efforts to be done efficiently in practice.

As $F = f(2\cos\frac{\pi}{n})$ with $f(x) = f_0 + \sum_{k=1}^{d} f_k T_k(x)$, where $T_k(x) \in \mathbf{Z}[x]$ is the k-th monic Chebyshev polynomial, evaluating F can be seen as evaluating $f \in \mathbf{Q}[x]$ at the real value $2\cos\frac{\pi}{n}$, testing if $F = 0$ can resume to testing if the minimal polynomial $M_n \in \mathbf{Q}[x]$ of $2\cos\frac{\pi}{n}$ divides f and, finally, computing the minimal polynomial $M_F \in \mathbf{Q}[x]$ of F remains to computing the minimal polynomial M_f of f in $\mathbf{Q}[x]/(M_n)$.

Our primary objective was thus to find a way for computing efficiently $M_n \in \mathbf{Q}[x]$, the minimal polynomial of $2\cos\frac{\pi}{n}$ in order, for example, to use state of the art algorithms for testing if $F = 0$ (which then resumes to a single polynomial division) or computing M_F (for example using [15]).

The algorithm we propose in section 4 for computing M_n is an adaptation of the algorithm from [1] that basically computes Φ_n, the minimal polynomial of the n-th primitive root of unity. The relation between Φ_n and M_n is simple: $\Phi_{2n} = \mathcal{D}(M_n)$, where $\mathcal{D}$ is the transformation $\mathbf{Q}[x]$ defined by $\mathcal{D}(P) = x^{\deg P} P(x + \frac{1}{x})$.

This simple transformation is also used in Section 2 to set simple and fast algorithms for multiplying, dividing, computing greatest common divisors for polynomials expressed in the Chebyshev basis. In particular, we provide, for the multiplication, a very simple alternative to [7] with an equivalent bit complexity. More generally, the complexity of these operations is the same as for polynomials in the monomial basis (see [16]).

It turns out that expressing the polynomials in the Chebyshev basis instead of in the usual monomial basis simplifies some parts of the algorithms and make easier the computation of complexity bounds. For example, we show that the computation of M_n can be done in $\widetilde{\mathcal{O}}(n_0^2)$ bit operations where $n_0 < n$ is the odd squarefree part of n.

The efficiency of this algorithm depends also on the way the transformations from the Chebyshev basis to the monomial basis and vice-versa are computed. In order not to reinvent the wheel, we have considered these transformations as a particular case of composition method developed in [9] with additional refinements linked to the properties of Chebyshev polynomials.

In section 3, we show that passing from the monomial basis to the Chebyshev basis consists essentially to inject in the algorithm from [9] the fast operations we have introduced in Section 2. For the reverse operation, we make use of a the linear recurrence introduced in [4] and essentially keep the structure of the composition algorithm from [9], thanks to some remarkable properties of Chebyshev polynomials. We finally show that the two transformations we propose perform $\widetilde{\mathcal{O}}(d)$ arithmetic operations in $\mathbf{Z}$ and $\widetilde{\mathcal{O}}(d(\tau + d))$ bit operations if τ is the maximal bitsize of the f_k's.

In section 5, we first come back to our initial problem and propose some algorithms together with their complexity analysis. We show that we can test whether $F = 0$ in $\mathcal{O}(n^2 + n\tau)$ binary operations and can decide the sign of F in $\widetilde{\mathcal{O}}(n^2\tau)$ binary operations, using an algorithm of [6] for a

fast evaluation of $\cos x$.

As F could also be used inside other algebraic expressions, we propose in section 6 an algorithm for computing efficiently the minimal polynomial of F in $\widetilde{\mathcal{O}}(n^3\tau)$ binary operations.

2. CHEBYSHEV POLYNOMIALS

As seen in the introduction, Chebyshev polynomials and their algebraic properties play a central role in the present contribution. We consider the *Chebyshev monic polynomials* defined by

$$T_n(x) = 2\cos nt, \quad U_n(x) = \frac{\sin nt}{\sin t}$$

where $x = 2\cos t$. Both of them satisfy the linear recurrence relation:

$$P_{n+1}(x) = xP_n(x) - P_{n-1}(x), \tag{1}$$

but their first terms differ:

$$\{T_0(x) = 2, T_1(x) = x\}, \quad \{U_0(x) = 0, U_1(x) = 1\}.$$

The roots of T_n and U_n are real roots and we have:

$$T_n(x) = \prod_{k=0}^{n-1}(x - 2\cos\tfrac{(2k+1)\pi}{2n}), \tag{2a}$$

$$U_n(x) = \prod_{k=1}^{n-1}(x - 2\cos\tfrac{k\pi}{n}). \tag{2b}$$

By derivating the relation $T_n(2\cos t) = 2\cos nt$, we obtain $\sin t\, T_n'(2\cos t) = n\sin nt$ that is

$$T_n' = nU_n. \tag{3}$$

By derivating once again we get

$$4(\sin t)^2\, T_n''(2\cos t) + 2\cos t\, T_n'(2\cos t) = 2n^2\cos nt,$$

or

$$(4 - x^2)T_n''(x) - xT_n'(x) + n^2T_n(x) = 0. \tag{4}$$

It means that T_n is an hypergeometric function. By identifying the coefficients, we get

$$T_n(x) = \sum_{k=0}^{\lfloor\frac{n}{2}\rfloor}(-1)^k\tfrac{n}{n-k}\tbinom{n-k}{k}x^{n-2k}. \tag{5}$$

On the other hand, from that $x = 2\cos t$ we get $x^n = \Big(\exp it + \exp -it\Big)^n$ whose expansion gives

$$x^n = \sum_{k=0}^{\lfloor\frac{n}{2}\rfloor-1}\tbinom{n}{k}T_{n-2k}(x) + \tfrac{1+(-1)^n}{2}, \tag{6}$$

From Equation (3) and Formula (5), we also obtain

$$U_{n+1}(x) = \sum_{k=0}^{\lfloor\frac{n}{2}\rfloor}(-1)^k\tbinom{n-k}{k}x^{n-2k}. \tag{7}$$

The family $\mathcal{T} = (1, T_j, j \geq 1)$ is therefore a basis of $\mathbf{Z}[x]$. We will also denote by $\mathcal{X}$ the monomial basis $(x^n, n \geq 0)$.

Given a basis $\mathcal{B} = (B_i)_{i\geq 0}$ of $\mathbf{Q}[x]$ and a polynomial P of degree d, we denote by $\tau_{\mathcal{B}}(P)$ the maximum bitsize of its coefficients $[\mathcal{B}]_iP$ in the basis $\mathcal{B}$.

Proposition 1. *We have $\tau_{\mathcal{X}}(T_n) \leq n$, $\tau_{\mathcal{X}}(U_n) \leq n-1$, $\tau_{\mathcal{T}}(x^n) \leq n$.*

Proof. From $\binom{n-k}{k} \leq \frac{n}{n-k}\binom{n-k}{k}$, we obtain first that $\tau_{\mathcal{X}}(U_{n+1}) \leq \tau_{\mathcal{X}}(T_n)$, using Formulas (5) and (7). The sum of the absolute values of the coefficients of T_n is $\sum_k \frac{n}{n-k}\binom{n-k}{k} = \phi^n + (-1/\phi)^n$, where ϕ is the golden ratio. We thus deduce that $[\mathcal{X}]_kT_n \leq \varphi^n + 1 \leq 2^n$ and $\tau_{\mathcal{X}}(T_n) \leq n$. We get $\tau(x^n) \leq n$ because that $\sum_k \binom{n}{k} = 2^n$. $\square$

We thus deduce

Corrolary 2. *Let P be a polynomial of degree d. Then we have $\tau_{\mathcal{X}}(P) \leq \tau_{\mathcal{T}}(P) + d + \log_2 d$, and $\tau_{\mathcal{T}}(P) \leq \tau_{\mathcal{X}}(P) + d + \log_2 d$.*

Chebyshev forms

Let $f \in \mathbf{Q}[x]$. We will say that f is in *Chebyshev form* when it is expressed in the Chebyshev basis: $[\mathcal{T}]f = f_0 + \sum_{j=1}^d f_jT_j$. The set of Chebyshev forms with rational coefficients is an Euclidean ring and we focus on computing efficiently in this ring, starting with elementary operations.

Let us introduce a simple shortcut that will straightforwardly offer a simple link between basic operations in the Chebyshev basis and in the usual monomial basis:

Definition 3. *$\mathcal{D}$ is the transformation from $\mathbf{Q}[x]$ to the space of even degree reversible polynomials given by*

$$\mathcal{D}(P) = x^{\deg P}\, P(x + \tfrac{1}{x}).$$

Note that $\mathcal{D}$ satisfies the following relation:

$$\mathcal{D}(f \cdot g) = \mathcal{D}(f) \cdot \mathcal{D}(g). \tag{8}$$

so that, as $T_n(x + \frac{1}{x}) = x^n + \frac{1}{x^n}$, then

$$\mathcal{D}(a_0 + \sum_{j=1}^n a_jT_j) = a_0x^n + \sum_{j=1}^n a_j(x^{n-j} + x^{n+j}). \tag{9}$$

Note that Equation (9) shows that passing from $[\mathcal{X}]\mathcal{D}(P)$ to $[\mathcal{T}]P$ and vice versa is just a way of changing the numbering of the coefficients, having in mind that $[\mathcal{X}]\mathcal{D}(P)$ is a reversible polynomial. In particular, $\tau_{\mathcal{X}}(\mathcal{D}(P)) = \tau_{\mathcal{T}}(P)$ and $\deg \mathcal{D}(P) = 2\deg P$. Also, passing from $[\mathcal{X}]\mathcal{D}(P)$ to $[\mathcal{T}]P$ and vice-versa can be done in $\mathcal{O}(d)$ bit operations.

From this remark, it is then sufficient to apply the classical fast multiplication for polynomials in $\mathbf{Q}[x]$ after applying $\mathcal{D}$ on the operands. We will denote by $M(\delta, \tau) = \widetilde{\mathcal{O}}(\delta\tau)$ the number of bit operations for multiplying two polynomials of $\mathbf{Z}[x]$ with degrees at most δ and coeffitients of bitsize at most t with respect to the usual monomial basis $\mathcal{X}$ and by $M(\tau) = \widetilde{\mathcal{O}}(\tau)$ the cost of the multiplication of two integers of bitsize dominated by τ. We obtain

Proposition 4. *Let $f, g \in \mathbf{Z}[x]$ given by their Chebyshev forms $[\mathcal{T}]f$ and $[\mathcal{T}]g$. Then we have $\tau_{\mathcal{T}}(f \cdot g) \leq \tau_{\mathcal{T}}(f) + \tau_{\mathcal{T}}(g) + \log_2 \min(\deg f, \deg g) + 1$.*

Suppose that $\tau_{\mathcal{T}}(f)$, $\tau_{\mathcal{T}}(g) \leq \tau$ and $\deg f, \deg g \leq d$ then computing $[\mathcal{T}](f \cdot g)$ claims $M(2d, \tau) + \mathcal{O}(d)$ bit operations.

Note that this very simple algorithm might be replaced by the one from Giorgi ([7]) which claims the equivalent of two multiplications between polynomials of degree d with coefficients of bitsize in $\mathcal{O}(\tau)$.

Let us now extend these results to the division. Applying the transformation $\mathcal{D}$ on both side of the relation $f = g \cdot q + r$, we then get:

$$\mathcal{D}(f) = \mathcal{D}(g)\mathcal{D}(q) + x^{\deg f - \deg r}\mathcal{D}(r). \tag{10}$$

In particular, Formulas (8) and (10) imply that $g \mid f$ iff $\mathcal{D}(g) \mid \mathcal{D}(f)$. Also, applying the same strategy as for the product of Chebyshev forms, we immediately get:

Corrolary 5. *Let $f, g \in \mathbf{Z}[x]$ given by their Chebyshev forms $[\mathcal{T}]f$ and $[\mathcal{T}]g$. Suppose that $\deg f, \deg g \leq d$ and $\tau_{\mathcal{T}}(f), \tau_{\mathcal{T}}(g) \leq \tau$. Then,*

1. (a) *deciding if g divides f knowing $[\mathcal{T}]f$ and $[\mathcal{T}]g$ can be done in $\widetilde{\mathcal{O}}(d^2 + d\tau)$ bit operations;*
   (b) *if g divides f then computing $[\mathcal{T}]Quo(f, g)$ from $[\mathcal{T}]f$ and $[\mathcal{T}]g$ can be done in $\widetilde{\mathcal{O}}(d^2 + d\tau)$ bit operations;*
   (c) *computing $[\mathcal{T}]gcd(f, g)$ from $[\mathcal{T}]f$ and $[\mathcal{T}]g$ can be done in $\widetilde{\mathcal{O}}(d^2\tau)$ bit operations.*
2. (a) *if $g|f$ then $\tau_{\mathcal{T}}(g) = \mathcal{O}(\tau_{\mathcal{T}}(f) + d)$.*
   (b) *$\tau_{\mathcal{T}}(\gcd(f, g)) \leq \mathcal{O}(\tau + d)$.*

Proof. The first items are direct consequences of [16, exercice 9.14] while the third item is a direct consequence of [16, Corollary 11.20]. The two last items come from the Mignotte bound ([16, p. 166]). □

3. BASES CHANGE

It is a classical result that $\mathcal{T}$ is an orthogonal basis. Bostan *et al.* (see [4]) give a fast general conversion method from the monomial basis to orthogonal bases. The alternatives we propose are based on the general method of Hart & Novocin for polynomial composition (see [9]) which lead to simplier descriptions and complexity analysis. Given $f = \sum_{i=0}^{d} a_i x^i$ in the Chebyshev basis and $f = f_0 + \sum_{i=1}^{d} f_i T_i$ in the monomial basis, in both cases, the method is based on the same divide and conquer principle as in [9]: splitting f into $\lceil \frac{d}{2} \rceil$ sub-polynomials of length 2.

Suppose that we have to evaluate the sum $S = \sum_{j=0}^{d} u_j v^j$. We first define:

$$d_0 = d,\ v_0 = v,\ S_{0,j} = u_j,\ j = 0, \ldots, d.$$

Let $c = \lceil \log_2 d \rceil - 1$. Then we define for $k = 0, \ldots, c$:

$$\begin{aligned} d_{k+1} &= \lceil \tfrac{d_k}{2} \rceil, v_{k+1} = v_k^2, \\ S_{k+1,j} &= S_{k,2j} + S_{k,2j+1} \cdot v_k,\ j = 0, \ldots, d_{k+1}. \end{aligned}$$

Then, for $k = 0, \ldots, c$,

$$S = \sum_{j=0}^{d_k} S_{k,j} \cdot v_k^j.$$

At the end, we get $S = S_{c,0}$.

Algorithm 1 is a direct application of this strategy for the evaluation of $f = \sum_{j=1}^{d} a_j x^j$ in the Chebyshev basis $\mathcal{T}$.

Proposition 6. *Let $f = \sum_{i=0}^{d} a_i x^i$ be a polynomial of bitsize τ. One can compute the Chebyshev form of f in $\widetilde{\mathcal{O}}(d)$ arithmetic operations, or in binary complexity $\widetilde{\mathcal{O}}(d^2 + d\tau)$. The Chebyshev form $[\mathcal{T}]f$ has bitsize $\tau + \mathcal{O}(d)$.*

Algorithm 1: From $\mathcal{X}$ to $\mathcal{T}$

Input: $f = \sum_{i=0}^{n} a_i x^i$
Output: $[\mathcal{T}]f = f_0 + \sum_{i=1}^{n} f_i T_i$

```
begin
    d ← ⌊n/2⌋; c ← 0; v ← T_1 ;
    for j ← 0 to d do
        S_{c,j} ← a_{2j} + a_{2j+1} · T_1
    v ← ([T]v)^2;
    while d > 0 do
        d ← ⌊d/2⌋;
        for j ← 0 to d do
            S_{c+1,j} ← [T]S_{c,2j} + [T]S_{c,2j+1} · [T]v
        c ← c + 1; v ← ([T]v)^2;
    return S_{c,0}
```

Proof. We use Algorithm 1. Let $c = \lceil \log_2 d \rceil$. We start with $v = [\mathcal{T}]x = T_1$, and compute successively $v = x^{2^k}$, $k = 0, \ldots, c$. We know that x^{2^k} has size $\mathcal{O}(2^k)$ in the Chebyshev basis. From Proposition 4, one can compute $[\mathcal{T}]x^{2^{k+1}}$ from $[\mathcal{T}]x^{2^k}$ in $\widetilde{\mathcal{O}}(2^{2k})$ bit operations.

At each step, the bitsize τ_{k+1} of the Chebyshev form $S_{k+1,j} = [\mathcal{T}]S_{k,2j} + [\mathcal{T}]S_{k,2j+1} \cdot [\mathcal{T}]x^{2^k}$ satisfies $\tau_{k+1} \leq \tau_k + 2^k + k + 1$. We therefore obtain $\tau_k = \tau + \mathcal{O}(2^k)$. One can compute the Chebyshev form $S_{k+1,j} = S_{k,2j} + S_{k,2j+1} \cdot x^{2^k}$ in $\widetilde{\mathcal{O}}(2^k \tau_k)$ binary operations from Proposition 4.

The total computing time is bounded by $\sum_{k=1}^{c} \tau_k 2^k = \widetilde{\mathcal{O}}(d^2 + d\tau)$. □

Conversely, given a Chebyshev form $f = f_0 + \sum_{j=1}^{d} f_j T_j$, we have to find the expression $f = \sum_{j=0}^{d} a_i x^i$. We use a key idea from the general method of [4], considering the characteristic matrix of the Chebyshev polynomials:

$$X = \begin{bmatrix} 0 & 1 \\ -1 & x \end{bmatrix}.$$

Using the recurrence (1), we have:

$$\begin{bmatrix} T_j \\ T_{j+1} \end{bmatrix} = X^j \begin{bmatrix} T_0 \\ T_1 \end{bmatrix},\ j \geq 0.$$

Algorithm 2 computes the matrix polynomial

$$M(f) = \sum_{i=0}^{\lfloor \frac{d}{2} \rfloor} \begin{bmatrix} f_{2i} & f_{2i+1} \end{bmatrix} X^{2i}$$

using the composition method in [9]. Then, following [4], we get (because $T_0 = 2$)

$$f = -f_0 + M(f) \cdot \begin{bmatrix} T_0 \\ T_1 \end{bmatrix}.$$

Proposition 7. *Let $f = f_0 + \sum_{i=1}^{d} f_i T_i$ be a Chebyshev form of bitsize τ. One can compute f in the monomial basis in $\widetilde{\mathcal{O}}(d)$ arithmetic operations, or a binary complexity $\widetilde{\mathcal{O}}(d^2 + d\tau)$. $[\mathcal{X}]f$ has bitsize $\tau + \mathcal{O}(d)$.*

Proof. An induction shows that $X^n = \begin{bmatrix} -U_{n-1} & U_n \\ -U_n & U_{n+1} \end{bmatrix}$. Thus the bitsize of X^n is $\mathcal{O}(n)$, using Proposition 1.

Algorithm 2: From $\mathcal{T}$ to $\mathcal{X}$

Input: $f = \sum_{i=0}^{n} f_i T_i$
Output: $[\mathcal{X}]f = \sum_{i=0}^{n} a_i x^i$

```
begin
    d ← ⌊n/2⌋; c ← 0; V ← X^2 ;
    for j ← 0 to d do
        S_{c,j} ← [f_{2j} f_{2j+1}]
    while d > 0 do
        d ← ⌊d/2⌋;
        for j ← 0 to d do
            S_{c+1,j} ← S_{c,2j} + S_{c,2j+1} · V
        c ← c + 1; V ← [X]V^2;
    return S_{c,0} · [2, x]
```

Let $c = \lceil \log_2 d \rceil$ and $0 < k \leq c$. We compute X^{2^k} from $X^{2^{k-1}}$ in $\mathcal{O}(2^{2k})$ binary operations. We thus compute X^{2^k} for $k = 0, \ldots, c$ in $\mathcal{O}(d^2)$ binary operations.

The bitsize $\tau_{k+1,j}$ of the 1×2 matrix $S_{k+1,j} = S_{k,2j} + S_{k,2j+1} \cdot X^{2^k}$ is bounded by $\tau_{k,j} + \mathcal{O}(2^k)$. It follows that $\tau_{k,j} \leq \tau_k \leq \tau + \mathcal{O}(2^k)$. At the end the bitsize of $[\mathcal{X}]f$ is bounded by $\tau + \mathcal{O}(d)$.

S_{k+1} is deduced from S_k by evaluating the $S_{k+1,j}$ for $j = 0, \ldots, \lfloor \frac{d}{2^k} \rfloor$. This claims $d/2^k \times \widetilde{\mathcal{O}}(\tau_k 2^k) = \widetilde{\mathcal{O}}(d(\tau + 2^k))$ binary operations.

At the end we compute $S_{c,0}$ in $\widetilde{\mathcal{O}}(d^2 + d\tau)$ binary operations. $\square$

4. THE MINIMAL POLYNOMIAL

The degree of the minimal polynomial of $\cos \frac{2\pi}{n}$ is $\frac{1}{2}\varphi(n)$ (see Rivlin, [14, Chapter 5] or Watkin & Zeitlin ,[17]). This minimal polynomial appears as a factor of the Chebyshev monic polynomials in [14]. It is a factor of $T_{\lfloor \frac{n}{2} \rfloor + 1} - T_{\lfloor \frac{n}{2} \rfloor}$ in [17]. Bayard & Cangul ([3]) use this formula to get an induction formula for the minimal polynomial of $2\cos\frac{\pi}{n}$ with the help of the Möbius inversion formula.

Here, we will show that the minimal polynomial M_n of $2\cos\frac{\pi}{n}$ satisfies $\Phi_{2n} = \mathcal{D}(M_n)$. We will adapt the algorithm given in [1] for the computation of Φ_n to the computation of M_n in the Chebyshev basis.

Definition 8. *Let $n > 2$, M_n denotes the minimal polynomial of $2\cos\frac{\pi}{n}$ in $\mathbf{Q}[x]$. We set $M_1 = 1$.*

Let ζ_n be a primitive n-th root of the unity, its minimal polynomial is the cyclotomic polynomial Φ_n. It is monic and of degree $\varphi(n)$ where φ is the Euler function. The roots of Φ_n are all the primitive n-th roots and we deduce that $\Phi_n(1/\zeta_n) = 0$ and thus Φ_n is reversible. We have $\mathbf{Q}(\cos\frac{2\pi}{2n}) = \mathbf{Q}(\zeta_n) \cap \mathbf{R}$ and the minimal polynomial over $\mathbf{Q}$ of $\cos\frac{2\pi}{2n}$ has degree $\frac{1}{2}\varphi(2n)$, when $n > 2$. Its roots are the $\cos\frac{k\pi}{n}$ where k and $2n$ are coprime. Note that the minimal polynomial of $\cos 2\frac{\pi}{n}$ is $\pm M_n(-t)$.

Lemma 9. *If $m \geq 3$ is odd, then $M_{2^k m} = M_m(T_{2^k})$.*

Proof. As $M_m \circ T_{2^k}(\cos\frac{\pi}{2^k m}) = 0$ and $(2^k, m) = 1$, then $M_{2^k m} | M_m(T_{2^k})$. We conclude since $M_{2^k m}$ and $M_m(T_{2^k})$ have same leading term. $\square$

The number of factors of T_n is given in [10]. We give here the complete factorization of the Chebyshev polynomials T_n and U_n as products of polynomials M_n, see also [11]:

Proposition 10. *We have*

$$U_{2^k(2m+1)} = (-1)^m \prod_{d|2m+1} \left(M_d(-t) \prod_{i=0}^{k} M_d(T_{2^i}) \right)$$

$$T_{2^k(2m+1)} = \prod_{d|2m+1} M_d(T_{2^{k+1}})$$

where M_n is the minimal polynomial of $\cos\frac{\pi}{n}$.

Proof. The factorization of U_n is obtained by comparing its roots with those of $M_d(\pm t)$, when $d|2m+1$ and using the Euler identity $n = \sum_{d|n} \varphi(d)$.

Let d be an odd divisor of n. We write $n = 2^k \cdot d_1 \cdot d$, where d_1 is odd. $\cos\frac{d_1\pi}{2n} = \cos\frac{\pi}{2^{k+1}d}$ is a root of T_n so $M_{2^{k+1}d} | T_n$. We conclude by comparing the leading terms. $\square$

From $M_n(\zeta_{2n} + 1/\zeta_{2n}) = 0$ when $n > 2$, we deduce a relation between the Chebyshev form of M_n and Φ_{2n}, see also [14, Chapter 5]:

Proposition 11. *We have $\Phi_{2n} = \mathcal{D}(M_n)$, for $n > 2$.*

Let $x = 2\cos t$ and $V_n(x) = \dfrac{\sin(n+\frac{1}{2})t}{\cos\frac{t}{2}}$ then $V_n = \prod_{d|2n+1} M_d$. This is an analogous formula to $X^n - 1 = \prod_{d|n} \Phi_d$, see [12].

Computation of M_n

The relation $T_n \circ T_m = T_n \circ T_m = T_{mn}$ makes the Chebyshev basis very convenient for the computation of the minimal polynomial M_n. Lemma 12 is a generalization of Lemma 9:

Lemma 12. *Let $n = 2^\alpha p_1^{\alpha_1} \ldots p_k^{\alpha_k}$ where the p_i are distinct odd primes. Let $n_0 = p_1 \cdots p_k$. Then we have*

$$\begin{cases} M_n = M_{n_0}(T_{n/n_0}), & \text{if } k \geq 1 \\ M_{2^\alpha} = T_{2^{\alpha-1}}. \end{cases}$$

Proof. If $n = 2^\alpha > 1$ then $T_{n/2}(2\cos\pi/n) = 2\cos\pi/2 = 0$. On the other hand $T_{n/2}$ and M_n are monic with the same degree. Thus they are equal.

If $n_0 > 1$ then $M_{n_0}(T_{n/n_0}(2\cos\pi/n)) = M_{n_0}(2\cos\pi/n_0) = 0$. We thus have $M_n | M_{n_0}(T_{n/n_0})$. On the other hand, M_n and $M_{n_0}(T_{n/n_0})$ are monic with the same degree. Thus they are equal. $\square$

Lemma 12 shows that the Chebyshev basis is particularly adapted for the computation of M_n.

Lemma 13. *$n = 2^\alpha p_1^{\alpha_1} \ldots p_k^{\alpha_k}$ where the p_i are distinct odd primes. Let $n_0 = p_1 \cdots p_k$. Then the coefficients of M_n (in the monomial basis, as well as in the Chebyshev basis) have bitsize $\mathcal{O}(n_0)$.*

Proof. As $M_{n_0} | U_{n_0}$ the conclusion is a consequence of Proposition 5, because we deduce from

$$U_n = \sum_{k=1}^{\lfloor \frac{n}{2} \rfloor} T_{n-2k-1} + \tfrac{1}{2}(1 - (-1)^n),$$

that the Chebyshev form U_n has maximum bitsize 1. $\square$

When n is an odd prime, we get :

LEMMA 14. *Let p a prime odd number, then*

$$M_p = T_k - T_{k-1} + \cdots + (-1)^{k-1}T_1 + (-1)^k,$$

where $k = 1/2(p-1)$.

PROOF. Let $p = 2k+1$ and $h = T_k - T_{k-1} + \cdots + (-1)^{k-1}T_1 + (-1)^k$. We have $\sin\frac{\pi}{2k+1} \cdot h(2\cos\frac{\pi}{2k+1}) = 0$.

But $\deg h = k = \frac{1}{2}\varphi(p)$ and thus $h = M_p$. □

LEMMA 15. *Let p be an odd prime and n such that $p \nmid n$, then $M_{np} = \dfrac{M_n(T_p)}{M_n}$.*

PROOF. We just need to show that both M_{np} and M_n divide $M_n(T_p)$ because these polynomials are monic and their degrees satisfy $2(\deg M_n + \deg M_{np}) = \varphi(2n) + \varphi(2pn) = p\varphi(n) = 2\deg M_n(T_p)$.

If $t = 2\cos\frac{\pi}{np}$ then $T_p(t) = 2\cos\frac{\pi}{n}$ and $M_n(T_p)(t) = 0$ so that $M_{np}|M_n(T_p)$. If $t = 2\cos\frac{\pi}{n}$ then $T_p(t) = 2\cos\frac{p\pi}{n}$ is a root of M_n. □

Following Arnold and Monagan in [1] for the computation of Φ_n, we deduce Algorithm 3 to compute the Chebyshev form of M_n:

Algorithm 3: Compute $[\mathcal{T}]M_n$

```
Input: n
Output: [T]M_n
begin
    Factorize n = 2^{α_0} p_1^{α_1} ··· p_k^{α_k}; n_0 ← p_1 ··· p_k;
    if k = 0 then
        if α_0 = 0 then
            return 1
        else
            return T_{2^{α_0 - 1}}
    i ← (1/2)(p_1 - 1);
    m ← T_i - T_{i-1} + ··· + (-1)^{i-1} T_1 + (-1)^i;
    for j ← 2 to k do
        m ← [T](m(T_{p_j}) / m)
    return m(T_{n/n_0})
```

PROPOSITION 16. *One computes M_n in the Chebyshev basis in complexity $\widetilde{\mathcal{O}}(n_0)$ or $\widetilde{\mathcal{O}}(n_0^2)$ bit operations.*

PROOF. We will use Algorithm 3. The factorization of n requires $\widetilde{\mathcal{O}}(\log n)$ binary operations, see [16, Chap. 19].

We have to compute successively $M_{q_{i+1}} = (M_{q_i} \circ T_{p_{i+1}})/M_{q_i}$, where $q_i = p_1 \cdots p_i$.

We know that $\tau(M_{q_i}) = \mathcal{O}(q_i)$, $\tau(M_{q_i}(T_{p_{i+1}})) = \tau(M_{q_i})$ and $\deg(M_{q_i}(T_{p_{i+1}})) = \varphi(2q_i)p_{i+1}/2 \le q_{i+1}$. By Proposition 4, we compute $M_{q_{i+1}}$ from M_{q_i} in $\widetilde{\mathcal{O}}(q_{i+1})$ arithmetic operations or $\widetilde{\mathcal{O}}(q_{j+1}^2)$ binary operations.

Finally we compute M_{n_0} in $\mathcal{O}(q_2 + \cdots + q_k) = \widetilde{\mathcal{O}}(n_0)$ arithmetic operations and $\widetilde{\mathcal{O}}(q_1^2 + \cdots + q_k^2) = \widetilde{\mathcal{O}}(n_0^2)$ binary operations. □

The analogy between M_n and the cyclotomic polynomial Φ_n allows us to deduce the following complexity bound for the computation of Φ_n from the algorithm proposed by Arnold and Monagan in [1] :

CORROLARY 17. *One can compute Φ_n in the complexity $\widetilde{\mathcal{O}}(n_0)$ and in the running time $\widetilde{\mathcal{O}}(n_0^2)$, where n_0 is the odd squarefree part of n.*

Example

The minimal polynomial M_{1260} is

$$\begin{aligned} M_{105}(T_{12}) &= T_{288} - T_{276} + T_{264} + T_{228} - T_{216} + 2\,T_{204} \\ &\quad -T_{192} + T_{180} + T_{144} - T_{132} + T_{120} - T_{108} \\ &\quad +T_{96} - T_{84} - T_{48} - T_{24} - 1. \end{aligned}$$

The maximum bitsize of $[\mathcal{X}]M_{1260}$ is $\tau([X]_{128}M_{1260}) = 197$ while the total bitsize of its coefficients is 20329.

The minimal polynomial M_{936} is

$$\begin{aligned} M_{39}(T_{24}) &= T_{288} + T_{264} - T_{216} - T_{192} \\ &\quad +T_{144} + T_{120} - T_{72} - T_{48} + 1. \end{aligned}$$

Its maximum bitsize is also 197 while the total bitsize of its coefficients is 20338.

5. SUM OF COSINES

The aim of this section is to give methods for determining the sign of $F = f(\gamma)$ where γ is one root of M_n, for evaluating F and for getting the minimal polynomial of F.

We consider here the expression

$$F = f_0 + 2\sum_{j=1}^{d} f_j \cos\tfrac{j\pi}{n}, \tag{11}$$

where $d \le n$ and $f_i \in \mathbf{Z}$. It is convenient to write F as $f(\gamma)$ where γ is the algebraic number $2\cos\frac{\pi}{n}$ and $f = f_0 + \sum_{k=1}^{d} f_k T_k \in \mathbf{Z}[x]$.

Evaluating the sum

The first step is the approximation of $\gamma = 2\cos k\frac{\pi}{n}$, using Brent's methods ([5, 6])

LEMMA 18. *Let $0 \le k \le n$ and $\gamma = 2\cos k\frac{\pi}{n}$. Let $\ell \in \mathbf{Z}_{>0}$. One can compute $c \in \mathbf{Q}$, of bitsize $\tau(c) \le \ell$ such that $|c - \gamma| \le 2^{-\ell}$ in $\widetilde{\mathcal{O}}(\ell + \log n)$ bit operations.*

PROOF. We first compute $r \in \mathbf{Q}$ of bitsize $2(\ell + 2\log n)$ such that $\left|r - k\frac{\pi}{n}\right| < 2^{-\ell-1}$. It claims $\widetilde{\mathcal{O}}(l + \log n)$ bit operations, using [5].

Then we compute c such that $|c - \cos r| < 2^{-\ell-1}$, using [6], in $\widetilde{\mathcal{O}}(\ell + \log n)$ binary operations. □

We thus deduce

CORROLARY 19. *Let $0 \le k \le n$ and $\gamma = 2\cos k\frac{\pi}{n}$. Let $\ell \in \mathbf{Z}_{>0}$. Let f be the Chebyshev form $f_0 + \sum_{i=1}^{n-1} f_i T_i$. One computes $\widetilde{F} \in \mathbf{Q}$ of bitsize $\mathcal{O}(n\tau(f) + \ell)$ such that $\left|\widetilde{F} - f(\gamma)\right| \le 2^{-\ell}$ in $\widetilde{\mathcal{O}}(n\ell + n\tau)$ bit operations.*

One computes F^- and F^+ of bitsize $\mathcal{O}(n\tau(f) + \ell)$ such that $F^- \le F \le F^+$ and $F^+ - F^- \le 2^{-\ell}$ in $\widetilde{\mathcal{O}}(n\ell + n\tau)$ bit operations.

PROOF. Using Lemma 18, one can compute the rational numbers $c_i, i = 1, \ldots, n-1$ such that $\left|c_i - \cos ik\frac{\pi}{n}\right| \le 2^{-\ell-\tau}/n$. They are of bitsize $\mathcal{O}(\tau + \ell + \log n)$ and we can compute them in $\widetilde{\mathcal{O}}(n\tau + n\ell)$ bit operations.

If $\widetilde{F} = f_0 + 2\sum_{i=1}^{n-1} f_i c_i$, then

$$\left|\tilde{F} - F\right| \leq 2\sum_{i=1}^{n-1} |f_i| \left|c_i - \cos ik\tfrac{\pi}{n}\right| \leq 2^{-\ell}.$$

Here, the $f_i c_i$ have bitsize $\tau + \ell + \log n$ and each can be computed in $\widetilde{\mathcal{O}}(\tau + \ell + \log n)$ bit operations, using the fast multiplication described in [16, Chap. 8]. F has bitsize $\mathcal{O}(\tau + \ell + \log n)$ and can be computed in $\widetilde{\mathcal{O}}(n\tau + n\ell)$ bit operations.

The proof of the second part of the Lemma is similar. □

Evaluating the sign of the sum

Here we want to evaluate the sign of $f(\gamma)$ where $\gamma = 2\cos k\frac{\pi}{n}$. and f is the Chebyshev form $f_0 + \sum_{i=1}^{n-1} f_i T_i$. Without loss of generality, we can suppose that $(k, 2n) = 1$, that is to say that γ is a root of M_n. If $(k, n) = d > 1$, then we can change n by n/d. If k is even, then γ will be a root of $M_n(-t)$.

As M_n is the minimal polynomial of γ then $f(\gamma) = 0$ iff $M_n \mid f$. If M_n does not divide f then we have $|\mathrm{Res}\,(M_n, f)| \geq 1$. As M_n is monic, we have also

$$\mathrm{Res}\,(M_n, f) = \prod_{M_n(\gamma)=0} f(\gamma).$$

Let γ be a root of M_n, then we have $|f(\gamma)| \leq ||f||_1$, where $||f||_1 = |f_0| + 2\sum_{i=1}^{d} |f_i|$. We thus obtain for every root γ of M_n:

$$|f(\gamma)| \geq (||f||_1)^{1-\deg M_n} = \delta. \tag{12}$$

Then, it is then sufficient to compute an approximation $\tilde{F}$ of $f(\gamma)$ such that $\left|\tilde{F} - f(\gamma)\right| < \frac{\delta}{2}$ in order to ensure that this approximation has the same sign than $f(\gamma)$. We thus deduce

Proposition 20. *Let f of degree $d < n$ with $\tau_{\mathcal{T}}(f) \leq \tau$. Let $\gamma = 2\cos k\frac{\pi}{n}$ where $(k, 2n) = 1$.*

1. *we can decide whether $f(\gamma) = 0$ in $\widetilde{\mathcal{O}}(n^2 + n\tau)$ bit operations,*
2. *we can compute* $\mathrm{sign}\, f(\gamma)$ *in $\widetilde{\mathcal{O}}(n^2\tau)$ bit operations,*
3. *we can evaluate $f(\gamma)$ with precision $2^{-\ell}$ in the complexity $\widetilde{\mathcal{O}}(n\ell + n\tau)$.*

Proof. We first decide if $f(\gamma) = 0$ by testing if $M_n | f$. It claims $\widetilde{\mathcal{O}}(n^2 + n\tau)$ bit operations by Propositions 16 and 5.

If $f(\gamma) \neq 0$, then we compute an approximation F of $f(\gamma)$ with accuracy $\frac{1}{2}\delta$ where $\delta = (||f||_1)^{1-\deg M_n}$. But $-\log_2 \delta \leq (\deg M_n - 1)(\log_2(2n+1) + \tau) = \widetilde{\mathcal{O}}(n\tau)$. We deduce, using Corollary 19, that F can be computed in $\widetilde{\mathcal{O}}(n^2\tau)$ bit operations.

The last point of the Proposition is a consequence of Corollary 19. □

The estimate of δ in Equation (12) may be too sharp. The alternative we propose is to explicitly compute $[\mathcal{T}]M_n$, then test if $[\mathcal{T}]M_n$ divides F using Corollary 5, and then compute $\tilde{F}$ using interval arithmetic, starting in low precision, and doubling the precision if the resulting interval contains 0. According to the above result, Algorithm 4 will end, performing $\widetilde{\mathcal{O}}(n^2\tau)$ bit operations in the worst case, but will use much less precision in generic situations.

Algorithm 4: Determine the sign of $f(\gamma)$

Input: $f = f_0 + \sum_i^n f_i T_i$, k such that $(k, 2n) = 1$
Output: sign $f(\gamma)$ where $\gamma = 2\cos k\frac{\pi}{n})$

```
begin
    Compute [T]M_n ;
    if M_n | f then
        return f(2 cos k π/n) = 0
    ℓ ← 1, F^- ← −1, F^+ ← 1;
    while F^+ · F^- < 0 do
        Compute [F^-, F^+] of length 2^{-ℓ}, containing
        f(γ);
        ℓ ← 2ℓ;
return sign (F^+)
```

Example

In [13], Myerson pointed out that

$$a = 16\,\sin\frac{\pi}{9}\,\sin\frac{5\pi}{18}\,\sin\frac{11\pi}{39}\,\sin\frac{3\pi}{8}$$

is very closed to 3. We write a as $f(\gamma)$ where

$$f = -T_{771} - T_{459} - T_{451} - T_{277} + T_{251} + T_{243} + T_{69} - T_{43},$$

and $\gamma = 2\cos\frac{\pi}{936}$ is the biggest root of M_{936}. Equation (12) asserts that

$$|f(\gamma) - 3| \geq ||f - 3||_1^{-288} = 19^{-288} \simeq 2^{-1224}.$$

Indeed, $\min_{M_n(x)=0} |f(x) - 3| \geq 2^{-29}$ and Algorithm 4 provides the value and the sign of $f(\gamma)$ very quickly.

6. THE MINIMAL POLYNOMIAL OF A SUM

We give an algorithm for the minimal polynomial of $F = f(\gamma)$. It is classical to consider the monic polynomial (because M_n is monic)

$$P_f(z) = \mathrm{Res}_x(z - f(x), M_n(x)) = \prod_{M_n(\gamma)=0} (z - f(\gamma)). \tag{13}$$

The roots of P_F are $z_k = f(2\cos k\frac{\pi}{n}) = f(T_k(2\cos\frac{\pi}{n}))$ where $(k, 2n) = 1$. We thus deduce that

Proposition 21. *There exists $\nu \in \mathbf{N}^*$ such that $P_f = M_f^\nu$, where M_f is the minimal polynomial of $f(2\cos\frac{\pi}{n})$.*

Proof. Let $\gamma_k = 2\cos k\frac{\pi}{n}$, be a root of M_n. The minimal polynomial μ_k of $f(\gamma_k)$ satisfies $\mu_k \circ f(\gamma_k) = 0$. Thus M_n divides $\mu_k \circ f$ and $\mu_k(f(\gamma_j)) = 0$ for every root γ_j of M_n. We thus deduce that μ_j divides μ_k and that $\mu_k = \mu_j = M_f$. P_f factorizes into $P_f = M_f^\nu$. □

We will compute M_f by computing first P_f and then $M_f = P_f / \gcd(P_f, P_f')$.

On might directly compute P_f using a general algorithm, following [16, Section 11.2]. It would then require $\widetilde{\mathcal{O}}(\delta\tau_s)$ bit operations where $\delta = deg(f) + deg(M_n)$ and τ_s is a bound on the total bitsize required to store any principal subresultant coefficient which are all polynomials of degree in $\mathcal{O}(\delta)$ in z with coefficients of bitsizes in $\widetilde{\mathcal{O}}(\delta\tau)$ (see Proposition 8.50 in [2]). Such an algorithm would then perform, in our case, $\widetilde{\mathcal{O}}(n^3\tau)$ bit operations, and we do not pretend to give a better theoretical upperbound. However, in pratice, this classical strategy is based on the *Half-Gcd* algorithm and

makes use if fast operations on univariate polynomials, both starting to be being efficient for rather high degrees (several hundred). The algorithm we propose in the sequel runs efficiently even for small degrees.

It is natural to consider P_f as a polynomial whose coefficients are in $\mathbf{Q}[x]/(M_n)$.

As $T_{n+i}(\gamma) = 2\cos(n+i)k\frac{\pi}{n} = (-1)^k T_i(\gamma)$ we have

$$T_{n+i} \equiv T_{n-i} \equiv -T_i \pmod{M_n}.$$

The Chebyshev basis $(1, T_1, \dots T_{\lfloor \frac{n-1}{2} \rfloor})$ is then particularly adapted as generating family of $\mathbf{Q}[x]/(M_n)$:

LEMMA 22. *$f = f_0 + \sum_{i=1}^{\lfloor \frac{n-1}{2} \rfloor} f_i T_i$ and $g = g_0 + \sum_{i=1}^{\lfloor \frac{n-1}{2} \rfloor} g_i T_i$ be Chebyshev forms. Then one can compute $h = h_0 + \sum_{i=1}^{\lfloor \frac{n-1}{2} \rfloor} h_i T_i$ where $h \equiv f \cdot g \pmod{M_n}$ in $\widetilde{\mathcal{O}}(n\tau)$ bit operations and $\tau(h) \le \tau(f) + \tau(g) + \log_2 n + 1$.*

PROOF. We compute $h = [\mathcal{T}](f \cdot g)$ in $\widetilde{\mathcal{O}}(n\tau)$ bit operations. We obtain $h = h_0 + \sum_{i=1}^{n-1} h_i T_i = h_0 + \sum_{i=1}^{\lfloor \frac{n-1}{2} \rfloor} (h_i - h_{n-i}) T_i$. $\square$

Let $N = \frac{1}{2}\varphi(2n)$ be the degree of P_f. The N roots of P_F are the $f(T_k(x))$ where $(k, 2n) = 1$.

We shall consider the i-th Newton sum:

$$S_i(P_f) = \sum_{\substack{1 \le k \le n \\ (k,2n)=1}} f(T_k(x))^i.$$

Consider first $f^i = f(x)^i = f_{i,0} + \sum_{j=1}^{\lfloor \frac{n-1}{2} \rfloor} f_{i,j} T_j$ in $\mathbf{Q}[x]/(M_n)$, then $f(T_k(x))^i \equiv f_{i,0} + \sum_{j=1}^{\lfloor \frac{n-1}{2} \rfloor} f_{i,j} T_{jk} \pmod{M_n}$.

We thus deduce that

$$S_i(P_f) \equiv N f_{i,0} + \sum_{j=1}^{\lfloor \frac{n-1}{2} \rfloor} f_{i,j} \Big(\sum_{\substack{1 \le k \le n \\ (k,2n)=1}} T_{jk} \Big) \pmod{M_n}$$

LEMMA 23. *Let j and n be nonnegative integers, then*

$$\sum_{\substack{1 \le k \le n \\ (k,2n)=1}} T_{jk} \equiv 2 S_j(\Phi_{2n}) \pmod{M_n}$$

PROOF. Let $\gamma = 2\cos\frac{\pi}{n}$ then

$$\sum_{\substack{1 \le k \le n \\ (k,2n)=1}} T_{jk}(\gamma) = 2\,\mathrm{Re}\Big(\sum_{\substack{1 \le k \le n \\ (k,2n)=1}} e^{jki\frac{\pi}{n}} \Big) = 2 S_j(\Phi_{2n})$$

because $S_j(\Phi_{2n})$ is real. $\square$

Note that the sums $S_j(\Phi_n)$ are known as *Ramanujan sums*, see Remark 25. We thus deduce that one can obtain $S_j(P_f)$ from coefficients $f_{i,j}$ of $f^j(x)$ in $\mathbf{Q}[x]/(M_n)$ with

$$S_i(P_f) = f_{i,0} N + 2 \sum_{j=1}^{\lfloor \frac{n-1}{2} \rfloor} f_{i,j} S_j(\Phi_{2n}) \tag{14}$$

Equation (14) gives a linear relation between the Newton sums of Φ_{2n} and the Newton sums of P_f. The associated matrix is $(f_{i,j}) = [\mathcal{I}]_j(f^i)$. Note that $(f_{i,j})$ are not unique but the Formula (14) does not depend on them. We thus deduce Algorithm 5.

Algorithm 5: Compute P_f

Input: $[\mathcal{T}]f = f_0 + \sum_{j=1}^{n-1} f_i T_i$
Output: $P_f = \mathrm{Res}_x(z - f(x), M_n(x))$

```
begin
    Compute [T]M_n = σ_0 + Σ_{i=1}^{N} σ_i T_i;
    for i ← 2 to N do
        Compute the Newton sums
        S_i = iσ_i − Σ_{j=1}^{i−1} σ_j S_{i−j}
    for i ← 2 to N do
        Compute f^i = f_{i,0} + Σ_{j=1}^{⌊(n−1)/2⌋} f_{i,j} T_j;
        Compute the Newton sums
        S_i(P_f) = f_{i,0} · N + 2 Σ_{j=1}^{⌊(n−1)/2⌋} f_{i,j} S_j;
        Compute
        σ_i(P_f) = −(1/i) ( S_i(P_f) + Σ_{j=1}^{i−1} σ_{i−j}(P_f) S_j(P_f) )
return z^N + Σ_{i=1}^{N} σ_i(P_f) z^{N−i}
```

PROPOSITION 24. *We can compute P_f in $\widetilde{\mathcal{O}}(n^3\tau)$ bit operations.*

PROOF. We first compute $[\mathcal{T}]M_n$ in $\widetilde{\mathcal{O}}(n^2)$ bit operations. We obtain $[\mathcal{T}]M_n = \sigma_0 + \sum_{i=1}^{N} \sigma_i T_i$. As $\Phi_{2n} = \mathcal{D}(M_n)$ we deduce that $\Phi_{2n} = \sum_{i=0}^{2N} b_i x^i$ where $b_i = b_{2N-i} = \sigma_i$.

We then obtain the $S_i(\Phi_{2n})$ using the Newton relations:

$$S_k + \sigma_1 S_{k-1} + \dots + \sigma_{k-1} S_1 + k\sigma_k = 0, \quad 1 \le k \le N. \tag{15}$$

Here the σ_i's have bitsize $\mathcal{O}(n)$. We thus deduce by induction that all the S_k have bitsize $\mathcal{O}(n)$. S_k is computed from $S_1, \dots, S_{k-1}$ in $n\widetilde{\mathcal{O}}(n)$ bit operations. In conclusion $S_1, \dots, S_N$ may be computed in $\widetilde{\mathcal{O}}(n^3)$ bit operations.

By induction $f^i \pmod{M_n}$ has bitsize $\mathcal{O}(i\tau)$ and can be computed from $f^{i-1} \pmod{M_n}$ in $\widetilde{\mathcal{O}}(in\tau)$ bit operations, using Lemma 22. We can compute all $f_{i,j}$, $1 \le i \le N$, $1 \le j \le \lfloor \frac{n-1}{2} \rfloor$ in $\widetilde{\mathcal{O}}(n^3\tau)$ bit operations.

We then compute each $S_j(P_f)$ in $\widetilde{\mathcal{O}}(n^2\tau)$ bit operations, using Formula (14). Il claims $\widetilde{\mathcal{O}}(n^3\tau)$ bit operations.

We then use the Newton formula (15) to get the coefficients $\sigma_1(P_f), \dots, \sigma_N(P_f)$ of P_f:

$$k\sigma_k(P_f) = -\sum_{i=1}^{k-1} S_i(P_f)\sigma_{k-i}(P_f).$$

Also, all $k\sigma_k$ have bitsize $\mathcal{O}(n\tau)$ and may be computed in $\widetilde{\mathcal{O}}(n^3\tau)$ bit operations. As $\sigma_k \in \mathbf{Z}$ they can all be obtained in $\widetilde{\mathcal{O}}(n^3\tau)$ bit operations $\square$

REMARK 25. *The Ramanujan sums are known (see [8, Th. 272]) to satisfy*

$$S_m(\Phi_n) = \mu(n/\gcd(n,m)) \frac{\varphi(n)}{\varphi(n/\gcd(n,m))},$$

where μ is the Möbius function and φ is the Euler function. It means that $S_m(\Phi_n)$ may be computed in $\mathcal{O}(n)$ binary operations (see [16, Chap. 19]). But this does not allow to compute Φ_n with the Newton formulas, which would require $\widetilde{\mathcal{O}}(n^3)$ binary operations.

Once P_f is computed, we can compute M_f as $P_f/\gcd(P_f, P_f')$.

Here P_f has degree N and bitsize $\mathcal{O}(n\tau)$. Then $h = \gcd(P_f, P_f')$ is calculated in time $\widetilde{\mathcal{O}}(n^3\tau)$, using Proposition 5.

Using Proposition 5, we know that $\tau_{\mathcal{T}}(h) = \widetilde{\mathcal{O}}(n\tau)$ so that the last operation $M_f = P_f/h$ claims $\widetilde{\mathcal{O}}(n^2\tau)$ bit operations.

Finally we deduce

PROPOSITION 26. *We can compute the minimal polynomial M_f of $f(2\cos\frac{\pi}{n})$ in $\widetilde{\mathcal{O}}(n^3\tau)$ binary operations.*

Example

Consider $F = 2a\cos\frac{\pi}{7} + 2b\cos\frac{\pi}{5}$. We write $F = f(2\cos\frac{\pi}{35})$ where $f = aT_5 + bT_7$.

The minimal polynomial of $2\cos\frac{\pi}{35}$ is

$$\begin{aligned} M_{35} &= T_{12} + T_{11} - T_7 - T_6 - T_5 - T_4 + T_2 + T_1 + 1 \\ &= t^{12} + t^{11} - 12\,t^{10} - 11\,t^9 + 54\,t^8 + 43\,t^7 \\ &\quad -113\,t^6 - 71\,t^5 + 110\,t^4 + 46\,t^3 - 40\,t^2 - 8\,t + 1. \end{aligned}$$

Computing $P_f = \mathrm{Res}\,(z - aT_5 - bT_7, M_{35})$ we obtain $P_f = M_f^2$ where

$$\begin{aligned} M_f &= z^6 - (3\,b + 2\,c)\,z^5 + c\,(-3\,c + 5\,b)\,z^4 \\ &\quad + \left(5\,b^3 + 6\,bc^2 + 6\,c^3\right) z^3 \\ &\quad -c\left(5\,b^3 + 7\,b^2c + 9\,bc^2 - 2\,c^3\right) z^2 \\ &\quad - \left(3\,b^5 - 4\,b^3c^2 - 7\,b^2c^3 + 2\,bc^4 + 4\,c^5\right) z \\ &\quad -b^6 + b^5c + 7\,c^2b^4 - 2\,b^3c^3 - 7\,b^2c^4 + 2\,bc^5 + c^6. \end{aligned}$$

7. REFERENCES

[1] Arnold, A., and Monagan, M. A high-performance algorithm for calculating cyclotomic polynomials. In *Proceedings of the 4th International Workshop on Parallel and Symbolic Computation* (2010), pp. 112–120.

[2] Basu, S., Pollack, R., and Roy, F.-M. *Algorithms in Real Algebraic Geometry*, vol. 10 of *Algorithms and Computation in Mathematics*. Springer-Verlag, 2006.

[3] Bayad, A., and Cangul, I. N. The minimal polynomial of $2\cos\frac{\pi}{q}$ and dickson polynomials. *Applied Mathematics and Computation 218*, 13 (Mars 2012), 7014–7022.

[4] Bostan, A., Salvy, B., and Schost, É. Fast conversion algorithms for orthogonal polynomials. *Linear Algebra and its Applications 431*, 1 (2010), 249–258.

[5] Brent, R. Multipleprecision zero-finding methods and the complexity of elementary function evaluation. *Analytic Computational Complexity (edited by J. F. Traub), Academic Press, New York* (1975), 151–176.

[6] Brent, R. Fast multiple precision evaluation of elementary functions. *Journal of the Association for Computing Machinery 23*, 2 (1976), 242–251.

[7] Giorgi, P. On polynomial multiplication in chebyshev basis. *IEEE Transactions on Computers 61*, 6 (2012), 780–789.

[8] Hardy, G. H., and Wright, E. M. *An Introduction to the Theory of numbers*. Oxford University Press, 2008.

[9] Hart, W. B., and Novocin, A. Practical divide-and-conquer algorithms for polynomial arithmetic. In *Lecture Notes in Computer Science* (september 2011), vol. 6885, CASC'11 Proceedings of the 13th international conference on Computer algebra in scientific computing, pp. 200–214.

[10] Hsiao, H. J. On factorization of chebyshev's polynomial of the first kind. *Bulletin of the Institute of Mathematics Academia Sinica 12*, 1 (1984), 89–94.

[11] Koseleff, P.-V., Pecker, D., and Rouillier, F. Computing chebyshev knot diagrams. http://arxiv.org/abs/1001.5192.

[12] Koseleff, P.-V., Pecker, D., and Rouillier, F. The first rational Chebyshev knots. *Journal of Symbolic Computation 45* (2010), 1341–1358.

[13] Myerson, G. Rational products of cosine of rational angles. *Aequationes Mathematicae, Univ. of Waterloo 45* (1993), 70–82.

[14] Rivlin, T. J. *Chebyshev Methods in Numerical Approximation*. John Wiley & Sons, Inc., 1990.

[15] Shoup, V. Efficient computation of minimal polynomials in algebraic extensions of finite fields. In *Proceedings of the 1999 International Symposium on Symbolic and Algebraic Computation* (New York, NY, USA, 1999), ISSAC '99, ACM, pp. 53–58.

[16] von jur Gathen, J., and Gerhard, J. *Modern Computer Algebra*, 03 ed. Cambridge University Press, juin 2013.

[17] Watkins, W., and Zeitlin, J. The minimal polynomial of $\cos\frac{2\pi}{n}$. *The American Mathematical Monthly 100*, 5 (May 1993), 471–474.