On Computing the Degree of a Chebyshev Polynomial from Its Value[1]

Erdal Imamoglu

Department of Mathematics, Kirklareli University
Kayali 39100, Kirklareli, Turkey

Erich L. Kaltofen

Department of Mathematics, North Carolina State University
Raleigh, North Carolina 27695-8205, USA

Abstract

Algorithms for interpolating a polynomial f from its evaluation points whose running time depends on the sparsity t of the polynomial when it is represented as a linear combination of t Chebyshev Polynomials of the First Kind with non-zero scalar coefficients are given by Lakshman Y. N. and Saunders [SIAM J. Comput., vol. 24 (1995)], Kaltofen and Lee [J. Symbolic Comput., vol. 36 (2003)] and Arnold and Kaltofen [Proc. ISSAC 2015]. The term degrees are computed from values of Chebyshev Polynomials of those degrees. We give an algorithm that computes those degrees in the manner of the Pohlig and Hellman algorithm [IEEE Trans. Inf. Theor., vol 24 (1978)] for computing discrete logarithms modulo a prime number p when the factorization of $p-1$ (or $p+1$) has small prime factors, that is, when $p-1$ (or $p+1$) is smooth. Our algorithm can determine the Chebyshev degrees modulo such primes in bit complexity $\log(p)^{O(1)}$ times the squareroot of the largest prime factor of $p-1$ (or $p+1$).

Keywords: Algorithms, Discrete Logarithms, Chebyshev Polynomials, Interpolation in terms of the Chebyshev Polynomials of the First Kind.

1. Introduction

Let $p \geq 3$ be a prime number and $T_n(x) \in \mathbb{Z}_p[x]$ be the Chebyshev Polynomial of the First Kind of degree n modulo p, which can be defined by the recurrence

$$T_0(x) = 1, T_1(x) = x, T_n(x) = (2xT_{n-1}(x) - T_{n-2}(x)) \bmod p \text{ for all } n \geq 2. \tag{1}$$

The following problem is addressed in our paper:

Problem 1. *From input p, $\beta \in \mathbb{Z}_p \setminus \{0\}$, $\zeta = T_\delta(\beta) \in \mathbb{Z}_p$ compute δ.*

[1] Supported by National Science Foundation CCF-1421128 and CCF-1708884.
Email addresses: eimamoglu@klu.edu.tr (Erdal Imamoglu), kaltofen@math.ncsu.edu (Erich L. Kaltofen)
URL: http://www.math.ncsu.edu/~kaltofen (Erich L. Kaltofen)

If one computes ω such that $\beta = (\omega + 1/\omega)/2$, where either $\omega \in \mathbb{Z}_p$ or

$$\omega = z \in \mathbb{Z}_p[z]/(z^2 - 2\beta z + 1), \quad z^2 - 2\beta z + 1 \text{ irreducible in } \mathbb{Z}_p[z], \tag{2}$$

one obtains by the property of $T_n(x)$,

$$T_n\Big(\frac{x + \frac{1}{x}}{2}\Big) = \frac{x^n + \frac{1}{x^n}}{2}, \tag{3}$$

that $\zeta = (\omega^\delta + 1/\omega^\delta)/2$. Therefore $\rho = \omega^\delta, \bar{\rho} = 1/\omega^\delta$ are the roots of $x^2 - 2\zeta x + 1 = 0$ and one can reduce Problem 1 to factoring

$$x^2 - 2\beta x + 1 = (x - \omega)(x - 1/\omega) \quad \text{and} \quad x^2 - 2\zeta x + 1 = (x - \omega^\delta)(x - 1/\omega^\delta), \tag{4}$$

the latter possibly over the quadratic extension (2), and the discrete logarithm problem $\omega^{\pm\delta} = \rho^{\pm 1}$.

We proceed to choose p such that the discrete logarithm problem is solvable efficiently, say in $(\log p)^{O(1)}$ bit complexity. If $p-1$ is "smooth", meaning $p-1$ has only prime factors of magnitude $(\log p)^{O(1)}$, the Pohlig-Hellman (1978) Algorithm can be used.[2] Our objective here is to apply the Pohlig-Hellman method directly to Problem 1 without factoring quadratic polynomials. When doing so, we cannot determine the order η of ω as input, but our algorithm can compute the order "on the fly" as an additional output. Note that in the case (2) we assume $p + 1$ to be smooth (see Assumption 6.ii below). Our Pohlig-Hellman approach for Problem 1 required us to unravel a double loop in the Pohlig-Hellman Algorithm and remove the Chinese remainder step. We essentially determine δ in mixed-radix representation one digit at a time (see (5) below). In Section 2, we present the original Pohlig-Hellman algorithm in the single loop form, which we transfer to the Chebyshev degree problem in Section 3. Neither algorithms require the order of ω on input.

Our work is motivated by sparse interpolation algorithms in Chebyshev basis (Lakshman and Saunders, 1995; Kaltofen and Lee, 2003; Giesbrecht et al., 2004; Potts and Tasche, 2014; Arnold and Kaltofen, 2015; Imamoglu et al., 2018). The first 4 papers compute for the non-zero terms of degree δ values $\zeta = T_\delta(\beta)$ and then deduce δ from ζ. Already Lakshman and Saunders (1995) give an algorithm, for integral β which is based on size comparisons. The trick above to compute an ω is in (Arnold and Kaltofen, 2015). In fact the sparse interpolation algorithms can choose ω and evaluate at $\beta = (\omega + 1/\omega)/2$. If β is chosen, one can factor $x^2 - 2\beta x + 1 = (x - \omega)(x - 1/\omega)$. Algorithm 3.2 below avoids such a factorization and works directly with Chebyshev polynomial evaluations. Numerically, one may use $T_n(\cos(\theta)) = \cos(n\theta)$ (Potts and Tasche, 2014). Our algorithm allows for very large degrees when representing a polynomial modulo a prime p, that is supersparse polynomials in Chebyshev basis in the language of (Kaltofen and Koiran, 2005).

2. Revisiting the Pohlig-Hellman Algorithm

Assumption 2. *Let p be a prime number and $\omega \in \mathbb{Z}_p \setminus \{0\}$ such that the order η of ω divides a known number P which divides $p - 1$. Here η is not known and P is smooth in the sense that all prime factors are reasonably sized. Our algorithm is linear in the squareroot of the largest prime factor of η.*

[2] In (Pohlig and Hellman, 1978) it is stated that R. Silver, R. Schroeppel, and H. Block had similar unpublished algorithms.

Consider the following problem:

Problem 3 (The Discrete Logarithm Problem)**.** *Assuming Assumption 2, compute $\delta \in \mathbb{Z}_\eta$ from input p, $\omega \in \mathbb{Z}_p \setminus \{0\}$, P and $\zeta = \omega^\delta \in \mathbb{Z}_p$.*

The Pohlig-Hellman (1978) Algorithm and Pollard's (1978) rho Algorithm solve Problem 3 (see (Menezes et al., 1996, Chapter 3) for a survey of discrete logarithm algorithms). The original Pohlig-Hellman Algorithm uses the known or precomputed order η of $\omega \in \mathbb{Z}_p \setminus \{0\}$. In this paper we introduce an algorithm (Algorithm 2.2) which is a variant of the original Pohlig-Hellman Algorithm and solves Problem 3 without precomputing the order η of $\omega \in \mathbb{Z}_p \setminus \{0\}$. Our algorithm uses the order of the cyclic multiplicative group $|\mathbb{Z}_p \setminus \{0\}| = p - 1$ instead of the order η of $\omega \in \mathbb{Z}_p \setminus \{0\}$. For our algorithm to be efficient η needs to be smooth.

We start with introducing the following notation:

Definition 4. *A sorted-prime-factorization of an integer $N \in \mathbb{Z}_{>0}$ is a prime-factorization*

$$N = \ell_1^{e_1} \cdots \ell_m^{e_m},\ e_k \in \mathbb{Z}_{>0} \textit{ for } k \in \{1, \ldots, m\} \textit{ and } m \in \mathbb{Z}_{>0}$$

of $N \in \mathbb{Z}_{>0}$ such that the prime numbers that appear in the factorization are sorted in the following way:

$$\ell_1 < \ell_2 < \cdots < \ell_m.$$

Note that the large prime factors (if any) of $N \in \mathbb{Z}_{>0}$ appear at the end of the list $[\ell_1, \ell_2, \ldots, \ell_m]$.

Definition 5. *Let $N \in \mathbb{Z}_{>0}$ and*

$$N = \ell_1^{e_1} \cdots \ell_m^{e_m},\ e_k \in \mathbb{Z}_{>0} \textit{ for } k \in \{1, \ldots, m\} \textit{ and } m \in \mathbb{Z}_{>0}$$

be the sorted-prime-factorization of N. We define the list $\mathcal{L}(N)$ associated with N as follows:

$$\mathcal{L}(N) \overset{\text{def}}{=} [\underbrace{\ell_1, \ldots, \ell_1}_{e_1 \textit{ times}}, \underbrace{\ell_2, \ldots, \ell_2}_{e_2 \textit{ times}}, \ldots, \underbrace{\ell_m, \ldots, \ell_m}_{e_m \textit{ times}}].$$

We denote the i-th element of $\mathcal{L}(N)$ as $\mathcal{L}(N)[i]$. Then we define

$$\mathcal{M}(N, \tau) \overset{\text{def}}{=} \prod_{i=1}^{\tau} \mathcal{L}(N)[i] \textit{ with } \mathcal{M}(N, 0) \overset{\text{def}}{=} 1,$$

and

$$\mathcal{I}(N, \tau) \overset{\text{def}}{=} k \textit{ if } \mathcal{L}(N)[\tau] = \ell_k.$$

In other words, $\mathcal{M}(N, \tau)$ is the product of the first τ elements of the list $\mathcal{L}(N)$ and $\mathcal{I}(N, \tau)$ is the index of the τ-th element of the list $\mathcal{L}(N)$.

2.1. Algorithm Description

Algorithm 2.2 below follows the idea of the Pohlig-Hellman (1978) Algorithm and solves Problem 3, that without precomputing the order η of $\omega \in \mathbb{Z}_p \setminus \{0\}$ and without Chinese remaindering the exponent. Our algorithm works as follows:

Let

$$P = \pi_1^{\epsilon_1} \cdots \pi_n^{\epsilon_n}\ \epsilon_\iota \in \mathbb{Z}_{>0} \text{ for } \iota \in \{1, \ldots, n\} \text{ and } n \in \mathbb{Z}_{>0}$$

be the sorted-prime-factorization of P, which divides $p-1$ (Definition 4) and

$$\eta = \ell_1^{e_1} \cdots \ell_m^{e_m},\ e_k \in \mathbb{Z}_{>0} \text{ for } k \in \{1, \ldots, m\} \text{ and } m \in \mathbb{Z}_{>0}$$

be the sorted-prime-factorization of η, which divides P. We have $m \le n$, $\{\ell_1, \ldots, \ell_m\} \subseteq \{\pi_1, \ldots, \pi_n\}$; and if $\ell_k = \pi_\iota$, then $e_k \le \epsilon_\iota$. Without precomputing the order η of ω, Algorithm 2.2 computes the mixed-mixed-radix representation of $\delta \in \mathbb{Z}_\eta$ with respect to base $\mathcal{M}(\eta, 0)$, $\mathcal{M}(\eta, 1)$, ..., $\mathcal{M}(\eta, e_1 + \cdots + e_m - 1)$ (the representation combines a mixed radix representation with a single radix representation). Namely, it computes

$$\delta = \sum_{\tau=0}^{e_1+\cdots+e_m-1} d_\tau \mathcal{M}(\eta, \tau), \quad d_\tau \in \mathbb{Z}_{\ell_{\mathcal{I}(\eta,\tau+1)}}. \tag{5}$$

Algorithm 2.2 uses the sorted-prime-factorization $P = \pi_1^{\epsilon_1} \cdots \pi_n^{\epsilon_n}$ instead of the sorted-prime-factorization $\eta = \ell_1^{e_1} \cdots \ell_m^{e_m}$ of η during the computation. In the iterative step of Algorithm 2.2, we compute the $(\tau+1)$'st digit $d_\tau \in \mathbb{Z}_{\pi_{\mathcal{I}(\eta,\tau+1)}}$ of (5).

2.2. Algorithm Pohlig-Hellman

Input:
- ▸ A prime number p.
- ▸ $\omega \in \mathbb{Z}_p \setminus \{0\}$.
- ▸ $\zeta = \omega^\delta \in \mathbb{Z}_p$.
- ▸ The factorization of $P \in \mathbb{Z}_{>0}$ such that P divides $p-1$ and such that the unknown multiplicative order η of ω divides P.

Output:
- ▸ The order η of ω.
- ▸ $\delta \bmod \eta$.

1. Let

$$P = \pi_1^{\epsilon_1} \cdots \pi_n^{\epsilon_n},\ \epsilon_\iota \in \mathbb{Z}_{>0} \text{ for } \iota \in \{1, \ldots, n\} \text{ and } n \in \mathbb{Z}_{>0}$$

be the sorted-prime-factorization of P (Definition 4). The deterministic bit complexity of our algorithm is $\max\{\sqrt{\pi_\nu} \mid \pi_\nu \text{ divides } \eta, 1 \le \nu \le n\} (\log p)^{O(1)}$.

$\tau \leftarrow 0$; $\Delta_{-1} \leftarrow 0$; $Q \leftarrow \pi_1^{\epsilon_1} \cdots \pi_n^{\epsilon_n}$ $(= P)$;

2. *While* $\tau \le \epsilon_1 + \cdots + \epsilon_n$ *do Steps* 2a–2c:
At this point the following will always be true.

$$\eta \text{ divides } Q; \tag{6}$$
$$\mathcal{M}(\eta, \tau) = \mathcal{M}(Q, \tau) \text{ (see Definition 5)}; \tag{7}$$
$$\Delta_{\tau-1} \equiv \delta \pmod{\mathcal{M}(Q, \tau)}; \tag{8}$$

2a. *If* $\tau > 0$ *and* $\omega^{\mathcal{M}(Q,\tau)} = 1$ *then return* $\eta \leftarrow \mathcal{M}(Q, \tau)$ *and* $\delta \leftarrow \Delta_{\tau-1}$.

2b. *Compute* $\zeta^{Q/\mathcal{M}(Q,\tau+1)}$ *and* $\omega^{Q/\mathcal{M}(Q,\tau+1)}$.
Here and in Step 2(c)iA we compute high powers, that by repeated squaring. One may pre-compute a list of 2^i'th powers which one can reuse in the repeated squaring algorithm.

2c. $S \leftarrow \{\}$.

2(c)i. *For* λ *from* 0 *to* $\pi_{\mathcal{I}(Q,\tau+1)} - 1$ *do Steps* 2(c)iA *and* 2(c)iB:

```
        2(c)(i)A. If $\zeta^{Q/\mathcal{M}(Q,\tau+1)} = (\omega^{Q/\mathcal{M}(Q,\tau+1)})^{\Delta_{\tau-1}+\lambda\mathcal{M}(Q,\tau)}$ then $S \leftarrow S \cup \{\lambda\}$.
        2(c)(i)B. If both $0, 1$ are in $S$ then $Q \leftarrow Q/\pi_{\mathcal{I}(Q,\tau+1)}$ (the $\epsilon_\nu$ change) and goto
                  Step 2.
    2(c)ii. If $S = \{\lambda\}$ then $d_\tau \leftarrow \lambda$; $\Delta_\tau \leftarrow \Delta_{\tau-1} + d_\tau\mathcal{M}(Q,\tau)$; $\tau \leftarrow \tau + 1$; goto Step 2.
            else the input specs are not satisfied.
```

We now prove that (6–8) are loop invariants. For $\tau = 0$ we have $\mathcal{M}(P, 0) = \mathcal{M}(\eta, 0) = 1$ (Definition 5) and $Q = P$, so the invariants are true. At each iteration, either τ was incremented in the previous iteration, which has produced a new mixed-mixed radix digit of η (5), or Q was divided by a prime. The first time the while loop is entered we can assume that a new digit was found before ($\Delta_{-1} = 0$). The loop attempts to compute d_τ from $\Delta_{\tau-1} + d_\tau\mathcal{M}(\eta, \tau) \equiv \delta \pmod{\mathcal{M}(\eta, \tau + 1)}$. Note that $\mathcal{M}(\eta, \tau) = \mathcal{M}(Q, \tau)$ by the loop invariant, but $\mathcal{M}(\eta, \tau + 1)$ is not known. One uses $\mathcal{M}(Q, \tau + 1)$ instead. Since η divides Q (6) we have

$$\begin{aligned}
\zeta^{\frac{Q}{\mathcal{M}(Q,\tau+1)}} = \omega^{\frac{Q}{\mathcal{M}(Q,\tau+1)}\delta} &= \omega^{\frac{Q}{\mathcal{M}(Q,\tau+1)}(\Delta_\tau+\Gamma_{\tau+1}\mathcal{M}(Q,\tau+1))} \\
&= \omega^{\frac{Q}{\mathcal{M}(Q,\tau+1)}\Delta_\tau}\omega^{Q\Gamma_{\tau+1}} \\
&= \omega^{\frac{Q}{\mathcal{M}(Q,\tau+1)}\Delta_\tau}1^{\Gamma_{\tau+1}} \\
&= \omega^{\frac{Q}{\mathcal{M}(Q,\tau+1)}\Delta_\tau} = \omega^{\frac{Q}{\mathcal{M}(Q,\tau+1)}(\Delta_{\tau-1}+d_\tau\mathcal{M}(\eta,\tau))},
\end{aligned} \tag{9}$$

where $\Gamma_{\tau+1}$ is the quotient obtained when δ is divided by $\mathcal{M}(Q, \tau+1)$. All $\lambda = d_\tau \in \mathbb{Z}_{\pi_{\mathcal{I}(Q,\tau+1)}}$ solve (9) when η divides $(Q/\mathcal{M}(Q, \tau + 1)) \cdot \mathcal{M}(\eta, \tau) = Q/\pi_{\mathcal{I}(Q,\tau+1)}$. If η does not divide $Q/\pi_{\mathcal{I}(Q,\tau+1)}$, then $\omega^{Q/\pi_{\mathcal{I}(Q,\tau+1)}}$ has order $\pi_{\mathcal{I}(Q,\tau+1)}$ (which is a prime) and only one $\lambda = d_\tau \in \mathbb{Z}_{\pi_{\mathcal{I}(Q,\tau+1)}}$ can satisfy (9), namely the $(\tau + 1)$'st digit of δ. Then we must also have $\mathcal{M}(\eta, \tau + 1) = \mathcal{M}(Q, \tau + 1)$, which is (7) for the next iteration, which concludes the proof for the invariants (6–8).

We remark that Algorithm 2.2 does not require the full factorization of $p - 1$, only the factor that is a multiple of the order η of ω. For the application in Prony-like algorithms with high degrees we therefore can select the modulus p such that there is a smooth factor P of $p - 1$ which is sufficiently large to capture the term degree δ. The base ω can then be selected as $\omega = \gamma^{(p-1)/P}$ with $\gamma \in \mathbb{Z}_p$, which guarantees an order which divides P. In summary, $p - 1$ needs not be fully factored, which makes the selection of p and factoring $p - 1$ via the elliptic curve factoring algorithm much easier.

One may deploy Shank's (1972) deterministic baby-steps/giant-steps algorithm or Pollard's (1978) randomized rho Algorithm (or Teske's (1998) version) for Step 2(c)iA at $\lambda = 2$ when S is still empty. Then by (9) d_τ is the discrete log of $Z = \zeta^{\frac{Q}{\mathcal{M}(Q,\tau+1)}}\omega^{-\frac{Q}{\mathcal{M}(Q,\tau+1)}\Delta_{\tau-1}}$ with base $\Omega = \omega^{Q/\pi_{\mathcal{I}(Q,\tau+1)}}$ of order $\pi_{\mathcal{I}(Q,\tau+1)}$, which has deterministic/expected bit complexity $\sqrt{\pi_{\mathcal{I}(Q,\tau+1)}}(\log p)^{O(1)}$. Index calculus-based algorithms can also be used. Because Pollard's rho randomized integer factoring algorithm (or Strassen's deterministic integer factoring algorithm) can compute the factorization of P in $O(\sqrt{\pi_n}(\log p)^{O(1)})$ bit-complexity, by Algorithm 2.1 we have the same asymptotic randomized/deterministic bit complexity

$$\max\{\sqrt{\ell} \mid \ell \text{ a prime divisor of } p - 1\}(\log p)^{O(1)}$$

for the discrete log problem modulo a prime p.

3. Pohlig-Hellman Algorithm for Computing Degree of a Chebyshev Polynomial of the First Kind from Its Value

In addition to (3) we recall further properties of the Chebyshev Polynomials of the First Kind T_n. An alternative to (1) is

$$\begin{bmatrix} T_n(x) \\ T_{n+1}(x) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & 2x \end{bmatrix}^n \begin{bmatrix} 1 \\ x \end{bmatrix} \quad \text{for } n \in \mathbb{Z}. \tag{10}$$

Note that (10) extends the subscript range n to negative integers and by computing the power of 2×2 coefficient matrix gives an algorithm for evaluating all T_n in $O(\log(n))$ scalar operations. We have the following additional property

$$T_n(T_m(x)) = T_{mn}(x) = T_m(T_n(x)) \text{ for } m, n \in \mathbb{Z},$$

which is easily derived from (3).

Assumption 6. *Let $p \geq 3$ be a prime number and $\beta = (\omega + 1/\omega)/2 \in \mathbb{Z}_p \setminus \{0\}$ such that the order η of ω divides a known number P such that:*

i. *If $\beta^2 - 1$ is a quadratic residue then P divides $p - 1$.* In this case $\omega \in \mathbb{Z}_p$.

ii. *If $\beta^2 - 1$ is a quadratic non-residue then P divides $p + 1$.* In this case $\omega = z \in \mathbb{F}_{p^2} = \mathbb{Z}[z]/(z^2 - 2\beta z + 1)$, note that then $z^2 - 2\beta z + 1$ is irreducible in $\mathbb{Z}[z]$, and the order of ω divides $p + 1$ because $1 = \text{Norm}_{\mathbb{F}_{p^2}/\mathbb{Z}_p}(z) = (z \cdot z^p) \bmod (z^2 - 2\beta z + 1)$.[3] Our algorithm does not compute in the extension.

Here η is not known and P is smooth.

The Lakshman-Saunders Algorithm chooses the β, and can do so by choosing $\omega \in \mathbb{Z}_p$. But, say, β is chosen randomly, then for $\beta = 2, 3, \ldots, p-2$ one has exactly $(p-3)/2$ quadratic residues of the form $\beta^2 - 1$ if $p \equiv 3 \pmod 4$, and exactly $(p-5)/2$ quadratic residues of the form $\beta^2 - 1$ if $p \equiv 1 \pmod 4$. The proof of the count can be based by considering the range of $(\beta + 1)/(\beta - 1)$. There are exactly $\phi(p + 1)/2$ of the β's in Assumption 6.ii that lead to z's of order $p + 1$, where ϕ is Euler's totient function. The latter follows because $\phi(p + 1)$ elements γ in $\mathbb{F}_{p^2}$ have order $p + 1$ and each pair γ, γ^p are the roots of one such (irreducible) polynomial.

Problem 7. *Assuming Assumption 6, compute δ from p, $\beta = (\omega + 1/\omega)/2 \in \mathbb{Z}_p \setminus \{0\}$, P and $\zeta = T_\delta(\beta) \in \mathbb{Z}_p$.*

3.1. Algorithm Description

Algorithm 3.2 below follows the pattern of Algorithm 2.2 and solves Problem 7, that without precomputing the order η of ω and without Chinese remaindering the degree of the Chebyshev Polynomial of the First Kind. Our algorithm works as follows:

As in Section 2.1, let

$$P = \pi_1^{\epsilon_1} \cdots \pi_n^{\epsilon_n},\ \epsilon_\iota \in \mathbb{Z}_{>0} \text{ for } \iota \in \{1, \ldots, n\} \text{ and } n \in \mathbb{Z}_{>0}$$
$$\eta = \ell_1^{e_1} \cdots \ell_m^{e_m},\ e_k \in \mathbb{Z}_{>0} \text{ for } k \in \{1, \ldots, m\} \text{ and } m \in \mathbb{Z}_{>0}$$

[3] We owe this observation to Mark van Hoeij.

be the sorted-prime-factorizations of P and η (Definition 4). Here η divides P and P divides $p-1$ (or $p+1$ depending on the quadratic residuosity of β^2-1). By Assumption 6, since $\omega^\eta = 1$ and $\omega^P = 1$, we have $T_\eta(\beta) = (\omega^\eta + 1/\omega^\eta)/2 = 1$ and $T_P(\beta) = (\omega^P + 1/\omega^P)/2 = 1$. Note that the opposite is true: $T_n(\beta) = 1 \implies \omega^n = 1$ because then $(\omega^n - 1)^2 = 0$. As was the case with Algorithm 2.2, without pre-computing the order η of ω, Algorithm 3.2 computes the mixed-mixed-radix representation of $\delta \in \mathbb{Z}_\eta$ (or $-\delta \in \mathbb{Z}_\eta$ because $\zeta = T_\delta(\beta) = T_{-\delta}(\beta)$ by (3)) with respect to base $\mathcal{M}(\eta,0), \mathcal{M}(\eta,1), \ldots, \mathcal{M}(\eta, e_1+\cdots+e_m-1)$:

$$\delta = \sum_{\tau=0}^{e_1+\cdots+e_m-1} d_\tau \mathcal{M}(\eta,\tau), \quad d_\tau \in \mathbb{Z}_{\ell_{\mathcal{I}(\eta,\tau+1)}}. \tag{11}$$

Like Algorithm 2.2 in Section 2, Algorithm 3.2 uses the sorted-prime-factorization $P = \pi_1^{\epsilon_1}\cdots\pi_n^{\epsilon_n}$ instead of the sorted-prime-factorization $\eta = \ell_1^{e_1}\cdots\ell_m^{e_m}$. In the iterative step we compute the $(\tau+1)$'st digit $d_\tau \in \mathbb{Z}_{\pi_{\mathcal{I}(\eta,\tau+1)}}$ of (11) (or $(\tau+1)$'st digit of $-\delta \bmod \eta$).

3.2. Algorithm Computing the Degree of a Chebyshev Polynomial of the First Kind

Input: ▸ A prime number $p \geq 3$.
▸ $\beta = (\omega + 1/\omega)/2 \in \mathbb{Z}_p \setminus \{0\}$.
▸ $\zeta = T_\delta(\beta) \in \mathbb{Z}_p$.
▸ The factorization of $P \in \mathbb{Z}_{>0}$ such that
if $\beta^2 - 1$ is a quadratic residue then P divides $p-1$,
if $\beta^2 - 1$ is a quadratic non-residue then P divides $p+1$
and such that the unknown multiplicative order η of ω divides P.

Output: ▸ The order η of ω.
▸ $\delta \bmod \eta$ or $-\delta \bmod \eta$.

1. Let

$$P = \pi_1^{\epsilon_1}\cdots\pi_n^{\epsilon_n},\ \epsilon_\iota \in \mathbb{Z}_{>0} \text{ for } \iota \in \{1,\ldots,n\} \text{ and } n \in \mathbb{Z}_{>0}$$

be the sorted-prime-factorization of P (Definition 4).

$\tau \leftarrow 0$; $\Delta_{-1} \leftarrow 0$; $Q \leftarrow \pi_1^{\epsilon_1}\cdots\pi_n^{\epsilon_n}$ $(= P)$;

2. *While* $\tau \leq \epsilon_1 + \cdots + \epsilon_n$ *do* Steps 2a–2c:
At this point the following will always be true.

$$\eta \text{ divides } Q; \tag{12}$$

$$\mathcal{M}(\eta,\tau) = \mathcal{M}(Q,\tau) \text{ (see Definition 5)}; \tag{13}$$

$$\Delta_{\tau-1} \equiv \delta \pmod{\mathcal{M}(Q,\tau)} \quad \text{or} \quad \Delta_{\tau-1} \equiv -\delta \pmod{\mathcal{M}(Q,\tau)}; \tag{14}$$

2a. *If* $\tau > 0$ *and* $T_{\mathcal{M}(Q,\tau)}(\beta) = 1$ *then return* $\eta \leftarrow \mathcal{M}(Q,\tau)$ *and* $\delta \leftarrow \Delta_{\tau-1}$.

2b. *Compute* $T_{Q/\mathcal{M}(Q,\tau+1)}(\zeta)$ *and* $T_{Q/\mathcal{M}(Q,\tau+1)}(\beta)$.
Here and in Step 2(c)iA we compute high degree values of the Chebyshev polynomials by computing the matrix powers in (10) by repeated squaring. One may pre-compute a list of 2^i'th powers of the matrices which one can reuse in the repeated squaring algorithm.

2c. $S \leftarrow \{\}$.

2(c)i. *For* λ *from* 0 *to* $\pi_{\mathcal{I}(Q,\tau+1)} - 1$ *do* Steps 2(c)iA and 2(c)iB:

2(c)(i)A. *If*

$$T_{Q/\mathcal{M}(Q,\tau+1)}(\zeta) = T_{\Delta_{\tau-1}+\lambda\mathcal{M}(Q,\tau)}(\,T_{Q/\mathcal{M}(Q,\tau+1)}(\beta)\,) \tag{15}$$

then $S \leftarrow S \cup \{\lambda\}$.

2(c)(i)B. *If* $0, 1, 2$ *are in* S *then* $Q \leftarrow Q/\pi_{\mathcal{I}(Q,\tau+1)}$ *(the* ϵ_ν *change) and goto Step* 2.

2(c)ii. *If* $S = \{\lambda\}$ *then* $d_\tau \leftarrow \lambda$; $\Delta_\tau \leftarrow \Delta_{\tau-1} + d_\tau\mathcal{M}(Q,\tau)$; $\tau \leftarrow \tau + 1$; *goto Step* 2. *Else the input specs are not satisfied.*

2(c)iii. *If* $S = \{\lambda_1, \lambda_2\}$ *then* $d_\tau \leftarrow \min(\lambda_1, \lambda_2)$; $\Delta_\tau \leftarrow \Delta_{\tau-1} + d_\tau\mathcal{M}(Q,\tau)$; $\tau \leftarrow \tau + 1$; *goto Step* 2. *Else the input specs are not satisfied.* Note that this step chooses one of the $\delta \bmod \eta$ and $-\delta \bmod \eta$ to proceed, and it appears only once during the computation.

As by our discussion at the end of Section 2, we now prove that (12–14) are loop invariants. For $\tau = 0$ we have $\mathcal{M}(P,0) = \mathcal{M}(\eta,0) = 1$ (Definition 5) and $Q = P$, so the invariants are true. At each iteration, either τ was incremented in the previous iteration, which has produced a new mixed-mixed radix digit of η (11), or Q was divided by a prime. The first time the while loop is entered we can assume that a new digit was found before ($\Delta_{-1} = 0$). The loop attempts to compute d_τ from $\Delta_{\tau-1} + d_\tau\mathcal{M}(\eta,\tau) \equiv \delta \pmod{\mathcal{M}(\eta,\tau+1)}$ (or attempts to compute d_τ from $\Delta_{\tau-1} + d_\tau\mathcal{M}(\eta,\tau) \equiv -\delta \pmod{\mathcal{M}(\eta,\tau+1)}$). Note that $\mathcal{M}(\eta,\tau) = \mathcal{M}(Q,\tau)$ by the loop invariant, but $\mathcal{M}(\eta,\tau+1)$ is not known. One uses $\mathcal{M}(Q,\tau+1)$ instead. Since η divides Q (12) we have

$$\begin{aligned}
T_{\frac{Q}{\mathcal{M}(Q,\tau+1)}}(\zeta) = T_{\frac{Q}{\mathcal{M}(Q,\tau+1)}}(T_\delta(\beta)) &= T_{\frac{Q}{\mathcal{M}(Q,\tau+1)}\delta}(\beta)\\
&= T_{\frac{Q}{\mathcal{M}(Q,\tau+1)}(\Delta_\tau+\Gamma_{\tau+1}\mathcal{M}(Q,\tau+1))}(\beta)\\
&= \frac{1}{2}\left(\omega^{\frac{Q}{\mathcal{M}(Q,\tau+1)}(\Delta_\tau+\Gamma_{\tau+1}\mathcal{M}(Q,\tau+1))} + \omega^{-\frac{Q}{\mathcal{M}(Q,\tau+1)}(\Delta_\tau+\Gamma_{\tau+1}\mathcal{M}(Q,\tau+1))}\right)\\
&= \frac{1}{2}\left(\omega^{\frac{Q}{\mathcal{M}(Q,\tau+1)}\Delta_\tau}\omega^{Q\Gamma_{\tau+1}} + \omega^{-\frac{Q}{\mathcal{M}(Q,\tau+1)}\Delta_\tau}\omega^{-Q\Gamma_{\tau+1}}\right)\\
&= \frac{1}{2}\left(\omega^{\frac{Q}{\mathcal{M}(Q,\tau+1)}\Delta_\tau}1^{\Gamma_{\tau+1}} + \omega^{-\frac{Q}{\mathcal{M}(Q,\tau+1)}\Delta_\tau}1^{-\Gamma_{\tau+1}}\right)\\
&= \frac{1}{2}\left(\omega^{\frac{Q}{\mathcal{M}(Q,\tau+1)}\Delta_\tau} + \omega^{-\frac{Q}{\mathcal{M}(Q,\tau+1)}\Delta_\tau}\right)\\
&= T_{\frac{Q}{\mathcal{M}(Q,\tau+1)}\Delta_\tau}(\beta) = T_{\frac{Q}{\mathcal{M}(Q,\tau+1)}(\Delta_{\tau-1}+d_\tau\mathcal{M}(Q,\tau))}(\beta),
\end{aligned} \tag{16}$$

where $\Gamma_{\tau+1}$ is the quotient obtained when δ is divided by $\mathcal{M}(Q,\tau+1)$. All $\lambda = d_\tau \in \mathbb{Z}_{\pi_{\mathcal{I}(Q,\tau+1)}}$ solve (16) when η divides $(Q/\mathcal{M}(Q,\tau+1)) \cdot \mathcal{M}(\eta,\tau) = Q/\pi_{\mathcal{I}(Q,\tau+1)}$. If η does not divide $Q/\pi_{\mathcal{I}(Q,\tau+1)}$, then at most two $\lambda = d_\tau \in \mathbb{Z}_{\pi_{\mathcal{I}(Q,\tau+1)}}$ can satisfy (16), namely the $(\tau+1)$'st digit of δ or the $(\tau+1)$'st digit of $-\delta \bmod \eta$. If there are two λ that satisfy (16) then we choose the smallest one (one can also chose the greater one). That choice appears only once while Algorithm 3.2 proceeds. After that choice, there is only one $\lambda = d_\tau \in \mathbb{Z}_{\pi_{\mathcal{I}(Q,\tau+1)}}$ that satisfies (16) in each next iteration. Then we must also have $\mathcal{M}(\eta,\tau+1) = \mathcal{M}(Q,\tau+1)$, which is (13) for the next iteration.

We do not know how to compute λ in Step 2(c)i by solving (15) via the matrix equations (10) in a Shanks's baby steps/giant steps manner, other than by computing $T_{\delta+1}(\beta)$ via retrieving ω and $\omega^{\pm\delta}$ from the quadratic equations (4). The sign in the exponent is determined by the choice of d_0 in Step 2(c)iii. When doing so, one may continue with Algorithm 2.2 instead.

References

Arnold, A., Kaltofen, E., 2015. Error-correcting sparse interpolation in the Chebyshev basis. In: ISSAC'15 Proc. 2015 ACM Internat. Symp. Symbolic Algebraic Comput. Association for Computing Machinery, New York, N. Y., pp. 21–28, URL: http://www.math.ncsu.edu/~kaltofen/bibliography/15/ArKa15.pdf.

Giesbrecht, M., Labahn, G., Lee, W., 2004. Symbolic-numeric sparse polynomial interpolation in Chebyshev basis and trigonometric interpolation. In: Proc. Workshop on Computer Algebra in Scientific Computation (CASC). pp. 195–205, https://cs.uwaterloo.ca/~mwg/files/triginterp.pdf.

Imamoglu, E., Kaltofen, E. L., Yang, Z., 2018. Sparse polynomial interpolation with arbitrary orthogonal polynomial bases. In: Proceedings of the 2018 ACM on International Symposium on Symbolic and Algebraic Computation. ISSAC '18. ACM, New York, NY, USA, pp. 223–230.
URL http://doi.acm.org/10.1145/3208976.3208999

Kaltofen, E., Koiran, P., 2005. On the complexity of factoring bivariate supersparse (lacunary) polynomials. In: Kauers, M. (Ed.), ISSAC'05 Proc. 2005 Internat. Symp. Symbolic Algebraic Comput. ACM Press, New York, N. Y., pp. 208–215, ACM SIGSAM's ISSAC 2005 Distinguished Paper Award. URL: http://www.math.ncsu.edu/~kaltofen/bibliography/05/KaKoi05.pdf.

Kaltofen, E., Lee, W., 2003. Early termination in sparse interpolation algorithms. J. Symbolic Comput. 36 (3–4), 365–400, special issue Internat. Symp. Symbolic Algebraic Comput. (ISSAC 2002). Guest editors: M. Giusti & L. M. Pardo. URL: http://www.math.ncsu.edu/~kaltofen/bibliography/03/KL03.pdf.

Lakshman, Y. N., Saunders, B. D., Apr. 1995. Sparse polynomial interpolation in nonstandard bases. SIAM J. Comput. 24 (2), 387–397.
URL http://dx.doi.org/10.1137/S0097539792237784

Menezes, A. J., Vanstone, S. A., Van Oorschot, P. C., 1996. Handbook of Applied Cryptography, 1st Edition. CRC Press, Inc., Boca Raton, FL, USA.

Pohlig, S., Hellman, M., Sep. 1978. An improved algorithm for computing logarithms over GF(p) and its cryptographic significance (corresp.). IEEE Trans. Inf. Theor. 24 (1), 106–110.
URL https://doi.org/10.1109/TIT.1978.1055817

Pollard, J. M., 1978. Monte carlo methods for index computation mod p. Mathematics of Computation 32 (143), 918–924.

Potts, D., Tasche, M., 2014. Sparse polynomial interpolation in Chebyshev bases. Linear Algebra and Applications 441, 61–87.

Shanks, D., 1972. Five number-theoretical algorithms. In: Proc. 2nd Manitoba Conf. Numerical Math. pp. 51–70.

Teske, E., 1998. Speeding up Pollard's rho method for computing discrete logarithms. In: International Algorithmic Number Theory Symposium. Springer, pp. 541–554.