The Frobenius FFT

Joris van der Hoeven, Robin Larrieu

The Frobenius FFT

Joris van der Hoeven[a], Robin Larrieu[b]
CNRS, Laboratoire d'informatique
École polytechnique
91128 Palaiseau Cedex
France
a. *Email:* vdhoeven@lix.polytechnique.fr
b. *Email:* larrieu@lix.polytechnique.fr

April 26, 2017

ABSTRACT

Let $\mathbb{F}_q$ be the finite field with q elements and let ω be a primitive n-th root of unity in an extension field $\mathbb{F}_{q^d}$ of $\mathbb{F}_q$. Given a polynomial $P \in \mathbb{F}_q[x]$ of degree less than n, we will show that its discrete Fourier transform $(P(1), P(\omega), ..., P(\omega^{n-1})) \in \mathbb{F}_{q^d}^n$ can be computed essentially d times faster than the discrete Fourier transform of a polynomial $Q \in \mathbb{F}_{q^d}[x]$ of degree less than n, in many cases. This result is achieved by exploiting the symmetries provided by the Frobenius automorphism of $\mathbb{F}_{q^d}$ over $\mathbb{F}_q$.

Keywords

FFT; finite field; complexity bound; Frobenius automorphism

A.C.M. SUBJECT CLASSIFICATION: G.1.0 Computer-arithmetic, F.2.1 Number-theoretic computations

A.M.S. SUBJECT CLASSIFICATION: 68W30, 68Q17, 68W40

1. INTRODUCTION

Let $n = 2^\ell$ and let $\omega = e^{2\pi i/n}$ be a primitive n-th root of unity in $\mathbb{C}$. The traditional algorithm for computing discrete Fourier transforms [7] takes a polynomial $P \in \mathbb{C}[x]$ of degree $<n$ on input and returns the vector $(P(1), P(\omega), ..., P(\omega^{n-1}))$. If P actually admits real coefficients, then we have $P(\omega^{-i}) = \overline{P(\omega^i)}$ for all i, so roughly $n/2$ of the output values are superfluous. Because of this symmetry, it is well known that such "real FFTs" can be computed roughly twice as fast as their complex counterparts [3, 25].

More generally, there exists an abundant literature on the computation of FFTs of polynomials with various types of symmetry. *Crystallographic* FFT algorithms date back to [26], with contributions as recent as [20], but are dedicated to crystallographic symmetries. So called *lattice FFTs* have been studied in a series of papers initiated by [11] and continued in [27, 4]. A more general framework due to [1] was recently revisited from the point of view of high-performance code generation [18]. Further symmetric FFT algorithms and their truncated versions were developed in [16].

In this paper, we focus on discrete Fourier transforms of polynomials $P \in \mathbb{F}_q[x]$ with coefficients in the finite field with q elements. In general, primitive n-th roots of unity ω only exist in extension fields of $\mathbb{F}_q$, say $\omega \in \mathbb{F}_{q^d}$. This puts us in a similar situation as in the case of real FFTs: our primitive root of unity ω lies in an extension field of the coefficient field of the polynomial. This time, the degree of the extension is $d = [\mathbb{F}_{q^d} : \mathbb{F}_q]$ instead of $2 = [\mathbb{C} : \mathbb{R}]$. As in the case of real FFTs, it is natural to search for algorithms that allow us to compute the DFT of a polynomial in $\mathbb{F}_q[x]$ approximately d times faster than the DFT of a polynomial in $\mathbb{F}_{q^d}[x]$.

The main purpose of this paper is to show that such a speed-up by an approximate factor of d can indeed be achieved. The idea is to use the Frobenius automorphism $\phi_q : \mathbb{F}_{q^d} \to \mathbb{F}_{q^d}$; $x \mapsto x^q$ as the analogue of complex conjugation. If $P \in \mathbb{F}_q[x]$, then we will exploit the fact that $P(\phi_q^k(\omega^i)) = \phi_q^k(P(\omega^i))$ for all $0 \leqslant i < n$ and $0 \leqslant k < d$. This means that it suffices to evaluate P at only one element of each orbit for the action of the Frobenius automorphism on the cyclic group $\langle \omega \rangle$ generated by ω. We will give an efficient algorithm for doing this, called the *Frobenius FFT*.

Our main motivation behind the Frobenius FFT is the development of faster practical algorithms for polynomial multiplication over finite fields of small characteristic. In [12], we have shown how to reduce multiplication in $\mathbb{F}_2[x]$ to multiplication in $\mathbb{F}_{2^{60}}[x]$, while exploiting the fact that efficient DFTs are available over $\mathbb{F}_{2^{60}}$. Unfortunately, this reduction involves an overhead of a factor 2 for zero padding. The Frobenius FFT can be thought of as a more direct way to use evaluation-interpolation strategies over $\mathbb{F}_{2^{60}}$ for polynomial multiplication over $\mathbb{F}_2$. We have not yet implemented the algorithms in the present paper, but the ultimate hope is to make such multiplications twice as efficient as in [12].

Let us briefly detail the structure of this paper. In section 2, we first recall some standard complexity results for finite field arithmetic. We pursue with a quick survey of the fast Fourier transform (FFT) in section 3. The modern formulation of this algorithm is due to Cooley and Tukey [7], but the algorithm was already known to Gauß [9].

In section 4, we introduce the *discrete Frobenius Fourier transform* (DFFT) and describe several basic algorithms to compute DFFTs of prime order. There are essentially two efficient ways to deal with composite orders. Both approaches are reminiscent of analogue strategies for real FFTs [25], but the technical details are more complicated.

The first strategy is to reduce the computation of d discrete Fourier transforms of polynomials $P_0, ..., P_{d-1} \in \mathbb{F}_q[x]$ of degree $<n$ to a single discrete Fourier transform of a polynomial $P \in \mathbb{F}_{q^d}[x]$ of degree $<n$. This elegant reduction is the subject of section 5, but it is only applicable if we need to compute many transforms. Furthermore, at best, it only leads to the gain of a factor $d / \log d$ instead of d, due to numerous applications of the Frobenius automorphism ϕ_q.

The second strategy, to be detailed in section 6, is to carefully exploit the symmetries in the usual Cooley–Tukey FFT; we call the resulting algorithm the *Frobenius FFT* (FFFT). The complexity analysis is more intricate than for the algorithm from section 5 and the complete factor d is only regained under certain conditions. Fortunately, these conditions are satisfied for the most interesting applications to polynomial multiplication in $\mathbb{F}_2[x]$ or when n becomes very large with respect to d.

In this paper, we focus on direct DFFTs and FFFTs. Inverse FFFTs can be computed in a straightforward way by reversing the main algorithm from section 6. Inverting the various algorithms from section 4 for DFFTs of prime orders requires a bit more effort, but should not raise any serious difficulties. The same holds for the algorithms to compute multiple DFFTs from section 5.

Besides inverse transforms, it is possible to consider truncated Frobenius FFTs, following [14, 15, 21], but this goes beyond the scope of this paper. It might also be worth investigating the use of Good's algorithm [10] as an alternative to the Cooley–Tukey FFT, since the order n usually admits many distinct prime divisors in our setting.

2. FINITE FIELD ARITHMETIC

Throughout this paper and unless stated otherwise, we will assume the bit complexity model for Turing machines with a finite number of tapes [23]. Given a field $\mathbb{K}$, we will write $\mathbb{K}[x]_{<n}$ for the set of polynomials in $\mathbb{K}[x]$ of degree $<n$.

Let us write $q = p^\delta$ for some prime number p and $\delta \geqslant 1$. Elements in the finite field $\mathbb{F}_q$ are usually represented as elements of $\mathbb{F}_p[x] / (\mu(x))$, where $\mu \in \mathbb{F}_p[x]$ is a monic irreducible polynomial of degree δ. We will denote by $\mathsf{M}_q(n)$ the cost to multiply two polynomials in $\mathbb{F}_q[x]_{<n}$. Multiplications in $\mathbb{F}_q$ can be reduced to a constant number of multiplications of polynomials in $\mathbb{F}_p[x]_{<\delta}$ via the precomputation of a "pre-inverse" of μ. The multiplication of two polynomials in $\mathbb{F}_q[x]_{<n}$ can be reduced to the multiplication of two polynomials in $\mathbb{F}_p[x]_{<2\delta n}$ using Kronecker substitution [8, Corollary 8.27]. In other words, we have $\mathsf{M}_q(n) = O(\mathsf{M}_p(\delta n))$.

The best currently known bound for $\mathsf{M}_q(n)$ was established in [13]. It was shown there that

$$\mathsf{M}_q(n) = O(n \log q \log(n \log q)\, 8^{\log^*(n \log q)}), \tag{2.1}$$

uniformly in q, where

$$\begin{aligned} \log^* x &:= \min \{k \in \mathbb{N} : \log^{\circ k} x \leqslant 1\}, \\ \log^{\circ k} &:= \log \circ \underset{k \times}{\cdots} \circ \log. \end{aligned} \tag{2.2}$$

Sometimes, we rather need to perform t multiplications $P_1\, Q, ..., P_t\, Q$ with $P_1, ..., P_t, Q \in \mathbb{F}_q[x]_{<n}$. The complexity of this operation will be denoted by $\mathsf{M}_q(n, t)$. Throughout this paper, it will be convenient to assume that $\mathsf{M}_q(n, t) / (n \log q)$ is increasing in both n and q. In other words, $\mathsf{M}_{p^\delta}(n, t) / (n\, \delta)$ is increasing in both n and δ.

The computation of the Frobenius automorphism ϕ_q in $\mathbb{F}_{q^d}$ requires $O(\log\ q)$ multiplications in $\mathbb{F}_{q^d}$ when using binary powering, so this operation has complexity $O(\mathsf{M}_q(d) \log q)$. This cost can be lowered by representing elements in $\mathbb{F}_{q^d}$ with respect to so-called normal bases [5]. However, multiplication in $\mathbb{F}_{q^d}$ becomes more expensive for this representation.

The discrete Frobenius Fourier transform potentially involves computations in all intermediate fields $\mathbb{F}_{q^e}$ where e divides d. The necessary minimal polynomials μ for these fields and the corresponding pre-inverses can be kept in a table. The bitsize of this table is bounded by $2 \lceil \log_2 q \rceil \sum_{e|d} d$. It is known [22] that $\sigma(d) := \sum_{e|d} d = O(d \log \log d)$. Consequently, the bitsize of the table requires $O(d \log \log d \log q)$ storage.

Another important operation in finite fields is *modular composition*: given $P, Q \in \mathbb{F}_q[x]_{<n}$ and a monic polynomial $R \in \mathbb{F}_q[x]$ of degree n, the aim is to compute the remainder $(P \circ Q)$ rem R of the euclidean division of $P \circ Q$ by R. If R is the minimal polynomial of the finite field $\mathbb{F}_{q^n} \equiv \mathbb{F}_q[x] / (R(x))$, then this operation also corresponds to the evaluation of the polynomial $P \in \mathbb{F}_q[x]_{<n}$ at a point in $\mathbb{F}_{q^n}$. If R and R' are two distinct monic irreducible polynomials in $\mathbb{F}_q[x]$ of degree n, then the conversion of an element in $\mathbb{F}_q[x] / (R(x))$ to an element in $\mathbb{F}_q[x] / (R'(x))$ also boils down to one modular composition of degree n over $\mathbb{F}_q$.

We will denote by $\mathsf{C}_q(n)$ the cost of a modular composition as above. Theoretically speaking, Kedlaya and Umans proved that $\mathsf{C}_q(n) = (n \log q)^{1+o(1)}$ [19]. From a practical point of view, one rather has $\mathsf{C}_q(n) = O(n^2 \log q)$. If n is smooth and one needs to compute several modular compositions for the same modulus, then algorithms of quasi-linear complexity do exist [17].

Given primitive elements $\alpha \in \mathbb{F}_{q^d}$ and $\beta \in \mathbb{F}_{q^e}$ with $e \,|\, d$, the paper [17] also contains efficient algorithms for conversions between $\mathbb{F}_q[\alpha]$ and $\mathbb{F}_q[\beta][\alpha]$. Using modular composition, we notice that this problem reduces in time $\mathsf{C}_q(d) + (d / e)\ \mathsf{C}_q(e) + \mathsf{C}_{q^e}(d / e)$ to the case where we are free to choose primitive elements α and β that suit us. We let $\mathsf{V}_q(d, e)$ be a cost function for conversions between $\mathbb{F}_q[\alpha]$ and $\mathbb{F}_q[\beta][\alpha]$ and denote $\mathsf{V}_q(d) = \max_{e|e'|d} (d / e') \mathsf{V}_q(e', e)$.

3. FAST FOURIER TRANSFORMS

3.1 The discrete Fourier transform

Let $\mathbb{K}$ be any field and let $\omega \in \mathbb{K}$ be a primitive n-th root of unity for some $n > 0$. Now consider a polynomial $P \in \mathbb{K}[x]_{<n}$, where

$$\mathbb{K}[x]_{<n} \ := \ \{P \in \mathbb{K}[x] : \deg P < n\}.$$

The *discrete Fourier transform* $\mathrm{DFT}_\omega(P)$ of P with respect to ω is the vector

$$\mathrm{DFT}_\omega(P) \ = \ (P(1), P(\omega), ..., P(\omega^{n-1})) \in \mathbb{K}^n.$$

If n is a small number, then $\mathrm{DFT}_\omega(P)$ can be computed by evaluating $P(\omega^i)$ directly for $i = 0, ..., n-1$, using Horner's method for instance. If n is a prime number, then Rader and Bluestein have given efficient methods for the computation of discrete Fourier transforms of order n; see [24, 6] and subsection 3.2 below.

In subsection 3.4, we are interested in the case when n is composite. In that case, one may compute $\mathrm{DFT}_\omega(P)$ using the *fast Fourier transform* (or FFT). The modern formulation of this algorithm is due to Cooley and Tukey [7], but the algorithm was already known to Gauß [9]. The output $(P(\omega^{\bar{\imath}}))_{i<n}$ of the FFT uses a mirrored indexation $\bar{\imath}$ that we will introduce in subsection 3.3 below.

We will denote by $\mathsf{F}_{q^d}(n)$ the cost of computing a FFT in the field $\mathbb{F}_{q^d}$.

3.2 Rader reduction

Assume that we wish to compute a DFT of large prime length n. The multiplicative group $\mathbb{F}_n^*$ is cyclic, of order $n-1$. Let $g \in \{1, ..., n-1\}$ be such that $g \bmod n$ is a generator of this cyclic group. Given a polynomial $P = p_0 + \cdots + p_{n-1} x^{n-1}$ and $j \in \{1, ..., n-1\}$ with $j = g^l \operatorname{rem} n$, we have

$$\begin{aligned}
\mathrm{DFT}_\omega(P)_{g^l \operatorname{rem} n} &= p_0 + \sum_{i=1}^{n-1} p_i \, \omega^{i (g^l \operatorname{rem} n)} \\
&= p_0 + \sum_{k=0}^{n-2} p_{g^{-k} \operatorname{rem} n} \, \omega^{(g^{-k} \operatorname{rem} n)(g^l \operatorname{rem} n)} \\
&= p_0 + \sum_{k=0}^{n-2} p_{g^{-k} \operatorname{rem} n} \, \omega^{g^{l-k} \operatorname{rem} n}.
\end{aligned}$$

Setting

$$\begin{aligned}
u_k &= p_{g^{-k} \operatorname{rem} n} & U &= u_0 + \cdots + u_{n-2} \, y^{n-2} \\
v_l &= \omega^{g^l \operatorname{rem} n} & V &= v_0 + \cdots + v_{n-2} \, y^{n-2} \\
w_l &= \mathrm{DFT}_\omega(P)_{g^l \operatorname{rem} n} - p_0 & W &= w_0 + \cdots + w_{n-2} \, y^{n-2},
\end{aligned}$$

it follows that $W = U \, V$, when regarding U, V and W as elements of

$$\mathbb{K}[y]^\circ_{n-1} := \mathbb{K}[y] / (y^{n-1} - 1).$$

In other words, we have essentially reduced the computation of a discrete Fourier transform of length n to a so-called *cyclic convolution* of length $n-1$.

3.3 Generalized bitwise mirrors

Let n be a positive integer with a factorization $n = m_0 \cdots m_{\ell-1}$ such that $m_j > 1$ for each j. Then any index $i < n$ can uniquely be written in the form

$$i = i_0 \cdot m_1 \cdots m_{\ell-1} + \cdots + i_{\ell-2} \cdot m_{\ell-1} + i_{\ell-1},$$

where $i_j < m_j$ for all $j < \ell$. We call $i_0, ..., i_{\ell-1}$ the *digits* of i when written in the *mixed base* $v = (m_0, ..., m_{\ell-1})$. We also define the *$v$-mirror* of i by

$$[i]_v = i_0 + i_1 \cdot m_0 + i_2 \cdot m_0 \, m_1 + \cdots + i_{\ell-1} \cdot m_0 \cdots m_{\ell-2}.$$

Whenever v is clear from the context, we simply write $\bar{\imath}$ instead of $[i]_v$.

If $v = (m_0)$ has length one, then we notice that

$$[i]_{(m_0)} = i.$$

Setting $\hat{v} = (m_{\ell-1}, ..., m_0)$, we also have

$$[[i]_v]_{\hat{v}} = i.$$

Finally, given $h \leqslant d$, let $v^\sharp = (m_0, ..., m_{h-1})$, $v^\flat = (m_h, ..., m_{\ell-1})$, $n^\sharp = m_0 \cdots m_{h-1}$ and $n^\flat = m_h \cdots m_{\ell-1}$, so that $v = (v^\sharp, v^\flat)$ and $n = n^\sharp n^\flat$. Then it is not hard to see that for all $i^\sharp < n^\sharp$ and $i^\flat < n^\flat$, we have

$$[i^\sharp n^\flat + i^\flat]_{v^\sharp, v^\flat} = [i^\flat]_{v^\flat} n^\sharp + [i^\sharp]_{v^\sharp}.$$

3.4 The Cooley–Tukey FFT

We are now in a position to recall how the FFT works. Let $v^\sharp = (m_0, ..., m_{h-1})$, $v^\flat = (m_h, ..., m_{\ell-1})$, $n^\sharp = m_0 \cdots m_{h-1}$ and $n^\flat = m_h \cdots m_{\ell-1}$ be as at the end of the previous subsection. We denote $\omega^\sharp = \omega^{n^\flat}$ and notice that $\omega^\sharp$ is a primitive $n^\sharp$-th root of unity. Similarly, $\omega^\flat := \omega^{n^\sharp}$ is a primitive $n^\flat$-th root of unity. Recall that $\bar{\imath} \equiv [i]_v$ for $i < n$. For any $i < n$ and $j < n$, we have

$$\omega^{\bar{\imath} j} = \omega^{(\overline{i^\flat} n^\sharp + \overline{i^\sharp})(j^\sharp n^\flat + j^\flat)} = \omega^{\overline{i^\sharp} j^\flat} (\omega^\flat)^{\overline{i^\flat} j^\flat} (\omega^\sharp)^{\overline{i^\sharp} j^\sharp},$$

so that

$$P(\omega^{\bar{\imath}}) = \sum_{j^\flat < n^\flat} \left[\omega^{j^\flat \overline{i^\sharp}} \sum_{j^\sharp < n^\sharp} P_{j^\sharp n^\flat + j^\flat} \, (\omega^\sharp)^{j^\sharp \overline{i^\sharp}} \right] (\omega^\flat)^{j^\flat \overline{i^\flat}}. \tag{3.1}$$

In this formula, one recognizes two polynomial evaluations of orders $n^\sharp$ and $n^\flat$. More precisely, we may decompose the polynomial P as

$$P = P_0^\sharp(x^\sharp) + P_1^\sharp(x^\sharp) \, x + \cdots + P_{n^\flat - 1}^\sharp(x^\sharp) \, x^{n^\flat - 1},$$

where $x^\sharp = x^{n^\flat}$ and $P_0^\sharp, ..., P_{n^\flat-1}^\sharp \in \mathbb{K}[x^\sharp]_{<n^\sharp}$. This allows us to rewrite (3.1) as

$$P(\omega^{\bar{\imath}}) = \sum_{j^\flat < n^\flat} (\omega^{j^\flat \overline{i^\sharp}} P_{j^\flat}^\sharp((\omega^\sharp)^{\overline{i^\sharp}})) \, (\omega^\flat)^{j^\flat \overline{i^\flat}}. \tag{3.2}$$

For each $i^\sharp < n^\sharp$, we may next introduce the polynomial $P_{i^\sharp}^\flat \in \mathbb{K}[x]_{<n^\flat}$ by

$$P_{i^\sharp}^\flat(x) = P_0^\sharp((\omega^\sharp)^{\overline{i^\sharp}}) + \cdots + P_{n^\flat-1}^\sharp((\omega^\sharp)^{\overline{i^\sharp}}) \, x^{n^\flat - 1}$$

The relation (3.2) now further simplifies into

$$P(\omega^{\bar{\imath}}) = P_{i^\sharp}^\flat(\omega^{\overline{i^\sharp}} (\omega^\flat)^{\overline{i^\flat}}). \tag{3.3}$$

This leads to the following recursive algorithm for the computation of discrete Fourier transforms.

Algorithm $\mathbf{FFT}_\omega(P)$
INPUT: $P \in \mathbb{K}[x]_{<n}$
OUTPUT: $(P(\omega^{\bar{\imath}}))_{i<n}$.

> **if** $\ell = 1$ **then** compute the answer using a direct algorithm and return it
> Take $0 < h < \ell$, $n^\sharp = m_0 \cdots m_{h-1}$ and $n^\flat = m_h \cdots m_{\ell-1}$
> Let $\omega^\sharp := \omega^{n^\flat}$, $\omega^\flat := \omega^{n^\sharp}$, $x^\sharp := x^{n^\flat}$
> Decompose $P(x) = P_0^\sharp(x^\sharp) + \cdots + P_{n^\flat-1}^\sharp(x^\sharp) \, x^{n^\flat - 1}$
> **for** $i^\flat < n^\flat$ **do**
> Compute $(P_{i^\flat}^\sharp((\omega^\sharp)^{\overline{i^\sharp}}))_{i^\sharp < n^\sharp} := \mathrm{FFT}_{\omega^\sharp}(P_{i^\flat}^\sharp)$
> **for** $i^\sharp < n^\sharp$ **do**
> Let $P_{i^\sharp}^\flat(x) := P_0^\sharp((\omega^\sharp)^{\overline{i^\sharp}}) + \cdots + P_{n^\flat-1}^\sharp((\omega^\sharp)^{\overline{i^\sharp}}) \, x^{n^\flat - 1}$
> Let $\tilde{P}_{i^\sharp}^\flat(x) := P_{i^\sharp}^\flat(\omega^{\overline{i^\sharp}} x)$
> Compute $(P_{i^\sharp}^\flat(\omega^{\overline{i^\sharp}} (\omega^\flat)^{\overline{i^\flat}}))_{i^\flat < n^\flat} := \mathrm{FFT}_{\omega^\flat}(\tilde{P}_{i^\sharp}^\sharp)$
> **return** $(P(\omega^{\bar{\imath}}))_{i<n} = (P_{i^\sharp}^\flat(\omega^{\overline{i^\sharp}} (\omega^\flat)^{\overline{i^\flat}}))_{i = i^\sharp n^\flat + i^\flat < n}$

If the m_i are all bounded, then this algorithm requires $O(n \log n)$ operations in $\mathbb{K}$. For practical purposes it is best though to pick h such that $n^\flat$ and $n^\sharp$ are as close to $\sqrt{n}$ as possible. This approach is most "cache friendly" in the sense that it keeps data as close as possible in memory [2].

4. FROBENIUS FOURIER TRANSFORMS

4.1 The discrete Frobenius Fourier transform

Let $\mathbb{F}_q \subseteq \mathbb{F}_{q^d}$ be an extension of finite fields. Let ϕ_q: $\mathbb{F}_{q^d} \to \mathbb{F}_{q^d}$; $x \mapsto x^q$ denote the Frobenius automorphism of $\mathbb{F}_{q^d}$ over $\mathbb{F}_q$. We recall that the group $\langle \phi_q \rangle$ generated by ϕ_q has size d and that it coincides with the Galois group of $\mathbb{F}_{q^d}$ over $\mathbb{F}_q$.

Let $\omega \in \mathbb{F}_{q^d}$ be a primitive n-th root of unity with $\mathbb{F}_{q^d} = \mathbb{F}_q[\omega]$. Recall that $n \mid q^d - 1$ and $\gcd(n, q) = 1$. We will write $\langle \omega \rangle$ for the cyclic group of size n generated by ω. The Frobenius automorphism ϕ_q naturally acts on $\langle \omega \rangle$ and we denote by $r_q(\omega^i)$ the order of an element ω^i under this action. Notice that $r_q(\omega^i) \mid d$. A subset $\mathcal{S} \subseteq \langle \omega \rangle$ is called a *cross section* if for every $\omega^j \in \langle \omega \rangle$, there exists exactly one $\omega^i \in \mathcal{S}$ such that $\omega^i = \phi_q^k(\omega^j)$ for some $k \in \mathbb{N}$. We will denote this element ω^i by $\pi_{\mathcal{S}}(\omega^j)$. We will also denote the corresponding set of indices by $\mathcal{I} = \{i < n : \omega^i \in \mathcal{S}\}$.

Now consider a polynomial $P \in \mathbb{F}_q[x]_{<n}$. Recall that its discrete Fourier transform with respect to ω is defined to be the vector

$$\mathrm{DFT}_\omega(P) \;=\; (P(1), P(\omega), ..., P(\omega^{n-1})).$$

Since P admits coefficients in $\mathbb{F}_q$, we have

$$P(\phi_q^k(\omega^j)) \;=\; \phi_q^k(P(\omega^j))$$

for all $j, k \in \mathbb{N}$. For any $\omega^j \in \langle \omega \rangle$, this means that we may retrieve $P(\omega^j)$ from $P(\pi_{\mathcal{S}}(\omega^j))$. Indeed, setting $\omega^i = \pi_{\mathcal{S}}(\omega^j)$ with $\omega^i = \phi_q^k(\omega^j)$, we obtain $P(\omega^j) = \phi_q^{-k}(P(\omega^i))$. We call

$$\mathrm{DFFT}_{\omega,\mathcal{S}}(P) \;=\; (P(\omega^i))_{i \in \mathcal{I}}$$

the *discrete Frobenius Fourier transform* of P with respect to ω and the section $\mathcal{S}$.

Let $\omega^i \in \mathcal{S}$ and $r = r_q(\omega^i)$. Then ϕ_q^r leaves ω^i invariant, whence $\phi_q^r(P(\omega^i)) = P(\phi_q^r(\omega^i)) = P(\omega^i)$. Since $\mathbb{F}_{q^r}$ is the subfield of $\mathbb{F}_{q^d}$ that is fixed under the action of ϕ_q^r, this yields

$$P(\omega^i) \;\in\; \mathbb{F}_{q^{r_q(\omega^i)}}.$$

It follows that the number of coefficients in $\mathbb{F}_q$ needed to represent $\mathrm{DFFT}_{\omega,\mathcal{S}}(P)$ is given by

$$\sum_{i \in \mathcal{I}} r_q(\omega^i) \;=\; n. \tag{4.1}$$

In particular, the output and input size n of the discrete Frobenius Fourier transform coincide.

4.2 Twiddled transforms

In the Cooley–Tukey algorithm from section 3.4, the second wave of FFTs operates on "twiddled" polynomials $\tilde{P}^\flat_{i^\sharp}(x) := P^\flat_{i^\sharp}(\omega^{\overline{i^\sharp}}\, x)$ instead of the polynomials $P^\flat_{i^\sharp}(x)$. An alternative point of view is to directly consider the evaluation of $P^\flat_{i^\sharp}$ at the points $\eta\,(\omega^\flat)^{i^\flat}$ with $i^\flat < n^\flat$ and where $\eta = \omega^{\overline{i^\sharp}}$. We will call this operation a *twiddled DFT*.

The twiddled DFFT is defined in a similar manner. More precisely, assume that we are given an s-th root of unity $\eta \in \mathbb{F}_{q^d}$ such that the set $\eta\,\langle \omega \rangle$ is stable under the action of ϕ_q. Assume also that we have a cross section $\mathcal{S}$ of $\eta\,\langle \omega \rangle$ under this action and the corresponding set of indices $\mathcal{I} = \{i < n : \eta\,\omega^i \in \mathcal{S}\} \subseteq \{0, ..., n-1\}$. Then the *twiddled DFFT* computes the family

$$\mathrm{DFFT}_{\eta,\omega,\mathcal{S}}(P) \;:=\; (P(\eta\,\omega^i))_{i \in \mathcal{I}}.$$

Again, we notice that

$$\sum_{i \in \mathcal{I}} r_q(\eta\,\omega^i) \;=\; n. \tag{4.2}$$

We will denote by $\Phi_{q,d}(n)$ the complexity of computing a twiddled DFFT of this kind.

4.3 The naive strategy

If n is a small number, then we recall from section 3.1 that it is most efficient to compute ordinary DFTs of order n in a naive fashion, by evaluating $P(\omega^i)$ for $i < n$ using Horner's method. From an implementation point of view, one may do this using specialized codelets for various small n.

The same naive strategy can also be used for the computation of DFFTs and twiddled DFFTs. Given a cross section $\mathcal{S} \subseteq \eta\,\langle \omega \rangle$ and the corresponding set of indices $\mathcal{I} = \{i < n : \eta\,\omega^i \in \mathcal{S}\}$, it suffices to evaluate $P(\eta\,\omega^i)$ separately for each $i \in \mathcal{I}$. Let $r_i = r_q(\eta\,\omega^i) \mid d$ be the order of ω^i under the action of ϕ_q. Then $\eta\,\omega^i$ actually belongs to $\mathbb{F}_{q^{r_i}}$, so the evaluation of $P(\eta\,\omega^i)$ can be done in time $n\,\mathsf{M}_{q^{r_i}}(1) + O(n\,r_i \log q)$. With the customary assumption that $\mathsf{M}_{q^n}(1)/n$ is increasing as a function of n, it follows using (4.2) that the complete twiddled DFFT can be computed in time

$$\begin{aligned}\Phi_{q,d}(n) \;&\leqslant\; n \sum_{i \in \mathcal{I}} \mathsf{M}_{q^{r_i}}(1) + O(n^2 \log q)\\ &\leqslant\; \frac{n^2}{d}\,\mathsf{M}_{q^d}(1) + O(n^2 \log q).\end{aligned}$$

Assuming that full DFTs of order n over $\mathbb{F}_{q^d}$ are also computed using the naive strategy, this yields

$$\Phi_{q,d}(n) \;\leqslant\; \tfrac{1}{d}\,\mathsf{F}_{q^d}(n)\,(1 + o(1)). \tag{4.3}$$

In other words, for small n, we have achieved our goal to gain a factor d.

4.4 Full Frobenius action

The next interesting case for study is when n is prime and the action of ϕ_q on $\eta\,\langle \omega \rangle$ is either transitive, or has only two orbits and one of them is trivial. If $\eta \in \langle \omega \rangle$, then $\omega^0 = 1$ always forms an orbit of its own in $\eta\,\langle \omega \rangle$, but we can have $|\mathcal{S}| = 2$. If $\eta \notin \langle \omega \rangle$, then it can happen that $\mathcal{S}$ is a singleton. This happens for instance for a 9-th primitive root of unity $\eta \in \mathbb{F}_{2^6}$, for $\omega = \eta^3 \in \mathbb{F}_4$, and $n = 3$.

If $\mathcal{S}$ is a singleton, say $\mathcal{S} = \{\eta\}$, then we necessarily have $r_q(\eta) = n = d$ and the DFFT reduces to a single evaluation $P(\eta)$ with $P \in \mathbb{F}_q[x]_{<n}$ and $\eta \in \mathbb{F}_{q^n}$. This is precisely the problem of modular composition that we encountered in section 2, whence $\Phi_{q,d}(n) = \mathsf{C}_q(n)$. The speed-up $\mathsf{F}_{q^n}(n)/\mathsf{C}_q(n)$ with respect to a full DFT is comprised between 1 and n, depending on the efficiency of our algorithm for modular composition. Theoretically speaking, we recall that $\mathsf{C}_q(n) = (n \log q)^{1+o(1)}$, which allows us to gain a factor $n/(n \log q)^\epsilon$ for any $\epsilon > 0$.

In a similar way, if $\eta \in \langle\omega\rangle$ and $|\mathcal{S}| = 2$, then the computation of one DFFT reduces to n additions (in order to compute $P(1)$) and one composition modulo a polynomial of degree $n-1$ (instead of n).

Remark 4.1. Assume that n is prime, $\eta \in \langle\omega\rangle$ and $|\mathcal{S}| = 2$. If we are free to choose the representation of elements in $\mathbb{F}_{q^d}$, then the DFFT becomes particularly efficient if we take $\mathbb{F}_{q^d} = \mathbb{F}_q[\omega]$. In other words, if $\mu \in \mathbb{F}_q[x]$ is the monic minimal polynomial of ω over $\mathbb{F}_q$ (so that $\deg \mu = d = n-1$), then we represent elements of $\mathbb{F}_{q^d}$ as polynomials in $\mathbb{F}_q[x]$ modulo μ. Given $P \in \mathbb{F}_q[x]_{<n}$, just writing down $P(\omega) = (P - P_d\ \mu)(\omega)$ then yields the evaluation of P at ω, whence $\Phi_{q,d}(n) = \mathsf{M}_q(1,n) + O(n \log q)$.

4.5 Rader reduction

For large prime orders n and $\eta \in \langle\omega\rangle$, let us now show how to adapt Rader reduction to the computation of DFFTs in favourable cases. With the notations from section 3.2, we have

$$(\omega^{g^\ell \operatorname{rem} n})^q \;=\; \omega^{(g^\ell q) \operatorname{rem} n} \;=\; \omega^{g^{\ell+\chi} \operatorname{rem} n}$$

for all ℓ, where $\chi < n-1$ is such that $g^\chi = q$. The action of ϕ_q therefore induces an action $\psi_q : \ell \mapsto \ell + \chi$ on $\mathbb{Z}/((n-1)\,\mathbb{Z})$.

Let us assume for simplicity that the order κ of the additive subgroup generated by χ in $\mathbb{Z}/((n-1)\,\mathbb{Z})$ is comprime with $\lambda := (n-1)/\kappa$. Then there exists an element $\tau < n-1$ of order λ in $\mathbb{Z}/((n-1)\,\mathbb{Z})$ and $\mathbb{Z}/((n-1)\,\mathbb{Z}) \cong \langle\chi\rangle \times \langle\tau\rangle$. Consequently,

$$\mathbb{F}_{q^\kappa}[y]^\circ_{n-1} \;=\; \mathbb{F}_{q^\kappa}[y^\tau]^\circ_\lambda[y^\chi]^\circ_\kappa.$$

We may thus regard the product $W = U\,V$ as a bivariate polynomial in y^τ and y^χ. Let $\tilde{W} \in \mathbb{F}_q[y^\tau]^\circ_\lambda$ be the constant term in y^χ of this bivariate polynomial. From $\tilde{W}$, we can recover $P(\omega^{g^\ell \operatorname{rem} n})$ for every $\ell \in \langle\tau\rangle$. By construction, the $\omega^{g^\ell \operatorname{rem} n}$ with $\ell \in \langle\tau\rangle$ actually form a section of $\langle\omega\rangle \setminus \{1\}$ under the action of ϕ_q. This reduces the computation of a DFFT for the section $\mathcal{S} := \{\omega^{g^\ell \operatorname{rem} n} : l \in \langle\tau\rangle\} \cup \{1\}$ to the computation of $\tilde{W}$.

As to the computation of $\tilde{W}$, assume for simplicity that $\mathbb{F}_q$ contains a primitive $(n-1)$-th root of unity. Recall that U has coefficients in $\mathbb{F}_q$ whereas V admits coefficients in $\mathbb{F}_{q^\kappa}$. Notice also that V does not depend on the input P, whence it can be precomputed. We compute $\tilde{W}$ as follows in the FFT-model with respect to y^τ. We first transform U and V into two vectors $\hat{U} \in (\mathbb{F}_q[y^\chi]^\circ_\kappa)^\lambda$ and $\hat{V} \in (\mathbb{F}_{q^\kappa}[y^\chi]^\circ_\kappa)^\lambda$. The computation of $\hat{V}$ may again be regarded as a precomputation. The computation of the Fourier transform $\hat{\tilde{W}} \in \mathbb{F}_{q^\kappa}^\lambda$ now comes down to λ modular compositions of degree κ over $\mathbb{F}_q$. We finally transform the result back into $\tilde{W}$. Altogether, this yields

$$\Phi_{q,\kappa}(n) \;\leqslant\; 2\,\kappa\,\mathsf{F}_q(\lambda) + \lambda\,\mathsf{C}_q(\kappa) + O(n \log q).$$

It is instructive to compare this with the cost of a full DFT of length n over $\mathbb{F}_{q^{n-1}}$ using Rader's algorithm:

$$\mathsf{F}_{q^{n-1}}(n) \;\leqslant\; 2\,\kappa\,\mathsf{F}_{q^{n-1}}(\lambda) + 2\,\lambda\,\mathsf{F}_{q^{n-1}}(\kappa) + O(n^2 \log q).$$

Depending on the efficiency of modular composition, the speed-up for DFFTs therefore varies between $n\,/\,\kappa$ and n. Notice that the algorithm from the previous subsection becomes a special case with parameters $\kappa = n-1$ and $\lambda = 1$.

5. MULTIPLE FOURIER TRANSFORMS

5.1 Parallel lifting

Let $\mathbb{F}_{q^d}$ be an extension of $\mathbb{F}_q$ and assume that its elements are represented as polynomials in a primitive element $\alpha \in \mathbb{F}_{q^d}$ of degree d. Given d polynomials $P_0, \ldots, P_{d-1} \in \mathbb{F}_q[x]_{<n}$, we may then form the polynomial

$$P \;=\; P_0 + P_1\,\alpha + \cdots + P_{d-1}\,\alpha^{d-1} \;\in\; \mathbb{F}_{q^d}[x].$$

If $n \mid q-1$ and ω is a primitive n-th root of unity in the base field $\mathbb{F}_q$, then we notice that

$$\operatorname{DFT}_\omega(P) \;=\; \operatorname{DFT}_\omega(P_0) + \cdots + \operatorname{DFT}_\omega(P_{d-1})\,\alpha^{d-1}. \qquad (5.1)$$

In other words, the discrete Fourier transform operates coefficientwise with respect to the basis $1, \alpha, \ldots, \alpha^{d-1}$ of $\mathbb{F}_{q^d}$.

Recall that $\mathsf{F}_q(n,\,t)$ stands for the cost of computing t Fourier transforms of length n over $\mathbb{F}_q$. The maps $(P_0, \ldots, P_{d-1}) \mapsto P_0 + \cdots + P_{d-1}\,\alpha^{d-1}$ and its inverse boil down to matrix transpositions in memory. On a Turing machine, they can therefore be computed in time $O(n\,d \log d \log q)$. From (5.1), it follows that

$$\mathsf{F}_q(n,d) \;\leqslant\; \mathsf{F}_{q^d}(n) + O(n\,d \log d \log q). \qquad (5.2)$$

This relation does not give us anything really new, since we made the assumption in section 2 that $\mathsf{M}_q(n,t)\,/\,(n \log q)$ is increasing in both n and q. Nevertheless, it provides us with an elegant and explicit algorithm in a special case.

5.2 Symmetric multiplexing

Let us next consider an s-th root of unity η and a primitive n-th root of unity $\omega \in \mathbb{F}_{q^d}$. Given $u = \eta\,\omega^i$, we have for all k,

$$\phi_q^k(P(\phi_q^{-k}(u))) \;=\; P_0(u)\,\phi_q^k(1) + \cdots + P_{d-1}(u)\,\phi_q^k(\alpha^{d-1}).$$

Abbreviating $\Phi_{P,k}(u) := \phi_q^k(P(\phi_q^{-k}(u)))$ and setting

$$B = \begin{pmatrix} 1 & \cdots & \alpha^{d-1} \\ \vdots & & \vdots \\ \phi_q^{d-1}(1) & \cdots & \phi_q^{d-1}(\alpha^{d-1}) \end{pmatrix}$$

$$\Phi_P(u) = \begin{pmatrix} \Phi_{P,0}(u) \\ \vdots \\ \Phi_{P,d-1}(u) \end{pmatrix} \qquad V_P(u) = \begin{pmatrix} P_0(u) \\ \vdots \\ P_{d-1}(u) \end{pmatrix}$$

it follows that

$$\Phi_P(u) \;=\; B\,V_P(u).$$

Now given the twiddled discrete Fourier transform $\operatorname{DFT}_{\eta,\omega}(P) := (P(\eta\,\omega^i))_{i<n}$ of P, we in particular know the values $P(u), \ldots, P(\phi_q^{d-1}(u))$. Letting the Frobenius automorphism ϕ_q act on these values, we obtain the vector $\Phi_P(u)$. Using one matrix-vector product $V_P(u) = B^{-1}\,\Phi_P(u)$, we may then retrieve the values of the individual twiddled transforms $\operatorname{DFT}_{\eta,\omega}(P_i)$ at u. Doing this for each $u \in \mathcal{S}$ in a cross section of $\eta\ \langle\omega\rangle$ under the action of ϕ_q, this yields a way to retrieve the twiddled DFFTs of $P_0, \ldots, P_{d-1}$ from the twiddled DFT of P.

Recall that $\Phi_{q,d}(n,\ t)$ denotes the complexity of computing the DFFTs of t elements of $\mathbb{F}_q[x]_{<n}$ over $\mathbb{F}_{q^d}$. Since B is a Vandermonde matrix, the matrix-vector product $B^{-1}\,\Phi_P(u)$ can be computed in time $O(\mathsf{M}_q(d)\log d)$ using polynomial interpolation [8, Chapter 10]. Given $\mathrm{DFT}_{\eta,\omega}(P)$, we may compute each individual vector $\Phi_P(u)$ in time $O(d\,\mathsf{M}_q(d)\log(d\,q))$ using the Frobenius automorphism ϕ_q. Since there are $|\mathcal{S}|$ orbits in $\langle\omega\rangle$ under the action of ϕ_q, we may thus retrieve the $\mathrm{DFFT}_{\eta,\omega}(P_i)$ from $\mathrm{DFT}_{\eta,\omega}(P)$ in time $O(|\mathcal{S}|\,d\,\mathsf{M}_q(d)\log q^d)$. In other words,

$$\Phi_{q,d}(n,d)\ \leqslant\ \mathsf{F}_{q^d}(n)+O(|\mathcal{S}|\,d\,\mathsf{M}_q(d)\log q^d). \tag{5.3}$$

5.3 Multiplexing over an intermediate field

In the extreme case when $\eta,\omega\in\mathbb{F}_q$, every $\eta\,\omega^i\in\eta\,\langle\omega\rangle$ has order one under the action of ϕ_q and $|\mathcal{S}|=n$. In that case, the bound (5.3) is not very sharp, but (5.2) already provides us with a good alternative bound for this special case. For $r\notin\{1,d\}$ with $r\mid d$, let us now consider the case when $u=\eta\,\omega^i$ has order at most r under the action of ϕ_q for all i, so that $u\in\mathbb{F}_{q^r}$. Let $\beta\in\mathbb{F}_{q^r}$ be a primitive element in $\mathbb{F}_{q^r}$ over $\mathbb{F}_q$, so that $\mathbb{F}_{q^r}=\mathbb{F}_q[\beta]$ and $\mathbb{F}_{q^d}=\mathbb{F}_{q^r}[\alpha]$. Given our d polynomials $P_0,\ldots,P_{d-1}\in\mathbb{F}_q[x]_{<n}$, we may form

$$\begin{aligned}P_{[i]}\ &=\ P_{ir}+\cdots+P_{ir+r-1}\,\beta^{r-1}\ \in\ \mathbb{F}_{q^r}[x]_{<n}\qquad(i<d/r)\\ P\ &=\ P_{[0]}+P_{[1]}\,\alpha+\cdots+P_{[d/r-1]}\,\alpha^{d/r-1}\ \in\ \mathbb{F}_{q^d}[x]_{<n}.\end{aligned}$$

Then we have

$$\begin{aligned}&\mathrm{DFT}_{\eta,\omega}(P)\\ =\ &\mathrm{DFT}_{\eta,\omega}(P_{[0]})+\cdots+\mathrm{DFT}_{\eta,\omega}(P_{[d/r-1]})\,\alpha^{d/r-1}.\end{aligned}$$

Moreover, we may compute $(\mathrm{DFFT}_{\eta,\omega}(P_{ir+j}))_{j<r}$ from $\mathrm{DFT}_{\eta,\omega}(P_{[i]})$ for each $i<d/r$, using the algorithm from the previous subsection, but with the additional property that at least one $u\in\eta\,\langle\omega\rangle$ has maximal order $r=[\mathbb{F}_{q^r}\colon\mathbb{F}_q]$ under the action of ϕ_q.

If n is prime and $\eta\in\langle\omega\rangle$, then the above discussion shows that, without loss of generality, we may assume that ω has order d under the action of ϕ_q. This means that ω^i has maximal order d for every $i\in\{1,\ldots,n-1\}$, whereas $\omega^0=1$ has order one. Hence, $|\mathcal{S}|=(n-1)/d+1$. Similarly, if n is prime and $\eta\notin\langle\omega\rangle$, then we obtain a reduction to the case when $\eta\,\omega^i$ has maximal order d for all $i\in\{0,\ldots,n-1\}$, whence $|\mathcal{S}|=n/d$. In both cases, we obtain the following:

Proposition 5.1. *If n is prime, then*

$$\Phi_{q,d}(n,d)\ =\ \mathsf{F}_{q^d}(n)+O(n\,\mathsf{M}_q(d)\log q^d).$$

Remark 5.2. We recall that $n\leqslant q^d-1$, whence $\log n\leqslant\log q^d$. When using the best known bound (2.1) for $\mathsf{M}_q(n)$ and $\mathsf{F}_{q^d}(n)\asymp\mathsf{M}_{q^d}(n)$, it follows for some constant $C>0$ that

$$\begin{aligned}\frac{\Phi_{q,d}(n,d)}{\mathsf{F}_{q^d}(n)}\ &\geqslant\ 1+C\,\frac{d\log q\log(d\log q)\,8^{\log^*(d\log q)}}{\log(n\,d\log q)\,8^{\log^*(n\,d\log q)}}\\ &\geqslant\ 1+8\,C\log q\log(d\log q).\end{aligned}$$

In other words, we cannot hope to gain more than an asymptotic factor of $d/\log d$ with respect to a full DFT.

If n is not necessarily prime, then the technique from this section can still be used for every individual u. However, the rewritings of elements in $\mathbb{F}_{q^d}$ as elements in $\mathbb{F}_{q^r}[\alpha]$ and $\mathbb{F}_q[\beta][\alpha]$ have to be done individually for each u using modular compositions. Denoting by r_i the order of $\eta\,\omega^i$ under ϕ_q, it follows that

$$\begin{aligned}&\Phi_{q,d}(n,d)\\ =\ &\mathsf{F}_{q^d}(n)+\sum_{i\in\mathcal{I}}\mathsf{V}_q(d,r_i)+\sum_{i\in\mathcal{I}}O(r_i\,\mathsf{M}_{q^{r_i}}(d/r_i)\log q^d)\\ \leqslant\ &\mathsf{F}_{q^d}(n)+n\,\mathsf{V}_q(d)+O(n\,\mathsf{M}_q(d)\log q^d).\end{aligned}$$

Notice that conversions of the same type correspond to modular compositions with the same modulus. If n is smooth, then it follows that we may use the algorithms from [17] and keep the cost of the conversions quasi-linear.

6. THE FROBENIUS FFT

Let $n=m_1\cdots m_\ell$ and $v=(m_1,\ldots,m_\ell)$ be as in section 3.3 and let $\omega\in\mathbb{F}_{q^d}$ be a primitive n-th root of unity. In this section we present an analogue of the FFT from section 3.4: the *Frobenius FFT* (or FFFT). This algorithm uses the same mirrored indexation as the Cooley–Tukey FFT: given $P\in\mathbb{F}_q[x]_{<n}$, it thus returns the family $(P(\omega^{\bar\imath}))_{i\in\mathcal{I}}$. We start with the description of the index set $\mathcal{I}=\{i<n\colon\omega^{\bar\imath}\in\mathcal{S}\}$ that corresponds to the so-called "privileged cross section" $\mathcal{S}$ of $\langle\omega\rangle$.

6.1 Privileged cross sections

Given $i<n$, consider the orbit $\mathcal{O}_q(\omega^{\bar\imath}):=\langle\phi_q\rangle(\omega^{\bar\imath})$ of $\omega^{\bar\imath}$ under the action of $\langle\phi_q\rangle$. We define

$$\begin{aligned}\pi_{q,v}(\omega^{\bar\imath})\ &:=\ \omega^{\bar\jmath}\\ j\ &:=\ \min\{0\leqslant j<n\colon\omega^{\bar\jmath}\in\mathcal{O}_q(\omega^{\bar\imath})\}.\end{aligned}$$

Then $\mathcal{S}:=\operatorname{im}\pi_{q,v}$ is a cross section of $\langle\omega\rangle$ under the action of ϕ_q; we call it the *privileged cross section* for ϕ_q (and v).

Now let $h\leqslant d$, $v^\sharp=(m_0,\ldots,m_{h-1})$, $v^\flat=(m_h,\ldots,m_{\ell-1})$, $n^\sharp=m_0\cdots m_{h-1}$ and $n^\flat=m_h\cdots m_{\ell-1}$ be as above. Then $\omega^\sharp:=\omega^{n^\flat}$ is a primitive $n^\sharp$-th root of unity. Let $\mathcal{S}^\sharp:=\operatorname{im}\pi_{q,v^\sharp}$ be the corresponding privileged cross section of $\langle\omega^\sharp\rangle$.

Proposition 6.1. *We have*

$$\mathcal{S}^\sharp\ =\ \mathcal{S}^{n^\flat}\ :=\ \{u^{n^\flat}\colon u\in\mathcal{S}\}.$$

Proof. Let $i=i^\sharp\,n^\flat+i^\flat$ and $j=j^\sharp\,n^\flat+j^\flat$ with $i^\sharp,j^\sharp<n^\sharp$ and $i^\flat,j^\flat<n^\flat$ be such that $\omega^{\bar\jmath}=\phi_q^k(\omega^{\bar\imath})$ for some k. Since

$$(\omega^{\bar\imath})^{n^\flat}=(\omega^{\overline{i^\flat}n^\sharp+\overline{i^\sharp}})^{n^\flat}=(\omega^{\overline{i^\sharp}})^{n^\flat}=(\omega^\sharp)^{\overline{i^\sharp}},$$

and similarly $(\omega^{\bar\jmath})^{n^\flat}=(\omega^\sharp)^{\overline{j^\sharp}}$, it follows that $(\omega^\sharp)^{\overline{j^\sharp}}=\phi_q^k((\omega^\sharp)^{\overline{i^\sharp}})$.

Inversely, given $j^\sharp<n^\sharp$ with $(\omega^\sharp)^{\overline{j^\sharp}}=\phi_q^k((\omega^\sharp)^{\overline{i^\sharp}})$ for some k, we claim that there exists a $j^\flat<n^\flat$ such that $\omega^{\bar\jmath}=\phi_q^k(\omega^{\bar\imath})$ for $j=j^\sharp\ n^\flat+j^\flat$. Indeed, $(\omega^\sharp)^{\overline{j^\sharp}}=\phi_q^k((\omega^\sharp)^{\overline{i^\sharp}})$ implies that $(\omega^{\overline{j^\sharp}})^{n^\flat}=(\phi_q^k(\omega^{\bar\imath}))^{n^\flat}$, whence $\omega^{\overline{j^\sharp}}/\,\phi_q^k(\omega^{\bar\imath})$ is an $n^\flat$-th root of unity. This means that there exists a $j^\flat<n^\flat$ with $\omega^{\overline{j^\sharp}}/\,\phi_q^k(\omega^{\bar\imath})=(\omega^{n^\sharp})^{-\overline{j^\flat}}$, i.e. $\omega^{\bar\jmath}=\omega^{\overline{j^\sharp}+\overline{j^\flat}n^\sharp}=\phi_q^k(\omega^{\bar\imath})$.

Now assume that j is minimal with $\omega^{\bar\jmath}=\phi_q^k(\omega^{\bar\imath})$ for some k. Given $\xi^\sharp<n^\sharp$ such that $(\omega^\sharp)^{\overline{\xi^\sharp}}\in\mathcal{O}_q((\omega^\sharp)^{\overline{i^\sharp}})$, the above claim shows that there exists a $\xi^\flat<n^\flat$ such that $\omega^{\bar\xi}\in\mathcal{O}_q(\omega^{\bar\imath})$ for $\xi=\xi^\sharp\ n^\flat+\xi^\flat$. But then $j\leqslant\xi$, whence $j^\sharp\leqslant\xi^\sharp$. This shows that $\omega^{\bar\jmath}\in\mathcal{S}$ implies $(\omega^{\bar\jmath})^{n^\flat}=(\omega^\sharp)^{\overline{j^\sharp}}\in\mathcal{S}^\sharp$.

Inversely, assume that $j^\sharp < n^\sharp$ is minimal such that $(\omega^\sharp)^{\overline{j^\sharp}} \in \mathcal{O}_q((\omega^\sharp)^{\overline{i^\sharp}})$. Then the above claim implies that there exists a $j^\flat < n^\flat$ such that $\omega^{\overline{j}} \in \mathcal{O}_q(\omega^{\overline{i}})$ for $j = j^\sharp n^\flat + j^\flat$. Without loss of generality we may assume that $j^\flat$ was chosen minimal while satisfying this property. Given $\xi^\sharp < n^\sharp$, $\xi^\flat < n^\flat$ and $\xi = \xi^\sharp\, n^\flat + \xi^\flat$ with $\omega^{\overline{\xi}} \in \mathcal{O}_q(\omega^{\overline{i}})$, we have $(\omega^\sharp)^{\overline{\xi^\sharp}} \in \mathcal{O}_q((\omega^\sharp)^{\overline{i^\sharp}})$. Consequently, $j^\sharp \leqslant \xi^\sharp$ and $j^\flat \leqslant \xi^\flat$ whenever $j^\sharp = \xi^\sharp$. In other words, $j \leqslant \xi$. This shows that $(\omega^\sharp)^{\overline{j^\sharp}} \in \mathcal{S}^\sharp$ implies the existence of a $j^\flat < n^\flat$ with $\omega^{\overline{j}} \in \mathcal{S}$ and $(\omega^{\overline{j}})^{n^\flat} = (\omega^\sharp)^{\overline{j^\sharp}}$. □

6.2 The main algorithm

We are now in a position to adapt the Cooley–Tukey FFT. In the case when $\ell = 1$ or n is prime, we will use one of the algorithms from section 4 for the computation of twiddled DFFTs.

Algorithm $\mathbf{FFFT}_\omega(P)$
INPUT: $P \in \mathbb{F}_q[x]_{<n}$
OUTPUT: $(P(\omega^{\overline{i}}))_{i \in \mathcal{I}}$.

if $\ell = 1$ **then return** $\mathrm{DFFT}_{1,\omega}(P)$
Take $h = \ell - 1$, $n^\sharp = m_0 \cdots m_{h-1}$ and $n^\flat = m_h \cdots m_{\ell-1}$
Let $\omega^\sharp := \omega^{n^\flat}$, $\omega^\flat := \omega^{n^\sharp}$, $x^\sharp := x^{n^\flat}$
Let $\mathcal{I}^\sharp := \{i^\sharp < n^\sharp : (\omega^\sharp)^{\overline{i^\sharp}} \in \mathcal{S}^{n^\flat}\}$
Decompose $P(x) = P_0^\sharp(x^\sharp) + \cdots + P_{n^\flat-1}^\sharp(x^\sharp)\, x^{n^\flat - 1}$
for $i^\flat < n^\flat$ **do**
  $(P_{i^\flat}^\sharp((\omega^\sharp)^{\overline{i^\sharp}}))_{i^\sharp \in \mathcal{I}^\sharp} := \mathrm{FFFT}_{\omega^\sharp}(P_{i^\flat}^\sharp)$
for $i^\sharp \in \mathcal{I}^\sharp$ **do**
  Let $P_{i^\sharp}^\flat(x) := P_0^\sharp((\omega^\sharp)^{\overline{i^\sharp}}) + \cdots + P_{n^\flat-1}^\sharp((\omega^\sharp)^{\overline{i^\sharp}})\, x^{n^\flat-1}$
    Notice that $P_{i^\sharp}^\flat \in \mathbb{F}_{q^r}[x] := \mathbb{F}_q\big[(\omega^\sharp)^{\overline{i^\sharp}}\big][x]$,
    where r is the order of $(\omega^\sharp)^{\overline{i^\sharp}}$
  Let $\mathcal{I}_{i^\sharp}^\flat := \{i^\flat < n^\flat : \omega^{\overline{i^\sharp n^\flat + i^\flat}} \in \mathcal{S}\}$
  if $\omega^{\overline{i^\sharp}}, \omega^\flat \in \mathbb{F}_q[(\omega^\sharp)^{\overline{i^\sharp}}]$ **then**
    Let $\tilde{P}_{i^\sharp}^\flat(x) := P_{i^\sharp}^\flat(\omega^{\overline{i^\sharp}} x)$
    Compute $(P_{i^\sharp}^\flat(\omega^{\overline{i^\sharp}} (\omega^\flat)^{\overline{i^\flat}}))_{i^\flat \in \mathcal{I}_{i^\sharp}^\flat} := \mathrm{FFT}_{\omega^\flat}\big(\tilde{P}_{i^\sharp}^\sharp\big)$
  else
    Compute $\big(P_{i^\sharp}^\flat(\omega^{\overline{i^\sharp}} (\omega^\flat)^{\overline{i^\flat}})\big)_{i^\flat \in \mathcal{I}_{i^\sharp}^\flat} := \mathrm{DFFT}_{\omega^{\overline{i^\sharp}}, \omega^\flat, \mathcal{S}_{i^\sharp}^\flat}(P_{i^\sharp}^\sharp)$
return $(P(\omega^{\overline{i}}))_{i \in \mathcal{I}} = (P_{i^\sharp}^\flat(\omega^{\overline{i^\sharp}} (\omega^\flat)^{\overline{i^\flat}}))_{i^\sharp \in \mathcal{I}^\sharp, i^\flat \in \mathcal{I}_{i^\sharp}^\flat}$

For the usual FFT, it was possible to choose h in such a way that $n^\sharp \approx n^\flat$, and we recall that this improves the cache efficiency of practical implementations. However, this optimization is more problematic in our setting since it would require the development of an efficient recursive algorithm for twiddled FFFTs. This is an interesting topic, but beyond the scope of the present paper.

6.3 Complexity analysis

Let us now perform the complexity analysis of FFFT_ω. For $k \in \{0, ..., \ell - 1\}$, we first focus on all FFTs and twiddled DFFTs that are computed at "stage k" using a fallback algorithm. These FFTs and twiddled DFFTs are all of length m_k. Given $e \,|\, e' \,|\, d$, let $\nu_{k,e,e'}$ be the number of transforms of a polynomial in $\mathbb{F}_{q^e}[x]_{<m_k}$ over $\mathbb{F}_{q^{e'}}$. From (4.2), it follows that

$$\sum_{e|e'|d} m_k\, \nu_{k,e,e'}\, e \;=\; n.$$

Now a naive bound for the cost of an FFT or twiddled DFFT of a polynomial in $\mathbb{F}_{q^e}[x]_{<m_k}$ over $\mathbb{F}_{q^{e'}}$ is

$$\Phi_{q^e, e'/e}(m_k) \;\leqslant\; \mathsf{F}_{q^{e'}}(m_k) + m_k\, \mathsf{M}_{q^e}(1). \tag{6.1}$$

This means that the cost of all FFTs and twiddled DFFTs at stage k is bounded by

$$\begin{aligned}\Phi_k \;&:=\; \sum_{e|e'|d} \nu_{k,e,e'}\, \Phi_{q^e, e'/e}(m_k)\\ &\leqslant\; \sum_{e|e'|d} \nu_{k,e,e'}\, \mathsf{F}_{q^{e'}}(m_k) + n\,\frac{\mathsf{M}_{q^d}(1)}{d}.\end{aligned}$$

Now $\nu_{k,e,e'}$ can only be non zero if $e' \leqslant m_k\, e$. Consequently,

$$\begin{aligned}\Phi_k \;&\leqslant\; \sum_{e|e'|d} \nu_{k,e,e'} \cdot e \cdot \frac{e'}{e} \cdot \frac{\mathsf{F}_{q^{e'}}(m_k)}{e'} + n\,\frac{\mathsf{M}_{q^d}(1)}{d}\\ &\leqslant\; \sum_{e|e'|d} \nu_{k,e,e'}\, e\, m_k\, \frac{\mathsf{F}_{q^d}(m_k)}{d} + n\,\frac{\mathsf{M}_{q^d}(1)}{d}\\ &\leqslant\; \frac{n}{d}\,(\mathsf{F}_{q^d}(m_k) + \mathsf{M}_{q^d}(1)).\end{aligned}$$

The total cost of all FFTs and twiddled DFFTs of prime length is therefore bounded by

$$\Phi_0 + \cdots + \Phi_{l-1} \;\leqslant\; \frac{n\,\ell}{d}\,(\mathsf{F}_{q^d}(\bar{m}) + \mathsf{M}_{q^d}(1)),$$

where $\bar{m} = \max(m_1, ..., m_k)$.

Strictly speaking, if e is a proper divisor of e', then the output of a twiddled DFFT of a polynomial in $\mathbb{F}_{q^e}[x]_{<m_k}$ over $\mathbb{F}_{q^{e'}}$ does not use the "standard" representation for $\mathbb{F}_{q^{e'}}$, so the result must be converted. The cost of all such conversions at stage k is bounded by

$$\begin{aligned}\mathsf{C}_k \;&:=\; \sum_{e \neq e'|d} \nu_{k,e,e'}\, \mathsf{V}_q(e', e)\\ &\leqslant\; \sum_{e \neq e'|d} m_k\, \nu_{k,e,e'}\, e\, \frac{\mathsf{V}_q(d)}{d}\end{aligned}$$

For a number a with prime factorization $a = p_1^{i_1} \cdots p_r^{i_r}$, let $\lambda(a) := i_1 + \cdots + i_r$. We also denote $N_{k,e} = \sum_{e|e'} m_k\, \nu_{k,e,e'}\, e$ and notice that $N_{k+1,e'} = \sum_{e|e'} m_k\, \nu_{k,e,e'}\, e$. Let us show by induction over k that

$$\mathsf{C}_0 + \cdots + \mathsf{C}_{k-1} \;\leqslant\; \sum_{e|d} N_{k,e}\, \lambda(e)\, \frac{\mathsf{V}_q(d)}{d}.$$

This is clear for $k = 0$, since $N_{0,1} = n$ and $N_{0,e} = 0$ for $e > 1$. Assuming that the relation holds for a given k, we get

$$\begin{aligned}\mathsf{C}_0 + \cdots + \mathsf{C}_k \;&\leqslant\; \sum_{e|e'|d} m_k\, \nu_{k,e,e'}\, e\, \lambda(e)\, \frac{\mathsf{V}_q(d)}{d} +\\ &\qquad \sum_{e \neq e'|d} m_k\, \nu_{k,e,e'}\, e\, \frac{\mathsf{V}_q(d)}{d}\\ &\leqslant\; \sum_{e|e'|d} m_k\, \nu_{k,e,e'}\, e\, \lambda(e')\, \frac{\mathsf{V}_q(d)}{d}\\ &=\; \sum_{e'|d} N_{k+1,e'}\, \lambda(e')\, \frac{\mathsf{V}_q(d)}{d}.\end{aligned}$$

This completes the induction and we conclude that the total conversion cost is bounded by

$$\mathsf{C}_0 + \cdots + \mathsf{C}_{\ell-1} \;\leqslant\; n\,\lambda(d)\,\frac{\mathsf{V}_q(d)}{d}.$$

This concludes the proof of the following result:

THEOREM 6.2. *If $m_0, ..., m_{\ell-1}$ are all prime, then*

$$\Phi_{q,d}(n) \;\leqslant\; \frac{n}{d}\left(\ell\,\mathsf{F}_{q^d}(\bar{m}) + \ell\,\mathsf{M}_{q^d}(1) + \lambda(d)\,\mathsf{V}_q(d)\right),$$

where $\bar{m} = \max(m_1, ..., m_{\ell-1})$.

Remark 6.3. If the m_k are very small, then we have shown in section 4.3 that (6.1) can be further sharpened into

$$\Phi_{q^e,e'/e}(m_k) \;\sim\; \frac{e}{e'}\,\mathsf{F}_{q^{e'}}(m_k).$$

As a consequence, the bound for $\Phi_{q,d}(n)$ becomes

$$\Phi_{q,d}(n) \;\lesssim\; \frac{n}{d}\left(\ell\,\mathsf{F}_{q^d}(1) + \lambda(d)\,\mathsf{V}_q(d)\right).$$

Remark 6.4. It is also possible to treat the case when $e = e'$ apart in the bound for Φ_k, thereby avoiding the factor $m_k \geqslant e'/e$ whenever possible. A similar analysis as for the cost of the conversions then yields

$$\begin{aligned}&\Phi_{q,d}(n)\\ &\leqslant\; \frac{n}{d}\left(\ell\,\frac{\mathsf{F}_{q^d}(\bar{m})}{\bar{m}} + \ell\,\mathsf{M}_{q^d}(1) + \lambda(d)\,\mathsf{F}_{q^d}(\bar{m}) + \lambda(d)\,\mathsf{V}_q(d)\right).\end{aligned}$$

Whenever we can manage to keep $\bar{m}$ and $\mathsf{V}_q(d)/d$ small with respect to ℓ, this means that we gain a factor d with respect to a full DFT.

6.4 Frobenius FFTs over $\mathbb{F}_{2^{60}}$

Let us now study the special case when $q = 2$ and $d = 60$. This case is useful for the multiplication of polynomials P, $Q \in \mathbb{F}_2[x]$ with $PQ \in \mathbb{F}_2[x]_{<n}$. The idea is to compute the Frobenius FFTs $(P(\omega^{\bar{\imath}}))_{i\in\mathcal{I}}$ and $(Q(\omega^{\bar{\imath}}))_{i\in\mathcal{I}}$, to perform the pointwise multiplications $((PQ)(\omega^{\bar{\imath}}))_{i\in\mathcal{I}}$ and to retrieve PQ by doing one inverse Frobenius FFT.

Modulo zero padding, we have some flexibility in the choice of n, which should divide $2^{60} - 1$. In particular, for polynomials P and Q of large degree, we may choose n to be a multiple of 61 and take $m_1 = 61$ and $h = 1$ (instead of $h = \ell - 1$) for the top call of the algorithm FFFT from section 6.2. By selecting the representation of $\mathbb{F}_{2^{60}}$ as explained in Remark 4.1, this means that $\mathbb{F}_2[(\omega^\sharp)^{\overline{i^\sharp}}] = \mathbb{F}_2$ for a single $i^\sharp = 0$ and $\mathbb{F}_2[(\omega^\sharp)^{\overline{i^\sharp}}] = \mathbb{F}_{2^{60}}$ for the remaining 60 indices $0 < i^\sharp < 61$. In other words, one Frobenius FFT of length n reduces to one Frobenius FFT of length $n/61$ and one full FFT over $\mathbb{F}_{2^{60}}$ of length $n/61$.

We choose the remaining m_2, ..., m_ℓ to be small prime numbers, so as to minimize the cost of the full FFT over $\mathbb{F}_{2^{60}}$ of length $n/61$. For the remaining FFFT of length $n/61$, we may then apply Theorem 6.2 and Remark 6.3. From a practical point of view, the computation of this remaining FFFT should take about $c/60$ times the time the computation of a full FFT of length $n/61$ for some small constant $c > 1$.

We conclude that one entire FFFT of length n can be performed approximately $61 - c$ times faster as a full FFT of length n. We hope that this will lead to practical implementations for polynomial multiplication in $\mathbb{F}_2[x]$ that are approximately twice as efficient as the implementation from [12].

7. REFERENCES

[1] L. Auslander, J. R. Johnson, and R. W. Johnson. An equivariant Fast Fourier Transform algorithm. Drexel University Technical Report DU-MCS-96-02, 1996.

[2] David H Bailey. Ffts in external of hierarchical memory. In *Proceedings of the 1989 ACM/IEEE conference on Supercomputing*, pages 234–242. ACM, 1989.

[3] G.D. Bergland. A fast Fourier transform algorithm for real-valued series. *Communications of the ACM*, 11(10):703–710, 1968.

[4] R. Bergmann. The fast Fourier transform and fast wavelet transform for patterns on the torus. *Applied and Computational Harmonic Analysis*, 2012. In Press.

[5] Dieter Blessenohl. On the normal basis theorem. *Note di Matematica*, 27(1):5–10, 2007.

[6] Leo I. Bluestein. A linear filtering approach to the computation of discrete Fourier transform. *IEEE Transactions on Audio and Electroacoustics*, 18(4):451–455, 1970.

[7] J.W. Cooley and J.W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Math. Computat.*, 19:297–301, 1965.

[8] J. von zur Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, New York, NY, USA, 3rd edition, 2013.

[9] C. F. Gauss. Nachlass: Theoria interpolationis methodo nova tractata. In *Werke*, volume 3, pages 265–330. Königliche Gesellschaft der Wissenschaften, Göttingen, 1866.

[10] I. J. Good. The interaction algorithm and practical Fourier analysis. *Journal of the Royal Statistical Society*, Series B. 20(2):361–372, 1958.

[11] A. Guessoum and R. Mersereau. Fast algorithms for the multidimensional discrete Fourier transform. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 34(4):937–943, 1986.

[12] D. Harvey, J. van der Hoeven, and G. Lecerf. Fast polynomial multiplication over $\mathbb{F}_{2^{60}}$. In *Proc. ISSAC '16*, pages 255–262, New York, NY, USA, 2016. ACM.

[13] D. Harvey, J. van der Hoeven, and G. Lecerf. Faster polynomial multiplication over finite fields. *J. ACM*, 63(6), 2017. Article 52.

[14] J. van der Hoeven. The truncated Fourier transform and applications. In J. Gutierrez, editor, *Proc. ISSAC 2004*, pages 290–296, Univ. of Cantabria, Santander, Spain, July 4–7 2004.

[15] J. van der Hoeven. Notes on the Truncated Fourier Transform. Technical Report 2005-5, Université Paris-Sud, Orsay, France, 2005.

[16] J. van der Hoeven, R. Lebreton, and É. Schost. Structured FFT and TFT: symmetric and lattice polynomials. In *Proc. ISSAC '13*, pages 355–362, Boston, USA, June 2013.

[17] J. van der Hoeven and G. Lecerf. Modular composition via factorization. Technical report, HAL, 2017. http://hal.archives-ouvertes.fr/hal-01457074.

[18] J. Johnson and X. Xu. Generating symmetric DFTs and equivariant FFT algorithms. In *ISSAC'07*, pages 195–202. ACM, 2007.

[19] K. S. Kedlaya and C. Umans. Fast polynomial factorization and modular composition. *SIAM J. Comput.*, 40(6):1767–1802, 2011.

[20] A Kudlicki, M. Rowicka, and Z. Otwinowski. The crystallographic Fast Fourier Transform. Recursive symmetry reduction. *Acta Cryst.*, A63:465–480, 2007.

[21] R. Larrieu. The truncated fourier transform for mixed radices. Technical report, HAL, 2016. http://hal.archives-ouvertes.fr/hal-01419442.

[22] Jean-Louis Nicolas and Guy Robin. Highly composite numbers by Srinivasa Ramanujan. *The Ramanujan Journal*, 1(2):119–153, 1997.

[23] C.H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[24] C.M. Rader. Discrete Fourier transforms when the number of data samples is prime. *Proc. IEEE*, 56:1107–1108, 1968.

[25] H.V. Sorensen, D.L. Jones, M.T. Heideman, and C.S. Burrus. Real-valued fast Fourier transform algorithm. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 35(6):849–863, 1987.

[26] L. F. Ten Eyck. Crystallographic Fast Fourier Transform. *Acta Cryst.*, A29:183–191, 1973.

[27] A. Vince and X. Zheng. Computing the Discrete Fourier Transform on a hexagonal lattice. *Journal of Mathematical Imaging and Vision*, 28:125–133, 2007.