Improved Upper Complexity Bounds for the Discrete Fourier Transform

Ulrich Baum, Michael Clausen, Benno Tietz
Institut für Informatik, Universität Bonn

January 18, 1990

Abstract

The linear complexity $L_s(G)$ of a finite group G is the minimal number of additions, subtractions and multiplications needed to evaluate a suitable Fourier transform of $\mathbf{C}G$. Combining and modifying several classical FFT-algorithms, we show that $L_s(G) \leq 8|G|\log_2|G|$ for any finite abelian group G.

1 Introduction

The design and analysis of efficient algorithms for Discrete Fourier Transforms (DFT) on finite non-abelian groups has been the subject of several recent investigations, see the references. The present paper re-investigates the classical FFT-algorithms for finite abelian groups. Our improved upper complexity bounds for the DFT on abelian groups automatically lead to improved bounds for the DFT on metabelian groups, see [6].

The set $\mathbf{C}G := \{\, a \mid a : G \to \mathbf{C}\}$ of all complex valued functions on the finite group G becomes a $|G|$-dimensional C-space by pointwise addition and scalar multiplication. A natural C-basis is given by the indicator functions of the group elements. Identifying each group element with its indicator function, $\mathbf{C}G$ can be viewed as the space of all formal sums $\sum_{g\in G} a_g g$ with complex coefficients. The multiplication in G can be extended to $\mathbf{C}G$:

$$\Big(\sum_{g\in G} a_g g\Big)\cdot\Big(\sum_{h\in G} b_h h\Big) = \sum_{k\in G}\Big(\sum_{gh=k} a_g b_h\Big)k\ .$$

Thus $\mathbf{C}G$ becomes a C-algebra, the so-called *group algebra* of G over C. For instance the group algebra of the cyclic group C_n of order n can be identified with the algebra $\mathbf{C}[X]/(X^n-1)$ of polynomials of degree $< n$ with multiplication modulo X^n-1. By Chinese Remaindering we know that $\mathbf{C}[X]/(X^n-1)$

is isomorphic to the algebra of n-square diagonal matrices. This isomorphism, known as the (cyclic) *Discrete Fourier Transform* (DFT_n), can be viewed as a structural transition from the *signal domain* $\mathbf{C}C_n$ into the *spectral domain* $\mathbf{C}^n$; in particular, the DFT_n is multiplicative and thus links the convolution in $\mathbf{C}[X]/(X^n-1)$ and the multiplication of n-square diagonal matrices. More generally, if G is abelian of order n, then $\mathbf{C}G$ is isomorphic to $\mathbf{C}^n$ as well.

Every algebra isomorphism $W: \mathbf{C}G \to \mathbf{C}^n$ is called a *Fourier transform* for $\mathbf{C}G$. With respect to natural bases, W can be viewed as a $|G|$-square complex matrix. E.g., if $G = C_n$ is the cyclic group of order n then $W = (\omega^{ab})_{0\le a,b<n}$ with $\omega = \exp(2\pi i/n)$. It is well known that for an abelian group G of order n there are exactly n distinct group morphisms $G \to \mathbf{C}^*$, the so-called (linear) characters $\chi_1, \ldots, \chi_n$. If $G = \{g_1, \ldots, g_n\}$, then $W = (\chi_i(g_j))_{1\le i,j\le n}$ is a Fourier transform for $\mathbf{C}G$. Up to permutations of rows and columns, W is uniquely determined by G.

The linear complexity $L_s(A)$ of a matrix $A \in \mathbf{C}^{n\times n}$ is the minimal number of $\mathbf{C}$-operations (= additions/subtractions/scalar multiplications) sufficient to compute Ax from a (generic) input vector $x \in \mathbf{C}^n$. If P and Q are n-square permutation matrices, it is easy to see that $L_s(A) = L_s(PAQ)$. Hence for an abelian group G, $L_s(G) := L_s(W)$, where W is a Fourier transform W for $\mathbf{C}G$, is well-defined. We call $L_s(G)$ the *linear complexity of* G.

For example, $L_s(C_2) = 2$, $L_s(C_3) \le 8$ and $L_s(C_5) \le 22$: $L_s(C_2) = 2$ is trivial. If ω is a primitive third root of unity, the computation scheme

$$\begin{array}{lllllll} A_0 &=& a_0 &+& a_1 &+& a_2 \\ A_1 &=& a_0 &+& (a_1-a_2)\omega &-& a_2 \\ A_2 &=& a_0 &-& a_1 &-& (a_1-a_2)\omega \end{array}$$

for the DFT_3 yields $L_s(C_3) \le 8$. Winograd [20] gives an algorithm for the computation of DFT_5 using 22 linear operations.

In [5], Büchi implicitly shows that $L_s(G) \le 38|G|\log|G|$ for every finite abelian group G. (Throughout this paper, $\log = \log_2$.) The constant 38 has been improved to 15 in [6]. Combining and modifying several classical FFT-algorithms, we will show in the present paper that

$$L_s(G) \le 8|G|\log|G|$$

for any finite abelian group G. However, for special classes of finite groups, this last result can be improved, see section 2.

2 Classical FFT algorithms revisited

In this section we are going to describe several basic algorithms for the efficient computation of Fourier transforms for abelian groups.

2.1 Good-Thomas

If $K = G \times H$ is a direct product of finite groups, we can construct a Fourier transform for K from several Fourier transforms for the factors G and H. This construction is an important tool in the proof of our result, as it leads to upper bounds for the linear complexity of direct products of finite abelian groups. In order to prove those upper bounds, we first cite without proof a fact from linear complexity theory.

Lemma 1 *For the Kronecker product $A \otimes B$ of $A \in \mathbf{C}^{a \times a}$ and $B \in \mathbf{C}^{b \times b}$, we have*

$$L_s(A \otimes B) \leq b \cdot L_s(A) + a \cdot L_s(B) \ .$$

The proof can be found in [6], for instance. We can now state the following important

Lemma 2 *If G and H are finite abelian groups, then*

$$L_s(G \times H) \leq |G| L_s(H) + |H| L_s(G) \ .$$

In particular, if $L_s(H) \leq c|H| \log |H|$ and $L_s(G) \leq c|G| \log |G|$ for some $c > 0$, then $L_s(G \times H) \leq c|G \times H| \log |G \times H|$.

PROOF. If W_G and W_H are Fourier transforms for G and H, respectively, then the Kronecker product $W_G \otimes W_H$ is a Fourier transform for $G \times H$, see [11, p. 516]. By the previous lemma, we have:

$$L_s(G \times H) = L_s(W_G \otimes W_H) \leq |G| L_s(W_H) + |H| L_s(W_G) = |G| L_s(H) + |H| L_s(G)$$

The second claim follows by an easy calculation.

Algorithmically, Lemma 2 means that $W_{G \times H}$ can be computed by $|G|$ evaluations of W_H followed by an intermediate permutation (which is free of cost in our model) and $|H|$ evaluations of W_G. This algorithm is known as the Good-Thomas-FFT.

For elementary abelian 2-groups $G = C_2^n$, repeated application of Lemma 2 yields the Fast Walsh-Hadamard transform with $L_s(C_2^n) \leq n2^n$.

Every finite abelian group is isomorphic to a direct product of cyclic groups of prime power order. Thus Lemma 2 allows us to restrict our analysis of the linear complexity of finite abelian groups to cyclic groups of prime power order. From now on, let $L(n) := L_s(C_n)$ denote the linear complexity of the cyclic group of order n, i.e. the linear complexity of the matrix $DFT_n = (\omega^{ab})_{0 \leq a,b < n}$.

2.2 Cooley-Tukey

The method of Cooley and Tukey [8] reduces the computation of DFT_n, n composite, to several computations of smaller DFTs. For $n = pq$, let ω be a primitive n-th root of unity. For a given input vector $(a_0, \ldots, a_{n-1})$ we have to compute the spectral coefficients

$$A_\mu = \sum_{\nu=0}^{n-1} a_\nu \omega^{\mu\nu}, \quad 0 \le \mu \le n-1.$$

As $n = pq$, the indices μ and ν have a unique representation $\mu = kq + l$ and $\nu = ip + j$ where $0 \le k, j < p$ and $0 \le l, i < q$. Rewriting the above equation, we obtain

$$\begin{aligned} A_{kq+l} &= \sum_{i<q} \sum_{j<p} a_{ip+j} \omega^{(ip+j)(kq+l)} \\ &= \sum_{j<p} \Big(\sum_{i<q} a_{ip+j} (\omega^p)^{il} \Big) \omega^{jl} (\omega^q)^{jk} . \end{aligned}$$

In order to evaluate $A_0, \ldots, A_{n-1}$ we first compute, for each $0 \le j < p$, the coefficients $b_{j,l} := \sum_{i<q} a_{ip+j} (\omega^p)^{il}$ $(0 \le j < p, 0 \le l < q)$ by a DFT_q. Then we multiply the $b_{j,l}$ by the corresponding 'twiddle factors' ω^{jl} and finally obtain for each $0 \le l < q$ the spectral coefficients A_{kq+l} $(0 \le k < p)$ by a DFT_p.

Altogether we have to perform p evaluations of DFT_q, q evaluations of DFT_p and $n - p - q + 1$ multiplications by the non-trivial twiddle factors. Thus

$$L(pq) \le p \cdot L(q) + q \cdot L(p) + pq - p - q + 1.$$

By induction, it follows that, for $m \in \mathbb{N}$,

$$L(p^m) \le mp^{m-1} \cdot L(p) + (m-1)p^m - mp^{m-1} + 1.$$

For $p = 2$, this yields

$$L(2^m) \le \frac{3}{2} m 2^m - 2^m + 1 .$$

2.3 Rader

If p is prime, Rader [16] suggested computing the DFT_p via a cyclic convolution of length $p-1$, which in turn can be computed using the DFT_{p-1}. As $\mathbb{Z}_p$ is a field, the multiplicative group $\mathbb{Z}_p^* = \mathbb{Z}_p \setminus \{0\}$ is cyclic of order $p-1$. Let g be a generator of $\mathbb{Z}_p^*$. Then the mapping $a \mapsto g^a \pmod p$ is an isomorphism of $(\mathbb{Z}_{p-1}, +)$ onto $(\mathbb{Z}_p^*, \cdot)$.

In order to compute $A_\mu = \sum_{\nu=0}^{p-1} a_\nu \omega^{\mu\nu}$, $0 \le \mu < p$, we first compute $A_0 = \sum_{\nu=0}^{p-1} a_\nu$, which takes at most $p-1$ operations. Next we consider

$$A'_\mu = A_\mu - a_0 = \sum_{\nu=1}^{p-1} a_\nu \omega^{\mu\nu}, \qquad 1 \le \mu < p$$

and change the order of summation using the isomorphism described above:

$$\begin{aligned} \nu &\mapsto g^\nu \\ \mu &\mapsto g^{-\mu} \end{aligned}$$

We obtain

$$A'_{g^{-\mu}} = \sum_{\nu=0}^{p-2} a_{g^\nu} \omega^{g^{\nu-\mu}}, \qquad 0 \le \mu \le p-2.$$

Defining $\tilde{A}_\mu := A'_{g^{-\mu}}$, $\tilde{a}_\nu := a_{g^\nu}$ and $\tilde{\omega}_\lambda := \omega^{g^{-\lambda}}$, we get

$$\tilde{A}_\mu = \sum_{\nu=0}^{p-2} \tilde{a}_\nu \tilde{\omega}_{\mu-\nu}$$

by a cyclic convolution of length $p-1$. We compute this convolution using the DFT_{p-1}:

$$(\tilde{A}_\mu) = DFT_{p-1}^{-1}\left(DFT_{p-1}((\tilde{a}_\nu)) \cdot DFT_{p-1}((\tilde{\omega}_\lambda))\right)$$

As $(\tilde{\omega}_\lambda)$ is input-independent, we have to perform a DFT_{p-1} as well as $p-1$ multiplications and a DFT_{p-1}^{-1}. But DFT_{p-1}^{-1} is (up to an output permutation) equivalent to $1/(p-1) \cdot DFT_{p-1}$, and we can combine the factor $1/(p-1)$ and the multiplications with $DFT_{p-1}(\tilde{\omega}_\lambda)$. To compute the A_μ from $\tilde{A}_\mu$, we need $p-1$ additional operations. Altogether, we have

$$L(p) \le 2 \cdot L(p-1) + 3p - 3\ .$$

If p is relatively small, the cyclic convolution of length $p-1$ can be more efficiently computed using interpolation at a suitable set of points, see Winograd [19].

2.4 Bluestein

For arbitrary n, Bluestein's Algorithm [4] computes the DFT_n using a cyclic convolution of a suitable length $N \ge 2n$, which again can be computed using a fast algorithm for DFT_N.

We rewrite the formula

$$A_\mu = \sum_{0 \le \nu < n} a_\nu \omega^{\mu\nu} \qquad (0 \le \mu < n)$$

using $2\mu\nu = \mu^2 + \nu^2 - (\mu - \nu)^2$ to

$$A_\mu = \omega^{\mu^2/2} \sum_{\nu<n} a_\nu \omega^{\nu^2/2} \omega^{-(\mu-\nu)^2/2}$$

and substitute $u_j := a_j \omega^{j^2/2}$ and $v_k := \omega^{-k^2/2}$. We obtain

$$\omega^{-\mu^2/2} A_\mu = \sum_{0\le\nu<n} u_\nu v_{\mu-\nu} \ .$$

Thus we can compute the DFT_n via a cyclic convolution of length n. Quite surprisingly, it turns out that it can be more efficient to simulate this convolution by a cyclic convolution of increased length. For any length $N \ge 2n$ (the choice of N will be discussed later), the simulation works as follows, modifying Bluestein's original approach: Let $C_N = \langle X | X^N = 1 \rangle$ be the cyclic group of order N. Consider the convolution $U * V$ of the two elements $U := \sum_{0\le i<n} u_i X^i$ and $V := \sum_{0\le j<n} v_j (X^j + (-1)^n X^{N-n+j})$ in the group algebra $\mathbf{C}C_N$. Noticing that $N \ge 2n$, $X^N = 1$ and $v_{\mu-\nu+n} = (-1)^n v_{\mu-\nu}$, we can write

$$\begin{aligned} U * V &= \sum_{\mu<n} \Big(\sum_{\substack{i,j<n \\ i+j=\mu}} u_i v_j + (-1)^n \sum_{\substack{i,j<n \\ i+j-n=\mu}} u_i v_j \Big) X^\mu + X^n(\ldots) \\ &= \sum_{\mu<n} \Big(\sum_{\nu=0}^{\mu} u_\nu v_{\mu-\nu} + (-1)^n \sum_{\nu=\mu+1}^{n-1} u_\nu v_{\mu-\nu+n} \Big) X^\mu + X^n(\ldots) \\ &= \sum_{\mu<n} \Big(\sum_{\nu=0}^{n-1} u_\nu v_{\mu-\nu} \Big) X^\mu + X^n(\ldots) \quad . \end{aligned}$$

Thus the first n coefficients of $U * V$ are just the desired numbers $\omega^{-\mu^2/2} A_\mu$, $0 \le \mu < n$.

Now let us look at the complexity of this algorithm. We need $n - 1$ multiplications to compute U from the input vector (a_ν). If we compute $U * V = DFT_N^{-1}(DFT_N(U) \cdot DFT_N(V))$ using the DFT_N, this takes at most $2 \cdot L(N) + N$ operations, see the last section. Finally, we need $n - 1$ multiplications to recover the spectral coefficients A_μ from $U * V$. Altogether, we have

$$L(n) \le 2 \cdot L(N) + N + 2n - 2 \ .$$

Of course, N should be chosen such that DFT_N can be efficiently computed, e.g. $N = 2^{\lceil \log 2n \rceil}$, which was Bluestein's original choice. On the other hand, our selection of N should be as small as possible, i.e. close to $2n$. Depending on n, we will choose $N = q2^k$ with $q \in \{1, 3, 5, 9, 27\}$.

From the definition of U, it is obvious that the last $N - n$ coefficients of U are equal to zero. In our case where $N = q2^k$, we can compute $DFT_N(U)$ by first computing q DFT_{2^k}'s and then 2^k DFT_q's, see Lemma 2. If we use

the Cooley-Tukey-algorithm to compute DFT_{2^k}, the $N-n$ zeroes in the input (which are in fixed positions) save at least $3N-4n$ operations. As $N-n$ outputs of the final DFT_N^{-1} are irrelevant, we can again save at least $3N-4n$ operations. Thus

$$\begin{aligned} L(n) &\leq 2 \cdot \left(2^k L(q) + q(\frac{3}{2}k2^k - 2^k + 1)\right) - 5N + 10n - 2 \\ &\leq 2^{k+1}L(q) + (3k-7)q2^k + 10n + 2q - 2 \ . \end{aligned}$$

3 The main result

We will now combine the algorithms described in the previous section to prove the following

Theorem 3 *For any finite abelian group G, $L_s(G) \leq 8|G| \log |G|$.*

PROOF. According to Lemma 2, it suffices to prove our assertion for cyclic groups of prime power order n. We use Bluestein's algorithm and choose N as follows:

interval for n	choice of N
$(\frac{1}{2}2^k; \frac{9}{16}2^k]$	$9 \cdot 2^{k-3}$
$(\frac{9}{16}2^k; \frac{5}{8}2^k]$	$5 \cdot 2^{k-2}$
$(\frac{5}{8}2^k; \frac{3}{4}2^k]$	$3 \cdot 2^{k-1}$
$(\frac{3}{4}2^k; \frac{27}{32}2^k]$	$27 \cdot 2^{k-4}$
$(\frac{27}{32}2^k; 2^k]$	2^{k+1}

In the five cases, we obtain the following upper bounds for $L(n)$ (recall that $L(2) = 2$, $L(3) \leq 8$ and $L(5) \leq 22$):

Case 1. $n \in (\frac{1}{2}2^k, \frac{9}{16}2^k]$, $N = 9 \cdot 2^{k-3}$. Then

$$n \log n \geq (k-1)2^{k-1} \ .$$

Thus

$$\begin{aligned} L(n) &\leq 2^{k-2}L(9) + (3k-16)9 \cdot 2^{k-3} + 10n + 16 \\ &\leq 52 \cdot 2^{k-2} + 27k2^{k-3} - 144 \cdot 2^{k-3} + 10n + 16 \\ &\leq \frac{27}{4}((k-1)2^{k-1}) + (\frac{27}{4} - 10)2^{k-1} + 10n + 16 \\ &\leq \frac{27}{4}n \log n - \frac{13}{4}2^{k-1} + 10n + 16 \\ &\leq 6.75n \log n + 7.12n + 16 \ . \end{aligned}$$

Hence $L(n) \leq 8n \log n$ for $n \geq 61$.

Case 2. $n \in (\frac{9}{16}2^k, \frac{5}{8}2^k]$, $N = 5 \cdot 2^{k-2}$. Then

$$n \log n \geq \frac{9}{16}2^k(k - 4 + \log 9) \geq \frac{9}{16}k2^k - 0.47 \cdot 2^k \ .$$

Thus

$$\begin{aligned} L(n) &\leq 2^{k-1}L(5) + (3k - 13)5 \cdot 2^{k-2} + 10n + 8 \\ &\leq 11 \cdot 2^k + \frac{15}{4}k2^k - 65 \cdot 2^{k-2} + 10n + 8 \\ &\leq \frac{20}{3}(\frac{9}{16}k2^k - 0.47 \cdot 2^k) + (\frac{20}{3} \cdot 0.47 + 11 - \frac{65}{4})2^k + 10n + 8 \\ &\leq \frac{20}{3}n \log n - 2.11 \cdot 2^k + 10n + 8 \\ &\leq 6.67n \log n + 6.63n + 8 \end{aligned}$$

Hence $L(n) \leq 8n \log n$ for $n \geq 36$.

Case 3. $n \in (\frac{5}{8}2^k, \frac{3}{4}2^k]$, $N = 3 \cdot 2^{k-1}$. Then

$$n \log n \geq \frac{5}{8}2^k(k - 3 + \log 5) \geq \frac{5}{8}k2^k - 0.43 \cdot 2^k \ .$$

Thus

$$\begin{aligned} L(n) &\leq 2^k L(3) + (3k - 10)3 \cdot 2^{k-1} + 10n + 4 \\ &\leq 8 \cdot 2^k + \frac{9}{2}k2^k - 30 \cdot 2^{k-1} + 10n + 4 \\ &\leq \frac{36}{5}(\frac{5}{8}k2^k - 0.43 \cdot 2^k) + (\frac{36}{5} \cdot 0.43 + 8 - 15)2^k + 10n + 4 \\ &\leq \frac{36}{5}n \log n - 3.90 \cdot 2^k + 10n + 4 \\ &\leq 7.2 \log n + 4.8n + 4 \end{aligned}$$

Hence $L(n) \leq 8n \log n$ for $n \geq 68$.

Case 4. $n \in (\frac{3}{4}2^k, \frac{27}{32}2^k]$, $N = 27 \cdot 2^{k-4}$. Then

$$n \log n \geq \frac{3}{4}2^k(k - 2 + \log 3) \geq \frac{3}{4}k2^k - 0.32 \cdot 2^k \ .$$

Thus

$$\begin{aligned} L(n) &\le 2^{k-3}L(27) + (3k-19)27\cdot 2^{k-4} + 10n + 52 \\ &\le 244\cdot 2^{k-3} + \frac{81}{16}k2^k - 513\cdot 2^{k-4} + 10n + 52 \\ &\le \frac{81}{12}(\frac{3}{4}k2^k - 0.32\cdot 2^k) + (\frac{81}{12}\cdot 0.32 + \frac{244}{8} - \frac{513}{16})2^k + 10n + 52 \\ &\le \frac{81}{12}n\log n + 0.60\cdot 2^k + 10n + 52 \\ &\le 6.75n\log n + 10.72n + 52 \end{aligned}$$

Hence $L(n) \le 8n\log n$ for $n \ge 410$.

Case 5. $n \in (\frac{27}{32}2^k, 2^k]$, $N = 2^{k+1}$. Then

$$n\log n \ge \frac{27}{32}2^k(k-5+3\log 3) \ge \frac{27}{32}k2^k - 0.21\cdot 2^k\ .$$

Thus

$$\begin{aligned} L(n) &\le 2(\frac{3}{2}(k+1)2^{k+1} - 2^{k+1} + 1) - 5\cdot 2^{k+1} + 10n - 2 \\ &\le 6(k+1)2^k - 14\cdot 2^k + 10n \\ &\le 6k2^k - 8\cdot 2^k + 10n \\ &\le \frac{64}{9}(\frac{27}{32}k2^k - 0.21\cdot 2^k) + (\frac{64}{9}0.21 - 8)2^k + 10n \\ &\le 7.12n\log n + 3.5n \end{aligned}$$

Hence $L(n) \le 8n\log n$ for $n \ge 16$.

We now look at those small prime powers n that do not already satisfy $L(n) \le 8n\log n$ by the above arguments, i.e.

$$\begin{aligned} n \in M \ := \ \{&2,3,4,5,7,8,9,11,13,14,15,17,19,23,25,27,41,43,47,49,\\ &53,97,101,103,107,193,197,199,211,389,397,401,409\}\ . \end{aligned}$$

For $n \in M \setminus \{47,107\}$, recursive application of the Cooley-Tukey algorithm (for prime powers p^k, $k > 1$), Rader's method (for primes) and Lemma 2 (for composite numbers with more than one prime divisor) yields $L(n) \le 8n\log n$. This can be easily checked by a little computer program that successively computes upper bounds for $L(n)$, $n = 1,2,3,...$ using the bounds given in the previous section.

For $n = 47$ and $n = 107$, Bluestein's algorithm with $N = 96$ and $N = 256$, respectively, shows that also $L(n) \leq 8n \log n$. Altogether, we have shown that $L(n) \leq 8n \log n$ for all positive integers n, which proves our theorem. □

The Fourier transform for abelian groups can be generalized to arbitrary finite groups: By Wedderburn's Theorem, the group algebra $\mathbf{C}G$ of a finite group G is isomorphic to a suitable algebra of block diagonal matrices:

$$\mathbf{C}G \simeq \bigoplus_{i=1}^{h} \mathbf{C}^{d_i \times d_i} .$$

Every such isomorphism is called a (generalized) Fourier transform for G. Since a non-abelian group G has infinitely many Fourier transforms, we define the linear complexity of the group G by $L_s(G) := \min L_s(W)$, where the minimum is taken over all possible Fourier transforms W for G.

Combining Theorem 3 and the results in [6], we get

Theorem 4 *If G is a finite metabelian group, i.e. $G'' = E$, then*

$$L_s(G) \leq 8|G| \log |G| .$$

References

[1] M.D. ATKINSON, *The Complexity of Group Algebra Computations*, Theor. Comp. Sci., 5 (1977), pp. 205–209.

[2] T. BETH, *Verfahren der schnellen Fourier-Transformation*, Teubner, Stuttgart, 1984.

[3] T. BETH, *On the Computational Complexity of the General Discrete Fourier Transform*, Theor. Comp. Sci., 51 (1987), pp. 331–339.

[4] L.I. BLUESTEIN, *A Linear Filtering Approach to the Computation of the Discrete Fourier Transform*, IEEE Trans. AU-18 (1970), pp. 451–455.

[5] W. BÜCHI, *Die diskrete Fourier Transformation*, Diplomarbeit, Universität Zürich, 1979.

[6] M. CLAUSEN, *Fast Fourier Transforms for Metabelian Groups*, SIAM J. Comput., 18 (1989), pp. 584–593.

[7] M. CLAUSEN, *Fast Generalized Fourier Transforms*, Theor. Comp. Sci. 67 (1989), pp. 55–63.

[8] J.W. COOLEY, J.W. TUKEY, *An Algorithm for the Machine Calculation of Complex Fourier Series*, Math. Comp. 19 (1965), pp. 297–301.

[9] P. DIACONIS, *Spectral Analysis for Ranked Data*, Ann. Statistics, to appear.

[10] P. DIACONIS, D. ROCKMORE, *Efficient Computation of the Fourier Transform on Finite Groups*, Technical Report, Stanford University, April 1988.

[11] B. HUPPERT, *Endliche Gruppen I*, Springer, Berlin, 1967.

[12] S.L. HURST, D.M. MILLER, J.C. MUZIO, *Spectral Techniques in Digital Logic*, Academic Press, 1985.

[13] M.G. KARPOVSKY, *Fast Fourier Transforms on Finite Non-Abelian Groups*, IEEE Trans. Computers 26/10 (1977), pp. 1028–1030.

[14] M.G. KARPOVSKY (ed.), *Spectral Techniques and Fault Detection*, Academic Press, 1985.

[15] H.J. NUSSBAUMER, *Fast Fourier Transform and Convolution Algorithms*, Springer, Berlin, 1981.

[16] C.M. RADER, *Discrete Fourier Transform when the Number of Data Points is Prime*, Proc. IEEE 56 (1968), pp. 1107–1108.

[17] D. ROCKMORE, *Fast Fourier Analysis for Abelian Group Extensions*, preprint, Harvard University, Dec. 1988.

[18] D. ROCKMORE, *Computation of Fourier Transforms on the Symmetric Group*, in: E. Kaltofen and S.M. Watt (eds.): Computers in Mathematics, Springer, New York, 1989.

[19] S. WINOGRAD, *On Computing the Discrete Fourier Transform*, Proc. Nat. Acad. Sci. USA 73 (1976), pp. 1005–1006.

[20] S. WINOGRAD, *Arithmetic Complexity of Computations*, SIAM, 1980.