J. Symbolic Computation (2004) **11**, 1–18

Generating Fast Fourier Transforms of Solvable Groups

M. CLAUSEN AND M. MÜLLER

Department of Computer Science, University of Bonn, Germany

{clausen, meinard}@cs.uni-bonn.de

(Received 31 May 2000)

This paper presents a new algorithm for constructing a complete list of pairwise inequivalent ordinary irreducible representations of a finite solvable group G. The input of the algorithm is a pc-presentation corresponding to a composition series refining a chief series of G. Modifying the Baum-Clausen-Algorithm for supersolvable groups and combining this with an idea of Plesken for constructing intertwining spaces, we derive a worst-case upper complexity bound $O(p \cdot |G|^2 \log(|G|))$, where p is the largest prime divisor of $|G|$. The output of the algorithm is well-suited to perform a fast Fourier transform of G. For supersolvable groups there are composition series which are already a chief series. In this case the generation of DFTs can be done more efficiently than in the solvable case. We report on a recent implementation for the class of supersolvable groups.

1. Introduction

Since its (re-)discovery by Cooley and Tukey in 1965, the classical fast Fourier transform (FFT) has been successfully applied to a wide range of problems in mathematics, computer science and engineering, see (Holmes, 1988). Cooley and Tukey proved that the discrete Fourier transform (DFT) of a length n vector can be computed in $O(n \log n)$ arithmetic operations compared to the naive matrix-vector multiplication that solves this task in $O(n^2)$.

From an algebraic point of view, performing a DFT of length n amounts to evaluating a full set of pairwise inequivalent irreducible representations of the cyclic group C_n of order n. Wedderburn's structure theorem for split semisimple algebras yields the right generalization of the notion of the DFT to arbitrary finite groups G: according to this theorem, the complex group algebra $\mathbb{C}G := \{a \,|\, a : G \to \mathbb{C}\}$ (the signal domain) is isomorphic to an algebra of block diagonal matrices (the spectral domain),

$$D = \oplus_{k=1}^{h} D_k : \mathbb{C}G \longrightarrow \oplus_{k=1}^{h} \mathbb{C}^{d_k \times d_k}.$$

Here, the number h of blocks equals the number of conjugacy classes of G and the projections $D_1, \ldots, D_h$ form a complete set of pairwise inequivalent irreducible representations of $\mathbb{C}G$. (We also call $D_1, \ldots, D_h$ a transversal of the irreducible representations of G and

0747–7171/90/000000 + 00 $03.00/0

denote such a list by $\mathrm{Irr}(G)$.) Every such isomorphism D is called a DFT of G. Concerning these generalized DFTs for a given finite group G, there are two fundamental computational problems:

(1) How can a DFT of G be *generated efficiently*? Note that if G is non-abelian, there are infinitely many DFTs. As we are interested in a fast *generation* of $D = \oplus D_k$ we should choose the representatives D_k in the equivalence classes very carefully.
(2) Is there a *suitable* DFT of G which can be *performed efficiently*? In other words, how must the representations D_k in (1) be chosen in order to facilitate a DFT-evaluation faster than the obvious bound $O(n^2)$, which could then be called a fast(er) Fourier transform (FFT)?

Symmetry adaptation is a useful concept for solving both types of computational problems. This paper is mainly concerned with the first question for the class of solvable groups. Refining a *chief series*

$$\mathcal{C} = (G = G_n \rhd G_{n-1} \rhd \ldots \rhd G_1 \rhd G_0 = \{1\})$$

to a composition series $\mathcal{T}$ of a solvable group G we construct, based on Clifford Theory, in a bottom-up fashion along the *composition series* $\mathcal{T}$ a $\mathcal{T}$-adapted DFT of G. However, applying Clifford directly destroys the $\mathcal{T}$-adaptation. In order to recover adaptation on level i one has to know intertwining spaces between the irreducibles already computed on level $i-1$ and certain G-conjugates. As it turns out, the construction of intertwining spaces is the most expensive part of the algorithm determining the overall complexity. Computing intertwining spaces directly, i.e., by solving a system of linear equations, is too expensive. For this reason, we construct them again in a bottom-up fashion, this time, however, along the *chief series* $\mathcal{C}$, since the normality of the subgroups in question is crucial for our construction. We obtain an algorithm that computes a DFT of a pc-presented solvable group G with $O(p \cdot |G|^2 \log(|G|))$ arithmetic operations, where p is the largest prime divisor of $|G|$.

Generalized FFTs (see problem (2)) have been designed for solvable groups by Beth (1987), for general finite and symmetric groups by Clausen (1989) and by Diaconis and Rockmore (1990), and for supersolvable groups by Baum (1991). Some recent results and further links to the literature about generalized FFTs can be found in (Maslen and Rockmore, 1997). The concept of symmetry adaptation has its origin in Young's seminormal form and orthogonal form of the irreducible representations of symmetric groups, see, e.g., (James and Kerber, 1989), p. 124 ff., and (Bürgisser *et al.*, 1997), p. 343. For problem (1), there is a nearly optimal solution in case of supersolvable groups due to Baum and Clausen (1994). Püschel (1998, 1999) describes an algorithm decomposing the regular representation of any solvable group G, which amounts to computing a DFT of G adapted to a composition series. Unfortunately, he gives no theoretical worst-case running time estimate. His experimental results for small group sizes (up to 500) suggest an *average* running time (averaged over all isomorphism types of groups of a fixed size) which is quadratic in the group order. As far as we know, our paper presents a first *worst-case* upper complexity bound in terms of field operations for the case of solvable groups.

The rest of this paper is organized as follows. After some preparations in Section 2, we describe in Section 3 the general construction of $\mathcal{T}$-adapted DFTs. In Section 4 we present our main algorithm for solvable groups (in the following also refered to as *Algorithm M*) and give a rough analysis and worst case complexity bound. For supersolvable groups the DFT-generation is much easier and can be done in a time which is - up to logarithmic factors - proportional to the output length. The main features and an implementation of this algorithm are described in Section 5. We conclude with some final remarks and an outlook in Section 6.

2. Background from Representation Theory

This section briefly recalls basic notions and facts from representation theory. For a more detailed account, the reader is referred to Serre (1986).

Let G be a finite group. An (ordinary) *representation* of G of *degree* (or *dimension*) d is a group morphism $D : G \to \mathrm{GL}(d, \mathbb{C})$. The corresponding character $\chi : G \to \mathbb{C}$ is defined by $\chi(g) := \mathrm{trace}(D(g))$. Two representations D and D' are called *equivalent*, $D \sim D'$, iff for some invertible matrix X one has $D'(g) = D^X(g)$, where D^X is defined by $D^X(g) := XD(g)X^{-1}$, $g \in G$. As a matter of fact, two representations are equivalent iff their characters coincide. The *direct sum* $D \oplus D'$ of two representations D and D' of G is a representation as well, defined by $(D \oplus D')(g) := D(g) \oplus D'(g)$, for $g \in G$. With mD, $m \in \mathbb{N}$, we denote the m-fold direct sum of D. A representation D is *irreducible* iff D is not equivalent to a direct sum of two representations. Characters corresponding to irreducible representations are called *irreducible characters*. The number of irreducible characters (which is the number of equivalence classes of irreducible representations of G) equals the number of conjugacy classes of G. By Maschke's Theorem, every representation D is equivalent to a direct sum of irreducible representations: $D \sim D_1 \oplus \ldots \oplus D_r$. Moreover, if Δ is an irreducible representation of G with character δ, then the *multiplicity* $\langle \Delta | D \rangle := |\{i : D_i \sim \Delta\}|$ of Δ in D, depends only on their characters. More precisely, if χ denotes the character of D then

$$\langle \Delta | D \rangle = \langle \delta | \chi \rangle := |G|^{-1} \sum_{g \in G} \delta(g^{-1}) \chi(g).$$

Intertwining spaces are another useful concept to deal with multiplicities and, more generally, with direct sum decompositions of representations. The *intertwining space* of two representations D and D' of G is defined by

$$\mathrm{Int}(D, D') := \{X \in \mathbb{C}^{d' \times d} | XD(g) = D'(g)X \quad \text{for all } g \in G\},$$

where d and d' denote the degrees of the representations. By Schur's Lemma, $\mathrm{Int}(D, D)$ is one-dimensional iff D is irreducible. In that case, the intertwining space consists of all scalar multiples of the identity matrix Id_d. The following statements are straightforward consequences of Schur's Lemma.

Lemma 2.1. *Let $D_1, \ldots, D_h$ be pairwise inequivalent irreducible representations of G and let $D_i \sim F_i$. Then for any integers n_i, m_i*

$$\mathrm{Int}\left(\oplus_{i=1}^h m_i D_i, \oplus_{i=1}^h n_i F_i\right) = \bigoplus_{i=1}^h \mathrm{Int}\left(m_i D_i, n_i F_i\right) = \bigoplus_{i=1}^h \mathbb{C}^{n_i \times m_i} \otimes \mathrm{Int}\left(D_i, F_i\right).$$

Furthermore, if D, F are representations of G and Y, X invertible matrices of dimension $\deg(D), \deg(F)$ *respectively, then*

$$\mathrm{Int}\left(F^X, D^Y\right) = Y \,\mathrm{Int}\left(F, D\right) X^{-1}.$$

Let H be a subgroup of G, D a representation of G, and F a representation of H. If the *restriction* $D{\downarrow}H$ of D to H equals F, then D is called an *extension* of F. Starting from F of degree f and a complete set $T = (g_1, \ldots, g_t)$ of left coset representatives of H in G, we obtain a representation of G of degree $f \cdot t$, the *induced representation* $F{\uparrow}_T G$, as follows:

$$(F{\uparrow}_T G)(g) := (\dot{F}(g_i^{-1} g g_j))_{1 \le i,j \le t},$$

where $\dot{F}$ equals F on H and is identically equal to the $f \times f$ zero matrix outside H. According to the Frobenius Reciprocity Theorem, induction and restriction of representations are dual in the following sense: if D is an irreducible representation of G and F an irreducible representation of H then the multiplicity of F in $D{\downarrow}H$ equals the multiplicity of D in $F{\uparrow}_T G$. We abbreviate this common multiplicity by $\langle D|F \rangle$. Analogous results are valid for characters.

Now let $\mathcal{C} = (G = G_n > G_{n-1} > \ldots > G_1 > G_0 = \{1\})$ be a chain of subgroups of G. To this chain we associate a graph, the *$\mathcal{C}$-character graph* of G. Its set of nodes is partitioned into $n+1$ levels. The nodes of level i correspond to the irreducible characters of G_i. Only the nodes of consecutive levels are linked by weighted edges. If χ and ψ are irreducible characters of G_i and G_{i-1}, respectively, then the two nodes are connected by an edge of weight $\langle \chi|\psi \rangle$. This graph will serve as a fundamental data structure for constructing and storing irreducible representations.

There is a close connection between the representations of G and those of a normal subgroup N. This is based on the action of G on the set of irreducible characters of N via $(g * \psi)(n) := \psi(g^{-1}ng) =: \psi^g(n)$. Similarly, if F is a representation of N then for each $g \in G$, $F^g(n) := F(g^{-1}ng)$ defines another representation of N, a *G-conjugate* of F. The following version of Clifford's Theorem will be of importance for us.

Theorem 2.2. *Let $N \lhd G$ and let χ be an irreducible character of G. Let ψ be an irreducible constituent of $\chi{\downarrow}N$ occuring with multiplicity $m > 0$ and suppose $\psi = \psi_1, \ldots, \psi_q$, are the distinct conjugates of ψ in G. Then*

$$\chi{\downarrow}N = m \sum_{k=1}^{q} \psi_k.$$

A proof of this theorem can be found in Theorem (6.2) of (Isaacs, 1976). An analogous result holds for the corresponding representations.

Finally, we need some notation and basic complexity bounds when dealing with a certain kind of sparse matrices and representations. Let K be any field and $d = f \cdot r$ with $f, d, r \in \mathbb{N}$. A matrix M is called *f-block monomial* iff

$$\exists \sigma \in S_r \, \exists A_1, \ldots, A_r \in \mathrm{GL}(f, K) : M = (P_\sigma \otimes \mathrm{Id}_f) \cdot (A_1 \oplus \ldots \oplus A_r),$$

where P_σ denotes the permutation matrix of $\sigma \in S_r := \mathrm{Sym}(\{1, \ldots, r\})$. A representation D of G is called f-block monomial iff $D(g)$ is an f-block monomial matrix for every $g \in G$. Now, suppose that an operation is either a multiplication, addition, subtraction or inversion in K. Let in the following all matrices in question be $d \times d$ matrices over K. Then matrix multiplication and inversion can be done with $O(d^3)$ K-operations (asymptotically more efficient algorithms for matrix multiplications like Strassen's algorithm (Strassen, 1969) are not used). If $f|d$ and all matrices in question are f-block monomial, then the complexity of matrix multiplication and inversion reduces to

$$O\left(\frac{d}{f} f^3\right) = O\left(d \cdot f^2\right). \tag{2.1}$$

Multiplication of an f-block monomial matrix with a full matrix can be done in

$$O\left(\frac{d^2}{f^2} f^3\right) = O\left(d^2 \cdot f\right). \tag{2.2}$$

3. Basics for DFT-Generation of Solvable Groups

In this section we want to summarize the general ideas for an algorithm which constructs for a finite solvable group G, given by a pc-presentation, a DFT adapted to a composition series of G. A finite group G is called *solvable* iff there exists a composition series $\mathcal{T} = (G = G_n \rhd G_{n-1} \rhd \ldots \rhd G_1 \rhd G_0 = \{1\})$, in which all of its composition factors G_i/G_{i-1} are of prime order p_i. For $1 \leq i \leq n$, let g_i be an element in G_i not in G_{i-1}. With respect to $(g_1, \ldots, g_n)$ each element $g \in G$ can be expressed uniquely in *normal form*

$$g = g_n^{e_n} \cdot g_{n-1}^{e_{n-1}} \cdot \ldots \cdot g_1^{e_1} \quad (0 \leq e_i < p_i).$$

The multiplication in G is completely described, if the normal forms of all powers $g_i^{p_i}$ and all commutators $[g_i, g_j] := g_i^{-1} g_j^{-1} g_i g_j$ are known. More formally, every solvable group has a power-commutator presentation (*pc-presentation*) of the form

$$G = \langle g_1, \ldots, g_n \mid g_i^{p_i} = u_i \ (1 \leq i \leq n), [g_i, g_j] = w_{ij} \ (1 \leq i < j \leq n) \rangle,$$

with words $u_i \in G_{i-1}$ and $w_{ij} \in G_{j-1}$, all given in normal form. Moreover, we require the presentation to be consistent, i.e., every word in the generators has a unique normal form. Consistent pc-presentations of this kind exactly describe the class of solvable groups.

With respect to such a pc-presentation a d-dimensional representation D of G is fully described by the representing matrices $D(g_1), \ldots, D(g_n)$ on the generators. Then, for any $g \in G$ given in normal form, $D(g) = D(g_n)^{e_n} \cdot \ldots \cdot D(g_1)^{e_1}$ can be computed with

$$O(d^3 \log(|G|)) \tag{3.1}$$

arithmetic operations using the binary method. In case of f-block monomial representations, this complexity reduces by (2.1) to

$$O(d \cdot f^2 \log(|G|)). \tag{3.2}$$

The concept of *symmetry adaptation* of a representation D is crucial in view of an efficient algorithm. D is called $\mathcal{T}$-adapted iff for all $0 \leq i \leq n$ the following conditions hold:

(1) The restriction $D{\downarrow}G_i$ is *equal* to the direct sum of irreducible representations of G_i, i.e., $D{\downarrow}G_i = \oplus_q F_{iq}$, with irreducible representations F_{iq}.
(2) Equivalent irreducible constituents of $D{\downarrow}G_i$ are *equal*, i.e., if $F_{iq} \sim F_{it}$ then $F_{iq} = F_{it}$ (but not necessarily $q = t$).

If D is $\mathcal{T}$-adapted then, for all $0 \leq i \leq n$, $D{\downarrow}G_i$ is $\mathcal{T}_i$-adapted, where $\mathcal{T}_i$ denotes the chain $(G_i > \ldots > G_0)$. We also write $\mathrm{Irr}(G_i, \mathcal{T}_i)$ for a transversal of $\mathcal{T}_i$-adapted irreducible representations of G_i.

The central idea of the algorithm is based on Clifford's Theorem. In our special case it says that given an irreducible representation F of G_{i-1}, $0 < i \leq n$, then exactly one of the following cases applies.

Case 1. All $F^{g_i^k}$, $0 \leq k < p_i =: p$, are equivalent. Then F extends to p pairwise inequivalent irreducible representations $D_0, \ldots, D_{p-1}$ of G_i of the same degree $\deg(F)$. Moreover, if $\chi^0, \ldots, \chi^{p-1}$ are the linear characters of the cyclic group G_i/G_{i-1} in a suitable order, we have $D_k = \chi^k \otimes D_0$ for all k. Finally, $F{\uparrow}G_i \sim D_0 \oplus \ldots \oplus D_{p-1}$.

Case 2. All $F^{g_i^k}$, $0 \leq k < p$, are pairwise inequivalent. Then the induction $F{\uparrow}G_i$ is an irreducible representation of $\mathbb{C}G_i$ of degree $p \cdot \deg(F)$. Moreover, all representations $F^{g_i^k}{\uparrow}G_i$, $0 \leq k < p$, are equivalent and $(F{\uparrow}G_i){\downarrow}G_{i-1} = \bigoplus_{k=0}^{p-1} F^{g_i^k}$.

For a proof see, e.g., Theorem (6.20) of (Clausen and Baum, 1993). Up to equivalence all irreducible representations of G_i can be obtained this way. This allows us to construct the irreducible representations of G iteratively in a bottom-up fashion along the composition series $\mathcal{T}$. For an efficient construction of $\mathrm{Irr}(G_i)$ from $\mathrm{Irr}(G_{i-1})$ in step i of the iterative construction one should use as much as possible the information already computed on level $i-1$. This means, one should define a $D \in \mathrm{Irr}(G_i)$ in such a way that $D{\downarrow}G_{i-1}$ is not only *equivalent* but *equal* to the direct sum of irreducibles of $\mathrm{Irr}(G_{i-1})$. This is exactly the philosophy of symmetry adaptation defined before. As a consequence, a new representation $D \in \mathrm{Irr}(G_i, \mathcal{T}_i)$ in step i has just to be defined on the generator g_i, the value of D on the generators $g_1, \ldots, g_{i-1}$ can be copied from step $i-1$ without further computations.

However, for the equivalence test and symmetry adaptation we need to know for each $F \in \mathrm{Irr}(G_{i-1}, \mathcal{T}_{i-1})$ the relation between the conjugate representation F^{g_i} and the corresponding $F' \in \mathrm{Irr}(G_{i-1}, \mathcal{T}_{i-1})$ with $F^{g_i} \sim F'$. That is the reason, one needs to know the intertwining spaces $\mathrm{Int}(F^{g_i}, F')$. It turns out that computing these spaces is the most expensive part of a construction following these lines which determines the complexity of the algorithm. We suppose for the moment that we can decide equivalence of two given representations and can compute intertwining spaces. Then the construction can be summarized as follows:

Input: A pc-presentation of a finite solvable group G corresponding to a composition series $\mathcal{T}$ described as above. Note, that $\mathrm{Irr}(G_0, \mathcal{T}_0)$ is trivial.

Step i. $\mathrm{Irr}(G_i, \mathcal{T}_i)$ is computed from $\mathrm{Irr}(G_{i-1}, \mathcal{T}_{i-1})$, $1 \leq i \leq n$. By Clifford's Theorem, for each $F \in \mathrm{Irr}(G_{i-1}, \mathcal{T}_{i-1})$ we have to consider two cases:

Case 1. $F \sim F^{g_i}$. Then F has p_i extensions $D_0, \ldots, D_{p_i-1}$.

- Let ω be a primitive p_ith root of unity and $X \in \mathrm{Int}(F^{g_i}, F) \setminus \{0\}$.
- Determine a solution c_0 of the equation $c^{p_i} X^{p_i} = F(g_i^{p_i})$ in the variable c. Note that $g_i^{p_i}$ is a word in G_{i-1} given by the pc-presentation.
- Define $D_k(g_i) := c_0 \cdot \omega^k \cdot X$, $k = 0, \ldots, p_i - 1$.
- With the information of step $i-1$ we define $D_k{\downarrow}G_{i-1} := F$ to get $\mathcal{T}_i$-adapted extensions of F.

Case 2. $F \not\sim F^{g_i}$. Then $F \uparrow G_i$ is irreducible and $(F \uparrow G_i) \downarrow G_{i-1} = \bigoplus_{k=0}^{p_i-1} F^{g_i^k}$. Now we have to adapt $F \uparrow G_i$.

- Find $F_k \in \mathrm{Irr}(G_{i-1}, \mathcal{T}_{i-1})$ with $F_k \sim F^{g_i^k}$ for $k = 0, \ldots, p_i - 1$.
- Compute $X_k \in \mathrm{Int}(F^{g_i^k}, F_k) \setminus \{0\}$ and set $X := \bigoplus_{k=0}^{p_i-1} X_k$.
- Define $D(g_i) := X^{-1}(F \uparrow G_i)(g_i)X$.
- By setting $D(g_j) := \bigoplus_{k=0}^{p_i-1} F_k(g_j)$ for $j = 0, ..., i-1$, (already known from step $i-1$) D defines a $\mathcal{T}_i$-adapted representation.

Output: A transversal of irreducible $\mathcal{T}$-adapted representations $\mathrm{Irr}(G, \mathcal{T})$, where each $D \in \mathrm{Irr}(G, \mathcal{T})$ is given by the matrices $D(g_1), \ldots, D(g_n)$.

Further details and a verification of this algorithm can be found in (Clausen and Baum, 1993).

4. Algorithm M and Complexity Bounds

Our main algorithm presented in this paper (*Algorithm M*) constructs for any solvable group G given by a pc-presentation corresponding to a composition series $\mathcal{T}$ refining a chief series $\mathcal{C}$ of G a full set of $\mathcal{T}$-adapted pairwise inequivalent irreducible representations of G. Our algorithm works bottom-up along the *chief series*. Within each chief factor we use for the construction of the representations a subalgorithm which is a relative version of the Baum-Clausen-Algorithm for supersolvable groups (Baum and Clausen, 1994) and will be referred to as *Algorithm RBC*. To lift the necessary data from one subgroup of the chief series to the next higher subgroup, we describe an algorithm for testing equivalence (*Algorithm ET*) which is based on an idea of Plesken (Plesken, 1987).

4.1. Algorithm RBC

As a subroutine for Algorithm M we need a relative version of the Baum-Clausen-Algorithm for supersolvable groups (Algorithm RBC). Since the relative version is a straightforward generalization of the origial algorithm and follows the lines described in Section 3, we refer to (Baum and Clausen, 1994) for details and just state the result.

Let H be a finite solvable group with normal subgroup N such that H/N is supersolvable. Then we have a chain of subgroups

$$\mathcal{T} = (H = H_r \triangleright H_{r-1} \triangleright \ldots \triangleright H_1 \triangleright H_0 = N),$$

where $H_k \triangleleft H$ and each $[H_k : H_{k-1}] := p_k$ is prime. By definition, a *pc-presentation of H relative N corresponding to $\mathcal{T}$* is of the form

$$H/N = \langle h_1N, \ldots, h_rN \mid h_k^{p_k}N = u_kN \, (1 \le k \le r), [h_kN, h_\ell N] = w_{k\ell}N \, (1 \le k < \ell \le r) \rangle,$$

with *generators* $h_k \in H_k \setminus H_{k-1}$, $k = 1, \ldots, r$. Furthermore, $u_k = h_{k-1}^{a_{k,k-1}} \cdot \ldots \cdot h_1^{a_{k,1}}$ and $w_{k\ell} = h_k^{b_{k\ell,k}} \cdot \ldots \cdot h_1^{b_{k\ell,1}}$ with suitable exponents $0 \le a_{k,j} < p_j$, $1 \le j < k$, and $0 \le b_{k\ell,j} < p_j$, $1 \le j \le k$.

Suppose we have the following data:

(i) A pc-presentation of H relative N corresponding to $\mathcal{T}$ with generators $h_1, \ldots, h_r$.

(ii) A transversal $\mathrm{Irr}(N)$ of irreducible representations of N. Furthermore, there is an algorithm which can evaluate any $F \in \mathrm{Irr}(N)$ of degree $f = \deg(F)$ at any $n \in N$ in $O(f^3 \cdot \log(|N|))$ operations.
(iii) The h_k-operation of the generators h_k on $\mathrm{Irr}(N)$ given by a permutation π_{h_k} of the set $\mathrm{Irr}(N)$ such that $\pi_{h_k}(F) \sim F^{h_k}$ for all $F \in \mathrm{Irr}(N)$, $k = 1, \ldots, r$.
(iv) Intertwining matrices $X_{h_k,F} \in \mathrm{Int}(F^{h_k}, \pi_{h_k}(F)) \setminus \{0\}$.

Then Algorithm RBC constructs a transversal $\mathrm{Irr}(H, \mathcal{T})$ of irreducible $\mathcal{T}$-adapted representations along the subgroups H_k in a bottom-up fashion. An analysis of this algorithm along the lines of (Baum and Clausen, 1994) (see Appendix B for details) gives a complexity bound of

$$O\left(|H| \log^2(|H|) \sqrt{|N|}\right).$$

We note that for $N = \{1\}$ Algorithm RBC reduces to the original Baum-Clausen-Algorithm for supersolvable groups, which has the complexity bound $O\left(|H| \log^2(|H|)\right)$. (Preparing this paper, we discovered a bug in the complexity analysis in (Baum and Clausen, 1994), leading to an additional $\log(|H|)$ factor. See Appendix A for details.)

4.2. Algorithm ET

A second subroutine of Algorithm M tests two representations for equivalence and constructs a non-trivial intertwining matrix in case of equivalence. The following lemma generalizes an idea of Plesken (Plesken, 1987).

Lemma 4.1. *Let G be a finite group, H a subgroup of G of index $s := [G : H]$ and $g_1, \ldots, g_s$ representatives of the right cosets of H, i.e., $G = Hg_1 \sqcup \ldots \sqcup Hg_s$. Let K be a field with* $\mathrm{char}(K) \nmid s$ *and D, Δ be K-representations of G. Then*

$$\psi : Y \mapsto \frac{1}{s} \sum_{i=1}^{s} \Delta(g_i^{-1}) Y D(g_i)$$

defines a K-linear projection, mapping $\mathrm{Int}(D{\downarrow}H, \Delta{\downarrow}H)$ *onto* $\mathrm{Int}(D, \Delta)$.

Proof. Trivially, ψ is a K-linear map. Furthermore, it follows easily that $\mathrm{Int}(D, \Delta) \subset \mathrm{Int}(D{\downarrow}H, \Delta{\downarrow}H)$ and $\psi(Y) = Y$ for all $Y \in \mathrm{Int}(D, \Delta)$, i.e., ψ is surjective. We just need to show $\psi(Y) \in \mathrm{Int}(D, \Delta)$ for any $Y \in \mathrm{Int}(D{\downarrow}H, \Delta{\downarrow}H)$. Fix such a Y, then

$$Y D(h) = \Delta(h) Y \tag{4.1}$$

for all $h \in H$. Obviously, for every $g \in G$ there are $h_1 \ldots, h_s$ such that (as sets!)

$$\{g_1 g, \ldots, g_s g\} = \{h_1 g_1, \ldots, h_s g_s\}. \tag{4.2}$$

Hence for this g we have

$$\begin{aligned} \Delta(g^{-1})\psi(Y)D(g) &= \frac{1}{s}\sum_{i=1}^{s} \Delta((g_i g)^{-1}) Y D(g_i g) \overset{(4.2)}{=} \frac{1}{s}\sum_{i=1}^{s} \Delta((h_i g_i)^{-1}) Y D(h_i g_i) \\ &\overset{(4.1)}{=} \frac{1}{s}\sum_{i=1}^{s} \Delta(g_i^{-1}) Y D(g_i) = \psi(Y). \end{aligned}$$

□

We use this lemma to design an algorithm for testing two irreducible representations for equivalence and constructing the intertwining space in case of equivalence. We will refer to this equivalence test algorithm as *Algorithm ET*.

Let H be a finite solvable group with normal subgroup N and let

$$\mathcal{T} = (H = H_r \rhd H_{r-1} \rhd \ldots \rhd H_1 \rhd H_0 = N)$$

be a chain of subgroups with prime indices $[H_k : H_{k-1}] =: p_k$, $k = 1, \ldots, r$. In this section we do not assume that the H_k are normal in the whole group H. As usual, let $h_k \in H$ such that $h_k H_{k-1}$ generates H_k/H_{k-1}. Define for any two representations D, Δ of H the maps

$$\psi_k : \mathrm{Int}(D{\downarrow}H_{k-1}, \Delta{\downarrow}H_{k-1}) \to \mathrm{Int}(D{\downarrow}H_k, \Delta{\downarrow}H_k), \quad Y \mapsto \frac{1}{p_k} \sum_{t=0}^{p_k-1} \Delta(h_k^{-t}) Y D(h_k^t).$$

Then $\psi := \psi_r \circ \psi_{r-1} \circ \cdots \circ \psi_1$ defines a projection of $\mathrm{Int}(D{\downarrow}N, \Delta{\downarrow}N)$ onto $\mathrm{Int}(D, \Delta)$. In case D, Δ are irreducible representations then by Schur's Lemma $\psi(Y)$ is either 0 or invertible for all $Y \in \mathrm{Int}(D{\downarrow}N, \Delta{\downarrow}N)$. Now, let $\mathcal{B}$ be a basis of $\mathrm{Int}(D{\downarrow}N, \Delta{\downarrow}N)$. We can test two irreducible representations D, Δ for equivalence by computing all images $\psi(E)$, $E \in \mathcal{B}$. If there is an $\psi(E) \neq 0$ then $D \sim \Delta$ and $\psi(E)$ spans the one-dimensional intertwining space $\mathrm{Int}(D, \Delta)$. Otherwise $\psi(E) = 0$ for all $E \in \mathcal{B}$ and $\mathrm{Int}(D, \Delta) = \{0\}$ by surjectivity of ψ, which implies $D \not\sim \Delta$.

Let $d = \deg(D) = \deg(\Delta)$ and $Y \in \mathrm{Int}(D{\downarrow}N, \Delta{\downarrow}N)$. Using

$$D(h_k^t) Y \Delta(h_k^{-t}) = D(h_k) \left(D(h_k^{t-1}) Y \Delta(h_k^{-(t-1)}) \right) \Delta(h_k)^{-1}$$

for $t = 1, \ldots, p_k - 1$, it follows that $\psi_k(Y)$ can be computed with $O(p_k d^3)$ operations for $k = 1, \ldots, r$. Therefore, computing $\psi(Y)$ takes

$$\sum_{k=1}^{r} O\left(p_k d^3\right) = O\left(d^3 \sum_{k=1}^{r} p_k\right)$$

operations. For the equivalence test one has to compute $\psi(E)$ for all $E \in \mathcal{B}$. This can be done with $O(|\mathcal{B}| \cdot d^3 \sum_{k=1}^{r} p_k)$ operations. In case the irreducible representations D, Δ are f-block monomial, $f \mid d = \deg(D) = \deg(\Delta)$, then the computation of $\psi_k(Y)$ is cheaper (using (2.1) and (2.2)) and can be done in $O(p_k \cdot (\frac{d}{f})^2 \cdot f^3) = O(p_k \cdot d^2 \cdot f)$. This leads to an overall cost of

$$O\left(|\mathcal{B}| \cdot d^2 \cdot f \sum_{k=1}^{r} p_k\right) \tag{4.3}$$

operations for the equivalence test.

4.3. Algorithm M

We now describe Algorithm M for constructing a $\mathcal{T}$-adapted DFT for a finite solvable group G. Let

$$\mathcal{C} = (G = G_n \rhd G_{n-1} \rhd \ldots \rhd G_1 \rhd G_0 = \{1\})$$

be a chief series of G, i.e., $G_i \lhd G$. Furthermore, the chief factors are elementary abelian, i.e., there exist $r_i \in \mathbb{N}$ and prime numbers p_i such that $G_i/G_{i-1} \simeq C_{p_i}^{r_i}$ (see Theorem (9.13) of (Huppert, 1967)). We refine this chief series to a composition series $\mathcal{T}$ of G with suitable subgroups

$$G_i = G_{ir_i} \rhd G_{ir_i-1} \rhd \ldots \rhd G_{i1} \rhd G_{i0} = G_{i-1}.$$

Note that the G_{ik}, $1 \leq k < r_i$, are in general not normal in G. Furthermore, let G be given by a pc-presentation with generators $\{g_{ik} \in G \,|\, 1 \leq i \leq n, 1 \leq k \leq r_i\}$ corresponding to $\mathcal{T}$ such that $g_{ik}G_{ik-1}$ generates $G_{ik}/G_{ik-1} \simeq C_{p_i}$.

Algorithm M works bottom-up along the chief series $\mathcal{C}$. At level $i, 1 \leq i \leq n$, it takes the following input:

(1) $\mathcal{F} := \mathrm{Irr}(G_{i-1}, \mathcal{T}_{i-1})$, a full set of pairwise inequivalent $\mathcal{T}_{i-1}$-adapted irreducible representations of G_{i-1}. The corresponding character graph of G_{i-1}.
(2) For every $i - 1 < j \leq n$ and $1 \leq k \leq r_j$ the g-action, $g := g_{jk}$, on $\mathcal{F}$ given by a permutation π_g of $\mathcal{F}$ such that $F^g \sim \pi_g F$ for all $F \in \mathcal{F}$. Furthermore, intertwining matrices $X_{gF} \in \mathrm{Int}(F^g, \pi_g F)$ for every $F \in \mathcal{F}$.

and computes the following output:

(1) $\mathcal{D} := \mathrm{Irr}(G_i, \mathcal{T}_i)$, a full set of pairwise inequivalent $\mathcal{T}_i$-adapted irreducible representations of G_i. The corresponding character graph of G_i.
(2) For every $i < j \leq n$ and $1 \leq k \leq r_j$ the g-action, $g := g_{jk}$, on $\mathcal{D}$ given by a permutation τ_g of $\mathcal{D}$ such that $D^g \sim \tau_g D$ for all $D \in \mathcal{D}$. Furthermore, intertwining matrices $Y_{gD} \in \mathrm{Int}(D^g, \tau_g D)$ for every $D \in \mathcal{D}$.

Note that the input of level 0 is trivial. Level i of the algorithm consists of two phases. (See next section for the complexity analysis of these two phases.)

Phase 1. Let $H := G_i$, $N := G_{i-1}$, $r := r_i$ and $p := p_i$. Then N is normal in H and H/N is elementary abelian, in particular supersolvable. Set $H_k := G_{ik}$, $k = 0, \ldots, r$, and $h_k := g_{ik}$, $k = 1, \ldots, r$, then (i) of the Algorithm RBC holds. Condition (ii) holds, since by induction hypothesis (1) of level $i-1$, the set $\mathcal{F} := \mathrm{Irr}(N, \mathcal{T}_{i-1})$ has already been constructed, i.e., $F \in \mathcal{F}$ are given on the generators of N. Therefore, by (3.1), $F(n)$ can be computed in $O(f^3 \cdot \log |N|)$, $f := \deg(F)$, for any $n \in N$ given in normal form. The data (iii) and (iv) are given by induction hypothesis (2) of level $i - 1$. Therefore we can use Algorithm RBC to construct $\mathcal{D} := \mathrm{Irr}(H, \mathcal{T}_i)$, which is the output (1) of level i.
In Algorithm RBC all the data needed to extend the character graph from G_{i-1} to G_i has already been computed.

Phase 2. We fix any $g := g_{jk}$, $i < j \leq n$, $1 \leq k \leq r_j$, and $D \in \mathcal{D}$, $d := d(D) := \deg(D)$. In order to define $\tau_g D$, we need to find the representation $\Delta \in \mathcal{D}$ with $D^g \sim \Delta$.

We reduce the number of possible candidates in $\mathcal{D}$ by looking on level $i-1$, To this end, we consider the information of induction hypothesis (2) of level $i-1$.
Consider the restriction $D{\downarrow}N$, whose decomposition into irreducibles of $\mathcal{F}$ can be read off the character graph of G_i. Let $F \in \mathcal{F}$, $f := \deg(F)$, with $m := m(D) := \langle D|F\rangle > 0$ and $\{F_1 = F, F_2, \ldots, F_q\} \subset \mathcal{F}$, $q := q(D) \in \mathbb{N}$, the orbit of F under the action of H on $\mathcal{F}$. Then, by Clifford's Theorem 2.2,

$$D{\downarrow}N \sim m \cdot \bigoplus_{k=1}^{q} F_k.$$

Since D is $\mathcal{T}_i$-adapted, there is a permutation matrix P of the form $P = P_\sigma \otimes \mathrm{Id}_f$ with a permutation $\sigma \in S_{d/f}$ such that

$$D{\downarrow}N = P\left(\bigoplus_{k=1}^{q} m \cdot F_k\right)P^{-1}.$$

Now, since $D^g{\downarrow}N \sim m \cdot \bigoplus_{k=1}^{q} F_k^g \sim m \cdot \bigoplus_{k=1}^{q} \pi_g F_k$, we know that

$$\Delta \in \{\Delta_1, \Delta_2, \ldots, \Delta_\ell\} \subset \mathcal{D}, \quad \ell := \ell(D) \in \mathbb{N},$$

where, by definition, this set consists precisely of those representations of $\mathcal{D}$ whose restriction to N are equivalent to $m \cdot \bigoplus_{k=1}^{q} \pi_g F_k$. This information can be easily computed looking at the character graph of G_i. We now use Algorithm ET to decide which of Δ_λ, $1 \le \lambda \le \ell$, is equivalent to D^g. To this end, we need a basis $\mathcal{B}$ of $\mathrm{Int}(D^g{\downarrow}N, \Delta_\lambda{\downarrow}N)$. Since Δ_λ is $\mathcal{T}_i$-adapted, there is a permutation matrix Q_λ of the form $Q_\lambda = \sigma_\lambda \otimes \mathrm{Id}_f$ with a permutation $\sigma_\lambda \in S_{d/f}$ such that

$$\Delta_\lambda{\downarrow}N = Q_\lambda\left(\bigoplus_{k=1}^{q} m \cdot \pi_g F_k\right)Q_\lambda^{-1}.$$

Then it follows by Lemma 2.1 that

$$\begin{aligned}
\mathrm{Int}\left(D^g{\downarrow}N, \Delta_\lambda{\downarrow}N\right) &= \mathrm{Int}\left(P\left(\bigoplus_{k=1}^{q} m \cdot F_k^g\right)P^{-1}, Q_\lambda\left(\bigoplus_{k=1}^{q} m \cdot \pi_g F_k\right)Q_\lambda^{-1}\right)\\
&= Q_\lambda \mathrm{Int}\left(\bigoplus_{k=1}^{q} m \cdot F_k^g\,, \bigoplus_{k=1}^{q} m \cdot \pi_g F_k\right)P^{-1}\\
&= Q_\lambda\left[\bigoplus_{k=1}^{q} \mathbb{C}^{m\times m} \otimes \mathrm{Int}\left(F_k^g, \pi_g F_k\right)\right]P^{-1}
\end{aligned}$$

Note that all the $X_{gF_k} \in \mathrm{Int}(F_k^g, \pi_g F_k)$ are known by induction hypothesis (2) of level $i-1$ and therefore

$$\mathcal{B} = \left\{E_{abc} := Q_\lambda\left[\bigoplus_{k=1}^{q} \delta_{kc} \cdot (E_{ab} \otimes X_{gF_k})\right]P^{-1}, 1 \le a, b \le m, 1 \le c \le q\right\}$$

is a basis of $\mathrm{Int}(D^g{\downarrow}N, \Delta_\lambda{\downarrow}N)$, where E_{ab} denotes the $m \times m$-matrix with exactly one non-zero entry 1 at position (a, b). Obviously,

$$|\mathcal{B}| = q \cdot m^2.$$

The rest of Phase 2 is now a straightforward application of Algorithm ET. Using the basis $\mathcal{B}$ we can decide whether D^g and Δ_λ are equivalent or not. In case of equivalence, we have $\Delta = \Delta_\lambda$ and set $\tau_g D := \Delta_\lambda$. Furthermore, in this case Algorithm ET also constructs a non-trivial $Y_{gD} \in \mathrm{Int}(D^g, \tau_g D)$. This is exactly the data (2) of level i we had to compute.

4.4. Analysis of Algorithm M

In this section we analyse the Algorithm M and determine its asymptotic behaviour. In our complexity model an arithmetic operation is a basic field operation in K (multiplication, inversion, addition, subtraction, copy), which are assumed to cost $O(1)$. For a discussion arising when computing exactly over the cyclotomic field $K = \mathbb{Q}^{(e)}$ (instead over $K = \mathbb{C}$) we refer to Section 6.

For our analysis we need the following estimates. With the notation of the last subsection we have $\sum_{D\in\mathcal{D}} \deg(D)^2 = |H|$. Since $\{\Delta_1, \Delta_2, \dots, \Delta_\ell\} \subset \mathcal{D}$ and $d = \deg(\Delta_\lambda)$ for all $\lambda = 1, \dots, \ell(D)$, we get

$$\ell(D) \cdot d^2 = \sum_{\lambda=1}^{\ell(D)} \deg(\Delta_\lambda)^2 \leq |H|. \tag{4.4}$$

Now, let G be a finite group and $\mathcal{D} = \mathrm{Irr}(G)$ be a transversal of the irreducible representations of G. Then $|G| = \sum_{D\in\mathcal{D}} \deg(D)^2$ and $d := \max_{D\in\mathcal{D}}(\deg(D)) \leq |G|^{\frac{1}{2}}$. Hence for all real $s \geq 2$, we have

$$d^s(G) := \sum_{D\in\mathcal{D}} \deg(D)^s \leq d^{s-2} \sum_{D\in\mathcal{D}} \deg(D)^2 \leq |G|^{\frac{s}{2}}. \tag{4.5}$$

We analyse the number of operations needed for a fixed level i, $1 \leq i \leq n$, in Phase 1 and Phase 2 of Algorithm M.

Phase 1. In step i of Algorithm M we use Algorithm RBC for $H = G_i$ and $N = G_{i-1}$ which needs

$$O(|H| \log^2(|H|)\sqrt{|N|}) \tag{4.6}$$

operations. Building up the character graph of G_i from the one of G_{i-1} can be done with few operations not effecting the asymptotic behaviour of the overall complexity.

Phase 2. In step i we have fixed a $g = g_{jk}$ and a $D \in \mathcal{D}$. Determining the numbers $m(d), q(D), \ell(D)$, the representations F_k, $k = 1, \dots, q(D)$, the representations Δ_λ, $\lambda = 1, \dots, \ell(D)$ and the permutation matrices P and Q_λ are for the most part table lookups in the character graph of G_i and copy operations which can be done with a negligible number of operations (not increasing the overall complexity). The expensive part is Algorithm ET. Independent of ℓ we have to build up the basis $\mathcal{B}$, which contains f-block monomial matrices with just one non-zero f-block. This can be done in

$$O\left(|\mathcal{B}| \cdot f^3\right) = O\left(m^2 \cdot q \cdot f^3\right) = O\left(d^3\right), \tag{4.7}$$

using $|\mathcal{B}| = q \cdot m^2$ and $d = m \cdot q \cdot f$. Furthermore, one has to compute $D^g(h_k) = D(g^{-1}h_k g)$ for $k = 1, \dots, r$. Since $g^{-1}h_k g$ can be read off the pc-presentation with no cost and is a normalized word in $H_k < H$, using the f-block monomiality of D we can compute all $D^g(h_k)$ by (3.2) in

$$O\left(\sum_{k=1}^{r} d \cdot f^2 \log(|H_k|)\right) = O\left(r \cdot d^3 \cdot \log(|H|)\right). \tag{4.8}$$

For D one has to perform at most ℓ equivalence tests with Δ_λ, $\lambda = 1, \dots \ell$, respectively. Since D and all Δ_λ are f-block monomial, Algorithm ET for all ℓ tests can be done by (4.3) with

$$O\left(\ell \cdot |\mathcal{B}| \cdot d^2 \cdot f \cdot r \cdot p\right) \stackrel{(4.4)}{=} O\left(|H| \cdot r \cdot p \cdot |\mathcal{B}| \cdot f\right) = O\left(|H| \cdot r \cdot p \cdot d^2\right) \tag{4.9}$$

operations. Summing over all $D \in \mathcal{D}$, we get from (4.7),(4.8) and (4.9) the complexity bound of step i of Phase 2 for a fixed $g = g_{jk}$:

$$\begin{aligned} & \sum_{D \in \mathcal{D}} \left(O\left(d^3\right) + O\left(r \cdot d^3 \cdot \log(|H|)\right) + O\left(|H| \cdot r \cdot p \cdot d^2\right)\right) \\ \stackrel{(4.5)}{=} \quad & O\left(|H|^{\frac{3}{2}} + r \cdot |H|^{\frac{3}{2}} \log(|H|) + |H|^2 \cdot r \cdot p\right) = O\left(|H|^2 \cdot r \cdot p\right). \end{aligned}$$

Now, there are at most $\log([G : H])$ generators $g = g_{jk}$, $i < j \leq n$ and $1 \leq k \leq r_j$ which leads to the following complexity bound for Phase 2 of step i:

$$O\left(\log([G : H]) \cdot |H|^2 \cdot r \cdot p\right). \tag{4.10}$$

Altogether, we have proved the following:

Lemma 4.2. *The number of operations of Algorithm M needed in level i to compute the data (1) and (2) of G_i from the data (1) and (2) of G_{i-1} is for Phase 1*

$$O\left(|G_i| \log^2(|G_i|)\sqrt{|G_{i-1}|}\right)$$

and for Phase 2

$$O\left(\log([G : G_i]) \cdot |G_i|^2 \cdot r_i \cdot p_i\right).$$

Summing over all levels $1 \leq i \leq n$ we obtain, up to a suitable constant $\gamma \in \mathbb{R}$, the following upper bound for the number of operations of Algorithm M. Here we use $[G : G_i]|G_i| = |G|$, $|G_i| \leq |G_n| \cdot 2^{i-n}$ and $\log([G : G_i]) \leq [G : G_i]$.

$$\begin{aligned} & \sum_{i=1}^{n} \gamma \cdot \left(|G_i| \log^2(|G_i|)\sqrt{|G_{i-1}|} + \log([G : G_i]) \cdot |G_i|^2 \cdot r_i \cdot p_i\right) \\ \leq \quad & \gamma \cdot \log^2(|G_n|) \sum_{i=1}^{n} |G_n| \cdot 2^{i-n} \cdot |G_n|^{\frac{1}{2}} \cdot (2^{i-n})^{\frac{1}{2}} + \gamma \sum_{i=1}^{n} [G : G_i]|G_i||G_n| \cdot 2^{i-n} \cdot r_i \cdot p_i \\ \leq \quad & \gamma |G|^{\frac{3}{2}} \log^2(|G|) \sum_{i=1}^{n} 2^{i-n} + \gamma |G|^2 \max\{r_i \cdot p_i | 1 \leq i \leq n\} \sum_{i=1}^{n} 2^{i-n} \\ \leq \quad & 2\gamma \left(|G|^{\frac{3}{2}} \log^2(|G|) + |G|^2 \max\{r_i \cdot p_i | 1 \leq i \leq n\}\right). \end{aligned}$$

Note that the complexity of Phase 2 is asymptotically more expensive than the one of Phase 1. We summarize the result in the following theorem, where an operation is a field operation in $\mathbb{Q}^{(e)}$.

THEOREM 4.3. *The ordinary irreducible representations of a solvable group G can be computed from a power-commutator presentation of G corresponding to a composition series refining a chief series with*

$$O(\max\{r_i \cdot p_i | 1 \leq i \leq n\} \cdot |G|^2)$$

operations. Using $r_i \leq \log(|G|)$, $1 \leq i \leq n$, one gets the complexity bound

$$O(p \cdot |G|^2 \log(|G|)),$$

where p denotes the largest prime divisor of $|G|$.

We want to emphasize two important features of Algorithm M which are decisive for its efficiency.

(1) Within two successive subgroups G_{i-1} and G_i of the chief series, all occuring matrices and representations are block-monomial, the block sizes determined by level $i-1$. Computing with the sparse block-monomial matrices is much cheaper than computing with full matrices.
(2) Since the subgroups G_i of the chief series are normal in the entire group G, one has a G-operation on the respective sets $\mathrm{Irr}(G_i)$. This allows a bottom-up construction of the corresponding intertwining matrices along the G_i instead of, e.g., solving linear equations on each level separately.

We have not yet implemented Algorithm M. However, we have implemented the Baum-Clausen-Algorithm for supersolvable groups which shows its practicability.

5. Implementation for Supersolvable Groups

Before we give some details and running times of an implementation of the Baum-Clausen-Algorithm for supersolvable groups, we want to mention two additional features that hold for supersolvable groups but not in general for solvable groups.

- For supersolvable groups G every DFT adapted to a chief series of G turns out to be automatically *monomial*, i.e., 1-block monomial. Processing only monomial matrices is the main reason for the efficiency of the Baum-Clausen-Algorithm.
- Even better, it turns out that all non-zero entries of the matrices are eth roots of unity, e denoting the exponent of G. (We also call such matrices *e-monomial.*) Since all matrix manipulations are either multiplications or inversions, one can compute symbolically in the additive group $\mathbb{Z}_e := \mathbb{Z}/e\mathbb{Z}$, i.e., one never runs into numerical problems!

In the following we use the notation of Section 3 with G being a supersolvable group and $\mathcal{D}_i := \mathrm{Irr}(G_i, \mathcal{T}_i)$. Define $d^1(G) := \sum_{D \in \mathcal{D}} \deg(D)$ and $\Omega := \sum_{i=1}^{n} i \cdot d^1(G_i)$. Then Ω is the number of all non-zero matrix coefficients of the matrices $D(g_k)$, $D \in \mathcal{D}_i$, $1 \leq i \leq n$, $1 \leq k \leq i$, which is the output of the algorithm on all levels i. One can show that the number of operations of the algorithm is nearly proportional (up to a logarithmic factor) to this magnitude Ω, which gives in general a much better complexity bound than $O(|G| \log^2(|G|))$. In this sense the algorithm is nearly optimal.

The Baum-Clausen-Algorithm has been implemented in the programming language C/C++ and tests were run on an Intel Pentium II with 300 MHz. The efficiency of the implementation is based on the fact, that e-monomial matrices of size N can be multiplied or inverted with only N operations in $\mathbb{Z}_e$. Since any e-monomial matrix $M \in \mathbb{C}^{N\times N}$ can be written in the form

$$M = P_\pi \cdot \mathrm{diag}(\omega^{a_1}, \ldots, \omega^{a_N})$$

with a permutation $\pi \in S_N$ and non-zero coefficients $\omega^{a_1}, \ldots, \omega^{a_N}$, just the $2N$ integers $\pi(1), \ldots, \pi(N)$ and $a_1, \ldots, a_N$ have to be stored for M. The following table shows the running times of the implementation of the Baum-Clausen-Algorithm for some small supersolvable groups. Here $|G|$ is the order of G, h the number of conjugacy classes of G, Ω defined as above, T the running time in milliseconds (ms) and T/Ω the quotient of the last two quantities. The groups in the first three examples are direct products of the symmetric group S_3 and the last example is concerned with a Sylow 2-subgroup of the symmetric group S_{16}.

G	$\lvert G\rvert$	h	Ω	T (ms)	T/Ω
$(S_3)^5$	7776	243	13235	266	0.020
$(S_3)^6$	46656	729	63528	1125	0.018
$(S_3)^7$	279936	2187	296464	4250	0.014
$\mathrm{Syl}_2(S_{16})$	32768	230	30960	2156	0.069

Of course, the first three groups are of a very simple nature. However, the running time of the algorithm does not essentially depend on the complexity of the pc-presentation, but mainly on the number and degrees of the irreducible representations constituting the DFT. This is verified by the more complex example $\mathrm{Syl}_2(S_{16})$. Therefore, the actual running times for constructing a monomial DFT of G reflect very well the theoretical result concerning the output length Ω.

As we have remarked in the introduction, a $\mathcal{T}$-adapted DFT allows a fast Fourier transform of complex valued signals $G \to \mathbb{C}$. In this sense, the Baum-Clausen-Algorithm is a fast program generator for FFTs of supersolvable groups. We have also implemented (in C/C++) the $O(|G|\log(|G|))$-FFT algorithm and its inverse (IFFT) for supersolvable groups as described in (Baum, 1991). The *input* of the FFT is the *output* of the Baum-Clausen-Algorithm and a complex valued signal. The output is the FFT of the signal, which is again a complex valued signal of the same length. The following table shows the running times of the implemented FFT and IFFT, transforming randomly generated complex signals. As above, $|G|$ is the order of G, which is also the length of the complex signal. In the FFT-column and IFFT-column are the running times in milliseconds (ms) of the FFT and IFFT, respectively.

G	$\lvert G\rvert$	FFT (ms)	IFFT (ms)
$(S_3)^5$	7776	250	328
$(S_3)^6$	46656	1813	2406
$(S_3)^7$	279936	12109	16985
$\mathrm{Syl}_2(S_{16})$	32768	1844	1827

These results show that the running times do not explode, but are approximately linear in the group size, which reflects very well the theoretical $O(|G|\log(|G|))$-complexity bound. Readers interested in the source code of both programs should contact one of the authors.

6. Final Remarks and Future Work

So far we have been concerned with representations over the complex field. By R. Brauer's theorem on splitting fields, ordinary irreducible representations of a finite group G can be constructed over the cyclotomic field $\mathbb{Q}^{(e)}$, where e denotes the exponent of G. Even though $\mathbb{Q}^{(e)} = \mathbb{Q}[X]/(\Phi_e(X))$ allows *exact* arithmetic, $\Phi_e(X)$ denoting the eth cyclotomic polynomial, computing in $\mathbb{Q}^{(e)}$ can be very expensive as we have no control over the sizes of the coefficients of the polynomials.

This problem does not occur when computing over finite fields. If K is a finite field containing a primitive eth root of unity and $\mathrm{char}(K) \nmid |G|$, then K is a splitting field of G as well. Moreover, there is a close relation between ordinary irreducible representations and irreducible K-representations. Note that Algorithm M works over any such field K. Hence we can work over a finite field K to obtain structural information (like character graph, equivalences, etc.) concerning ordinary representations. To obtain representations over $\mathbb{Q}^{(e)}$ from those over K, lifting techniques generalizing Hensel's Lemma is the content of an ongoing research project of the authors.

Appendix A

In this appendix we fix the bug in the complexity analysis in (Baum and Clausen, 1994), leading to an additional $\log(|G|)$ factor.

We go into Section 4 of (Baum and Clausen, 1994), p. 357. In Phase 2 of the Baum-Clausen-Algorithm the permutation τ_j and the intertwining matrices Y_{jD} are computed for each $i < j \leq n$. In the analysis, summation over those j has been forgotten. Taking this into account, one gets an additional factor $(n-i)$ in the upper bounds for the number of operations in Case 1

$$(n-i) \cdot (4f\log(|G_i|) + pf + f(2i-2) + 5)$$

and in Case 2

$$(n-i) \cdot \left(2f - 2\frac{f}{p} + 5 - f\frac{5}{p}\right)$$

(compare with p. 358). Following the rest of the analysis, one easily sees that an upper bound for the number of basic operations is given by $O(n|G|\log(|G|))$. (Compare with Theorem 4.1 of (Baum and Clausen, 1994)). Furthermore, note that $n \leq \log(|G|)$).

Appendix B

In this appendix we derive the complexity bound of Algorithm RBC along the lines of (Baum and Clausen, 1994). As mentioned in Section 4.1, Algorithm RBC is - based on the assumptions (i) to (iv) - a straightforward generalization of the Baum-Clausen-Algorithm. The only difference is that one starts with the subgroup N instead of the trivial group $\{1\}$. This has consequences concerning the complexity bound, since the representations and intertwining matrices appearing in Algorithm RBC are not any longer monomial as in the Baum-Clausen-Algorithm. However, from the construction it follows easily that all appearing representations and intertwining matrices are at least block monomial, where the block sizes are bounded by the maximal degree over all representations in $\mathrm{Irr}(N)$. For example, if $F \in \mathrm{Irr}(H, \mathcal{T})$ is any representation and $\Gamma \in \mathrm{Irr}(N)$ with $\langle F|\Gamma\rangle > 0$, $\gamma := \deg(\Gamma)$, then F is γ-block monomial. Note that $\gamma \leq \sqrt{|N|}$.

We analyze level k of Algorithm RBC. Let $F \in \mathcal{F}$, $\mathcal{F} := \mathrm{Irr}(H_{k-1}, \mathcal{T}_{k-1})$, and $f := \deg(F)$. As mentioned before, F is block monomial of some block size $\mathrm{bs}(F)$ with

$$\mathrm{bs}(F) \leq \min(\sqrt{|N|}, f). \tag{6.1}$$

Let $\mathrm{m}(F) := f \cdot \mathrm{bs}(F)^2$, then (compare (2.1)) the number of operations needed for multiplication or inversion of matrices of this block structure is bounded by

$$2 \cdot \mathrm{m}(F) = 2 \cdot f \cdot \mathrm{bs}(F)^2 \leq 2 \cdot f^2 \cdot \mathrm{bs}(F) \leq 2 \cdot f^2 \cdot \sqrt{|N|}. \tag{6.2}$$

To obtain bounds for the number of operations in Phase 1 and Phase 2 of Algorithm RBC, one has just to replace the groups G_{i-1} by H_{k-1} and the factor f by $\mathrm{m}(F)$ in the analysis in Section 4 in (Baum and Clausen, 1994). Altogether one gets the following bounds (considering also the corrections described in Appendix A):

Phase 1, Case 1: $4\mathrm{m}(F)\log(|H_k|) + p_k\mathrm{m}(F)(k+1) + \mathrm{m}(F)(2k-4) + 2$

Phase 1, Case 2: $\frac{4}{p_k}\mathrm{m}(F)\log(|H_{k-1}|) + \mathrm{m}(F)(k+5) + \frac{\mathrm{m}(F)}{p_k}(2k-9)$

Phase 2, Case 1: $(n-k) \cdot (4\mathrm{m}(F)\log(|H_k|) + p_k\mathrm{m}(F) + \mathrm{m}(F)(2k-2) + 5)$

Phase 2, Case 2: $(n-k) \cdot \left(2\mathrm{m}(F) - 2\frac{\mathrm{m}(F)}{p_k} + 5 - \mathrm{m}(F)\frac{5}{p_k}\right)$

As the first cases of both phases are obviously more expensive than the corresponding second ones, our worst-case analysis will be based on Case 1. If we sum up over all representations $F \in \mathcal{F}$ and use the fact that

$$\sum_{F\in\mathcal{F}} \mathrm{m}(F) \;=\; \sum_{F\in\mathcal{F}} f \cdot \mathrm{bs}(F)^2 \;\overset{(6.2)}{\leq}\; \sum_{F\in\mathcal{F}} f^2\sqrt{|N|} \;=\; |H_{k-1}|\sqrt{|N|},$$

we obtain the upper bound

$$\begin{aligned}
& \sqrt{|N|}(4|H_{k-1}|\log(|H_k|) + p_k|H_{k-1}|(k+1) + |H_{k-1}|(2k-2)) \\
& \quad + \sqrt{|N|}(n-k)(4|H_{k-1}|\log(|H_k|) + p_k|H_{k-1}| + |H_{k-1}|(2k+3)) \\
\leq\ & \sqrt{|N|}(4(n-k+1)|H_{k-1}|\log(|H_k|) + p_k|H_{k-1}|(n+1) + |H_{k-1}|(2k+3)(n-k+1)) \\
\leq\ & \sqrt{|N|}(2(n-k+1)|H_k|\log(|H_k|) + |H_k|(n+1) + |H_k|(k+1.5)(n-k+1)) \\
\leq\ & \sqrt{|N|}(2n|H_k|\log(|H_k|) + n \cdot k \cdot |H_k| + 3n|H_k|) \\
\leq\ & \sqrt{|N|} \cdot 3n \cdot (|H_k|\log(|H_k|) + |H_k|)
\end{aligned}$$

for the number of operations in level k of Algorithm RBC. Summing up over all levels $1 \leq k \leq r$, we obtain - analogously to p. 359 of (Baum and Clausen, 1994) - as the upper bound

$$\sqrt{|N|} \cdot 6n \cdot (|H|\log(|H|) + |H|) = O\left(|H|\log^2(|H|)\sqrt{|N|}\right)$$

for the total number of operations of Algorithm RBC.

Acknowledgement

We would like to thank the anonymous referees for their valuable comments and suggestions.

References

Baum, U. (1991). Existence and Efficient Construction of Fast Fourier Transforms of Supersolvable Groups. Computational Complexity **1**, 235–256.

Baum, U., Clausen, M. (1994). Computing irreducible representations of supersolvable groups. Mathematics of Computation, Volume **63**, Number 207, 351–359.

Beth, T. (1987). On the Computational Complexity of the General Discrete Fourier Transform. Theor. Comp. Sci. **51**, 331–339.

Bürgisser, P., Clausen, M., Shokrollahi, M.A. (1997). Algebraic Complexity Theory. Grundlehren der mathematischen Wissenschaften, Volume **315**, Springer Verlag, Berlin.

Clausen, M. (1989). Fast Generalized Fourier Transforms. Theor. Comp. Sci. **67**, 55–63.

Clausen, M., Baum, U. (1993). Fast Fourier Transforms. BI-Wissenschaftsverlag, Mannheim.

Clausen, M., Müller, M. (1999). A Fast Program Generator of FFTs. Proceedings AAECC-13, Honolulu, LNCS **1719**, 29–42.

Diaconis, P., Rockmore, D. (1990). Efficient computation of the Fourier transform of finite groups. J. of the A.M.S. **3**(2), 297–332.

Holmes, R. (1988). Mathematics of Signal Processing I and II. MIT Lincoln Laboratory TR.

Huppert, B. (1967). Endliche Gruppen I. Grundlehren der mathematischen Wissenschaften, Volume **134**, Springer Verlag.

Isaacs, I. (1976). Character Theory of Finite Groups. Academic Press, Inc.

James, G., Kerber, A. (1989). The Representation Theory of the Symmetric Group. Cambridge University Press.

Maslen, D., Rockmore, D. (1997). Generalized FFTs. DIMACS Series in Disc. Math. Theor. Comp. Sci., L. Finkelstein and W. Kantor, Eds., vol. **28**, 183–237.

Plesken, W. (1987). Towards a Soluble Quotiont Algorithm. J. Symbolic Computation **4**, 111–122.

Püschel, M. (1998). Konstruktive Darstellungstheorie und Algorithmengenerierung. PhD thesis, Universität Karlsruhe, Fakultät für Informatik.

Püschel, M. (1999). Decomposing Monomial Representations of Solvable Groups. Technical Report Drexel-MCS-1999-2, Dept. of Mathematics and Computer Science, Philadelphia.

Serre, J.-P. (1986). Linear Representations of Finite Groups. Graduate Texts in Mathematics, Springer-Verlag.

Strassen, V. (1969). Gaussian elimination is not optimal. Num. Math. **13**, 354–356.