Fast Fourier Transform Algorithms for Finite Abelian Groups

L. Auslander
J. R. Johnson
and
R. W. Johnson

Technical Report DU-MCS-95-01
Dept. of Mathematics and Computer Science
Drexel University
Philadelphia, PA 19104

Fast Fourier Transform Algorithms for Finite Abelian Groups

L. Auslander*
Department of Mathematics
The City University of New York
New York, NY 10036

J. R. Johnson*,†
Department of Mathematics and Computer Science
Drexel University
Philadelphia, PA 19104

R. W. Johnson*,‡
Department of Computer Science
St. Cloud State University
St. Cloud, MN 56301

Phone: (612)654–9873 Fax: (612)654–9912

Email: `rwj@eeyore.stcloud.msus.edu`

Abstract[1]— This paper presents a divide and conquer algorithm to compute the Fourier transform of a finite Abelian group using the Fourier transform of an arbitrary subgroup and the Fourier transform of the corresponding quotient group. The construction uses an arbitrary choice of coset representatives for the quotient group. Different choices of coset representatives lead to different data flows in the algorithm and different twiddle factors.

The derivation of the algorithm generalizes the derivation of the FFT presented by Cooley and Tukey. Moreover, it can be used to obtain explicit algorithms for multidimensional Fourier transforms. The algorithm presented in this paper generalizes all of the known Cooley-Tukey type algorithms for multidimensional Fourier transforms and provides new algorithms with alternative data flow patterns. Some preliminary experiments suggest that in hierarchical memory computers these algorithms are more efficient than the standard "row-column" approach.

SP EDICS SP 4.1.3 — *Multidimensional Signal Processing*
Theory, Algorithms, and Systems - Transforms

---

*Supported in part by Advanced Research Projects Agency ARPA Order No. 6674, monitored by AFOSR under contract No. F49620-89-C-0020.

†Supported in part by NSF grant CCR-9211016.

‡Supported in part by Advanced Research Projects Agency ARPA Order No. 7898, monitored by NIST under contract No. 60NANB1D1151.

[1]Permission to publish this abstract separately is granted.

1 Introduction

In 1965, Cooley and Tukey [11] presented a divide and conquer algorithm, called the Fast Fourier Transform (FFT), for computing the Fourier transform of a sequence of n points. The FFT has a long and interesting history [10].

Cooley and Tukey considered the problem of computing

$$X(j) = \sum_{k=0}^{N-1} A(k)W^{jk}, \; j = 0, 1, \ldots, N-1, \; W = e^{2\pi i/N} \tag{1}$$

given a complex function $A(k)$.

Their algorithm is obtained by viewing the functions X and A as functions of two variables and rearranging the sum in Equation 1 as a nested sum. Since the material in this paper is closely modeled on the ideas in the paper by Cooley and Tukey, we reproduce their derivation.

> To derive the algorithm, suppose N is composite, i.e., $N = r_1 r_2$. Then let the indices in (1) be expressed
>
> $$\begin{aligned} j &= j_1 r_1 + j_0, j_0 = 0, 1, \ldots, r_1 - 1, \; j_1 = 0, 1, \ldots, r_2 - 1, \\ k &= k_1 r_2 + k_0, k_0 = 0, 1, \ldots, r_2 - 1, \; k_1 = 0, 1, \ldots, r_1 - 1. \end{aligned} \tag{2}$$
>
> Then, one can write
>
> $$X(j_1, j_0) = \sum_{k_0} \sum_{k_1} A(k_1, k_0) W^{jk_1 r_2} W^{jk_0}. \tag{3}$$
>
> Since
>
> $$W^{jk_1 r_2} = W^{j_0 k_1 r_2}, \tag{4}$$
>
> the inner sum, over k_1, depends only on j_0 and k_0 and can be defined as a new array,
>
> $$A_1(j_0, k_0) = \sum_{k_1} A(k_1, k_0) W^{j_0 k_1 r_2}. \tag{5}$$
>
> The new result can then be written
>
> $$X(j_1, j_0) = \sum_{k_0} A_1(j_0, k_0) W^{(j_1 r_1 + j_0) k_0}. \tag{6}$$

Alternative algorithms have been presented when r_1 and r_2 are relatively prime [13]. The approach of Cooley and Tukey has been used to obtain multidimensional Fourier Transform Algorithms. The "vector radix" algorithm

was presented in [15, 23], and a more general multidimensional algorithm was independently presented in [4] and [22].

Recently the authors have generalized the Cooley-Tukey algorithm to apply to an arbitrary finite Abelian group [1]. This algorithm computes the Fourier Transform of a finite Abelian group using the Fourier transform of an arbitrary subgroup and the Fourier transform of the quotient group obtained by moding out by the chosen subgroup. The construction uses an arbitrary choice of coset representatives for the quotient group. Different choices of coset representatives lead to different data flow in the algorithm and different twiddle factors. These results generalize all of the known Cooley-Tukey type algorithms, including those mentioned above, and provides new algorithms with alternative data flow.

The results in [1] were obtained using the theory of induced representations; similar in spirit to recent work on algorithms for the computation of Fourier transforms of non-Abelian groups [8, 9, 12, 24]. In this paper an alternative combinatorial proof is presented. This proof generalizes the original proof used by Cooley and Tukey. The benefit of this approach is that it provides an explicit formula that can be used for constructing multidimensional Fourier Transform algorithms. A meta-algorithm is provided that, given a presentation for an Abelian group, a presentation for a subgroup, a set of coset representatives for the quotient group, and a set of coset representatives for a related quotient group in the dual group, constructs a matrix factorization of the Fourier transform of A. This matrix factorization provides an algorithm for computing the given multidimensional Fourier transform.

In Section 2 the Fourier transform of a finite Abelian group is defined. A proof of the main result is presented in Section 3. Section 4 shows how to apply the results about Abelian groups to multidimensional Fourier transforms, and Section 5 presents some concrete examples. In particular, it is shown how to obtain the existing algorithms as special cases of the general algorithm. In addition a class of examples are presented that could not have been computed with existing algorithms. Finally, Section 6 discusses the consequences of the results in this paper to the implementation of high performance multi-dimensional Fourier transform algorithms.

2 The Fourier Transform of a Finite Abelian Group

Let A be a finite Abelian group whose order is denoted by $|A|$ (the group operation will be denoted by $+$). The set of complex valued functions on A with inner product

$$(f,g) = \frac{1}{|A|}\sum_{a\in A} f(a)\overline{g(a)}$$

form an inner product space denoted by $L^2(A)$.

A non-zero function $\chi \in L^2(A)$ such that

$$\chi(a_1 + a_2) = \chi(a_1)\chi(a_2) \text{ for } a_1, a_2 \in A,$$

is called a character. If $a \in A$ is of order n, then $\chi(a)^n = 1$, and therefore $|\chi(a)| = 1$ and $\chi(-a) = \overline{\chi(a)}$.

If χ_1 and χ_2 are characters then we can define $(\chi_1 + \chi_2)(a) = \chi_1(a)\chi_2(a)$. Under this operation the set of characters form a group, called the dual or character group, which we will denote by $\hat{A}$. It is well known that $\hat{A}$ is isomorphic to A (see Section 4 and [14]).

Suppose that $\chi \not\equiv 1$, then there exists $a_1 \in A$ such that $\chi(a_1) \neq 1$. Since,

$$\sum_{a\subset A}\chi(a) = \sum_{a\subset A}\chi(a + a_1) = \chi(a_1)\sum_{a\subset A}\chi(a),$$

$(1-\chi(a_1))\sum_{a\in A}\chi(a) = 0$, and since $\chi(a_1) \neq 1$, $\sum_{a\in A}\chi(a) = 0$. The following property easily follows from this calculation.

Lemma 2.1 *Let* $\chi_1, \chi_2 \in \hat{A}$.

$$(\chi_1, \chi_2) = \begin{cases} 1 & \text{if } \chi_1 = \chi_2 \\ 0 & \text{if } \chi_1 \neq \chi_2 \end{cases}$$

This lemma implies that $\hat{A}$ is an orthonormal basis for $L^2(A)$. Hence, an arbitrary function, $f \in L^2(A)$ can be written uniquely as

$$f(x) = \sum_{\chi\in\hat{A}} \hat{f}(\chi)\chi(x), \text{ where } \hat{f}(\chi) = (f,\chi). \tag{7}$$

This is the Fourier series expansion of the function f. The coefficients in this expansion are called the Fourier coefficients and are obtained from the Fourier transform of f.

Definition 2.1 (Fourier Transform)
The Fourier transform of A, $F(A) : L^2(A) \to L^2(\hat{A})$ is defined by

$$F(A)(f)(\chi) = \hat{f}(\chi) = \frac{1}{|A|} \sum_{a \in A} f(a)\overline{\chi(a)} = (f, \chi).$$

3 A Divide and Conquer Algorithm

Let $\langle\, , \rangle : A \times \hat{A} \to \mathbf{C}$ be the bilinear pairing of A with its dual $\hat{A}$ defined by

$$\langle a, \hat{a} \rangle = \hat{a}(a).$$

Let $B < A$ be a subgroup of A. Corresponding to B we can form a subgroup, $B^{\perp}$, of $\hat{A}$ equal to $\{\hat{a} \in \hat{A} | \ \ \langle b, \hat{a} \rangle = 1, \text{ for } b \in B\}$, the characters that are perpendicular, with respect to $\langle , \rangle$, to the subgroup B.

Since $\hat{a} \in B^{\perp}$ is equal to 1 on B, we can identify it with a character on $C = A/B$, by setting $\hat{a}(a + B) = \hat{a}(a)$. Likewise given a character, $\hat{c}$, of A/B, we can obtain a character in $B^{\perp}$, by composing the projection $A \to A/B$ with $\hat{c}$. Therefore $B^{\perp}$ is isomorphic to $\hat{C}$.

A character, $\hat{a} \in \hat{A}$, restricted to the elements of B is a character of $\hat{B}$ denoted by $\hat{a}|_B$. The restriction map, $\hat{a} \mapsto \hat{a}|_B$ is a homomorphism from $\hat{A} \to \hat{B}$ with kernel $B^{\perp}$. Since $|B^{\perp}| = |A/B|$ and $|A| = |\hat{A}|$, $|\hat{A}/B^{\perp}| = |\hat{B}|$ and the restriction map is onto; therefore

$$\hat{A}/B^{\perp} \cong \hat{B}.$$

As a result of this isomorphism there are $|B^{\perp}| = |C|$ characters in $\hat{A}$ that restrict to each character in $\hat{B}$. Moreover, these are the characters in the cosets of $\hat{A}/B^{\perp}$.

Let $C = A/B$ and $\hat{B} = \hat{A}/B^{\perp}$. Let $\xi : C \to A$ and $\hat{\eta} : \hat{B} \to \hat{A}$ be a choices of coset representatives. The following theorem shows how to compute $F(A)$ using $|C|$ copies of $F(B)$, $|B|$ copies of $F(C)$ and $|A|$ complex multiplications. The complex multiplications depend on the choices of coset representatives ξ and $\hat{\eta}$.

To simplify the notation used in the following theorem and proof we will remove the normalization constant $1/|A|$ and conjugation in the definition of the Fourier transform and compute

$$F(A)(\hat{a}) = \sum_{a \in A} f(a)\langle a, \hat{a} \rangle.$$

There is no harm in doing this since the constants and conjugation can easily be reinserted.

Theorem 3.1 *Let $B < A$, $C = A/B$, $\hat{C} = B^{\perp}$, $\hat{B} = \hat{A}/\hat{C}$, $\xi : C \to A$ be a choice of coset representatives for A/B, and $\hat{\eta} : \hat{B} \to \hat{A}$ be a choice of coset representatives for $\hat{A}/\hat{C}$. Then*

$$\hat{f}(\hat{a}) = \sum_{c \in C} \langle c, \hat{c} \rangle \left(\langle \xi(c), \hat{\eta}(\hat{b}) \rangle \sum_{b \in B} f_{\xi(c)}(b) \langle b, \hat{b} \rangle \right).$$

where $f_{\xi(c)}(b) = f(b + \xi(c))$ and $\hat{a} \in \hat{A} = \hat{\eta}(\hat{b}) + \hat{c}$, with $\hat{b} \in \hat{B}$ and $\hat{c} \in \hat{C}$.

Proof. Using the coset decompositions of A/B and $\hat{A}/\hat{C}$ the indexing sets can be written as $A = B \times \xi(C)$ and $\hat{A} = \hat{C} \times \hat{\eta}(\hat{B})$, and

$$\begin{aligned} \hat{f}(\hat{a}) &= \sum_{a \in A} f(a) \langle a, \hat{a} \rangle & (8) \\ &= \sum_{c \in C} \sum_{b \in B} f_{\xi(c)}(b) \langle b + \xi(c), \hat{c} + \hat{\eta}(\hat{b}) \rangle. & (9) \end{aligned}$$

Using the bilinearity of $<,>$ this is equal to

$$\sum_{c \in C} \sum_{b \in B} f_{\xi(c)}(b) \ \langle b, \hat{c} \rangle \ \langle \xi(c), \hat{c} \rangle \ \langle b, \hat{\eta}(\hat{b}) \rangle \ \langle \xi(c), \hat{\eta}(\hat{b}) \rangle, \tag{10}$$

Since $\hat{c} \in B^{\perp}$, $< b, \hat{c} >= 1$ and Equation 10 is equal to

$$\sum_{c \in C} \sum_{b \in B} f_{\xi(c)}(b) \langle \xi(c), \hat{c} \rangle \langle b, \hat{\eta}(\hat{b}) \rangle \langle \xi(c), \hat{\eta}(\hat{b}) \rangle. \tag{11}$$

Furthermore, since $\langle \xi(c), c \rangle$ and $\langle \xi(c), \hat{\eta}(\hat{b}) \rangle$ do not depend on the inner summation index b, this is equal to the nested sum

$$\sum_{c \in C} \langle \xi(c), \hat{c} \rangle \left(\langle \xi(c), \hat{\eta}(\hat{b}) \rangle \sum_{b \in B} f_{\xi(c)}(b) \langle b, \hat{\eta}(\hat{b}) \rangle \right). \tag{12}$$

Finally, since $\langle \xi(c), \hat{c} \rangle$ and $\langle b, \hat{\eta}(\hat{b}) \rangle$ are independent of the choice of coset representatives $\xi(c)$ and $\hat{\eta}(\hat{b})$ this completes the proof. ∎

This theorem is the basis for a divide and conquer algorithm for computing $F(A)$, which we now describe.

1. Compute $\hat{f}_{\xi(c)} = F(B) f_{\xi(c)}$ for $c \in C$.

2. Compute $g_{\hat{\eta}(\hat{b})}$ for $\hat{b} \in \hat{B}$, where $g_{\hat{\eta}(\hat{b})}(c) = \langle \xi(c), \hat{\eta}(\hat{b}) \rangle \hat{f}_{\xi(c)}(\hat{b})$ for $c \in C$.

3. Compute $\hat{g}_{\hat{\eta}(\hat{b})} = F(C) g_{\hat{\eta}(\hat{b})}$ for $\hat{b} \in \hat{B}$.

Observe that $\hat{f}_{\xi(c)} \in L^2(\hat{B})$, $g_{\hat{\eta}(\hat{b})} \in L^2(C)$, $\hat{g}_{\hat{\eta}(\hat{b})} \in L^2(\hat{C})$, and by Theorem 3.1 $\hat{g}_{\hat{\eta}(\hat{b})}(\hat{c}) = \hat{f}(\hat{\eta}(\hat{b}) + \hat{c})$. Since any $\hat{a}$ can be written as $\hat{\eta}(\hat{b}) + \hat{c}$ for some $\hat{b} \in \hat{B}$ and $\hat{c} \in \hat{C}$, all of the values of the function $\hat{f} = F(A) f$ have been computed.

To implement this algorithm the elements of A and $\hat{A}$ must be ordered. Ordering the elements of A and $\hat{A}$ fixes the representations of the functions f and $\hat{f}$, introducing data permutations or index computations when accessing the values of $f_{\xi(c)}$ and $g_{\hat{\eta}(\hat{b})}$, and storing the values of $\hat{g}_{\hat{\eta}(\hat{b})}$.

If the elements of A are ordered then f can be viewed as a vector whose indices are the elements of A. Step (1) of the above algorithm first creates $|C|$ functions of B denoted by $f_{\xi(c)}$. If the elements of B are ordered then the functions $f_{\xi(c)}$ can be represented by subvectors obtained from f. The subvectors $f_{\xi(c)}$ can be ordered using the order of the coset representatives $\xi(C)$. Therefore the first part of Step (1) corresponds to a permutation of the vector f. This permutation is determined by the order of the elements of A, the subgroup B, and the coset representatives $\xi(C)$. It corresponds to permuting the indexing set A to $\xi(C) \times B$.

After the vectors $f_{\xi(c)}$ are gathered, $\hat{f}_{\xi(c)} = F(B) f_{\xi(c)}$ is computed for each $c \in C$. Step (2) then creates $|\hat{B}| = |B|$ functions of C. For each $\hat{b} \in \hat{B}$ the function $g_{\hat{\eta}(\hat{b})}$ defines a function of C. If the functions $\hat{f}_{\xi(c)}$ are stored as vectors the computation of $g_{\hat{\eta}(\hat{b})}$ requires a stride permutation followed by a diagonal multiplication (called the "twiddle factor"). The stride permutation gathers the elements of each vector $\hat{f}_{\xi(c)}$ indexed by $\hat{b}$. The twiddle factor multiplies $\hat{f}_{\xi(c)}(\hat{b})$ by $\langle \xi(c), \hat{\eta}(\hat{b}) \rangle$.

Finally, Step (3) performs $|B|$ computations of the Fourier transform on C, $\hat{g}_{\hat{\eta}(\hat{b})} = F(C) g_{\hat{\eta}(\hat{b})}$. As mentioned above, the resulting functions combine to give all of the values of $\hat{f}$; however, the values are not necessarily in the order specified for $\hat{A}$. Instead, the functions are indexed by $\hat{\eta}(\hat{B}) \times \hat{C}$, and a permutation similar to the one in Step (1) is required so that the resulting function is ordered corresponding to the order of $\hat{A}$.

The Fourier transform $F(A)$ is a linear transformation from $L^2(A)$ onto $L^2(\hat{A})$. As soon as the elements of A and $\hat{A}$ are ordered, the linear computation can be written as a matrix. The above algorithm can be summarized by the following matrix factorization. The matrix factorization uses the notation $I_n \otimes M$, where I_n is the $n \times n$ identity matrix, and $I_n \otimes M$ denotes the direct sum of n copies of the matrix M. The symbol $\otimes$ denotes the tensor product (also called the Kronecker product), and in general, if A is an $m \times n$ matrix and B is a $p \times q$ matrix then $A \otimes B$ is the $mp \times nq$ block matrix obtained by replacing a_{ij} by the matrix $a_{ij}B$ for $0 \leq i < m$ and $0 \leq j < n$ (see [16]). In addition, the permutation $L_n^{mn} : in+j \to jm+i, \;\; 0 \leq i < m, 0 \leq j < n$, will be used. This permutation, called a stride permutation, gathers the elements of a vector of size mn into n segments of m elements. The elements of each segment are gathered at stride n, and the i-th segment begins with the i-th element of the input vector.

Corollary 3.2 *Let $B < A$, $C = A/B$, $\hat{C} = B^{\perp}$, $\hat{B} = \hat{A}/\hat{C}$, $\xi : C \to A$ be a choice of coset representatives for A/B, and $\hat{\eta} : \hat{B} \to \hat{A}$ be a choice of coset representatives for $\hat{A}/\hat{C}$. Let $P(\xi)$ be the permutation that reorders the A corresponding to $\xi(C) \times B$, $Q(\hat{\eta})$ be the permutation that reorders $\hat{A}$ corresponding $\hat{\eta}(\hat{B}) \times \hat{C}$. Let $T(\hat{\eta}, \xi)(\hat{a}, a) = \langle \xi(c), \hat{\eta}(\hat{b})$, where $a = \xi(c) + b$ and $\hat{a} = \hat{\eta}(\hat{b}) + \hat{c}$. Then*

$$T(\hat{\eta}, \xi)(\hat{a}, a) = \bigoplus_{c \in C} \bigoplus_{\hat{b} \in \hat{B}} \langle \xi(c), \hat{\eta}(\hat{b}) \rangle,$$

and

$$F(A) = Q(\hat{\eta})^{-1} \left(I_{|B|} \otimes F(C) \right) T(\hat{\eta}, \xi) L_{|B|}^{|A|} \left(I_{|C|} \otimes F(B) \right) P(\xi).$$

The Cooley-Tukey algorithm has been described as a matrix factorization by many authors (see the books by Van Loan [27] or Tolimieri, An and Lu [25, 26]) for a comprehensive survey. Corollary 3.2 provides a generalization of many of these results.

4 Application to Multidimensional Fourier Transforms

By the fundamental theorem of Abelian groups [17], any finite Abelian group is isomorphic to a direct product of cyclic groups, and hence is isomorphic to

$\mathbf{Z}/n_1\mathbf{Z} \times \cdots \times \mathbf{Z}/n_t\mathbf{Z}$, where $\mathbf{Z}/n_i\mathbf{Z}$ is the additive group of integers mod n. Such an isomorphism is called a presentation. Choice of presentation is not unique; however, it is possible to determine all possible presentations from the prime power factorization of the orders of the cyclic factors of any given presentation.

Suppose $\mathbf{Z}/n_1\mathbf{Z} \times \cdots \times \mathbf{Z}/n_t\mathbf{Z}$ is a presentation for A, and let $a = (a_1, \ldots, a_t)$ and $x = (x_1, \ldots, x_t)$ be arbitrary elements in A. Define $a \mapsto \hat{a}$ by

$$\hat{a}(x) = \hat{a}_1(x_1) \cdots \hat{a}_t(x_t),$$

where

$$\hat{a}_j(x_j) = e^{\frac{2\pi i}{n_j} a_j x_j}.$$

It is easy to verify that $\hat{a} \in \hat{A}$ and the mapping $a \to \hat{a}$ is an isomorphism from A onto $\hat{A}$.

Using this presentation and the corresponding isomorphism, the definition of $F(A)$ becomes:

$$\begin{aligned} \hat{f}(\hat{a}) &= \hat{f}(\hat{a_1}, \ldots, \hat{a_t}) \\ &= \sum_{(x_1,\ldots,x_t)} f(x_1, \ldots, x_t) e^{\frac{2\pi i}{n_1} a_1 x_1 + \cdots + \frac{2\pi i}{n_t} a_t x_t}. \end{aligned}$$

This is the standard definition of the t-dimensional Fourier transform of $n_1 \times \cdots \times n_t$ points, which we denote by $F_{(n_1,\ldots,n_t)}$.

In the application of Theorem 3.1 to the computation of $F_{(n_1,\ldots,n_t)}$, the Abelian group $A = \mathbf{Z}/n_1\mathbf{Z} \times \cdots \times \mathbf{Z}/n_t\mathbf{Z}$ and its dual $\hat{A} = \widehat{\mathbf{Z}/n_1\mathbf{Z}} \times \cdots \times \widehat{\mathbf{Z}/n_t\mathbf{Z}}$. The algorithm in Corollary 3.2 can be applied as soon as the following information has been obtained:

1. a basis for a subgroup $B < A$.
2. a set of coset representatives for $C = A/B$.
3. a basis for $B^{\perp}$.
4. a set of coset representatives for $\hat{B} = \hat{A}/B^{\perp}$.

A classification of all subgroups of a finite Abelian group is given in [7]. If only a generating set is given for B, a basis can be algorithmically computed

using a constructive version of the fundamental theorem of Abelian groups based on the Smith normal form of an integer matrix [17, 20]. A set of coset representatives for Step 2 can be computed using an algorithm to compute the Smith normal form of the relation matrix for A/B obtained by setting the generators for B equal to zero. The Smith normal form computation of the relation matrix determines a presentation for A/B.

The group $B^{\perp}$ can be computed by solving a system of linear equations mod N, where N is the least common multiple of $n_1, \ldots, n_t$. If $\omega_n = e^{\frac{2\pi i}{n}}$, then $\langle b, \hat{a} \rangle = 1$ implies

$$\omega_{n_1}^{b_1 a_1} \cdots \omega_{n_t}^{b_t a_t} = 1,$$

which is true if and only if

$$b_1 a_1 N/n_1 + \cdots + b_t a_t N/n_t \equiv 0 \pmod{N}.$$

Therefore, if $\{b_1, \ldots, b_r\}$, with $b_i = (b_{i1}, \ldots, b_{it})$, is a basis for B, then the elements of $B^{\perp}$ are the solutions, $0 \leq x_1 < n_1, \ldots 0 \leq x_t < n_t$ of the linear system

$$\begin{array}{lll} b_{11}N/n_1 x_1 + & \cdots & + b_{1t} x_t N/n_t \equiv 0 \pmod{N} \\ & \cdots & \\ b_{i1}N/n_1 x_1 + & \cdots & + b_{it} x_t N/n_t \equiv 0 \pmod{N} \\ & \cdots & \\ b_{r1}N/n_1 x_1 + & \cdots & + b_{rt} x_t N/n_t \equiv 0 \pmod{N}. \end{array}$$

Once a basis has been computed for $B^{\perp}$ coset representatives of $\hat{A}/B^{\perp}$ can be computed in the same way that coset representatives for A/B were determined in Step (2).

5 Examples

In this section, Corollary 3.2 is applied to derive matrix factorizations for several one-dimensional and two-dimensional Fourier transforms. These examples are first used to derive many of the existing algorithms as special cases. Section 5.1 contains a derivation of the algorithm presented in the paper of Cooley and Tukey [11] and the prime factor algorithm of Good and Thomas [13]. Section 5.2 contains a derivation of the standard row column

algorithm and the vector-radix algorithm for computing multidimensional Fourier transforms. Finally, Section 5.3 contains an example that can not be derived using existing methods. This example best illustrates the general techniques, described in Section 4, for using Corollary 3.2 to derive multidimensional Fourier transform algorithms.

5.1 Cyclic Groups

Let $A = \mathbf{Z}/mn\mathbf{Z}$, $B = \{0, m, 2m, \ldots, (n-1)m\}$. Then $B \cong \mathbf{Z}/n\mathbf{Z}$ and $A/B \cong \mathbf{Z}/m\mathbf{Z}$, and $\{0, 1, \ldots, m-1\}$ can be chosen as coset representatives (i.e. $\xi(j) = j$, for $0 \leq j < m$). An element $\hat{a} \in \hat{A} = \widehat{\mathbf{Z}/mn\mathbf{Z}}$ is in $B^{\perp}$ if and only if $\omega_{mn}^{am} = 1$. This in turn is true if and only if $am \equiv 0 \pmod{mn}$. Therefore, using the notation of Section 4, $B^{\perp} = \{\hat{0}, \hat{n}, \widehat{2n}, \ldots, \widehat{(m-1)n}\}$, which is isomorphic to $\mathbf{Z}/m\mathbf{Z}$. The quotient group $\hat{A}/B^{\perp} \cong \mathbf{Z}/n\mathbf{Z}$, and $\{\hat{0}, \hat{1}, \ldots, \widehat{n-1}\}$ can be chosen as coset representatives (i.e. $\hat{\eta}(\hat{k}) = \hat{k}$, for $0 \leq k < n$).

With these choices, Theorem 3.1 is identical to the algorithm derived by Cooley and Tukey (see the Introduction). Corollary 3.2, can be used to state the resulting algorithm as a matrix factorization.

The permutations are $P(\xi) = L_m^{mn}$, $Q(\hat{\eta}) = L_n^{mn}$, and $L_{|B|}^{|A|} = L_n^{mn}$, and since $\langle \xi(j), \hat{\eta}(\hat{k}) \rangle = \langle j, \hat{k} \rangle = \omega_{mn}^{jk}$,

$$T(\hat{\eta}, \xi)(\hat{a}, a) = \bigoplus_{c \in C} \bigoplus_{\hat{b} \in \hat{B}} \langle \xi(c), \hat{\eta}(\hat{b}) \rangle = \bigoplus_{j=0}^{m-1} \bigoplus_{k=0}^{n-1} \omega_{mn}^{jk} = T_m^{mn}.$$

With these choices Corollary 3.2 becomes

$$F_{mn} = L_m^{mn}(I_n \otimes F_m) T_m^{mn} L_n^{mn} (I_m \otimes F_n) L_m^{mn}. \tag{13}$$

See [18] or either of the books [25, 27] for alternative derivations of this matrix factorization.

If we assume that m and n are relatively prime, it is possible to chose coset representatives so that all of the twiddle factors are equal to one. Since m and n are relatively prime there exist integers λ and μ such that $\lambda m + \mu n = 1$. If we set $e_m = \mu n$ and $e_n = \lambda n$ then we observe that

$$\begin{aligned} e_m &\equiv 1 \pmod{m} \qquad & e_m &\equiv 0 \pmod{n} \\ e_n &\equiv 0 \pmod{m} \qquad & e_n &\equiv 1 \pmod{n}. \end{aligned}$$

Consequently, e_m and e_n are a pair of complementary orthogonal idempotents for $\mathbf{Z}/mn\mathbf{Z}$, i.e., $e_m^2 = e_m$, $e_n^2 = e_n$, $e_m e_n = 0$, and $e_m + e_n = 1$.

Since $e_m e_n \equiv 0 \pmod{mn}$, $\langle e_m, \hat{e}_n \rangle = \omega_{mn}^{e_m e_n} = 1$, and therefore choosing $\xi(j) = je_m$ and $\hat{\eta}(\hat{k}) = k\hat{e}_n$, causes $T(\hat{\eta}, \xi) = I_{mn}$. Let $P(m,n) = P(\xi) : im + je_m \mapsto jn + i, 0 \leq i < n, 0 \leq jm$, and $P(n,m) = Q(\hat{\eta}) : in + je_n \mapsto jm + i, 0 \leq i < m, 0 \leq jn$, then Corollary 3.2 implies

$$F_{mn} = P(n,m)^{-1}(I_n \otimes F_m)L_n^{mn}(I_m \otimes F_n)P(m,n). \tag{14}$$

5.2 Coherent Presentations

Given a presentation for an Abelian group, A, a subgroup is *coherent* with respect to the presentation if it has a basis whose elements are multiples of the basis elements in the presentation of A.

A special case of a coherent subgroup is obtained when the subgroup is equal to one of the summands in the presentation of A. For example, let $A = \mathbf{Z}/n_1\mathbf{Z} \times \mathbf{Z}/n_2\mathbf{Z}$ and let $B = \{(j,0)|\ 0 \leq j < n_1\}$. Then $B \cong \mathbf{Z}/n_1\mathbf{Z}$, $A/B \cong \mathbf{Z}/n_2\mathbf{Z}$, $B^\perp = \{(0,\hat{k})|0 \leq k < n_2\} \cong \mathbf{Z}/n_2\mathbf{Z}$, and $\hat{A}/B^\perp \cong \mathbf{Z}/n_1\mathbf{Z}$. In this example we may choose $\{(0,k)|\ 0 \leq k < n_2\}$ as coset representatives for A/B, and $\{(\hat{j},0)|\ 0 \leq j < n_1\}$ as coset representatives for $\hat{A}/B^\perp$.

Assuming that A and $\hat{A}$ are ordered lexicographically then in this example, $P(\xi) = L_{n_2}^{n_1 n_2}$, $Q(\hat{\eta}) = I_{n_1 n_2}$, $T(\hat{\eta}, \xi) = I_{n_1 n_2}$, and

$$F_{(n_1,n_2)} = (I_{n_1} \otimes F_{n_2})L_{n_1}^{n_1 n_2}(I_{n_2} \otimes F_{n_1})L_{n_2}^{n_1 n_2}. \tag{15}$$

The general case of a coherent subgroup can be illustrated with the following example. If $A = \mathbf{Z}/m_1 n_1\mathbf{Z} \times \mathbf{Z}/m_2 n_2\mathbf{Z}$ then $B = \{(j_1 m_1, j_2 m_2)|\ 0 \leq j_1 < n_1,\ 0 \leq j_2 < n_2\}$ is coherent.

$B \cong \mathbf{Z}/n_1\mathbf{Z} \times \mathbf{Z}/n_2\mathbf{Z}$ and $A/B \cong \mathbf{Z}/m_1\mathbf{Z} \times \mathbf{Z}/m_2\mathbf{Z}$. Moreover, $\{(j,k)|\ 0 \leq j < m_1, 0 \leq k \leq m_2\}$ can be chosen as coset representatives (i.e. $\xi(j,k) = (j,k)$).

$B^\perp = \{(\widehat{k_1 n_1}, \widehat{k_2 n_2}),\ 0 \leq k_1 < m_1,\ 0 \leq k_2 < m_2\}$, which is isomorphic to $\mathbf{Z}/m_1\mathbf{Z} \times \mathbf{Z}/n_2\mathbf{Z}$. $\hat{A}/B^\perp \cong \mathbf{Z}/n_1\mathbf{Z} \times \mathbf{Z}/n_2\mathbf{Z}$, and $\{(\hat{j},\hat{k})|\ 0 \leq j < n_1, 0 \leq k \leq n_2\}$ can be chosen as coset representatives (i.e. $\hat{\eta}(\hat{j},\hat{k}) = (\hat{j},\hat{k})$).

Assuming A and $\hat{A}$ along with B and $B^\perp$ and the coset representatives for A/B and $\hat{A}/B^\perp$ are ordered lexicographically. $P(\xi) - P$ is the permutation that gathers $m_1 m_2$ submatrices of dimension $n_1 \times n_2$. The elements of each submatrix are gathered at stride m_2 along the rows and stride m_1 along

the columns. $Q(\hat{\eta}) = Q$ is a similar block permutation. The twiddle factor $T = T(\hat{\eta}, \xi)$ is defined by

$$\langle (j_1, k_1), (\hat{j}_2, \hat{k}_2) \rangle = \omega_{m_1 n_1}^{j_1 j_2} \omega_{m_2 n_2}^{k_1 k_2}$$

for $0 \leq j_1 < m_1, 0 \leq k_1 < n_1, 0 \leq j_2 < m_2, 0 \leq k_2 < n_2$. Corollary 3.2 then implies the following factorization:

$$F_{(m_1 n_1, m_2 n_2)} = Q^{-1}(I_{n_1 n_2} \otimes F_{(m_1, m_2)}) T L_{n_1 n_2}^{m_1 n_1 m_2 n_2} (I_{m_1 m_2} \otimes F_{(n_1, n_2)}) P. \quad (16)$$

This is a variant of the vector radix algorithm mentioned in the introduction.

In the special case when $m_1 = n_1$ and $m_2 = n_2$ then $P = Q$, therefore the only essential permutation (conjugating by P just corresponds to relabeling the data) is the stride permutation in the middle. Comparing this to the algorithm in Equation 15 we see that, at the cost of T, we have eliminated one of the runtime permutations $L_{mn}^{m^2 n^2}$.

5.3 Incoherent Presentations

Given a subgroup $B < A$ it is not always possible to choose a presentation for A such that B is coherent with respect to that presentation. The following example from exercise 5 in Chapter 3 of "The Theory of Groups" by M. Hall [14] illustrates this situation. Let $A = \mathbf{Z}/p^3\mathbf{Z} \times \mathbf{Z}/p\mathbf{Z}$, where p is a prime number, and let B be the cyclic subgroup of order p^2 generated by $= (p, 1)$. Then the following calculation shows that A/B is isomorphic to $\mathbf{Z}/p^2\mathbf{Z}$. Clearly, if there were a presentation of A relative to which B were coherent, then A/B would be isomorphic to $\mathbf{Z}/p\mathbf{Z} \times \mathbf{Z}/p\mathbf{Z}$.

Indeed, a relation matrix R for A/B is obtained by adjoining the generator for B to the relations for A.

$$R = \begin{pmatrix} p^3 & 0 \\ 0 & p \\ p & 1 \end{pmatrix}.$$

Since the first row is equal to p^2 times the third row minus p times the second row it is a redundant relation. The resulting relations can be unimodularly transformed (i.e. by pre or post multiplying by integer matrices whose determinant is plus or minus one) into a diagonal matrix.

$$\begin{pmatrix} p^2 & 0 \\ 0 & 1 \end{pmatrix} = PRQ = \begin{pmatrix} -1 & p \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & p \\ p & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ -p & 1 \end{pmatrix}.$$

The column operations corresponding to Q require the linear change of variables on the basis of A: $(p,1) \mapsto (p,1)$ and $(0,1) \mapsto p(p,1)+(0,1) = (p^2, p+1)$. From this calculation we conclude that $A/B \cong \mathbf{Z}/p^2\mathbf{Z}$ and $\{j(p,1) | 0 \le j < p^2\}$ can be chosen as coset representatives.

Since $B^\perp \cong \widehat{A/B}$, $B^\perp$ is cyclic of order p^2. An element $(\hat{x}, \hat{y}) \in \hat{A}$ is in $B^\perp$ if and only if

$$\omega_{p^3}^{px}\omega_p^y = 1 \Longleftrightarrow px + p^2 y \equiv 0 \pmod{p^3}.$$

It is easy to verify that $(p^2-p, 1)$ is a solution, and moreover, it is a cyclic generator for all of the solutions (the order of $(p^2-p,1)$ is p^2 since $p(p^2-p,1) \neq 0$).

Another unimodular diagonalization shows that $\{k(\hat{p}, \hat{1}) | 0 \le k < p^2\}$ can be chosen as coset representatives for $\hat{A}/B^\perp$.

The permutation $P(\xi) = Q(\hat{\eta}) = P$ is the permutation $j(p+1)+k \mapsto k * p^2 + j$. The twiddle factor, $T = T(\hat{\eta}, \xi)$ is defined by

$$\langle j(p,1), k(\hat{p}, \hat{1}) \rangle = \omega_{p^3}^{p^2 jk}\omega_p^{jk}$$

and Corollary 3.2 implies the following factorization.

$$F_{(p^3,p)} = P^{-1}(I_{p^2} \otimes F_{p^2}) T L_{p^2}^{p^4} (I_{p^2} \otimes F_{p^2}) P. \tag{17}$$

6 Implementation Considerations

In this section, we outline some of the potential benefits of the results presented in this paper for implementing multidimensional Fourier transforms.

The goal of the implementer is to construct a program to efficiently evaluate the linear computation

$$y = Fx,$$

where F is generally a one, two, or three dimensional Fourier transform. The approach we have suggested, is to apply repeatedly the construction in Corollary 3.2 to factor F into small Fourier transforms that we may assume are efficiently implemented. Along the way we will pick up tensor products, diagonal multiplications (twiddle factors), and, most importantly, permutations. The code for each of the resulting factors can be combined to give a program

for the computation. The resulting formula can also be algebraically manipulated to produce many different algorithms with different performance characteristics. For a detailed discussion of how this code can be generated see [18] and [19].

One of the main features of this paper is to show how, possibly at the cost of non-standard twiddle factors, the class of resulting permutations may be enlarged from the standard approach. The performance bottleneck for implementations of the FFT for large data sets on modern computer architectures is the data flow [5]. These new permutations may enable us to find better implementations of the Fourier transform. This is especially true in the multidimensional case. To see why this is true, consider the choices the programmer has in applying the factorization procedure. In one dimension, the presentation of $B < A$ and hence, A/B is essentially unique. This is because of the fact that a cyclic group has a unique subgroup of a given order. So in one dimension, once the size of the smaller Fourier transform is fixed, the only free choice is the set of coset representatives. In two dimensions, the situation is considerably more complex and fruitful. Given A there are many non-isomorphic B's to choose of a given size. Furthermore, even if we choose isomorphic B's it can happen that the resulting quotients C's are *not* isomorphic. This together with the choice of coset representatives, gives the algorithm designer considerable flexibility in matching an algorithm to a specific machine to obtain a high performance implementation.

We report the results of an experiment we conducted which suggest that these ideas may have a practical value in implementing multidimensional Fourier transforms. Consider the problem of implementing a square two-dimensional Fourier transform $F_{N,N}$ with $N = 2^n$. Typically an algorithm based on Equation 15 is used. In this case the resulting factorization is

$$y = (I_N \otimes F_N) L_N^{N^2} (I_N \otimes F_N) L_N^{N^2} x$$

where the load-stride at stride N, $L_N^{N^2}$, corresponds to matrix transposition. This approach is called the row-column algorithm. Assuming that the initial data is stored in a matrix which is in row-major order, the algorithm converts the data to column-major order, applies F_N to the N columns of data, transposes the intermediate result back to row-major order, and then applies F_N to the N rows of intermediate data. Presuming that an efficient implementation of F_N is available, this reduces the problem to finding an efficient implementation of transposition.

Stride s	Time (secs)
1 (copy)	173
2	197
4	259
16	774
64	2819
4096 (transposition)	>72000

Table 1: Wall clock timings of Load-Stride.

On machines with a hierarchical or distributed memory the behavior of load-stride permutations for large data sizes varies with the stride. Our test machine was a Sun Sparc 10 model 41 running under Solaris 1.0 with the following memory hierarchy

$16-\mathrm{KB}$	on-chip cache
$1-\mathrm{MB}$	on-board cache
$64-\mathrm{MB}$	main memory
$400-\mathrm{MB}$	swapfile

For complex data (8 bytes per point) we obtained wall-clock timings for a straight forward implementation of $y \leftarrow L_s^{2^{24}} x$. These are shown in Table 1. Thus, our row-column evaluation of $F_{2^{12},2^{12}}$ takes more than 72,000 seconds!

One approach to handling the situation is to use a multipass algorithm to do the transposition [21]. In fact, a simple one would be to do $L_4^{2^{24}}$ six times based on the factorization $L_{st}^{rst} = L_s^{rst} L_t^{rst}$. However, the following observations suggest a faster implementation might be obtained by the methods we have developed in this paper.

Instead of building $F_{N,N}$ from one-dimensional transforms, we can use, for example, Equation 16 to build $F_{N,N}$ from smaller two-dimensional transforms. In this case we have

$$y = Q(I_4 \otimes F_{(N/2,N/2)}) T L_4^{N^2} (I_{N/4} \otimes F_{(2,2)}) P,$$

where P and Q are block permutations. Now for $N = 2^{24}$ we have

$$y = Q(I_4 \otimes F_{(2^{11},2^{11})}) T L_4^{2^{24}} (I_{2^{22}} \otimes F_{(2,2)}) P.$$

Factor	Time (secs)
Q	437
$I_4 \otimes F_{(2^{11},2^{11})}$	198
$TL_4^{2^{24}}$	259
$I_{2^{22}} \otimes F_{(2,2)}$	185
P	399

Table 2: Wall clock timings of 2-dimensional Cooley-Tukey.

For this case we have obtained timings shown in Table 2. Reference to the table shows that this two-dimensional Cooley-Tukey algorithm costs at worst about 1500 seconds which is very much less than a straight-forward row-column evaluation at more than 72,000 seconds.

This experiment was only a test of quick and dirty code, but we believe it strongly suggests that the methods developed in this paper warrant further practical study. The wide variety of new data flows introduced by the factorization procedure we have developed in this paper may have an implementation advantage over those of traditional multidimensional FFT's on modern computer architectures.

References

[1] L. Auslander, J. R. Johnson, and R. W. Johnson. Multidimensional Cooley-Tukey algorithms revisited. 1994. Preprint.

[2] L. Auslander, J. R. Johnson, and R. W. Johnson. Computing finite Fourier transforms exploiting group symmetries: Part I — the role of inner products. 1993. Preprint.

[3] L. Auslander, J. R. Johnson, and R. W. Johnson. Computing finite Fourier transforms exploiting group symmetries: Part II — orbital evaluation. 1993. Preprint.

[4] L. Auslander, R. Tolimieri, and S. Winograd. Hecke's theorem in quadratic reciprocity, finite nilpotent groups and the cooley-tukey algorithm. *Adv. in Math.*, 43(2):122–172, February 1982.

[5] D. H. Bailey. FFT's in external or hierarchical memory. *J. Supercomp.*, 4:23–35, 1990.

[6] T. Beth. *Vehrfahren der Schneller Fourier Transform.* Teubner, Stuttgart, 1984.

[7] G. Birkhoff. Subgroups of abelian groups. *Proc. London Math. Soc.*, 38(2):385–401, 1934.

[8] M. Clausen. Fast Fourier transforms for metabelian groups. *SIAM J. Comput.*, 18(3):584–593, 1989.

[9] M. Clausen and U. Baum. *Fast Fourier Transforms.* Wissshaftsverlag, Manheim, Germany, 1993.

[10] J. W. Cooley. The re-discovery of the fast Fourier transform algorithm. *Mikrochim. Acta*, 3:33–45, 1987.

[11] J. W. Cooley and J. W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Math. Comp.*, 19(90):297–301, April 1965.

[12] P. Diaconis and D. Rockmore. Efficient computation of the Fourier transform on finite groups. *J. Amer. Math. Soc.*, 3(2):297–332, April 1990.

[13] I. J. Good. The interaction algorithm and practical Fourier analysis. *J. Roy. Statist. Soc. Ser. B*, 20(2):361–372, 1958.

[14] M. Hall. *The Theory of Groups*. The Macmillan Company, New York, 1959.

[15] D. B. Harris, J. H. McClellan, D. S. K. Chan, and H. W. Schuessler. Vector radix fast Fourier transform. In *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, pages 548–551, Hartford, CT, May 1977.

[16] Roger A. Horn and Charles R. Johnson. *Topics in Matrix Analysis*. Cambridge University Press, Cambridge, 1991.

[17] N. Jacobson. *Basic Algebra I*. W. H. Freeman and Company, San Francisco, 1974.

[18] J. R. Johnson, R. W. Johnson, D. Rodriguez, and R. Tolimieri. A methodology for designing, modifying, and implementing Fourier Transform algorithms on various architectures. *Circuits Systems Signal Process.*, 9(4):449–500, 1990.

[19] R. W. Johnson, C.-H. Huang, and J. R. Johnson. Multilinear algebra and parallel programming. *The Journal of Supercomputing*, 5:189–217, 1991.

[20] R. Kannan and A. Bachem. Polynomial algorithms for computing the Smith and Hermite normal forms of an integer matrix. *SIAM J. Computing*, 8(4):499–507, November 1979.

[21] S. D. Kaushik, C.-H. Huang, J. R. Johnson, R. W. Johnson, and P. Sadayappan. Efficient transposition algorithms for large matrices. In *Proceedings of Supercomputing '93*, pages 656–665, Portland, OR, November 21–23, 1993.

[22] R. M. Mersereau and T. C. Speake. A unified treatment of Cooley-Tukey algorithms for the evaluation of the multimensional DFT. *IEEE Trans. Acoust., Speech, Signal Processing*, ASSP-29(5):1011–1018, October 1981.

[23] G. E. Rivard. Direct fast Fourier transforms of bivariate functions. *IEEE Trans. Acoust., Speech, Signal Processing*, ASSP-25(3):250–252, June 1977.

[24] D. Rockmore. Fast Fourier analysis for abelian group extensions. *Adv. in Appl. Math.*, 11:164–204, 1990.

[25] R. Tolimieri, M. An, and C. Lu. *Algorithms for the Discrete Fourier Transform and Convolution.* Springer-Verlag, New York, 1989.

[26] R. Tolimieri, M. An, and C. Lu. *Mathematics of Multidimensional Fourier Transform Algorithms.* Springer-Verlag, New York, 1993.

[27] Charles Van Loan. *Computational Frameworks for the Fast Fourier Transform. Frontiers in Applied Mathematics*, Society for Industrial and Applied Mathematics, Philadelphia, 1992. Vol. 10.