Bachelor Mathematics

Bachelor thesis

Algorithms for Computing Fourier Transforms on Finite Groups

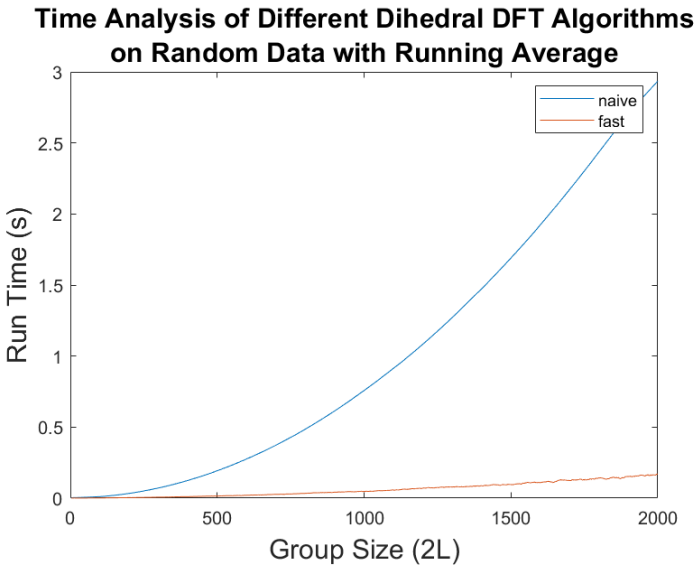
by

Madeleine Gignoux

July 1, 2023

Supervisor: dr. Thomas Rot

Second examiner: dr. Paulo Jorge de Andrade Serra



Department of Mathematics
Faculty of Sciences

VU UNIVERSITY AMSTERDAM

Abstract

This thesis constructs algorithms to compute Fast Fourier Transforms (FFTs) on several families of finite groups with arithmetic complexity in $\mathcal{O}(|G| \log |G|)$ operations. First, the Cyclic Discrete Fourier Transform (Cyclic DFT), which is a Fourier transform often found in literature for functions from $\mathbb{Z}/L\mathbb{Z}$ to $\mathbb{C}$ is introduced. Then, the representation theory of finite groups is used to provide a backbone to the Fourier theory used in this paper. A finite group Fourier transform for functions from G to $\mathbb{C}$—where G is a finite group—can then be defined, as well as a proof of the subsequent inverse transform, and some key theorems from Fourier analysis. The Cyclic DFT can then be related to the finite group Fourier transform, and we discuss two prominent Cyclic FFTs: The Mixed-Radix FFT and the Prime Factor Algorithm. Ultimately, FFTs for abelian, dihedral, and direct products of dihedral groups are written using the theory discussed, which have an arithmetic complexity in $\mathcal{O}(|G| \log |G|)$ operations.

Title: Algorithms for Computing Fourier Transforms on Finite Groups
Author: Madeleine Gignoux, m.gignoux@student.vu.nl, 2695118
Supervisor: dr. Thomas Rot
Second examiner: dr. Paulo Jorge de Andrade Serra
Date: July 1, 2023

Department of Mathematics
VU University Amsterdam
de Boelelaan 1081, 1081 HV Amsterdam
`http://www.math.vu.nl/`

Contents

1. Introduction

In his 19^{th}-century work on heat flow, Joseph Fourier claimed that *any* function can be written as a linear combination of sines and cosines [4]. Although this claim is wrong without additional conditions, it marks the beginning of Fourier analysis as a field of research. Nowadays, the specific work of Fourier is known as the Fourier series in which sufficiently nice periodic functions converge to a linear combination of sines and cosines.

The Fourier series can be calculated for all continuous signals, including sound waves. In the case of a sound wave, which can be represented by periodic functions of the signal's pressure over time, the Fourier series represents the signal as the amplitudes of its frequencies.

Reconstructing, compressing, and filtering signals is ubiquitous in modern technology, and as a result the Fourier series is an incredibly application-rich topic. However, when these signals are analyzed by a computer, the signal is digitized and discretized as the computer samples the signal. Therefore, instead of continuous signals, which use integration to find the Fourier series, summations are done. This resulting transformation is called the Cyclic Discrete Fourier Transform.

The Cyclic Discrete Fourier Transform (Cyclic DFT) is easy to work with because there are no issues of convergence when we reconstruct the function from its Fourier coefficients, which will be proven in Theorem 2.2. Thus, regardless of our function, the Cyclic Inverse Discrete Fourier Transform (Cyclic IDFT) always recovers the original input. However, to accurately represent the continuous signal, sampling theorems such as the Shannon-Nyquist sampling theorem set a lower bound for the number of samples necessary to accurately reconstruct the original signal [16].

Direct computation, which is colloquially referred to as naïve computation, of these Fourier coefficients is far too slow for modern demand, and faster algorithms called Cyclic Fast Fourier Transforms (Cyclic FFTs) are necessary. What makes the naïve implementation of the Cyclic DFT considered slow is that it has an arithmetic complexity in $\mathcal{O}(L^2)$ operations where L is the number of samples taken of the signal. In contrast, a Cyclic FFT invented by James Cooley and John Tukey known as the Mixed-Radix FFT [6], which is actually a modern rediscovery of work found in a posthumous publication of Gauss [10, 11], decreases the operational complexity to $\mathcal{O}(L \log L)$ for certain inputs. Besides this, the work of Irving Good called the Prime Factorization Algorithm (PFA) will also be covered [5].

In addition to mathematical methods of computing Cyclic FFTs, more computer science specific methods such as parallel computing, hard coding, lower-level programming, dynamic programming, and GPU computing can be done. While these are outside the scope of this paper, it would be remiss not to mention the work of Matteo Frigo and Steven Johnson known as the *Fastest Fourier Transform in the West* (FFTW) [2] which

is regarded as one of the fastest Cyclic FFTs [9], and used in MATLAB's built-in FFTW function [3].

The Cyclic DFT has theoretical underpinnings, originating from the representations of cyclic groups. Representation theory, which can be found in literature such as *Linear Representations of Finite Groups* by Jean-Pierre Serre [15], *Group Representations in Probability and Statistics* by Persi Diaconis [7], and *Representation Theory of Finite Groups* by Benjamin Steinberg [17] can be used to write a Fourier transform for functions from finite groups G to $\mathbb{C}$. Using this theory, we can prove the subsequent Fourier inversion theorem, as well as central theorems to Fourier analysis such as the Convolution Theorem and Plancherel's Theorem.

Finally, using the representation theory of cyclic and dihedral groups discussed in [15], we will write Fourier transforms for abelian groups, dihedral groups, and direct products of dihedral groups, along with fast algorithmic implementations of these transformations using our study of the aforementioned Cyclic FFTs.

In a 1991 paper titled Generalized FFTs - A Survey of Some Recent Results [12], it is conjectured whether any finite group Fourier transform can be computed in $\mathcal{O}(|G| \log |G|)$ arithmetic complexity where $|G|$ is the size of the group. This thesis aims to answer the question posed in this paper for the aforementioned families of finite groups.

Along with this paper, all algorithms discussed will be implemented in the numeric computing environment MATLAB Release 2023a with the code repository available at the clickable link here. To see an overview of the project, as well as necessary packages (referred to in MATLAB as Toolboxes), please click here.

2. The Discrete Fourier Transform

The most well-known finite group Fourier transform is the Fourier transform for cyclic groups. This is known in literature as the Discrete Fourier Transform (DFT) and for our purposes we will call this specific transform the Cyclic DFT as we also aim to study non-cyclic DFTs. This transform is defined as follows.

Definition 2.1 (The Cyclic Discrete Fourier Transform)**.** The Cyclic Discrete Fourier Transform (Cyclic DFT) of a complex valued function $f : \mathbb{Z}/L\mathbb{Z} \to \mathbb{C}$ is given by $\widehat{f} : \mathbb{Z}/L\mathbb{Z} \to \mathbb{C}$ where

$$\widehat{f}[n] = \frac{1}{L} \sum_{k=0}^{L-1} f[k] \omega_L^{-nk},$$

and $\omega_L = e^{\frac{2\pi i}{L}}$.

Accompanying every Fourier transform is its corresponding inverse Fourier transform. In a non-finite setting, the inverse Fourier transform converges to the original function under some conditions. However, in a finite setting any function $f : \mathbb{Z}/L\mathbb{Z} \to \mathbb{C}$ can be recovered by the Cyclic Inverse Discrete Fourier Transform.

Theorem 2.2 (The Cyclic Inverse Discrete Fourier Transform)**.** *The Cyclic Inverse Discrete Fourier Transform (Cyclic IDFT) is given by*

$$f[k] = \sum_{n=0}^{L-1} \widehat{f}[n] \omega_L^{nk}.$$

Proof. This can be seen by plugging in the formula for $\widehat{f}$ from Definition 2.1.

$$\begin{aligned} \sum_{n=0}^{L-1} \widehat{f}[n] \omega_L^{nk} &= \sum_{n=0}^{L-1} \left[\frac{1}{L} \sum_{m=0}^{L-1} f[m] \omega_L^{-nm} \right] \omega_L^{nk} \\ &= \frac{1}{L} \sum_{m=0}^{L-1} f[m] \sum_{n=0}^{L-1} \omega_L^{nk} \omega_L^{-nm} = \sum_{m=0}^{L-1} f[m] \delta_{km} = f[k] \end{aligned}$$

where δ_{km} is the Kronecker delta function and the proof that

$$\sum_{n=0}^{L-1} \omega_L^{nk} \omega_L^{-nm} = \begin{cases} L & \text{if } k = m, \\ 0 & \text{else,} \end{cases}$$

is excluded. This will be generalized in Theorem 3.13 where the orthogonality of representations is discussed. □

These specific transforms are central to this paper, and will show up in the context of non-cyclic groups. For now, we turn to defining a general finite group Fourier transform.

3. The Fourier Transform on Finite Groups

Our aim is to understand the Fourier transform of a function $f : G \to \mathbb{C}$ for finite groups G using representation theory. This uses group theory and linear algebra to provide a backbone for all finite group Fourier transforms, including the Cyclic DFT from Chapter 2. We will begin by explaining representations of finite groups, and then combine this with some linear algebra to derive a Fourier transform for finite groups and prove the subsequent Fourier inversion. Hereinafter, the group G will always denote a finite group.

3.1. Representation Theory

Representations are a way of understanding the structure of a group G as automorphisms of a vector space V. Thus, a representation is a combination of a vector space V and a function $\phi : G \to GL(V)$. As the name implies, they represent the group, meaning they maintain the structure of the group, i.e. ϕ is a homomorphism. They are defined as follows.

Definition 3.1 (Representation)**.** A representation of G is a pair (ϕ, V) where V is a finite dimensional vector space over the field $\mathbb{C}$, and ϕ is a homomorphism $\phi : G \to GL(V)$. The set of all representations of G is denoted $\mathrm{Rep}(G)$.

Definition 3.2 (Homomorphism)**.** Let $(\phi, V), (\mu, W) \in \mathrm{Rep}(G)$. A homomorphism from (ϕ, V) to (μ, W) is a linear map $T : V \to W$ such that $T\phi(g) = \mu(g)T$ for all $g \in G$. The set of homomorphisms from (ϕ, V) to (μ, W) is denoted $\mathrm{Hom}_G(\phi, \mu)$ where the vector spaces are left out for readability. If T is an isomorphism, then $T, T^{-1} \in \mathrm{Hom}_G(\phi, \mu)$.

Thinking about representations in such an abstract way is not necessary for our purposes. Using the isomorphism $GL(V) \cong GL_d(C)$ where $d = \dim(V)$, we define a matrix representation as follows.

Definition 3.3 (Matrix Representation)**.** A matrix representation $(\phi, \mathbb{C}^d)$ of G is a homomorphism $\phi : G \to GL_d(\mathbb{C})$. The set of all matrix representations of G is denoted $\mathrm{MatRep}(G)$.

As an example, when discussing the Cyclic DFT in Chapter 2, the group $\mathbb{Z}/L\mathbb{Z}$ had L one-dimensional matrix representations indexed by $n = 0, \ldots, L-1$ denoted $(\phi^n, \mathbb{C}) \in \mathrm{MatRep}(\mathbb{Z}/L\mathbb{Z})$ given by $\phi^n : \mathbb{Z}/L\mathbb{Z} \to GL_1(\mathbb{C}) \cong \mathbb{C}^*$ where $[k] \mapsto \omega_L^{nk}$. To show this is a matrix representation, it can be checked that ϕ^n is a homomorphism.

To relate representations to matrix representations, as well as relate representations to each other in general, there is the following definition of equivalence of representations.

Definition 3.4. Two representations $(\phi, V), (\mu, W) \in \text{Rep}(G)$ are equivalent if there exists an isomorphism T from (ϕ, V) to (μ, W). Equivalence is denoted $\phi \sim \mu$ where the respective vector spaces are again excluded for readability.

Theorem 3.5. *The equivalence given in Definition 3.4 is an equivalence relation.*

Proof. To show $\sim$ is an equivalence relation, we show (1) $\phi \sim \phi$, (2) $\phi \sim \mu$ if and only if $\mu \sim \phi$, and (3) if $\phi \sim \mu$ and $\mu \sim \rho$ then $\phi \sim \rho$.

1. Let $(\phi, V) \in \text{Rep}(G)$. Observe that the identity map I is an invertible linear map such that $I\phi(g) = \phi(g)I$ for all $g \in G$, so $\phi \sim \phi$.

2. Let $(\phi, V), (\mu, W) \in \text{Rep}(G)$. Moreover, let T be an isomorphism $T : V \to W$ such that $T\phi(g) = \mu(g)T$. Left and right multiplying by T^{-1} we have $T^{-1}T\phi(g)T^{-1} = T^{-1}\mu(g)TT^{-1}$ which reduces to $\phi(g)T^{-1} = T^{-1}\mu(g)$, so $\mu \sim \phi$. The other direction can be proved without loss of generality.

3. Let (ϕ, V), $(\mu, W), (\rho, U) \in \text{Rep}(G)$. Moreover, let T be an isomorphism $T : V \to W$ such that $T\phi(g) = \mu(g)T$ and R be an isomorphism $R : W \to U$ such that $R\mu(g) = \rho(g)R$. Then observe that

$$RT\phi(g) = R\mu(g)T = \rho(g)RT.$$

   Thus since the composition of two isomorphisms is an isomorphism, $P = RT$ is an isomorphism such that $P\phi(g) = \rho(g)P$, thus $\phi \sim \rho$.

□

Now, we can relate representations to matrix representations using Theorem 3.5 as follows.

Theorem 3.6. *Every representation $(\phi, V) \in Rep(G)$ is equivalent to a matrix representation $(\mu, \mathbb{C}^d) \in MatRep(G)$ where $d = dim(V)$.*

Proof. Given $(\phi, V) \in \text{Rep}(G)$ and a basis $(e_1, \ldots, e_d)$ of V, any element $v \in V$ can be written $v = \sum_{i=1}^{d} a_i e_i$. Thus the map $T : V \to \mathbb{C}^d$ where $T(v) = (a_1, \ldots, a_d)$ is an isomorphism of vector spaces. Now, let $\mu(g) = T\phi(g)T^{-1}$, then μ is a homomorphism since

$$\mu(g_1g_2) = T\phi(g_1g_2)T^{-1} = T\phi(g_1)\phi(g_2)T^{-1} = T\phi(g_1)T^{-1}T\phi(g_2)T^{-1} = \mu(g_1)\mu(g_2),$$

meaning $(\phi, \mathbb{C}^d) \in \text{MatRep}(G)$ and $\mu \sim \phi$ by construction. □

A group can have many different representations, as well as representations in different vector spaces. This leads to an idea called *reducibility*, finding representations that build all other representations. Defining reducibility requires the definition of G-invariance, which we define first.

Definition 3.7 (G-Invariance)**.** Let $(\phi, V) \in \text{Rep}(G)$. A subspace W of V is G-invariant if for all $g \in G$ and $w \in W$, $\phi(g)w \in W$.

Definition 3.8 (Reducibility)**.** Let $(\phi, V) \in \text{Rep}(G)$, then (ϕ, V) is irreducible if the only G-invariant sub-spaces are V and $\{0\}$.

Note that every one-dimensional matrix representation $(\phi, \mathbb{C})$ where $\phi : G \to GL_1(\mathbb{C})$ is irreducible since there are no nontrivial sub-spaces of $\mathbb{C}$. Thus in the case of the aforementioned representations $(\phi^n, \mathbb{C}) \in \text{MatRep}(\mathbb{Z}/L\mathbb{Z})$ where $[k] \mapsto \omega_L^{nk}$, each $\phi^n : \mathbb{Z}/L\mathbb{Z} \to GL_1(\mathbb{C}) \cong \mathbb{C}^*$ is irreducible.

Theorem 3.9. *If $(\phi, V) \in Rep(G)$ is equivalent to an irreducible representation $(\mu, W) \in Rep(G)$, then (ϕ, V) is irreducible.*

Proof. Let $(\phi, V) \in \text{Rep}(G)$ be equivalent to $(\mu, W) \in \text{Rep}(G)$ which is irreducible. Thus there exists an isomorphism $T : V \to W$ such that

$$T\phi(g) = \mu(g)T.$$

Let A be a G-invariant subspace of V, thus for for all $a \in A$, $\phi(g)a \in A$. We will show that $A = \{0\}$ or $A = V$. By the equivalence relation,

$$T\phi(g)a = \mu(g)Ta.$$

Let $Z = TA$ and $z = Ta$. By the above equation $\mu(g)z \in Z$. Since (μ, W) is irreducible, $Z = \{0\}$ or $Z = W$. Since T is an isomorphism, this means $A = \{0\}$ or $A = V$. □

Theorem 3.10 (Schur's Lemma)**.** *Let $(\phi, V), (\mu, W) \in Rep(G)$ be irreducible representations of G and let $T \in Hom_G(\phi, \mu)$. Then $T = 0$ or T is an isomorphism. Also:*

1. *If $\phi \nsim \mu$, then $T = 0$.*
2. *If $\phi = \mu$ and $V = W$, then $T = \lambda I$ where $\lambda \in \mathbb{C}$.*

Proof. The proof of this can be found in an un-numbered theorem entitled *Schur's Lemma* in Chapter 2B of [7]. □

Recall that to prove the Cyclic IDFT recovered the original function in Theorem 2.2 we claimed that the representations were orthogonal. This is exactly what we are on track to stating. To begin with, we will define our inner product.

Definition 3.11. The group algebra of G, denoted $L(G) = \{f | f : G \to \mathbb{C}\}$ is an inner product space with an inner product

$$\langle f, h \rangle = \frac{1}{|G|} \sum_{g \in G} f(g)\overline{h(g)}.$$

Another property of $(\phi^n, \mathbb{C}) \in \text{MatRep}(\mathbb{Z}/L\mathbb{Z})$ where $\phi^n[k] \mapsto \omega_L^{nk}$ is that $\phi^n[k]\overline{\phi^n[k]} = 1$. This is also necessary for direct proof of Theorem 2.2. This property is generalized as a matrix representation being unitary, which is defined as follows.

Definition 3.12 (Unitary Representation)**.** A unitary representation $(\phi, \mathbb{C}^d) \in \text{MatRep}(G)$ is a homomorphism $\phi : G \to U_d(\mathbb{C})$ where $U_d(\mathbb{C})$ is the group of $d \times d$ unitary matrices. Thus $\phi(g)\phi(g)^* = \phi(g)^*\phi(g) = I_d$ where $\phi(g)^*$ is the *conjugate transpose* of $\phi(g)$, and I_d is the $d \times d$ identity matrix.

From here, all of these concepts can be combined to show that the coefficients of non-equivalent, irreducible, unitary representations are orthogonal. This is known as *The Schur Orthogonality Relations.*

Theorem 3.13 (The Schur Orthogonality Relations)**.** *Let $(\phi, \mathbb{C}^d), (\mu, \mathbb{C}^{d'}) \in MatRep(G)$ be non-equivalent, irreducible, unitary representations of G. Let $\phi_{ij} : G \to \mathbb{C}$ where $g \mapsto \phi(g)_{ij}$ (and μ_{kl} defined similarly). Then, ϕ_{ij} and μ_{kl} are orthogonal in $L(G)$, that is*

$$\langle \phi_{ij}, \mu_{kl} \rangle = 0,$$

and

$$\langle \phi_{ij}, \phi_{kl} \rangle = \begin{cases} \frac{1}{d} & i = k, j = l, \\ 0 & else. \end{cases}$$

Proof. The proof of this can be found in Corollary 2 and Corollary 3 of an un-numbered theorem entitled *Schur's Lemma* in Chapter 2B of [7]. □

Thus the coefficients of all non-equivalent, irreducible, unitary representations are orthogonal. We will define this set of representations as follows.

Definition 3.14. Let the set of unitary representatives of the equivalence classes of irreducible representations of G be denoted $S(G)$. We can index the elements of this set by n as $S(G)$ is finite, thus the elements of $S(G)$ have the form $(\phi^n, \mathbb{C}^{d_n}) \in S(G)$.

Thus by Theorem 3.13, given $(\phi^n, \mathbb{C}^{d_n}) \in S(G)$, we have that $\sqrt{d_n}\phi^n_{ij}$ forms an orthonormal set on $L(G)$. In fact, this is an orthonormal basis of $L(G)$.

Theorem 3.15. *Given all $(\phi^n, \mathbb{C}^{d_n}) \in S(G)$, the summation $\sum_n d_n^2 = |G|$ and thus $\sqrt{d_n}\phi^n_{ij}$ forms an orthonormal basis of $L(G)$.*

Proof. The proof of this can be found in Corollary 2(a) of Proposition 5 in Chapter 2C of [7]. □

Thus the size of $S(G)$ is finite since the number of coefficients of elements of $S(G)$ is the group size $|G|$ which is finite. Since there exists an orthonormal basis of $L(G)$, for any function f in $L(G)$

$$f = \sum_{n,i,j} \left\langle f, \sqrt{d_n}\phi^n_{ij} \right\rangle \sqrt{d_n}\phi^n_{ij},$$

which we can use to define our Fourier transform.

Definition 3.16 (Finite Group Fourier Transform)**.** The Fourier transform of a function $f \in L(G)$ is given by $\widehat{f} : S(G) \to \bigoplus_n \mathrm{Mat}(d_n)$ where $\mathrm{Mat}(d_n)$ is the set of $d_n \times d_n$ matrices. For each representation $(\phi^n, \mathbb{C}^{d_n}) \in S(G)$, the Fourier transform $\widehat{f}(\phi^n) = (\widehat{f}_{ij}(\phi^n))_{i,j=1}^{d_n}$ where

$$\widehat{f}_{ij}(\phi^n) = \frac{1}{|G|} \sum_{g \in G} f(g) \overline{\phi_{ij}^n(g)}.$$

Again, the vector space is excluded for readability.

Thus the Fourier transform on a finite group takes a function $f \in L(G) = \{f | f : G \to \mathbb{C}\}$ and maps it to $\widehat{f} \in F(G) = \{\widehat{f} | \widehat{f} : S(G) \to \bigoplus_n \mathrm{Mat}(d_n)\}$, where we will call $F(G)$ the Fourier space of G. The Fourier transform is therefore a map $f \mapsto \widehat{f}$, from $L(G) \to F(G)$.

Now, the following Fourier inversion theorem provides a way to map back to our original function f from our matrices $\widehat{f}$.

Theorem 3.17 (Inverse Finite Group Fourier Transform)**.** *Given the Fourier coefficients defined in Theorem 3.16, the Fourier inversion is given by*

$$f(g) = \sum_{n,i,j} d_n \widehat{f}_{ij}(\phi^n) \phi_{ij}^n(g).$$

Proof. Observe that

$$\begin{aligned} f(g) &= \sum_{n,i,j} \left\langle f, \sqrt{d_n}\phi_{ij}^n \right\rangle \sqrt{d_n}\phi_{ij}^n(g) \\ &= \sum_{n,i,j} \left[\frac{1}{|G|} \sum_{g \in G} f(g) \overline{\sqrt{d_n}\phi_{ij}^n(g)} \right] \sqrt{d_n}\phi_{ij}^n(g) \\ &= \sum_{n,i,j} d_n \left[\frac{1}{|G|} \sum_{g \in G} f(g) \overline{\phi_{ij}^n(g)} \right] \phi_{ij}^n(g) \\ &= \sum_{n,i,j} d_n \widehat{f}_{ij}(\phi^n) \phi_{ij}^n(g). \end{aligned}$$

□

3.2. Fourier Theorems

The Fourier transform turning the convolution into multiplication, being unitary, and being an isometry are all central theorems to Fourier theory in its discrete and non-discrete forms. We will prove these now.

First, the Fourier transform is linear, which will be useful in later proofs.

Theorem 3.18 (Linearity)**.** *The finite group Fourier transform is a linear transform, i.e. given $f, g \in L(G)$, then $\widehat{(bf + cg)} = b\widehat{f} + c\widehat{g}$.*

Proof. This can easily be checked by applying Definition 3.16. □

3.2.1. The Convolution Theorem

Definition 3.19. The convolution of two functions $f, g \in L(G)$ is given by

$$(f * g)(a) = \sum_{b \in G} f(b)g(b^{-1}a).$$

Using this, the convolution theorem can be proven.

Theorem 3.20 (Convolution Theorem)**.** *Given $f, g \in L(G)$ and $(\phi^n, \mathbb{C}^{d_n}) \in S(G)$ then,*

$$\widehat{(f * g)}(\phi^n) = \widehat{f}(\phi^n)\widehat{g}(\phi^n).$$

Proof. Observe the following

$$\begin{aligned}
\widehat{(f * g)}_{ij}(\phi^n) &= \sum_{a \in G} \left[\sum_{b \in G} f(b)g(b^{-1}a) \right] \overline{\phi^n_{ij}(a)} \\
&= \sum_{ba \in G} \sum_{b \in G} f(b)g(b^{-1}ba)\overline{\phi^n_{ij}(ba)} \\
&= \sum_{a \in G} \sum_{b \in G} f(b)g(a) \sum_{k=0}^{d_n} \overline{\phi^n_{ik}(b)\phi^n_{kj}(a)} \\
&= \sum_{k=0}^{d_n} \left[\sum_{b \in G} f(b)\overline{\phi^n_{ik}(b)} \right] \left[\sum_{a \in G} g(a)\overline{\phi^n_{kj}(a)} \right] \\
&= \sum_{k=0}^{d_n} \widehat{f}_{ik}(\phi^n)\widehat{g}_{kj}(\phi^n).
\end{aligned}$$

Which is simply ij-th element of the matrix multiplication of $\widehat{f}(\phi^n)$ and $\widehat{g}(\phi^n)$, thus $\widehat{(f * g)}(\phi^n) = \widehat{f}(\phi^n)\widehat{g}(\phi^n)$. □

3.2.2. Plancherel's Theorem

Plancherel's Theorem equates f and $\widehat{f}$, through the inner products of the function spaces of f and $\widehat{f}$. Recall that $f \in L(G)$ and $\widehat{f} \in F(G)$. The inner product of $L(G)$ has been covered already in Definition 3.21, but the inner product of $F(G)$ has not been discussed so far.

Definition 3.21. Let the inner product of $F(G)$ be

$$\langle \widehat{f}, \widehat{g} \rangle = \sum_{n,i,j} \widehat{f}_{ij}(\phi^n)\overline{\widehat{g}_{ij}(\phi^n)}.$$

It is not hard to verify that this defines a Hermitian inner product. From here, we prove Parseval's Theorem, which states that the Fourier transform is a unitary operation, i.e. the norms of f and $\widehat{f}$ are equal.

Theorem 3.22 (Parseval's Theorem)**.** *Given the inner products*

$$\langle f, g\rangle_1 = \frac{1}{|G|}\sum_{a\in G} f(a)\overline{g(a)} \qquad \langle \widehat{f}, \widehat{g}\rangle_2 = \sum_{n,i,j} d_n \widehat{f}_{ij}(\phi^n)\overline{\widehat{g}_{ij}(\phi^n)}.$$

We have that $||f||_1 = ||\widehat{f}||_2$

Proof. Observe the following

$$\begin{aligned}
||f||_1^2 &= \frac{1}{|G|}\sum_{a\in G} f(a)\overline{f(a)} = \frac{1}{|G|}\sum_{a\in G}|f(a)|^2 \\
&= \frac{1}{|G|}\sum_{a\in G}\left|\sum_{n,i,j} d_n \widehat{f}_{ij}(\phi^n)\phi^n_{ij}(a)\right|^2 \\
&= \frac{1}{|G|}\sum_{a\in G}\left[\sum_{n,i,j} d_n \widehat{f}_{ij}(\phi^n)\phi^n_{ij}(a)\right]\left[\overline{\sum_{m,k,l} d_m \widehat{f}_{kl}(\phi^m)\phi^m_{kl}(a)}\right] \\
&= \sum_{n,i,j}\sum_{m,k,l} d_n d_m \widehat{f}_{ij}(\phi^n)\overline{\widehat{f}_{kl}(\phi^m)}\left[\frac{1}{|G|}\sum_{a\in G}\phi^n_{ij}(a)\overline{\phi^m_{kl}(a)}\right] \\
&= \sum_{n,i,j}\sum_{m,k,l} d_n \widehat{f}_{ij}(\phi^n)\overline{\widehat{f}_{kl}(\phi^m)}\delta_{nm}\delta_{ik}\delta_{jl} \\
&= \sum_{n,i,j} d_n \widehat{f}_{ij}(\phi^n)\overline{\widehat{f}_{ij}(\phi^n)} = ||\widehat{f}||_2^2.
\end{aligned}$$

□

We can now make the following extension.

Theorem 3.23 (Plancherel's Theorem)**.** *Given the inner products defined above, we have that* $\langle f, g\rangle_1 = \langle \widehat{f}, \widehat{g}\rangle_2$.

Proof. Since both $\langle\cdot,\cdot\rangle_1$ and $\langle\cdot,\cdot\rangle_2$ are Hermitian inner products, for either inner product

$$\langle f, g\rangle_k = \frac{1}{4}(||f+g||_k^2 - ||f-g||_k^2 + i||f+ig||_k^2 - i||f-ig||_k^2).$$

Combining this with Theorem 3.18 and Theorem 3.22 we get $\langle f, g\rangle_1 = \langle \widehat{f}, \widehat{g}\rangle_2$. □

4. Cyclic Groups

To connect the Cyclic DFT from Chapter 2 to the finite group Fourier transform from Chapter 3, recall that the Cyclic DFT and IDFT were defined as

$$\widehat{f}[n] = \frac{1}{L}\sum_{k=0}^{L-1} f[k]\omega_L^{-nk} \quad \text{and} \quad f[k] = \sum_{n=0}^{L-1} \widehat{f}[n]\omega_L^{nk}. \tag{4.1}$$

Now for any cyclic group of size L which we denote $C_L = \langle r | r^L = e \rangle$, the group will be isomorphic to $\mathbb{Z}/L\mathbb{Z}$ by $r^k \mapsto [k]$, which had a complete set of representations $(\phi^n, \mathbb{C}) \in S(C_L)$ for $n = 0, \ldots, L-1$ where $\phi^n[k] = \omega_L^{nk}$ [15]. Thus by Definition 3.16

$$\widehat{f}(\phi^n) = \frac{1}{L}\sum_{r^k \in C_L} f(r^k)\omega_L^{-nk} \quad \text{and} \quad f(r^k) = \sum_{n=0}^{L-1} \widehat{f}(\phi^n)\omega_L^{nk}.$$

Letting $f(r^k)$ be written as $f[k]$ and $\widehat{f}(\phi^n)$ be written $\widehat{f}[n]$, the Cyclic DFT is the finite group Fourier transform for cyclic groups, and likewise the Cyclic IDFT is the inverse finite group Fourier transform for cyclic groups.

4.1. The Naïve Cyclic DFT

We want to count the total number of operations necessary to compute the Cyclic DFT. First, we must define what an operation is. To do so, we proceed by the definition of Cooley and Tukey [6].

Definition 4.1 (Arithmetic Complexity)**.** An *operation* is a complex addition or a complex multiplication. The *arithmetic complexity* of an algorithm refers to the total number of operations it takes to compute.

Moreover, we assume any operations done for indexing or calculating roots of unity do not incur any arithmetic complexity. In reality, calculating indices, allotting space for new variables, and computing roots of unity does take variable amounts of time, which will be analyzed in Section 4.4.

So far, we have discussed Fast Fourier Transforms as being *faster* than their respective Discrete Fourier Transform, but now we have the tools to define this clearly.

Definition 4.2 (Fast Fourier Transform)**.** A Fast Fourier Transform (FFT) is any algorithm that reduces the arithmetic complexity of the naïve implementation of the corresponding DFT.

Although this paper focuses on computing the DFT, the Cyclic IDFT is computed the same way up to a normalization and conjugation of the ω_L^{-nk} terms and thus algorithms and complexity analysis for Cyclic IDFTs will follow from their corresponding Cyclic DFT algorithms. Moreover, Cyclic IDFTs will be important in Section 6.2 and Section 7.2. Cyclic IFFTs can be found in GitHub under the folder of the corresponding Cyclic FFT.

Before discussing the algorithms that reduce the arithmetic complexity of the Cyclic DFT, we begin with the naïve implementation, which refers to implementing the Cyclic DFT directly from Equation 4.1.

☞ *The code and documentation is available here.*

Proposition 4.1.1. *The arithmetic complexity of naïvely computing the Cyclic DFT for a cyclic group C_L is $2L^2 \in \mathcal{O}(|C_L|^2)$ operations.*

In the computation of each Fourier coefficient $\widehat{f}[n]$, there are L terms summed which is $L-1$ additions, where each term includes one multiplication, resulting in a total of L multiplications. Including normalization, this results in $L-1+L+1=2L$ operations. Since we have to compute L coefficients, we have in total $2L^2$ operations which has an asymptotic arithmetic complexity in $\mathcal{O}(|C_L|^2)$ operations.

Using the Naïve Cyclic DFT as a point of reference, we can now look at Cyclic FFTs which reduce this arithmetic complexity.

4.2. The Mixed-Radix FFT

☞ *The code and documentation is available here.*

The Mixed-Radix FFT, theorized by James Cooley and John Tukey in 1965 marks the historical beginning of Cyclic DFTs being considered possible to compute in under $\mathcal{O}(|C_L|^2)$ operations [6, 11].

The main idea of the Mixed-Radix FFT is to use a factorization of the group size $L = L_1 \dots L_m$ to divide the Cyclic DFT into smaller Cyclic DFTs. To start with, let $L = L_1 L_2$, then we reintepret f as a multi-variable function by rewriting the indices n and k using two variables as follows

$$\begin{aligned} k &= k_1 + k_2 L_1, \\ n &= n_2 + n_1 L_2. \end{aligned}$$

The indices n and k now consist of two components, meaning $f[k] = f[k_1, k_2]$, and likewise $\widehat{f}[n] = \widehat{f}[n_1, n_2]$. Now, to prevent duplicated values but range over all $n, k \in \mathbb{Z}/L\mathbb{Z}$ we have

$$\begin{aligned} 0 \le n_1 \le L_1 - 1, \qquad & 0 \le k_1 \le L_1 - 1, \\ 0 \le n_2 \le L_2 - 1, \qquad & 0 \le k_2 \le L_2 - 1. \end{aligned}$$

These ranges will help to reinterpret the summation given in Equation 4.1. Observe that the Cyclic DFT can be split into three steps as follows

$$\begin{aligned}\widehat{f}[n_1, n_2] &= \frac{1}{L_1 L_2} \sum_{k_1=0}^{L_1-1} \sum_{k_2=0}^{L_2-1} f[k_1, k_2] \omega_L^{-(n_2+n_1 L_2)(k_1+k_2 L_1)} \\ &= \underbrace{\frac{1}{L_1} \sum_{k_1=0}^{L_1-1} \underbrace{\left[\underbrace{\left[\frac{1}{L_2} \sum_{k_2=0}^{L_2-1} f[k_1, k_2] \omega_{L_2}^{-n_2 k_2} \right]}_{\text{Step 1}} \omega_L^{-n_2 k_1} \right]}_{\text{Step 2}} \omega_{L_1}^{-n_1 k_1}}_{\text{Step 3}}.\end{aligned}$$

In Step 1 the Cyclic DFT is done over the values of k_2 of which there are L_2, giving a new sequence of results dependent on k_1 and n_2. In Step 2, these results are multiplied by the *twiddle factor* $\omega_L^{-n_2 k_1}$, and in Step 3, a Cyclic DFT is done over the values of k_1 of which there are L_1. This can be written as a three-step algorithm as follows.

Step 1. Compute Cyclic DFTs: $\widehat{t}[k_1, n_2] = \frac{1}{L_2} \sum_{k_2=0}^{L_2-1} f[k_1, k_2] \omega_{L_2}^{-n_2 k_2}$

Step 2. Multiply by twiddle factors: $\widetilde{t}[k_1, n_2] = \widehat{t}[k_1, n_2] \omega_L^{-n_2 k_1}$

Step 3. Compute Cyclic DFTs: $\widehat{f}[n_1, n_2] = \frac{1}{L_1} \sum_{k_1=0}^{L_1-1} \widetilde{t}[k_1, n_2] \omega_{L_1}^{-n_1 k_1}$

There are two hidden steps here. First reinterpreting $f[k]$ as $f[k_1, k_2]$ and secondly uninterpreting $\widehat{f}[n_1, n_2]$ back to $\widehat{f}[n]$. However, by our assumptions these do not incur any computational cost.

Using recursion in these algorithms to break the problem into smaller sub-problems, known as the divide-and-conquer paradigm, we can speed this algorithm up further. Therefore, the Cyclic DFTs in Step 1 and Step 3 provide an opportunity for recursion. The base case (i.e. when we can no longer recurse) is hit when we can no longer factorize the group size (i.e. L is prime). In this case, the naïve implementation given in Section 4.1 is the only algorithm discussed in this paper that can handle this. In practice, an algorithm known as Rader's FFT is also often used to handle prime group sizes, but will not be covered in this thesis [13].

Proposition 4.2.1. *The arithmetic complexity of computing the Mixed-Radix FFT for a cyclic group C_L is given by the recurrence relation*

$$T(L) = \begin{cases} 2L^2 & \text{if } L \text{ is prime,} \\ L_1 T(L_2) + L + L_2 T(L_1) & \text{where } L_1 L_2 = L. \end{cases}$$

To understand the arithmetic complexity of this algorithm, consider when $L = L_1 L_2$. Then, Step 1 computes L_1 Cyclic DFTs of size L_2, Step 2 multiplies these results by

$L_1 L_2 = L$ twiddle factors, and Step 3 computes L_2 Cyclic DFTs of size L_1. Now, if recursion occurs, the algorithm is computed again by factorizing L_1 and L_2, giving a recurrence of $L_1 T(L_2) + L + L_2 T(L_1)$ as long as L is not prime. However, if L is prime we resort to computing the Cyclic DFT naïvely, which has an arithmetic complexity of $2L^2$ per Proposition 6.1.1.

As a side note, if L is a power of 2, letting $L_1 = L/2$ and $L_2 = 2$ we get $T(L) = 2T(L/2) + L + (L/2)T(2) = 2T(L/2) + 5L$ which can be solved to get $T(L) \in \mathcal{O}(L \log L)$. This particular case has been heavily studied due to its $\mathcal{O}(L \log L)$ complexity and is known as the Radix-2 FFT [6].

4.3. The Prime Factor Algorithm

☞ *The code and documentation is available here.*

The Prime Factor Algorithm (PFA) improves upon the Mixed-Radix FFT for group sizes with co-prime factors. Developed by Thomas Good in 1963, prior to the Cooley Tukey algorithm, but only wide-spread later, the two algorithms were initially considered the same [5]. However, the PFA offers slight improvements in arithmetic complexity over the Mixed-Radix FFT for co-prime factorizations of groups sizes.

When looking at the Mixed-Radix FFT algorithm, Step 2, i.e. the twiddle factor step, requires L multiplications by roots of unity. This can not be done recursively, and incurs L operations every time the Mixed-Radix FFT is used. The PFA removes this step by using a different rewriting of f as a multi-variable function. Thus, we start by reinterpretting f as a multi-variable function, this time using Bézout's identity to rewrite the index k as

$$k = k_1 L_2 + k_2 L_1.$$

Similarly, let the index n be rewritten using n_1 and n_2, which are given by the Chinese Remainder Theorem

$$\begin{aligned} n_1 &\equiv n \mod L_2, \\ n_2 &\equiv n \mod L_1. \end{aligned}$$

Thus the ranges are the same as before, which are

$$\begin{aligned} 0 \leq n_1 \leq L_1 - 1, \qquad & 0 \leq k_1 \leq L_1 - 1, \\ 0 \leq n_2 \leq L_2 - 1, \qquad & 0 \leq k_2 \leq L_2 - 1, \end{aligned}$$

and n is the solution to the Chinese Remainder Theorem, so

$$n \equiv L_1 a n_2 + L_2 b n_1 \mod L,$$

where

$$\begin{aligned} L_1 a &\equiv 1 \mod L_2, \\ L_2 b &\equiv 1 \mod L_1. \end{aligned}$$

This gives the following rewrite of Equation 4.1.

$$\begin{aligned}\widehat{f}[n_1, n_2] &= \frac{1}{L_1 L_2} \sum_{k_1=0}^{L_1-1} \sum_{k_2=0}^{L_2-1} f[k_1, k_2] \omega_L^{-(L_1 a n_2 + L_2 b n_1)(L_2 k_1 + L_1 k_2)} \\ &= \underbrace{\frac{1}{L_1} \sum_{k_1=0}^{L_1-1} \underbrace{\left[\frac{1}{L_2} \sum_{k_2=0}^{L_2-1} f[k_1, k_2] \omega_{L_2}^{-n_2 k_2} \right]}_{\text{Step 1}} \omega_{L_1}^{-n_1 k_1}}_{\text{Step 2}} .\end{aligned}$$

Note that the twiddle factors are now gone! So the PFA is given by the following two-step algorithm.

Step 1. Compute Cyclic DFTs: $\widehat{t}[k_1, n_2] = \frac{1}{L_2} \sum_{k_2=0}^{L_2-1} f[k_1, k_2] \omega_{L_2}^{-n_2 k_2}$

Step 2. Compute Cyclic DFTs: $\widehat{f}[n_1, n_2] = \frac{1}{L_1} \sum_{k_1=0}^{L_1-1} \widehat{t}[k_1, n_2] \omega_{L_1}^{-n_1 k_1}$

Of course, we must again reinterpret $f[k]$ as $f[k_1, k_2]$ before Step 1 and uninterpret $\widehat{f}[n_1, n_2]$ as $\widehat{f}[n]$ after Step 2 which by our assumptions incurs no arithmetic complexity.

Proposition 4.3.1. *The arithmetic complexity of computing the PFA for a cyclic group C_L is given by the recurrence relation*

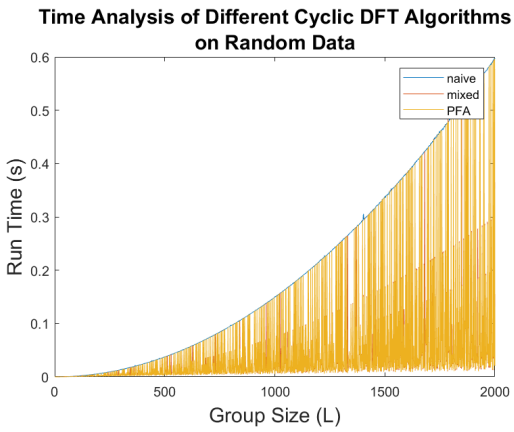
$$T(L) = \begin{cases} 2L^2 & \text{if no co-prime factors,} \\ L_1 T(L_2) + L_2 T(L_1) & \text{where } L_1 L_2 = L \text{ and } L_1, L_2 \text{ co-prime.} \end{cases}$$

The arithmetic complexity of the PFA is simply that of the Mixed-Radix FFT except without the twiddle factor step, which incurred a computational cost of L multiplications by twiddles. Therefore, the PFA has L fewer operations than the Mixed-Radix FFT in each recurrence.

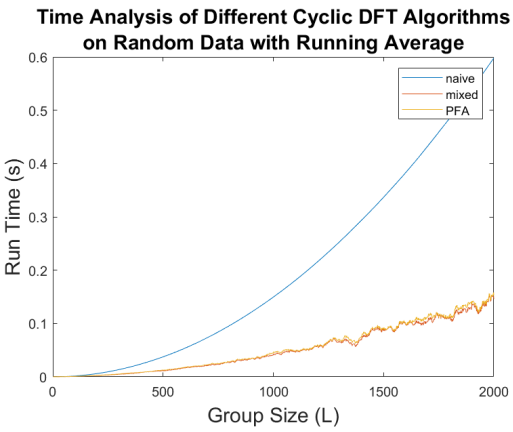
4.4. Analysis

☞ *The driver used to produce the following results is available here.*

As mentioned, all results about arithmetic complexity are theoretic as among other things the processor must compute or locate roots of unity in memory, store data, reinterpret f as a multi-variable function, and uninterpret $\widehat{f}$ back to a single-variable function. We can use MATLAB's built-in `timeit` function [3] to compute how long our encodings of these algorithms actually take to compute. To test the run time of the discussed algorithms for computing the Cyclic DFTs, we can create functions $f : C_L \to \mathbb{C}$ for $L = 1$ to 2000 consisting of pseudo-random uniformly distributed complex numbers generated by MATLAB's `rand` function [3].



(a) Time Analysis

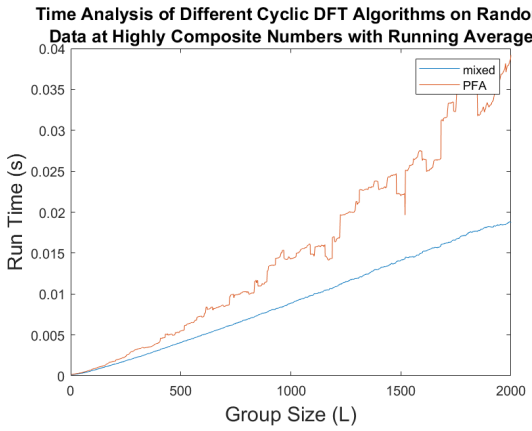


(b) Time Analysis with Moving Mean

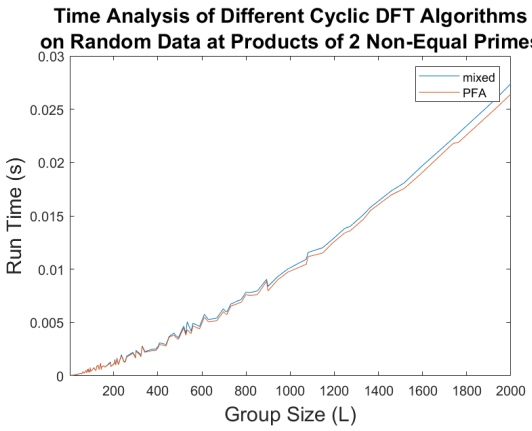
Figure 4.1.: Time Analysis of Cyclic DFT Algorithms

In Figure 4.1(a) we can see that the naïve implementation of the Cyclic DFT has a steadily increasing run time as the group size increases. However, what we find with both the Mixed-Radix FFT and the PFA is that the run time varies depending on the group size. This is exactly what we expect to find based on Proposition 4.2.1 and Proposition 4.3.1, from which we see that when the group size L is prime, we have to run the Naïve Cyclic DFT for both algorithms. To make the results clearer, we take the 100-point running average of the run time data resulting in Figure 4.1(b).

From this, we can see that on average the Mixed-Radix FFT runs slightly faster than the PFA despite the recurrence relations in Proposition 4.2.1 and Proposition 4.3.1, pointing to the PFA having a better arithmetic complexity. This is to be expected as prime factorizations are far more common than prime factorizations where each factor is co-prime. To analyze this we compare the Mixed-Radix FFT and PFA for highly composite group sizes and for co-prime group sizes separately.



(a) Highly Composite Group Sizes



(b) Co-prime Group Sizes

Figure 4.2.: Comparisons of Mixed-Radix FFT and PFA Algorithms

From Figure 4.2(a) we can see that the Mixed-Radix FFT handles highly composite group sizes better than the PFA. A major disadvantage of the PFA is that it runs group sizes such as $L = 2^n$ for $n \in \mathbb{Z}$ naïvely in $\mathcal{O}(L^2)$ operations whereas the Mixed-Radix FFT can run this in $\mathcal{O}(L \log L)$ operations. However, the PFA handles products of non-equal primes marginally better than the Mixed-Radix FFT as seen in Figure 4.2(b). This makes sense as the complexity is reduced by a factor of the group size L in each recurrence when comparing Proposition 4.2.1 and Proposition 4.3.1.

To improve these results, modern FFTs combine these algorithms with many others to create an algorithm that begins by analyzing the group size, and then applies a sequence of different smaller Cyclic FFTs to efficiently compute the Cyclic FFT. For instance, one popular Cyclic FFT algorithm known as the FFTW claims a $\mathcal{O}(L \log L)$ complexity for inputs of all lengths–including primes [1].

Going into this project, my goal was to write a Cyclic FFT that I could show was in $\mathcal{O}(L \log L)$ operations. However looking at the sheer quantity of research on the topic, I realized that without learning more FFT algorithms, and studying how they are combined using number theory, I would not create anything remotely new. As an example, the FFTW website states "In one way or another, [the] FFTW uses the Cooley-Tukey algorithm, the prime factor algorithm, Rader's algorithm for prime sizes, and a split-radix algorithm... [The] FFTW's code generator also produces new algorithms that we do not completely understand" [1]. Moreover, many commercial FFT algorithms such as the FFTW use computer science specific ideas such as parallel computing, hard coding, lower-level programming, dynamic programming, and GPU computing making their algorithms run incomparably faster that my own.

However, studying the Mixed-Radix FFT and the PFA were extremely helpful for this paper. As we will see, splitting a DFT into smaller DFTs using methods learned from studying the Mixed-Radix FFT and PFA will come back when studying Abelian FFTs and Dihedral Product FFTs.

Lastly, although we will explore FFTs for other finite groups, Cyclic DFTs and IDFTs will show up in these contexts. Based on our analysis of Figure 4.2(a) and Figure 4.2(b), the Mixed-Radix FFT ultimately ran faster in general and had drastic improvements for highly composite group sizes. Thus, moving forward, whenever we compute a Cyclic DFT we shall use our implementation of the Mixed-Radix FFT.

5. Abelian Groups

So far, we have only worked with the complete set of representations of cyclic groups. Combining the complete set of representations of cyclic groups with the Fundamental Theorem of Abelian Groups gives us a passage towards writing Abelian DFTs.

Theorem 5.1. *Every finite abelian group is a direct product of cyclic groups.*

Proof. The proof of this can be found in Theorem 3 in Chapter 5.2 of [8]. □

Thus, given a finite abelian group G, there exist cyclic groups C_{L_i} for $1 \leq i \leq s$ such that $G = C_{L_1} \times \cdots \times C_{L_s}$. From here, a complete set of representations of abelian groups can be constructed using the complete set of representations of these cyclic groups.

Theorem 5.2. *Given the complete set of representations of C_{L_i} with elements $(\phi_i^{n_i}, \mathbb{C}) \in S(C_{L_i})$ for $1 \leq i \leq s$, then $(\phi^{n_1,\ldots,n_s}, \mathbb{C})$ where*

$$\begin{aligned} \phi^{n_1,\ldots,n_s} : C_{L_1} \times \cdots \times C_{L_s} &\to \mathbb{C}^* \\ (g_1, \ldots, g_s) &\mapsto \prod_{i=1}^{s} \phi_i^{n_i}(g_i) \end{aligned}$$

forms a complete set of representations of $C_{L_1} \times \cdots \times C_{L_s}$.

Proof. To prove this, we must show the following. We follow a similar method to Proposition 4.5.1 in Chapter 4 of [17].

1. each $(\phi^{n_1,\ldots,n_s}, \mathbb{C})$ is a representation
2. each $(\phi^{n_1,\ldots,n_s}, \mathbb{C})$ is irreducible
3. each $\phi^{n_1,\ldots,n_s}(g_1, \ldots, g_s)$ is unitary for all $g_i \in C_{L_i}$
4. each $(\phi^{n_1,\ldots,n_s}, \mathbb{C})$ is distinct

To the first point, we show that $\phi^{n_1,\ldots,n_s}$ is a homomorphism. To see this, observe that

$$\begin{aligned} \phi^{n_1,\ldots,n_s}(g_1\widetilde{g}_1, \ldots, g_s\widetilde{g}_s) &= \prod_{i=1}^{s} \phi_i^{n_i}(g_i\widetilde{g}_i) \\ &= \prod_{i=1}^{s} \phi_i^{n_i}(g_i)\phi_i^{n_i}(\widetilde{g}_i) \\ &= \prod_{i=1}^{s} \phi_i^{n_i}(g_i) \prod_{i=1}^{s} \phi_i^{n_i}(\widetilde{g}_i) \\ &= \phi^{n_1,\ldots,n_s}(g_1, \ldots, g_s)\phi^{n_1,\ldots,n_s}(\widetilde{g}_1, \ldots, \widetilde{g}_s). \end{aligned}$$

To the second point, each $(\phi^{n_1,\ldots,n_s}, \mathbb{C})$ is irreducible since there are no non-trivial sub-spaces of $\mathbb{C}$.

Now, to the third point, these representation are unitary since conjugation is distributive over multiplication. Thus,

$$\begin{aligned}\phi^{n_1,\ldots,n_s}(g_1,\ldots,g_s)\phi^{n_1,\ldots,n_s}(g_1,\ldots,g_s)^* &= \left[\prod_{i=1}^{s}\phi_i^{n_i}(g_i)\right]\left[\prod_{i=1}^{s}\phi_i^{n_i}(g_i)\right]^* \\ &= \prod_{i=1}^{s}\phi_i^{n_i}(g_i)\overline{\phi_i^{n_i}(g_i)} = 1,\end{aligned}$$

and $\phi^{n_1,\ldots,n_s}(g_1,\ldots,g_s)^*\phi^{n_1,\ldots,n_s}(g_1,\ldots,g_s) = 1$ can be shown without loss of generality.

Finally, to the fourth point, it is important to show that these representations are distinct, since a complete set of representations only contains non-equivalent representations. In a one-dimensional setting this equates to checking if the representations are not equal. To state this clearly, if $\phi^{n_1,\ldots,n_s} = \phi^{n'_1,\ldots,n'_s}$, then $n_i = n'_i$ for $1 \leq i \leq s$.

To prove this, observe that

$$\phi_i^{n_i}(g_i) = \phi^{n_1,\ldots,n_s}(e,\ldots,e,g_i,e,\ldots,e) = \phi^{n'_1,\ldots,n'_s}(e,\ldots,e,g_i,e,\ldots,e) = \phi_i^{n'_i}(g_i)$$

for all $1 \leq i \leq s$. Thus $\phi_i^{n_i} = \phi_i^{n'_i}$, which is only true if $n_i = n'_i$ since $\phi_i^{n_i}$ and $\phi_i^{n'_i}$ are not equivalent.

Therefore, $(\phi^{n_1,\ldots,n_s}, \mathbb{C})$ for $n_i = 0,\ldots,L_i - 1$ forms a complete set of representations of $C_{L_1} \times \cdots \times C_{L_s}$. This is a complete set of representations by Theorem 3.15 since

$$\sum_{n_1,\ldots,n_s} d_{n_1}\ldots d_{n_s} = \sum_{n_1,\ldots,n_s} 1 = L_1\ldots L_s = |C_{L_1}|\ldots|C_{L_s}| = |C_{L_1}\times\cdots\times C_{L_s}|.$$

□

Thus, since $(\phi_i^{n_i}, \mathbb{C}) \in S(C_{L_i})$ where $\phi_i^{n_i}(r^{k_i}) = \omega_{L_i}^{n_i k_i}$ for $n_i = 0,\ldots,L_i - 1$ forms a complete set of representations of C_{L_i}, by Theorem 5.2 we now have that $C_{L_1}\times\cdots\times C_{L_s}$ has a complete set of representations with elements $(\phi^{n_1,\ldots,n_s}, \mathbb{C})$ where

$$\phi^{n_1,\ldots,n_s}(r^{k_1},\ldots,r^{k_s}) = \omega_{L_1}^{n_1 k_1}\ldots\omega_{L_s}^{n_s k_s}.$$

Now, let $f(r^{k_1},\ldots,r^{k_s})$ be denoted $f[k_1,\ldots,k_s]$ and $\widehat{f}(\phi^{n_1,\ldots,n_s})$ be denoted $\widehat{f}[n_1,\ldots,n_s]$, then by Definition 3.16 the Abelian DFT is

$$\widehat{f}[n_1,\ldots,n_s] = \frac{1}{L_1\ldots L_s}\sum_{k_1=0}^{L_1-1}\cdots\sum_{k_s=0}^{L_s-1} f[k_1,\ldots,k_s]\overline{\omega_{L_1}^{n_1 k_1}\ldots\omega_{L_s}^{n_s k_s}}. \tag{5.1}$$

5.1. The Naïve Abelian Fourier Transform

☞ *The code and documentation is available here.*

Proposition 5.1.1. *The arithmetic complexity of naïvely computing the Abelian DFT for an abelian group $G = C_{L_1} \times \cdots \times C_{L_s}$ is $|G|^2(s+1)$ operations.*

Using Equation 5.1 to compute one Fourier coefficient of the Abelian DFT we see there are a total of $L_1 \dots L_s = |G|$ terms summed, and in each term there is an additional multiplication by s roots of unity, which with normalization amounts to a total of $s|G| + |G|$ operations per Fourier coefficient. Thus, to compute all $|G|$ Fourier coefficients, there are a total of $|G|^2(s+1)$ operations.

As a side note, observe that when $s = 1$, i.e. $G = C_{L_1}$, this is simply the arithmetic complexity of the Naïve Cyclic DFT stated in Proposition 4.1.1.

5.2. A Fast Abelian Fourier Transform

☞ *The code and documentation is available here.*

Using the methods of the Mixed-Radix FFT in Section 4.2 and the Prime Factor Algorithm in Section 4.3 which split Cyclic DFTs into smaller Cyclic DFTs, we can rewrite the Abelian DFT such that it consists of smaller Cyclic DFTs by rewriting Equation 4.1 as follows

$$\widehat{f}[n_1, \dots, n_s] = \underbrace{\frac{1}{L_s} \sum_{k_s=0}^{L_s-1} \cdots \underbrace{\left[\frac{1}{L_s} \sum_{k_2=0}^{L_2-1} \underbrace{\left[\frac{1}{L_1} \sum_{k_1=0}^{L_1-1} f[k_1, \dots, k_s] \omega_{L_1}^{-n_1 k_1} \right]}_{\text{Step 1}} \omega_{L_2}^{-n_2 k_2} \right]}_{\text{Step 2}} \cdots \omega_{L_s}^{-n_s k_s}}_{\text{Step } s}.$$

Since this has been broken down into Cyclic DFTs, any Cyclic FFT applies now to improve the arithmetic complexity of computing the Abelian DFT. This can be written as an s step algorithm as follows.

Step 1. Compute Cyclic DFTs:

$$\widehat{t}_1[n_1, k_2, \dots, k_s] = \frac{1}{L_1} \sum_{k_1=0}^{L_1-1} f[k_1, \dots, k_s] \omega_{L_1}^{-n_1 k_1}$$

Step 2. Compute Cyclic DFTs:

$$\widehat{t}_2[n_1, n_2, k_3, \dots, k_s] = \frac{1}{L_2} \sum_{k_2=0}^{L_2-1} \widehat{t}_1[n_1, k_2, \dots, k_s] \omega_{L_2}^{-n_2 k_2}$$

$$\vdots$$

Step i. Compute Cyclic DFTs:

$$\widehat{t}_i[n_1,\ldots,n_i,k_{i+1},\ldots,k_s] = \frac{1}{L_i}\sum_{k_i=0}^{L_i-1}\widehat{t}_{i-1}[n_1,\ldots,n_{i-1},k_i,\ldots,k_s]\omega_{L_i}^{-n_ik_i}$$

$$\vdots$$

Step s. Compute Cyclic DFTs:

$$\widehat{f}[n_1,\ldots,n_s] = \widehat{t}_s[n_1,\ldots,n_s] = \frac{1}{L_s}\sum_{k_s=0}^{L_s-1}\widehat{t}_{s-1}[n_1,\ldots,n_{s-1},k_s]\omega_{L_s}^{-n_sk_s}$$

Proposition 5.2.1. *Using a Cyclic FFT with arithmetic complexity in $\mathcal{O}(L_i\log L_i)$ for a cyclic group C_{L_i}, the Abelian FFT for a group $G = C_{L_1}\times\cdots\times C_{L_s}$ has an arithmetic complexity in $\mathcal{O}(|G|\log|G|)$.*

Let $A(|G|)$ denote the complexity of computing an Abelian DFT of group size $|G|$ and $C(L_i)$ denote the complexity of computing a Cyclic DFT of group size L_i.

Consider the cost in Step i. In Step i, a Cyclic FFT of size L_i is computed $|G|/L_i$ times. Over a total of s steps this is

$$A(|G|) = (|G|/L_1)C(L_1) + \cdots + (|G|/L_s)C(L_s).$$

Using a Cyclic FFT with complexity $C(L_i)\in\mathcal{O}(L_i\log L_i)$ this means $C(L_i)\leq cL_i\log L_i$ for some $c\in\mathbb{R}$ as $L_i\to\infty$. Thus this results in $A(|G|)\leq c|G|(\log L_1+\cdots+\log L_s) = c|G|\log|G|\in\mathcal{O}(L\log L)$ since $|G| = L_1\ldots L_s$.

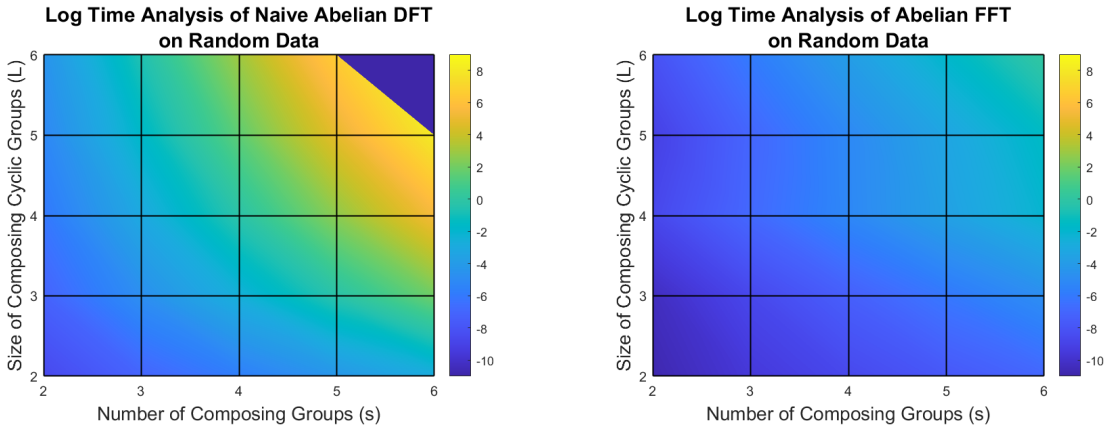
5.3. Analysis

☞ *The driver used to produce the following results is available here.*

We can test the run time of this algorithm using similar methods to Section 4.4. However, this time the group $G = C_{L_1}\times\cdots\times C_{L_s}$ is dependent on the number s of composing groups, and the size L_i of these composing groups. To test both of these factors, we can fix $L_i = L$ for all $1\leq i\leq s$ such that $G = C_L\times\cdots\times C_L$. Letting L and s range from 2 to 6 and measuring the run time of the Abelian DFT algorithms computed on randomly generated functions on G, we get run time data dependent on two factors, L and s, that we plot on a pseudo-color plot using MATLAB's `pcolor` function [3].

Although these may seem like small test cases, this scales quickly as the direct product of 6 cyclic groups of size 6 has $6^6 = 46,556$ elements. To make the results more visible, the logarithm of the run time data was taken since the pseudo-color plot assigns colors linearly. While the Abelian FFT could handle all these inputs in a matter of seconds, the Naïve Abelian DFT did not finish computing all of these Abelian DFTs after running

for more than 48 hours, and had to be terminated. The dark blue vertex indicates the test case where $L = 6$ and $s = 6$ which could not be computed.



(a) Log Time Analysis of Naïve Abelian DFT (b) Log Time Analysis of Abelian FFT

Figure 5.1.: Log Time Analysis of Abelian DFT Algorithms

Comparing Figure 5.1(a) and Figure 5.1(b), we see that the Abelian FFT runs far faster than the Naïve Abelian DFT in every test case. In short, the Abelian FFT algorithm was able to compute each Abelian DFT in under 2 seconds, whereas the Naïve Abelian DFT took over 50 minutes to run for the group where $s = 6$ and $L = 5$, and could not compute the Abelian DFT for the group where $s = 6$ and $L = 6$.

We can also analyze the dependence on s in the Naïve Abelian DFTs arithmetic complexity of $|G|^2(s + 1)$ operations given by Proposition 5.1.1. To do so, we compare the Naïve Abelian DFTs run time at $s = 3$ and $L = 4$ to the run time at $s = 6$ and $L = 2$. Although both of these cases compute an Abelian DFT for groups of size 64, the case where $s = 6$ took 1.19 times longer to compute than the case where $s = 3$. On the other hand, using the Abelian FFT, this transform was computed 1.11 times faster.

This makes sense as comparing Proposition 5.1.1 to Proposition 5.2.1, we see that our Abelian FFT has removed this dependence upon s, and moreover, the more our Abelian DFT is split up, the more Cyclic FFTs we can apply. Although Proposition 5.2.1 only holds for Cyclic FFTs in $\mathcal{O}(L \log L)$ operations, we can see that using our implementation of the Mixed-Radix FFT in our Abelian FFT already improves upon the run time of naïvely computing the Abelian DFT.

6. The Dihedral Group

The dihedral group D_{2L} is an interesting group for doing Fourier transforms as it has an easy to understand structure, but is not abelian for $L > 2$. As we will see, the complete set of representations of D_{2L} has one-dimensional and two-dimensional representations and different representations when L is even or odd. This paper will cover the even case, but the arithmetic complexity will asymptotically be the same for L odd. Recall that

$$D_{2L} = \left\langle s, r | r^L = s^2 = e, srs^{-1} = r^{-1} \right\rangle.$$

For our purposes, we will only consider $L > 2$. Using [15] and writing the elements of D_{2L} in the form $s^l r^k$ for $l = 0, 1$ and $r = 0, \ldots, L-1$, the complete set of irreducible representations of D_{2L} for L even is given by the following one-dimensional representations

$$\phi^{-3}(s^l r^k) = 1, \quad \phi^{-2}(s^l r^k) = (-1)^l, \quad \phi^{-1}(s^l r^k) = (-1)^k, \quad \phi^0(s^l r^k) = (-1)^{l+k}.$$

Next, splitting this up and writing rotations as r^k and reflections of rotations as sr^k, we have two-dimensional representations where

$$\phi^n(r^k) = \begin{pmatrix} \omega_L^{nk} & 0 \\ 0 & \omega_L^{-nk} \end{pmatrix} \quad \text{and} \quad \phi^n(sr^k) = \begin{pmatrix} 0 & \omega_L^{-nk} \\ \omega_L^{nk} & 0 \end{pmatrix},$$

for $n = 1, \ldots, L/2 - 1$. If we instead want to consider the odd case, then ϕ^{-1} and ϕ^0 are not included and $n = 1, \ldots, (L-1)/2 - 1$.

Since there are two different *types* of representations, one-dimensional and two-dimensional, and two different *types* of elements of D_{2L}, one for rotations, and one for reflections of rotations, the Fourier transform given in Definition 3.16 can be split up as follows

$$\widehat{f}_{ij}(\phi^n) = \begin{cases} \frac{1}{2L} \sum_{s^l r^k \in D_{2L}} f\left(s^l r^k\right) \overline{\phi^n(s^l r^k)} & \text{for } -3 \leq n \leq 0, \\ \frac{1}{2L} \left[\sum_{r^k \in D_{2L}} f\left(r^k\right) \overline{\begin{pmatrix} \omega_L^{nk} & 0 \\ 0 & \omega_L^{-nk} \end{pmatrix}_{ij}} + \sum_{sr^k \in D_{2L}} f\left(sr^k\right) \overline{\begin{pmatrix} 0 & \omega_L^{-nk} \\ \omega_L^{nk} & 0 \end{pmatrix}_{ij}} \right] & \text{for } 0 < n < L/2. \end{cases}$$

6.1. The Naïve Dihedral Fourier Transform

Similar to our assumptions in Section 4.1, since the coefficients of representations of Dihedral groups consist of roots of unity, 1, 0, and powers of -1, we will assume the computer can calculate these coefficients without incurring any computational cost.

☞ *The code and documentation is available here.*

Proposition 6.1.1. *The arithmetic complexity of naïvely computing the Dihedral DFT for a dihedral group D_{2L} is $8L^2 \in \mathcal{O}(|D_{2L}|^2)$ operations. If L is even, then computing the Fourier coefficients indexed by representations $-3 \leq n \leq 0$ takes $16L$ operations and computing the coefficients indexed by representations $0 < n < L/2$ takes $8L^2 - 16L$ operations.*

Consider the calculation of the Fourier coefficients for $-3 \leq n \leq 0$ and $0 < n < L/2$ separately. For $-3 \leq n \leq 0$, there is one summation of size $2L$ containing one multiplication, resulting in $4L$ operations with normalization per calculation of one coefficient, of which there are 4. In total this amounts to $16L$ operations.

For $0 < n < L/2$, there are two summations of size L each containing one multiplication, which is used to compute $L/2 - 1$ transforms which each contain 4 coefficients resulting in $16L(L/2-1)$ operations with normalization. Combining this with the above result ultimately gives an arithmetic complexity of

$$16L + 8L^2 - 16L = 8L^2 \in \mathcal{O}(|D_{2L}|^2).$$

Viewing this transform as two separate pieces, it becomes clear that the algorithm for $-3 \leq n \leq 0$ does not affect the asymptotic arithmetic complexity of the algorithm and is in fact linear, i.e. in $\mathcal{O}(L)$ operations. Thus to improve the arithmetic complexity of the algorithm, we focus on improving the arithmetic complexity of computing the Dihedral DFT for the coefficients of the representations indexed by $0 < n < L/2$.

6.2. A Fast Dihedral Fourier Transform

☞ *The code and documentation is available here.*

Since the Dihedral DFT uses roots of unity, the Dihedral FFT can be related to the Cyclic FFT through rewriting. Recall that for the two-dimensional representations, the Dihedral DFT is given by

$$\widehat{f}_{ij}(\phi^n) = \frac{1}{2L}\left[\sum_{r^k \in D_{2L}} f\left(r^k\right)\begin{pmatrix}\omega_L^{-nk} & 0\\ 0 & \omega_L^{nk}\end{pmatrix}_{ij} + \sum_{sr^k \in D_{2L}} f\left(sr^k\right)\begin{pmatrix}0 & \omega_L^{nk}\\ \omega_L^{-nk} & 0\end{pmatrix}_{ij}\right] \tag{6.1}$$

for $0 < n < L/2 - 1$. From the definition of D_{2L}, rotations (including the identity) $r^k \in D_{2L}$ are uniquely $e, r, \ldots, r^{L-1}$. Now, the index $r^k \in D_{2L}$ can be written r^k for $k = 0, \ldots, L-1$. Likewise, the rotations of reflections (including the reflection s) can be rewritten sr^k for $k = 0, \ldots, L-1$. Thus the summation in Equation 6.1 can be rewritten as

$$\widehat{f}_{ij}(\phi^n) = \frac{1}{2L}\left[\sum_{k=0}^{L-1} f\left(r^k\right)\begin{pmatrix}\omega_L^{-nk} & 0\\ 0 & \omega_L^{nk}\end{pmatrix}_{ij} + \sum_{k=0}^{L-1} f\left(sr^k\right)\begin{pmatrix}0 & \omega_L^{nk}\\ \omega_L^{-nk} & 0\end{pmatrix}_{ij}\right].$$

Which in a few steps can be rewritten as

$$\widehat{f}_{ij}\left(\phi^n\right) = \frac{1}{2L}\left[\begin{pmatrix} \sum_{k=0}^{L-1} f\left(r^k\right)\omega_L^{-nk} & 0 \\ 0 & \sum_{k=0}^{L-1} f\left(r^k\right)\omega_L^{nk} \end{pmatrix}_{ij} + \begin{pmatrix} 0 & \sum_{k=0}^{L-1} f\left(sr^k\right)\omega_L^{nk} \\ \sum_{k=0}^{L-1} f\left(sr^k\right)\omega_L^{-nk} & 0 \end{pmatrix}_{ij}\right]$$

$$= \frac{1}{2}\begin{pmatrix} \frac{1}{L}\sum_{k=0}^{L-1} f\left(r^k\right)\omega_L^{-nk} & \frac{1}{L}\sum_{k=0}^{L-1} f\left(sr^k\right)\omega_L^{nk} \\ \frac{1}{L}\sum_{k=0}^{L-1} f\left(sr^k\right)\omega_L^{-nk} & \frac{1}{L}\sum_{k=0}^{L-1} f\left(r^k\right)\omega_L^{nk} \end{pmatrix}_{ij}. \tag{6.2}$$

Therefore every matrix entry with a ω_L^{-nk} term is a Cyclic DFT and every matrix entry with a ω_L^{nk} term is a normalized Cyclic IDFT. However, we do not use the full range of results causing a trade-off, since for our purposes $1 < n < L/2$ but the Cyclic DFT and IDFT give results for $0 \leq n \leq L-1$.

Thus, any Cyclic FFT and IFFT can be applied to speed up the calculation of the Dihedral DFT resulting in a Dihedral FFT!

Proposition 6.2.1. *Using a Cyclic FFT with arithmetic complexity in $\mathcal{O}(L\log L)$ for a cyclic group C_L, the Dihedral FFT for a dihedral group D_{2L} has arithmetic complexity in $\mathcal{O}(|D_{2L}|\log|D_{2L}|)$.*

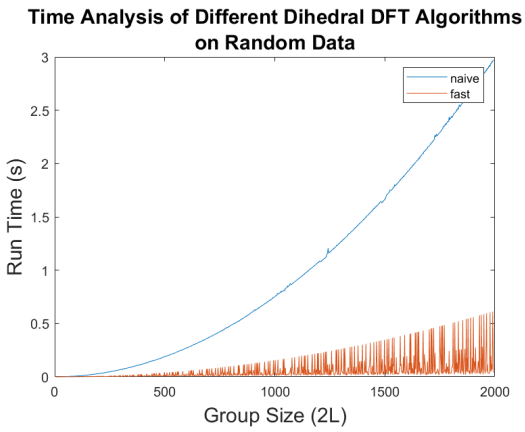
Let $D(2L)$ denote the arithmetic complexity of computing a Dihedral DFT of group size $2L$ and $C(L)$ denote the arithmetic complexity of computing a Cyclic DFT of group size L. By Proposition 6.1.1 the cost of computing the coefficients of representations indexed by $-3 \leq n \leq 0$ is $16L$. Now, using the methods discussed above, to compute the coefficients of the representations indexed by $0 < n < L/2$ amounts to two Cyclic DFTs and two Cyclic IDFTs of group size L. Recall that we can compute Cyclic IDFTs with the same methods as Cyclic DFTs as they are the same up to normalization and conjugation by the roots of unity.

Using a Cyclic FFT and IFFT in $\mathcal{O}(L\log L)$, this results in $D(2L) \leq 16L + 2c_1 L\log L + 2c_2\log L$ operations for some $c_1, c_2 \in \mathbb{R}$ as $L \to \infty$. Thus $D(2L) \in \mathcal{O}(L\log L)$ which is equivalent to saying $D(2L) \in \mathcal{O}(|D_{2L}|\log|D_{2L}|)$.

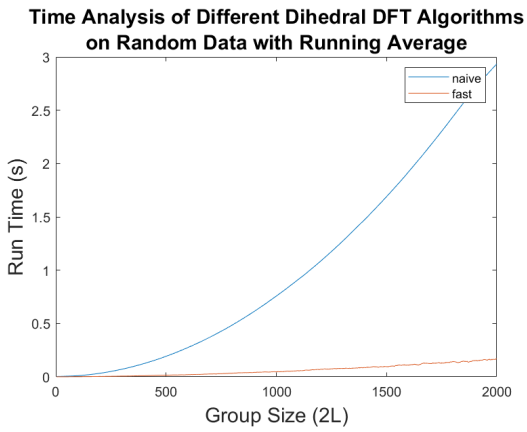
6.3. Analysis

☞ *The driver used to produce the following results is available here.*

To compare the naïve implementation of the Dihedral DFT with our Dihedral FFT, we follow the methods of Section 4.4 and randomly generate functions $f : D_{2L} \to \mathbb{C}$ from $L = 3$ to $L = 1000$ which allow us to test our algorithms for group sizes up to 2000. Recall from our analysis in Section 4.4 that our Mixed-Radix FFT was the fastest way of computing the Cyclic DFT, and therefore is the Cyclic FFT algorithm that we will use for computations of the Dihedral FFT. Computing the run times and a 100-point running average over these run times gives the following two plots.



(a) Time Analysis



(b) Time Analysis with Moving Mean

Figure 6.1.: Time Analysis of Dihedral DFT Algorithms

In Figure 6.1(a) we can see that the Dihedral FFT is consistently faster than the Dihedral DFT even at prime values of L. To understand why, we can compare the equation of coefficients $0 < n < L/2$ given in Equation 6.1 and our rewrite of this equation given in Equation 6.2. Looking at these, we see that Equation 6.1 had added arithmetic complexity in each step by multiplying the function by 0 within the summation, which added 4 unnecessary multiplications that we were ultimately able to remove using our rewriting in Equation 6.2. Moreover, on a non-theoretical note, implementing Equation 6.1 requires initializing two 2×2 matrices instead of one 2×2 matrix, which also makes our Dihedral FFT faster.

Taking the running average in Figure 6.1(b) helps us see how much faster our Dihedral FFT runs in general. From Section 4.4, we know that the Mixed-Radix FFT is not in $\mathcal{O}(L \log L)$ for all cyclic groups C_L, meaning this specific implementation of the Dihedral FFT using the Mixed-Radix FFT is not in $\mathcal{O}(|D_{2L}| \log |D_{2L}|)$ for all L. However, as discussed in Proposition 6.2.1, given a Cyclic FFT in $\mathcal{O}(L \log L)$, this is theoretically possible, and the Mixed-Radix FFT already makes large improvements in computing the Dihedral DFT as seen in both figures.

7. Direct Products of Dihedral Groups

Finally, we want to write FFTs for direct products of dihedral groups by combining the FFTs we have explored so far. To begin with, we introduce some representation theory for direct products of groups.

Theorem 7.1. *If $(\phi_1, \mathbb{C}^{d_n})$ and $(\phi_2, \mathbb{C}^{d_m})$ are irreducible matrix representations of G_1 and G_2 respectively, then $(\phi_1 \otimes \phi_2, \mathbb{C}^{d_n d_m})$ is an irreducible matrix representation of $G_1 \times G_2$.*

Proof. The proof of this can be found in Theorem 10 of Chapter 3.2 of [15]. □

We can now find the complete set of representations of a direct product of groups.

Theorem 7.2. *Given the complete set of representations of G_i with elements $(\phi_i^{n_i}, \mathbb{C}^{d_{n_i}}) \in S(G_i)$ for $1 \leq i \leq s$, then $(\phi^{n_1,\dots,n_s}, \mathbb{C}^{d_{n_1}\dots d_{n_s}})$ where*

$$\phi^{n_1,\dots,n_s} : G_1 \times \cdots \times G_s \to GL_{d_{n_1}\dots d_{n_s}}(\mathbb{C})$$
$$(g_1, \dots, g_s) \mapsto \bigotimes_{i=1}^{s} \phi_i^{n_i}(g_i)$$

forms a complete set of representations of $G_1 \times \cdots \times G_s$

Proof. Similar to Theorem 5.2, we must show the following.

1. each $\phi^{n_1,\dots,n_s}$ is a representation
2. each $\phi^{n_1,\dots,n_s}$ is irreducible
3. each $\phi^{n_1,\dots,n_s}(g_1, \dots, g_s)$ is unitary for all $(g_1, \dots, g_s) \in G_1 \times \cdots \times G_s$
4. each $\phi^{n_1,\dots,n_s}$ is distinct

We will prove this explicitly for $s = 2$. The rest follows inductively since the direct product of groups is a group. The first and the second point are then covered by Theorem 7.1.

To the third point we must show that given $\phi_1^{n_1}(g_1) \in U(d_{n_1})$ and $\phi_2^{n_2}(g_2) \in U(d_{n_2})$ we have that $\phi^{n_1,n_2}(g_1, g_2) \in U(d_{n_1} d_{n_2})$. To see this, observe that

$$\begin{aligned}
[\phi^{n_1,n_2}(g_1,g_2)]\,[\phi^{n_1,n_2}(g_1,g_2)]^* &= [\phi_1^{n_1}(g_1) \otimes \phi_2^{n_2}(g_2)]\,[\phi_1^{n_1}(g_1) \otimes \phi_2^{n_2}(g_2)]^* \\
&= [\phi_1^{n_1}(g_1) \otimes \phi_2^{n_2}(g_2)]\,[\phi_1^{n_1}(g_1)^* \otimes \phi_2^{n_2}(g_2)^*] \\
&= [\phi_1^{n_1}(g_1)\phi_1^{n_1}(g_1)^*] \otimes [\phi_2^{n_2}(g_2)\phi_2^{n_2}(g_2)^*] \\
&= I_{d_{n_1}} \otimes I_{d_{n_2}} = I_{d_{n_1} d_{n_2}},
\end{aligned}$$

where the second step uses that the conjugate transpose is distributive over the Kronecker product and the third step uses the mixed-product property of Kronecker products. Proving $[\phi^{n_1,n_2}(g_1,g_2)]^* [\phi^{n_1,n_2}(g_1,g_2)] = I_{d_{n_1}\dots d_{n_s}}$ can likewise be done without loss of generality.

Now, to the fourth point we must show that each ϕ^{n_1,n_2} is distinct. This means showing that if there is an isomorphism $T : \mathbb{C}^{d_{n_1}d_{n_2}} \to \mathbb{C}^{d_{n_1'}d_{n_2'}}$ such that

$$T\phi^{n_1,n_2}(g_1,g_2) = \phi^{n_1',n_2'}(g_1,g_2)T,$$

then $n_1 = n_1'$ and $n_2 = n_2'$. Considering when $g_1 = e$ and rewriting both sides of the equation we have the following.

$$T[\phi_1^{n_1}(e) \otimes \phi_2^{n_2}(g_2)] = [\phi_1^{n_1'}(e) \otimes \phi_2^{n_2'}(g_2)]T$$
$$T[I_{d_{n_1}} \otimes \phi_2^{n_2}(g_2)] = [I_{d_{n_1'}} \otimes \phi_2^{n_2'}(g_2)]T$$

Which can be written as a diagonal block matrix as follows.

$$T\begin{pmatrix} \phi_2^{n_2}(g_2) & & 0 \\ & \ddots & \\ 0 & & \phi_2^{n_2}(g_2) \end{pmatrix} = \begin{pmatrix} \phi_2^{n_2'}(g_2) & & 0 \\ & \ddots & \\ 0 & & \phi_2^{n_2'}(g_2) \end{pmatrix} T$$

From the top left term of the multiplication we can find $T_2 : \mathbb{C}^{d_{n_2}} \to \mathbb{C}^{d_{n_2'}}$ such that $T_2\phi_2^{n_2}(g_2) = \phi_2^{n_2'}(g_2)T_2$. However, since each $\phi_2^{n_2}$ is distinct, this means $n_2 = n_2'$. Similarly, setting $g_2 = e$ and using the canonical isomorphism $(A \otimes B) \cong (B \otimes A)$ (Proposition 20 in Section 10.4 of [8]) we get

$$T[I_{d_{n_2}} \otimes \phi_1^{n_1}(g_1)] = [I_{d_{n_2'}} \otimes \phi_1^{n_1'}(g_1)]T.$$

From this we can similarly find $T_1 : \mathbb{C}^{d_{n_1}} \to \mathbb{C}^{d_{n_1'}}$ such that $T_1\phi_1^{n_1}(g_1) = \phi_1^{n_1'}(g_1)T_1$ which is only true if $n_1 = n_1'$. Therefore if $\phi^{n_1,n_2} \sim \phi^{n_1'n_2'}$ then $n_1 = n_1'$ and $n_2 = n_2'$, and therefore each representation $(\phi^{n_1,n_2}, \mathbb{C}^{d_{n_1}d_{n_2}})$ is distinct.

Since each representation $(\phi^{n_1,n_2}, \mathbb{C}^{d_{n_1}d_{n_2}})$ is a non-equivalent, irreducible, unitary representation of $G_1 \times G_2$, by Theorem 3.15 we have a complete set of representations of $G_1 \times G_2$ since

$$\sum_{n_1,n_2} d_{n_1}d_{n_2} = \sum_{n_1} d_{n_1} \sum_{n_2} d_{n_2} = |G_1||G_2| = |G_1 \times G_2|.$$

□

Because we know the complete set of representations of dihedral groups, we can now use Theorem 7.2 to write the complete set of representations for direct products of dihedral groups, which we can then use to write a Fourier transform for this group which we will call the Dihedral Product DFT.

Let $G = D_{2L_1} \times \cdots \times D_{2L_s}$. Using Definition 3.16 and Theorem 7.2 the Dihedral Product DFT has Fourier coefficients

$$\widehat{f}_{kl}(\phi^{n_1,\ldots,n_s}) = \frac{1}{2^s L_1 \ldots L_s} \sum_{(g_1,\ldots,g_s)\in D_{2L_1}\times\cdots\times D_{2L_s}} f(g_1,\ldots,g_s)\overline{\left(\bigotimes_{i=1}^{s} \phi_i^{n_i}(g_i)\right)}_{kl}, \qquad (7.1)$$

where $(\phi_i^{n_i}, \mathbb{C}^{d_{n_i}}) \in S(D_{2L_i})$ for $1 \leq i \leq s$.

7.1. The Naïve Dihedral Product Transform

☞ *The code and documentation is available here.*

Proposition 7.1.1. *The arithmetic complexity of naïvely computing the Dihedral DFT for a dihedral group $G = D_{2L_1} \times \cdots \times D_{2L_s}$ is $|G|^2(s+1)$ operations.*

To compute one Fourier coefficient using Equation 7.1 we sum over the elements of $G = D_{2L_1} \times \cdots \times D_{2L_s}$ which amounts to $|G| - 1$ additions. In each of these additions we multiply $f(g_1, \ldots, g_s)$ by a coefficient of the Kronecker products of s matrices. Calculating the Kronecker product coefficient takes $s - 1$ multiplications so in total this results in s multiplications within one term of the summation and s$|G|$ multiplications in total. Thus, including normalization, computing one Fourier coefficients takes $|G| - 1 + s|G| + 1 = |G|(s+1)$ operations, since there are $|G|$ Fourier coefficients by Theorem 3.15, this amounts to $|G|^2(s+1)$ operations.

As a side note, when $s = 1$, i.e. $G = D_{2L_1}$, then the arithmetic complexity is $2|D_{2L_1}|^2 = 8L_1^2$ which is exactly the arithmetic complexity of the Naïve Dihedral DFT given in Proposition 6.1.1.

7.2. A Fast Dihedral Product Transform

☞ *The code and documentation is available here.*

To improve computation of the Dihedral Product DFT we aim to rewrite our Dihedral Product DFT as a combination of Dihedral DFTs similar to how we rewrote the Abelian DFT as a combination of Cyclic DFTs. This is not as straight forward since instead of taking a product of the cyclic representations as in the case of the abelian group, we have to take the tensor product of the representations since the dihedral group has two-dimensional representations. However, we can rewrite the elements of the Kronecker product term in Equation 7.1 as a product using the following theorem.

Theorem 7.3. *For a $c \times d$ matrix B, the ij-th coefficient of the Kronecker product $(A \otimes B)$ is equal to $A_{p_1q_1}B_{p_2q_2}$ where*

$$p_1 = \left\lceil \frac{i}{c} \right\rceil, \quad q_1 = \left\lceil \frac{j}{d} \right\rceil, \quad p_2 = (i-1) \,\%\, c + 1, \quad q_2 = (j-1) \,\%\, d + 1,$$

and $a \,\%\, b$ denotes the remainder of a/b.

Proof. See Theorem A.1 in Appendix A.1. □

Now, if we repeatedly do this, we can find a_i and b_i such that

$$\left(\bigotimes_{i=1}^{s} \phi_i^{n_i}(g_i)\right)_{kl} = \prod_{i=1}^{s} (\phi_i^{n_i})_{a_i b_i}(g_i).$$

Applying this to Equation 7.1 we have

$$\begin{aligned}\widehat{f}_{kl}(\phi^{n_1,\ldots,n_s}) &= \frac{1}{2^s L_1 \ldots L_s} \sum_{(g_1,\ldots,g_s)\in D_{2L_1}\times\cdots\times D_{2L_s}} f(g_1,\ldots,g_s)\overline{(\phi_1^{n_1})_{a_1b_1}(g_1)\ldots(\phi_s^{n_s})_{a_sb_s}(g_s)} \\ &= \underbrace{\frac{1}{2L_s}\sum_{g_s\in D_{2L_s}} \cdots \underbrace{\left[\frac{1}{2L_1}\sum_{g_1\in D_{2L_1}} f(g_1,\ldots,g_s)\overline{(\phi_1^{n_1})_{a_1b_1}(g_1)}\right]}_{\text{Step 1}} \ldots \overline{(\phi_s^{n_s})_{a_sb_s}(g_s)}}_{\text{Step } s}.\end{aligned}$$

Thus using the same method of splitting up the DFT that we used to write our Abelian FFT, we can split up the Dihedral Product FFT into smaller Dihedral FFTs, which in turn use Cyclic FFTs and IFFTs! This can be written as an s-step algorithm as follows.

Step 1. Compute Dihedral DFTs:

$$\widehat{t}_{a_1b_1}(\phi_1^{n_1}, g_2, \ldots, g_s) = \frac{1}{2L_1}\sum_{g_1\in D_{2L_1}} f(g_1,\ldots,g_s)\overline{(\phi_1^{n_1})_{a_1b_1}(g_1)}$$

$$\vdots$$

Step i. Compute Dihedral DFTs

$$\widehat{t}_{a_ib_i}(\phi_1^{n_1},\ldots,\phi_i^{n_i}, g_{i+1},\ldots,g_s) = \frac{1}{2L_i}\sum_{g_i\in D_{2L_i}} \widehat{t}_{a_{i-1}b_{i-1}}(\phi^{n_1},\ldots,\phi^{n_{i-1}}, g_i,\ldots,g_s)\overline{(\phi_i^{n_i})_{a_ib_i}(g_i)}$$

$$\vdots$$

Step s. Compute Dihedral DFTs

$$\widehat{f}_{kl}(\phi^{n_1,\ldots,n_s}) = \widehat{t}_{a_sb_s}(\phi_1^{n_1},\ldots,\phi_s^{n_s}) = \frac{1}{2L_s}\sum_{g_s\in D_{2L_s}} \widehat{t}_{a_{s-1}b_{s-1}}(\phi^{n_1},\ldots,\phi^{n_{s-1}}, g_s)\overline{(\phi_s^{n_s})_{a_sb_s}(g_s)}$$

Proposition 7.2.1. *Using a Cyclic FFT with arithmetic complexity in $\mathcal{O}(L_i \log L_i)$ operations for a cyclic group C_{L_i}, the Dihedral Product FFT for a group $G = D_{2L_1} \times \cdots \times D_{2L_s}$ has arithmetic complexity in $\mathcal{O}(|G| \log |G|)$ operations.*

Let $P(|G|)$ denote the arithmetic complexity of computing the Dihedral Product DFT of $G = D_{2L_1} \times \cdots \times D_{2L_s}$ and $D(2L_i)$ denote the arithmetic complexity of computing a Dihedral DFT of group size $2L_i$. To find the arithmetic complexity, consider the arithmetic complexity of computing a given step i. In this step, we compute a Dihedral FFT of size $2L_i$ for the coefficients of each representation $\phi_1^{n_1}, \ldots, \phi_{i-1}^{n_{i-1}}$ and each element of $G_{i+1}, \ldots, G_s$. By Theorem 3.15 there are $2L_1 \ldots 2L_{i-1}$ such coefficients of representations and there are $2L_{i+1} \ldots 2L_s$ elements. Thus Step i can be computed in $2L_1 \ldots 2L_{i-1} 2L_{i+1} \ldots 2L_s D(2L_i)$ operations.

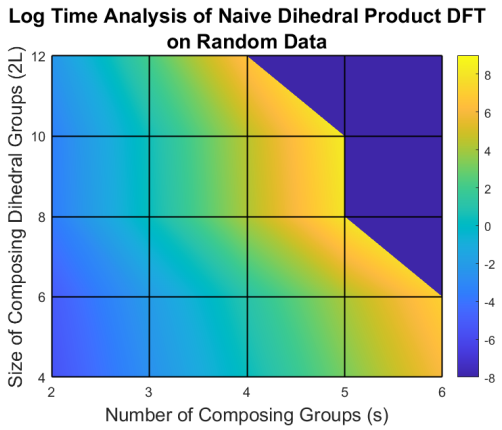
Now, by Proposition 5.2.1, given a Cyclic DFT in $\mathcal{O}(L_i \log L_i)$ operations for a cyclic group C_{L_i}, the Dihedral FFT for a group of size $2L_i$ can be computed in $\mathcal{O}(2L_i \log 2L_i)$ operations. Thus for some $c \in \mathbb{R}$ and $L_i \to \infty$, we can compute Step i in under $2L_1 \ldots 2L_{i-1} 2L_{i+1} \ldots 2L_s (c 2L_i \log 2L_i) = c|G| \log 2L_i$ operations since $|G| = 2^s L_1 \ldots L_s$. Doing this over s steps results in

$$P(|G|) \leq \sum_{i=1}^{s} c|G| \log 2L_i = c|G| \log 2^s L_1 \ldots L_s \in \mathcal{O}(|G| \log |G|) \text{ operations.}$$

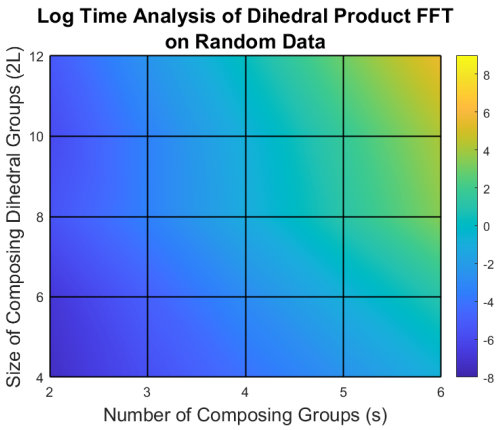
7.3. Analysis

☞ *The driver used to produce the following results is available here.*

To compare these algorithms, we use the pseudo-color plot method described in Section 5.3 for $G = D_{2L} \times \cdots \times D_{2L}$ giving the results shown in Figure 7.1. The Naïve Dihedral Product DFT also failed to compute some of the test cases in 48 hours, which are the cases with dark blue vertices in Figure 7.1(a).



(a) Log Time Analysis of Naïve Dihedral Product DFT



(b) Log Time Analysis of Dihedral Product FFT

Figure 7.1.: Log Time Analysis of Dihedral Product DFT Algorithms

Similar to the analysis of the Abelian FFT in Section 5.3, the Dihedral Product FFT

tested faster than the naïve implementation of the Dihedral Product DFT in every case. Moreover, we can see the influence of the s term in the $|G|^2(s+1)$ complexity for the Naïve Dihedral Product DFT given in Proposition 7.1.1. Take group size $4,096$. The Naïve Dihedral Product DFT took 6.5 times longer to compute the test case when $s = 4$ and $L = 4$ than when $s = 6$ and $L = 2$. Both of these test cases have the same group size since $(2 \times 4)^4 = (2 \times 2)^6 = 4,096$. In contrast, the Dihedral Product FFT took only 1.3 times longer to compute the case when $s = 6$ compared to the case when $s = 4$, which could arise from the Mixed Radix Cyclic FFT not being in $\mathcal{O}(L \log L)$ operations.

Therefore, given a fixed group size, the Dihedral Product DFT is heavily impacted by the number of composing groups. This can also be seen in the fact that when $s = 6$, the test cases with $L = 4, 5, 6$ failed to compute. On the contrary, the Dihedral Product FFT was able to run the case when $s = 6$ and $L = 6$, which has group size $(2 \times 6)^6 = 2,985,984$ in 285 seconds.

8. Conclusion

This project combined theory about Cyclic FFTs and representation theory of finite groups to efficiently compute Fourier transforms for finite abelian and non-abelian groups. To begin with, we introduced the Cyclic DFT as well as representation theory, and found a general Fourier transform for any finite group as well as its subsequent Fourier inversion. Using this, we were able to prove the Convolution Theorem, Parseval's Theorem, and Plancherel's Theorem for finite groups, theorems that are central to Fourier theory in discrete and non-discrete settings.

Then, using our finite group Fourier transform and referring to [15] for theory about representations of cyclic and dihedral groups, we were able to write Fourier transforms for abelian, dihedral, and direct products of dihedral groups. From here, we analyzed that naïvely computing these Fourier transforms takes $2|G|^2$ operations for dihedral groups and $|G|^2(s+1)$ operations for abelian and direct products of dihedral groups, where s was the number of groups used in the direct product. However, using our study of Cyclic FFTs, we could write our own algorithms that could be combined with a Cyclic FFT in $\mathcal{O}(L \log L)$ for a cyclic group C_L to write these FFTs in $\mathcal{O}(|G| \log |G|)$.

In terms of application, Cyclic DFTs and Abelian DFTs are extremely important in modern times, especially in the field of signal-processing. Specifically, the Cyclic DFT is used to reinterpret signals as their frequency domain. Moreover, Abelian DFTs do the same for signals of more dimensions, and as a result, Abelian DFTs are often referred to as multi-dimensional DFTs within this field. For instance, a photo composed of pixels dependent on an x and y coordinate can be analyzed using a DFT on a direct product of two cyclic groups, and a video composed of x and y values as well as a frame number z is a DFT on a direct product of three cyclic groups. Lastly, Fourier transforms on dihedral groups are less common, but have been explored in the field of optics in [18], however a different formulation of the Dihedral DFT is used.

Fourier transforms on direct products of dihedral groups do not appear application-rich. However, if a DFT for direct products of finite non-abelian groups was ever of interest, the method of splitting up the transform into smaller group transforms using coefficients of the Kronecker products could similarly be used to improve the efficiency of the transform.

In terms of further research, we can try to generalize these results further such that they apply to larger classes of groups, or instead look into other group Fourier transforms for specific groups of interest. To the first point, proving an upper bound on the number of operations necessary to compute a FFT for finite groups has been studied [12], as well as theoretical ways of computing FFTs for wreath products of groups [14]. To the second point, Fourier theory for finite groups is adaptable to compact groups by way of the Peter-Weyl Theorem and a Haar measure for the group [19].

Bibliography

[1] FFTW documentation. Accessed on May 24, 2023. URL: https://www.fftw.org/fftw3_doc/Introduction.html#Introduction/.

[2] FFTW homepage. Accessed on May 24, 2023. URL: https://www.fftw.org/.

[3] MATLAB documentation. Accessed on May 24, 2023. URL: https://nl.mathworks.com/help/.

[4] F. Alexander. *The Analytical Theory of Heat by Joseph Fourier Translated, With Notes by Alexander Freeman*. St. Johns College, Cambridge, 1878.

[5] J.W. Cooley, P.A.W. Lewis, and P.D. Welch. Historical notes on the fast fourier transform. *Proceedings of the IEEE*, 55(10):1675–1677, 1967. doi:10.1109/PROC.1967.5959.

[6] J.W. Cooley and J.W. Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of Computation*, 19:297–301, 1965.

[7] P. Diaconis. *Group Representations in Probability and Statistics*. Institute of Mathematical Statistics, 1st edition, 1988.

[8] D.S. Dummit and R.M. Foote. *Abstract Algebra*. John Wiley & Sons, 3rd edition, 1991.

[9] M. Frigo and S.G. Johnson. The design and implementation of fftw3. *Proceedings of the IEEE*, 93(2):216–231, 2005. doi:10.1109/JPROC.2004.840301.

[10] C.F. Gauss. *Theoria Interpolationis Methodo Nova Tractata*. Göttingen: Königliche Gesellschaft der Wissenschaften, 3rd edition, 1876.

[11] M. Heideman, D. Johnson, and C. Burrus. Gauss and the history of the fast fourier transform. *IEEE ASSP Magazine*, 1(4):14–21, 1984. doi:10.1109/MASSP.1984.1162257.

[12] D.K. Maslen and D.N. Rockmore. Generalized FFTs - a survey of some recent results. *Max Planck Institut für Mathematik*, 1991.

[13] C.M. Rader. Discrete fourier transforms when the number of data samples is prime. *Proceedings of the IEEE*, 56(6):1107–1108, 1968. doi:10.1109/PROC.1968.6477.

[14] D.N. Rockmore. Fast Fourier Transforms for Wreath Products. *Department of Mathematics and Computer Science, Dartmouth College*, 1994.

[15] J.P. Serre. *Linear Representations of Finite Groups.* Collection Méthodes Mathématiques. Springer-Verlag, 1st edition, 1977.

[16] C.E. Shannon. Communication in the presence of noise. *Proceedings of the IRE*, 37(1):10–21, 1949. doi:10.1109/JRPROC.1949.232969.

[17] B. Steinberg. *Representation Theory of Finite Groups: An Introductory Approach.* Universitext. Springer New York, 2011.

[18] M.A.G. Viana and V. Lakshminarayanan. *Dihedral Fourier Analysis: Data-analytic Aspects and Applications.* Lecture Notes in Statistics. Springer New York, 2012.

[19] A. Vollrath. The Nonequispaced Fast SO(3) Fourier Transform, Generalisations and Applications. *Aus dem Institut für Mathematik der Universität zu Lübeck*, 2010.

A. Appendix

A.1. Kronecker Products

Given a $m \times n$ matrix A and a $c \times d$ matrix B, the Kronecker product is defined as the $mc \times nd$ block matrix

$$(A \otimes B) = \begin{pmatrix} A_{11}B & \cdots & A_{1n}B \\ \vdots & \ddots & \vdots \\ A_{m1}B & \cdots & A_{mn}B \end{pmatrix}$$

which written out is

$$(A \otimes B) = \begin{pmatrix} \begin{pmatrix} A_{11}B_{11} & \cdots & A_{11}B_{1d} \\ \vdots & \ddots & \vdots \\ A_{11}B_{c1} & \cdots & A_{11}B_{cd} \end{pmatrix} & \cdots & \begin{pmatrix} A_{1n}B_{11} & \cdots & A_{1n}B_{1d} \\ \vdots & \ddots & \vdots \\ A_{1n}B_{c1} & \cdots & A_{1n}B_{cd} \end{pmatrix} \\ \vdots & \ddots & \vdots \\ \begin{pmatrix} A_{m1}B_{11} & \cdots & A_{m1}B_{1d} \\ \vdots & \ddots & \vdots \\ A_{m1}B_{c1} & \cdots & A_{m1}B_{cd} \end{pmatrix} & \cdots & \begin{pmatrix} A_{mn}B_{11} & \cdots & A_{mn}B_{1d} \\ \vdots & \ddots & \vdots \\ A_{mn}B_{c1} & \cdots & A_{mn}B_{cd} \end{pmatrix} \end{pmatrix}. \quad \text{(A.1)}$$

We can now prove Theorem 7.3 for any size matrix B.

Theorem A.1. *For a $c \times d$ matrix B, the ij-th coefficient of the Kronecker product $(A \otimes B)$ is equal to $A_{p_1 q_1} B_{p_2 q_2}$ where*

$$p_1 = \left\lceil \frac{i}{c} \right\rceil, \quad q_1 = \left\lceil \frac{j}{d} \right\rceil, \quad p_2 = (i-1) \,\%\, c + 1, \quad q_2 = (j-1) \,\%\, d + 1,$$

and $a \,\%\, b$ denotes the remainder of a/b.

Proof. Let $T = (A \otimes B)$. Looking at Equation A.1, observe that the j-th element of the x-th row of T is

$$T_{xj} = \begin{pmatrix} \begin{pmatrix} A_{y1}B_{z1} & \cdots & A_{y1}B_{zd} \end{pmatrix} & \cdots & \begin{pmatrix} A_{yn}B_{z1} & \cdots & A_{yn}B_{zd} \end{pmatrix} \end{pmatrix}_j,$$

for some y and z. In the above equation, the elements of the y-th row of A and the z-th row of B are being multiplied. Specifically the z-th row of B is fixed and multiplied by A_{y1} through A_{yn}. Thus the j-th element of this vector then consists of the $\lceil \frac{j}{d} \rceil$-th

entry of the y-th row of A, as the entries of A are fixed for every d elements and A is 1-indexed. Moreover, the j-th element of this vector consists of the $(j-1) \ \% \ d+1$ entry of the z-th row of B, as the elements of B repeat every d elements. Since these matrices use indices starting at 1, we can not have remainders of zero, which we avoid by subtracting one from j and adding one to the result. Thus,

$$T_{xj} = A_{y,\lceil \frac{j}{d} \rceil} B_{z,(j-1) \ \% \ d+1}.$$

Without loss of generality, the i-th element of the x-th column of T is

$$T_{ix} = A_{\lceil \frac{i}{c} \rceil,y} B_{(i-1) \ \% \ c+1,z},$$

for some y and z. Since the indices of the rows and columns of T are independent, we have

$$T_{ij} = A_{\lceil \frac{i}{c} \rceil,\lceil \frac{i}{d} \rceil} B_{(i-1) \ \% \ c+1,(j-1) \ \% \ d+1}.$$

Thus $(A \otimes B)_{ij} = A_{p_1 q_1} B_{p_2 q_2}$ where

$$p_1 = \left\lceil \frac{i}{c} \right\rceil, \quad q_1 = \left\lceil \frac{j}{d} \right\rceil, \quad p_2 = (i-1) \ \% \ c + 1, \quad q_2 = (j-1) \ \% \ d + 1.$$

□