

Midyear Report: Decimation-In-Frequency Fast Fourier Transforms on the Symmetric Group

Eric J. Malm

MATH 197: Senior Thesis
Advisor: Michael E. Orrison
Second Reader: Shahriar Shahriari

December 10, 2004

Contents

| | | |
|-----------|--|-----------|
| I | Literature Review: Recent Results in Generalized FFTs | 4 |
| 1 | Introduction | 4 |
| 2 | Group-Theoretical Fourier Transforms | 5 |
| 2.1 | Abelian Groups | 6 |
| 2.2 | Nonabelian Groups | 9 |
| 3 | Algorithmic Approaches to FFTs | 10 |
| 3.1 | Decimation-in-Time Algorithms | 10 |
| 3.2 | Decimation-in-Frequency Algorithms | 12 |
| 3.3 | Convolution Algorithms | 13 |
| 4 | The Symmetric Group | 13 |
| 4.1 | Representations of S_n | 14 |
| 4.2 | FFTs on S_n | 16 |
| 5 | Applications | 17 |
| 6 | Open Questions | 18 |
| II | Current and Future Work | 19 |
| 7 | Introduction | 19 |
| 8 | Representation Theory of S_n | 19 |
| 8.1 | DFTs as a Change of Basis | 19 |
| 8.2 | Combinatorial Objects and Techniques | 20 |
| 8.3 | Young's Seminormal Matrix Representations | 22 |
| 8.4 | Jucys-Murphy Elements and the Content of a Tableau | 24 |
| 9 | Decimation-In-Frequency Algorithm Theory | 25 |
| 9.1 | Choice of Basis Ordering | 26 |
| 9.2 | Computation of Frequency Separation Factors | 27 |
| 9.2.1 | Centrally Primitive Idempotents | 28 |
| 9.2.2 | Jucys-Murphy Eigenspace Projections | 28 |
| 9.2.3 | Computation of Projection Matrix Factors | 29 |
| 9.3 | Computation of Permutation and Scaling Matrices | 32 |

| | |
|--|-----------|
| 10 Initial Implementation and Results | 35 |
| 10.1 <i>Mathematica</i> Implementation | 35 |
| 10.2 Operation Counts and Precomputation Times | 35 |
| 11 Future Directions | 37 |
| 11.1 Improvement of Algorithmic Efficiency | 37 |
| 11.2 Choice of Basis | 38 |
| 11.3 SPIRAL Factorization | 38 |
| 11.4 Matlab, GAP implementations | 38 |
| 11.5 Recursive Structure of FFTs | 38 |
| 11.6 Computation of Inverse DFT | 39 |
| A Mathematica Code: FFT Generation Algorithm | 39 |

Part I

Literature Review: Recent Results in Generalized FFTs

1 Introduction

Spectral analysis methods play a crucial role in mathematics today and in the pure and applied sciences. For example, the study of Fourier series concerns itself with the decomposition of a periodic function f into a series of complex exponentials [11]:

$$f(x) = \sum_{n=-\infty}^{\infty} c_n e^{inx} \quad (1.1)$$

The amplitudes c_n of these exponentials then constitute the spectrum of the function. Such decompositions have applications to linear partial differential equations, where the exponential functions act as a basis of eigenfunctions of the linear operator associated with the equation. The eigenvalue spectrum of the operator then determines how the solution to the equation depends on the initial and boundary conditions [11, 13]. This series decomposition can be extended to nonperiodic functions by allowing a continuous distribution of frequencies for the constituent complex exponentials, so that a function $f(x)$ decomposes as

$$f(x) = \int_{-\infty}^{\infty} g(\omega) e^{i\omega x} d\omega \quad (1.2)$$

The spectrum of amplitudes $g(\omega)$ is then the Fourier transform of the function $f(x)$. This terminology also illustrates that the Fourier transform itself is a map from one space of functions to another space of functions, possibly over a different domain.

Spectral methods, and the Fourier transform in particular, have significant applications in the physical sciences. In quantum mechanics, for instance, the eigenvalue spectrum of a Hermitian operator such as the Hamiltonian or the angular momentum operator determines the range of observable values for the physical quantity corresponding to that operator. The position-space wave function $\psi(x)$ of a particle describes the distribution of its possible positions states, and can be rewritten in terms of its momentum-space wave function $\hat{\psi}(p)$ in what is essentially a Fourier transform [28]:

$$\psi(x) = \frac{1}{\sqrt{2\pi\hbar}} \int_{-\infty}^{\infty} \hat{\psi}(p) e^{ipx/\hbar} dp.$$

The physical significance of these transforms arises from the natural duality between quantities such as position and momentum and energy and time, which also underlies the famous Heisenberg uncertainty relations $\Delta x \Delta p \geq \hbar/2$ and $\Delta E \Delta t \geq \hbar/2$.

From an engineering perspective, much of modern signal processing concerns determining not only the spectrum of a given input signal, but also how that spectrum will change when the signal passes through particular systems and how to design systems that amplify or isolate certain portions

of the spectrum. Of particular importance to signal processing is the Discrete Fourier Transform (DFT), which converts a function on N evenly spaced points to N amplitudes associated to certain frequencies. In keeping with the terminology above, these amplitudes are called the Fourier coefficients of the function. This transform allows discrete samples of a continuous signal to yield some information about the spectrum of that signal. Because computational methods operate primarily on such discrete data, the DFT has become ubiquitous in modern signal processing.

Naïve implementations of the DFT require $O(N)$ operations for the construction of each coefficient, resulting in an overall $O(N^2)$ algorithm for the DFT. This $O(N^2)$ complexity severely limits the application of the DFT to large data sets and motivates the search for more efficient implementations of the transform. Any such efficient implementation of the DFT is called a Fast Fourier Transform (FFT). Such FFTs trace back even to Gauss, who determined an efficient interpolation of a planetary orbit between n points from its interpolation on two sets of $n/2$ points. Modern FFTs are derived from the algorithm that Cooley and Tukey developed in the 1960s [5, 24], which computes a DFT on $N = pq$ points first by p transforms of length q and then by q transforms of length p , for a total of $pq(p + q)$ operations. Applied recursively in the case where $N = 2^n$, this algorithm yields a complexity of $O(N \log N)$ for the N -point DFT, a significant improvement over the naïve $O(N^2)$ implementation.

2 Group-Theoretical Fourier Transforms

As discussed above, given a continuous periodic function $f : \mathbb{R} \rightarrow \mathbb{C}$, we can determine some of its frequency spectrum by sampling the function at N points x_0, x_1, \dots, x_{n-1} over one of its periods and computing the DFT on those points. One of the key features of the DFT is that the amplitudes and relative phases of the coefficients it yields do not depend on the time period over which the function was sampled. In particular, this indicates that the DFT is invariant under cyclic shifts of the points $X = \{x_0, x_1, \dots, x_{n-1}\}$. Such shifts correspond to the action of the group $\mathbb{Z}/n\mathbb{Z}$ on this set X . We can then replace the points of X with the elements of $\mathbb{Z}/n\mathbb{Z}$. Since $\mathbb{Z}/n\mathbb{Z}$ also shifts itself in the same manner, this replacement preserves that action of $\mathbb{Z}/n\mathbb{Z}$ on the set over which the function is defined. Finally, we can treat the function $f : \mathbb{Z}/n\mathbb{Z} \rightarrow \mathbb{C}$ as an element of the group \mathbb{C} -algebra $\mathbb{C}(\mathbb{Z}/n\mathbb{Z})$, where the coefficient of $g \in \mathbb{Z}/n\mathbb{Z}$ is $f(g)$. Therefore, there exists a natural mapping of the function $f : X \rightarrow \mathbb{C}$ into the $\mathbb{C}(\mathbb{Z}/n\mathbb{Z})$, and this mapping provides a representation-theoretic interpretation of the DFT and an avenue for its generalization to arbitrary groups.

Suppose G is a finite group with h conjugacy classes, and let V be a $\mathbb{C}G$ -module, so that V is a representation for G . Let \hat{G} denote the set of h equivalence classes of irreducible representations of G , and let ρ_1, \dots, ρ_h be representatives of these equivalence classes. Then V decomposes as

$$V = \bigoplus_{i=1}^h V^i, \tag{2.1}$$

where each V^i is isomorphic to a direct sum of n_i isomorphic copies of ρ_i , so that $V^i \cong n_i \rho_i$. These V^i are referred to as the isotypic subspaces of V and are unique for each $\mathbb{C}G$ -module V . In particular, $\mathbb{C}G$ is itself a $\mathbb{C}G$ -module by left multiplication, and so it too decomposes into a direct sum of

its irreducible representations. The isotypic components in this decomposition correspond to the minimal two-sided ideals of $\mathbb{C}G$, while the irreducible representations correspond to its minimal left ideals. Thus, each element $f \in \mathbb{C}G$ can be written as a unique sum of elements in the representations constituting this direct sum, and these elements provide a generalizations of the Fourier coefficients obtained by the usual DFT.

These coefficients can be written explicitly by considering each irreducible representation ρ_i as a vector space of \mathbb{C} of dimension equal to the degree d_i of the representation. Then each component of $f \in \mathbb{C}G$ corresponds to a vector in \mathbb{C}^{d_i} , each isotypic component of f corresponds to a matrix in $\mathbb{C}^{d_i \times d_i}$, and we can write our isomorphism $\mathbb{C}G = \bigoplus_{i=1}^h V^i \cong \bigoplus_{i=1}^h d_i \rho_i$ as

$$\mathbb{C}G \cong \bigoplus_{i=1}^h \mathbb{C}^{d_i \times d_i}. \quad (2.2)$$

This result restates Wedderburn's Theorem [5, 9]: every complex group algebra is isomorphic to a direct sum of matrix algebras. Because we have a choice of basis for each $\mathbb{C}^{d_i \times d_i}$, this isomorphism is not unique, and any choice of isomorphism D from $\mathbb{C}G$ into this direct sum is called a DFT for the group G . Combined with a choice of basis for $\mathbb{C}G$, we can reformulate the DFT as a $|G| \times |G|$ matrix from the coordinate representation of $f \in \mathbb{C}G$ to the coordinate representation of its transform $\hat{f} \in \bigoplus_{i=1}^h \mathbb{C}^{d_i \times d_i}$. This matrix form also indicates that the DFT is a linear transformation from the input function to the coefficients.

Given a particular DFT D for a group G and an element $f = \sum_{g \in G} f_g g \in \mathbb{C}G$, we can compute the Fourier coefficients in block j of the direct sum as

$$\hat{f} = \sum_{g \in G} f_g \rho_j(g), \quad (2.3)$$

where ρ_j is the irreducible representation corresponding to that block, expressed as a matrix in terms of the basis chosen for the DFT D .

2.1 Abelian Groups

We now illustrate how this concept of generalized Fourier transforms encompasses the familiar cases of spectral analysis discussed in Section 1, all of which then correspond to Fourier transforms on abelian groups. In the case of a finite abelian group G such as $\mathbb{Z}/N\mathbb{Z}$, each irreducible representation of G has degree 1, so that the isotypic subspaces of $\mathbb{C}G$ are all one-dimensional. Therefore, Wedderburn's Theorem states that

$$\mathbb{C}G \cong \bigoplus_{i=1}^{|G|} \mathbb{C}^{1 \times 1} \cong \mathbb{C}^{|G|}, \quad (2.4)$$

and the Fourier coefficients of $f \in \mathbb{C}G$ are simply the coordinates in $\mathbb{C}^{|G|}$ of the image of f under this isomorphism.

We can also reformulate the Cooley-Tukey FFT on $N = pq$ points in terms of a factorization of the group $\mathbb{Z}/N\mathbb{Z}$. In particular, we introduce the chain of subgroups $1 < \mathbb{Z}/p\mathbb{Z} < \mathbb{Z}/N\mathbb{Z}$.

The analysis presented here closely follows that presented by Maslen and Rockmore [16]. The N irreducible representations of $\mathbb{Z}/N\mathbb{Z}$ are all one-dimensional and hence are equal to its irreducible characters. These characters are given by $\zeta_k(j) = \omega^{jk}$, where ω is a primitive N th-root of unity. Thus, by the formula given above for the coefficients, the k th Fourier coefficient of an element $f = \sum_j f_j j \in \mathbb{C}(\mathbb{Z}/N\mathbb{Z})$ is given by

$$X_k = \sum_{j=0}^{N-1} f_j \zeta_k(j) = \sum_{j=0}^{N-1} f_j \omega^{jk}. \quad (2.5)$$

We can reindex the sum into a double sum via the group factorization specified above. Each $j \in \mathbb{Z}/N\mathbb{Z}$ can be written as an element of a coset of the subgroup $q\mathbb{Z}/N\mathbb{Z} \cong \mathbb{Z}/p\mathbb{Z}$, so that $j = i_1 q + i_2$. Let A be a complete set of the representatives for the cosets of $q\mathbb{Z}/N\mathbb{Z}$ in $\mathbb{Z}/N\mathbb{Z}$. Then the sum above becomes

$$X_k = \sum_{a \in A} \sum_{b \in q\mathbb{Z}/N\mathbb{Z}} \zeta_k(a+b) f_{a+b} = \sum_{a \in A} \sum_{b \in q\mathbb{Z}/N\mathbb{Z}} \zeta_k(a) \zeta_k(b) f_{a+b} = \sum_{a \in A} \zeta_k(a) \sum_{b \in q\mathbb{Z}/N\mathbb{Z}} \zeta_k(b) f_{a+b}. \quad (2.6)$$

We note that, in the inner sum, ζ_k acts only on elements of the subgroup $q\mathbb{Z}/N\mathbb{Z}$, so we need only consider the restrictions ($\zeta_k \downarrow q\mathbb{Z}/N\mathbb{Z}$) of the characters ζ_k to this subgroup. Since

$$\zeta_k(i_1 q) = \omega^{i_1 q k} = (\omega^q)^{i_1 k},$$

and ω^q is a primitive p th-root of unity, these restrictions correspond exactly to the p irreducible characters χ_m of $\mathbb{Z}/p\mathbb{Z}$. Consequently, we need compute the inner sum only for $k \in \mathbb{Z}/p\mathbb{Z}$.

We now reformulate the calculation of the Fourier coefficients in two stages, at the expense of some storage space:

- We compute and store $f_1(a, k) = \sum_{b \in q\mathbb{Z}/N\mathbb{Z}} \zeta_k(b) f_{a+b}$ for all $a \in A$ and for all $k \in \mathbb{Z}/p\mathbb{Z}$. Each sum requires p operations, for a total of $p^2 q$ operations.
- We then compute $\sum_{a \in A} \zeta_k f_1(a, k)$ for all $k \in \mathbb{Z}/N\mathbb{Z}$ to obtain the Fourier coefficients. Each sum requires q operations, for a total of $Nq = pq^2$ operations.

The final operation count for these two stages is then $pq(p+q)$ operations, while the total count for the one-stage calculations given by Equation (2.5) is $N^2 = p^2 q^2$. This algorithm therefore represents a significant improvement in complexity, and ultimately leads to the $O(N \log N)$ running time of the Cooley-Tukey FFT.

Abelian groups other than the cyclic groups present similar DFTs and FFTs. We consider an example from Maslen and Rockmore [16], called the 2^n -factorial design, which corresponds to functions over $(\mathbb{Z}/2\mathbb{Z})^n$. Such a set of data might arise from the effects of n independent factors on the growth of a crop of plants, where each factor can assume either a high value or a low value. The irreducible representations of $(\mathbb{Z}/2\mathbb{Z})^n$ are, as above, all one-dimensional, and are given by $\chi_v(w) = (-1)^{\langle v, w \rangle}$, where $v, w \in (\mathbb{Z}/2\mathbb{Z})^n$ and $\langle v, w \rangle$ represents the usual inner product taken mod 2.

We decompose the computation of the Fourier coefficients by Equation (2.3) into n stages according to the chain of subgroups

$$1 < \mathbb{Z}/2\mathbb{Z} < (\mathbb{Z}/2\mathbb{Z})^2 < \dots < (\mathbb{Z}/2\mathbb{Z})^{n-1} < (\mathbb{Z}/2\mathbb{Z})^n$$

in a manner similar to the separation performed in the Cooley-Tukey FFT to yield a transform in $3 \cdot 2^n \log 2^n$ operations. The DFT under consideration here is equivalent to the well-known Walsh-Hadamard transform, and the FFT algorithm we generate then represents a sparse factorization of the DFT matrix associated with the transform. Much work has already been done on FFTs for abelian groups. The key result, presented by Clausen and Baum [5], is that the combination of the methods of Cooley and Tukey and the so-called chirp- z and Rader transforms yields FFTs for all abelian groups G in fewer than $8|G| \log |G|$ operations.

We can even extend this group theoretic formulation to infinite groups, under certain restrictions which we detail below. Consider a function $f : \mathbb{R} \rightarrow \mathbb{C}$ that is 2π -periodic. We reformulate f as a function on the unit circle S^1 , which is a compact group. The associated group algebra also has irreducible degree-one representations, although in this case there are an infinite number of them, indexed by \mathbb{Z} . If the elements S^1 are represented by the angle $\theta \in [0, 2\pi)$, then the n th irreducible representation is given by $\chi_n(\theta) = \frac{1}{2\pi} e^{-in\theta}$. Thus, the Fourier coefficients f_n are determined by the integral

$$f_n = \int_0^{2\pi} f(\theta) \chi_n(\theta) d\theta = \frac{1}{2\pi} \int_0^{2\pi} f(\theta) e^{-in\theta} d\theta, \quad (2.7)$$

In general, this integral exists because S^1 is compact and thus has finite volume under the measure $d\theta$. The inverse transform is as specified in Equation (1.1). We often require that the function f be band-limited, so that $f_n = 0$ for $|n| > B$ for some B . This restriction allows us to replace the infinite sum in Equation (1.1) with a finite sum from $-B$ to B . In this case, there exists a sampling method that allows the exact computation of the coefficients from $2B + 1$ samples of the function f [24]; the finite-case FFT then makes such computations efficient.

Finally, the Fourier transform over \mathbb{R} detailed above in Equation (1.2) provides an example of a transform over a noncompact abelian group. As above, the Fourier coefficients are given by an integral, although this time over \mathbb{R} :

$$\hat{f}(\omega) = \frac{1}{2\pi} \int_{-\infty}^{\infty} f(x) e^{-i\omega x} dx. \quad (2.8)$$

This result reflects that there are an infinite number of irreducible representations of \mathbb{R} , as in the S^1 case, although now they are indexed by \mathbb{R} instead of by \mathbb{Z} . In order that these coefficients $\hat{f}(\omega)$ exist, we must place certain restrictions on f . These include that it be band-limited [24] or that it belong to a special class of function that ultimately decay faster than e^{-x^2} [13]. In the band-limited case, the Fourier coefficients have finite support, so as in the S^1 case, there exist sampling methods that then admit judicious use of the FFT to create efficient transforms.

2.2 Nonabelian Groups

We now explore the generalization of these Fourier transforms to the case of nonabelian groups G . Among the finite groups of particular interest are the symmetric groups S_n , the dihedral groups D_{2n} , solvable and supersolvable groups, and finite groups of Lie-type, including several types of matrix groups over finite fields [16, 24]. Such groups have applications that include the analysis of ranked data or the construction of error-correcting codes; see Section 5 for more details on the potential applications of spectral analysis on these groups.

Fourier transforms on finite nonabelian groups are even useful for understanding or manipulating the corresponding group algebras, as multiplication of elements in the group algebra corresponds to a convolution operation on the coefficients of the elements. In turn, this convolution becomes simple pointwise multiplication in the transform space, so it can be computed through two DFTs, a multiplication, and an inverse DFT. Implemented naïvely, either approach requires $O(|G|^2)$ operations, but an FFT algorithm can reduce the complexity of the transform approach to at worst $O(|G|^{3/2})$ and in some cases $O(|G| \log^c |G|)$, depending on the efficiency of the FFT algorithm itself.

Such applications therefore motivate us to determine how efficient the FFTs for a given group can be. Such questions are usually stated in terms of the complexity $L_s(G)$ of a group G , which is defined to be the minimum of the complexities of all the possible DFT matrices associated with G [3, 5, 16]. A number of bounds on the complexity of nonabelian FFTs have already been established. Clausen [3] states that, for a finite group G , the complexity $L_s(G)$ of a generalized FFT on G is bounded above by

$$L_s(G) \leq \min_C \{(s(C) - l(C)) \cdot |G| + 7\sqrt{q(C)}|G|^{3/2}\},$$

where the minimum is taken over all possible chains C of subgroups $1 = G_0 < \dots < G_n = G$ of G , where $l(C)$ is the length n of the chain, and where q and s are the maximum and sum of the indices $[G_{i+1} : G_i]$ determined by the chain. While this is a significant improvement over the trivial bound of $2|G|^2$ operations, the existence of $O(|G| \log |G|)$ FFTs for abelian groups demonstrates that this is by no means a sharp bound. Also of interest are lower bounds on the complexity of an FFT, so that we can determine when we have an optimal algorithm. Clausen and Baum [5] state that $O(|G|)$ is the best lower bound that has been proved so far in computational models that allow arbitrarily large multiplications, although if limits are placed on those multiplications the lower complexity bound grows to $O(|G| \log |G|)$.

Better complexity bounds have been determined for specific families of groups. Clausen and Baum [5] cite a result that if G is a solvable group with a monomial DFT, then its complexity is less than $8.5|G| \log |G|$. Since all supersolvable groups meet this criterion, this result applies to them as well. Maslen [14] proves that the symmetric group S_n has an FFT involving $O(|S_n| \log^2 |S_n|)$ operations; further discussion of symmetric group FFTs occurs in Section 4.2. Maslen and Rockmore [16] also demonstrate that the complexity of $GL_n(F_q)$ is bounded above by $\frac{1}{2}2^{2q}q^{2n-2}|GL_n(F_q)|$.

As in the commutative case, these Fourier transforms on finite groups can be extended to a compact group G provided certain constraints apply [24]. In particular, the irreducible representation

of G must be finite-dimensional, and square-integrable functions have a countable number of coefficients such that the Fourier decomposition converges. The Fourier coefficients are then computed as integrals over the group with respect to the Haar measure. Among the groups that meet these criteria are the classical compact Lie groups, such as $O(n)$, $SO(n)$, $U(n)$, $SU(n)$, and $Sp(n)$. Maslen [24] has made progress on bounds for transforms of band-limited functions on $U(n)$, $SU(n)$, and $Sp(n)$. Driscoll and Healy, furthermore, treat the 2-sphere S^2 as a homogeneous space of $SO(3)$ to construct an FFT that yields a spherical harmonic decomposition for a band-limited function on S^2 .

In the noncompact case, Chirikjian [2] has made some progress with respect to Fourier transforms for the Euclidean motion group $SE(3) = SO(3) \times R^3$, although a general theory of generalized FFTs on noncompact nonabelian groups has not yet been developed. Section 6 addresses current open questions in this and other aspects of FFT research.

3 Algorithmic Approaches to FFTs

3.1 Decimation-in-Time Algorithms

We now discuss different methods of constructing FFT algorithms. The majority of current FFT algorithms employ a decimation-in-time or separation of variables approach, in which the elements of the group G are factored according to a particular chain of subgroups $1 = G_0 < G_1 < \dots < G_n = G$. As in the Cooley-Tukey case, the frequencies are then computed through a series of nested sums. The factorization that the subgroup chain affords reduces the total number of operations that must be performed to compute the sums at each stage.

This separation of variables is expressed graphically in the so-called Bratteli diagram for the chain of subgroups, which depicts how the irreducible representations of each G_i factor when restricted to G_{i-1} . We borrow the notation Ram uses in his explanation of these concepts [23]. In general, if V^ρ is an irreducible $\mathbb{C}G_i$ -module, then $V^\rho \downarrow G_{i-1}$ will be a direct sum of irreducible $\mathbb{C}G_{i-1}$ -modules, so that

$$V^\rho \downarrow G_{i-1} \cong \bigoplus_{\sigma \in \hat{G}_{i-1}} c_\sigma^\rho V^\sigma,$$

where \hat{G}_{i-1} is the set of equivalence classes of irreducible representations of G_{i-1} , and c_σ^ρ is the multiplicity of the representation σ in ρ . In the Bratteli diagram, then, the irreducible representations of each subgroup in the chain are depicted as nodes arranged vertically, and c_σ^ρ arrows are drawn from each $\sigma \in \hat{G}_{i-1}$ to ρ . As an illustration, Figure 1 depicts a Bratteli diagram for S_4 , which clearly shows, for example, that the restriction of the (3, 1) irreducible representation of S_4 to S_3 yields the direct sum of a (3) and a (2, 1) irreducible representation for S_3 .

Consider a path drawn from $\sigma \in \hat{G}_i$ to $\rho \in \hat{G}_n$. This path then represents a G_i -invariant subspace of V^ρ with dimension d_σ . Consequently, the paths from \emptyset , the root node in the diagram, to the representations of G represent one-dimensional subspaces and hence correspond to basis vectors for the irreducible representations of G .

In addition, the basis vectors that these paths in the Bratteli diagram represent then interact particularly well with the process of restricting representations to subgroups in the chain. Suppose

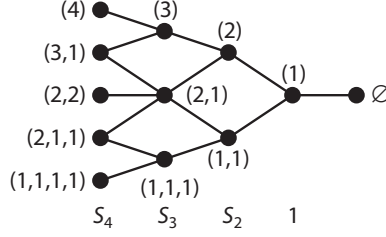


Figure 1: The Bratteli diagram for the subgroup chain $S_4 > S_3 > S_2 > 1$.

that $\rho \in \hat{G}$ and V^ρ is an irreducible $\mathbb{C}G_{n-1}$ corresponding to ρ with basis $B^\rho = \{v_L\}$ determined by these paths in the diagram. Then restricting V^ρ to G_{n-1} yields a decomposition

$$V^\rho = V^{\sigma_1} \oplus V^{\sigma_2} \oplus \dots \oplus V^{\sigma_k}$$

such that $\sigma_i \in \hat{G}_{n-1}$. Furthermore, B^ρ partitions into bases B^{σ_i} for these V^{σ_i} that correspond to the paths going from the \emptyset representation at G_0 to σ_i . These bases B^{σ_i} then can be similarly partitioned upon the restriction of the v^{σ_i} s to G_{n-2} , and so on down the chain of subgroups. This basis B^ρ is called a *seminormal basis*, a *Gelfand-Tsetlin basis*, or an *adapted basis* [16, 23]. This last name indicates that the basis is adapted to the chain of subgroups chosen above.

A seminormal basis for the representations of G provides a convenient form for the matrix representations of $\mathbb{C}G$ [23]. Each element in $\mathbb{C}G$ can be viewed as a \mathbb{C} -linear transformation on the irreducible $\mathbb{C}G$ -modules V^ρ , and hence has a matrix representation for each choice of basis for these modules. In particular, if $\mathbb{C}^{d_\rho \times d_\rho}$ is the matrix algebra corresponding to $\rho \in \hat{G}$ with respect to the basis $\{v_k\}$, then its ij th coefficient indicates how v_j maps onto v_i . Because the seminormal basis partitions under restriction to $\mathbb{C}G_i$, it gives a matrix representation for $\mathbb{C}G$ that decomposes into a direct sum of matrix representations for $\mathbb{C}G_i$ under this restriction. Furthermore, we can index these matrix coefficients by pairs of basis vectors, or equivalently pairs of paths in the Bratteli diagram that have the same endpoints.

Decimation-in-time algorithms rely upon writing elements of the group G as elements of the double cosets of G_{n-1} , where the coset representatives are drawn from a fixed set A [16]. These representatives are subsequently represented as elements of the double cosets of G_{n-2} and so on until an entire factorization of the group with respect to this chain is reached. Then, just as in the algebraic approach to the Cooley-Tukey algorithm presented in Section 2.1, the computation of the Fourier coefficients can be approached in stages relating to the chosen chain of subgroups. Finally, the seminormal basis ensures that the representations in the sum will restrict to direct sums of representations of subgroups, reducing the number of terms in each sum.

3.2 Decimation-in-Frequency Algorithms

Decimation-in-frequency algorithms present an approach to these FFTs that is essentially dual to the decimation-in-time approach. In particular, the same concepts of Bratteli diagrams and semi-normal bases are used, but in these algorithms the frequency space is systematically decomposed according to the irreducible representations of the chosen chain of subgroups.

We illustrate this approach with an algebraic description of the Gentleman-Sande FFT [15, 24]. In order to do so, we first discuss the notion of a separating set for a representation V of a group G . Recall that V decomposes into isotypic subspaces V^{ρ_i} , as in Equation (2.1). Consider a set of simultaneously diagonalizable linear transformations $\{T_1, \dots, T_k\}$ on V such that the eigenspaces of the T_j s are direct sums of the isotypic components of V . Then applying each T_j to each V^{ρ_i} yields a list $c_i = (\lambda_{i1}, \dots, \lambda_{ik})$ of the eigenvalues of each T_j on V^{ρ_i} . If $c_i = c_j$ implies that $V^{\rho_i} = V^{\rho_j}$, we say that the T_j s form a separating set for V . Essentially, then, the T_j s suffice to distinguish among the isotypic components of V .

Consider now the case of a group algebra $\mathbb{C}G$ acting on itself as a left- $\mathbb{C}G$ module. Then $\mathbb{C}G$ is also a representation of G , as discussed in Section 2, and the elements of $\mathbb{C}G$ act as linear transformations of $\mathbb{C}G$. Thus, we can represent a separating set for $\mathbb{C}G$ as a collection of elements of $\mathbb{C}G$. One such separating set is the collection of centrally primitive idempotents $\{e_1, \dots, e_h\}$ that correspond to the two-sided ideals of $\mathbb{C}G$, as e_i has eigenvalue 1 on the i th isotypic component and 0 elsewhere. These idempotents form a basis for the space of conjugacy class sums in $\mathbb{C}G$, so any linear combination of class sums is also a diagonalizable linear transform on $\mathbb{C}G$, and any set of them can be diagonalized simultaneously [26]. This result also recommends the set of class sums as a separating set for $\mathbb{C}G$ [15].

We now address the DFT from a separating set perspective. We borrow the algebraic formulation of the conventional DFT as an isomorphism of $\mathbb{C}(\mathbb{Z}/N\mathbb{Z})$ from Section 2.1. Since each of the representations of $\mathbb{C}(\mathbb{Z}/N\mathbb{Z})$ is one-dimensional, so are the isotypic subspaces of $\mathbb{C}(\mathbb{Z}/N\mathbb{Z})$. Hence, they correspond to the Fourier coefficients we seek to recover. The representations are given by $\zeta_j(i) = \omega^{ij}$, where ω is a primitive N th root of unity. Furthermore, the conjugacy class sum $T_1 = \bar{1}$ separates these isotypic components, since $\zeta_j(\bar{1}) = \omega^j$ and each of these is distinct for distinct j . Thus, each isotypic component V_j corresponds to the eigenspace of T_1 with eigenvalue ω^j . To isolate the Fourier coefficients of $f \in \mathbb{C}(\mathbb{Z}/N\mathbb{Z})$, we then compute the projections of f onto these spaces. Doing so by the projection formula

$$f_i = \frac{d_i}{|G|} \sum_{g \in G} \chi_i(g)^* \rho(g) \quad (3.1)$$

given in [15] or [26] requires $O(N)$ operations for each of the N coefficients, however, which yields an $O(N^2)$ algorithm.

The Gentleman-Sande FFT, and decimation-in-frequency algorithms in general, take advantage of a chain of subgroups of G to compute the Fourier coefficients in a series of projections, just as the decimation-in-time algorithms construct the coefficients in a series of sums. The idea in the Gentleman-Sande FFT is to first consider the effect of the class sum $T_q = \omega^q$, which is a separating

set for $\mathbb{C}(\mathbb{Z}/N\mathbb{Z})$ as a $\mathbb{C}(q\mathbb{Z}/N\mathbb{Z}) \cong \mathbb{C}(\mathbb{Z}/p\mathbb{Z})$ -module. Thus, we first project $f \in \mathbb{C}(\mathbb{Z}/N\mathbb{Z})$ onto the eigenspaces W_0, \dots, W_{q-1} of T_q , each of which then consists of a direct sum of q isotypic subspaces of $\mathbb{C}(\mathbb{Z}/N\mathbb{Z})$:

$$W_k = V_k \oplus V_{k+p} \oplus \dots \oplus V_{k+(q-1)p}. \quad (3.2)$$

Each projection then takes only $O(pq)$ operations to compute, for a total of $O(p^2q)$ operations. Then the projections via T_1 onto the $N = pq$ isotypics of $\mathbb{C}(\mathbb{Z}/N\mathbb{Z})$ as a $\mathbb{C}(\mathbb{Z}/N\mathbb{Z})$ -module take only $O(q)$ operations per coefficient, for a total of $O(pq^2)$ operations [15]. The overall complexity is $O(pq(p+q))$, the same as for the Cooley-Tukey FFT.

Decimation-in-frequency algorithms for a finite group G follow a pattern similar to that of the Gentleman-Sande FFT. If G is nonabelian, however, some of the isotypic subspaces of CG will have dimension greater than one and will no longer correspond directly to the Fourier coefficients of CG . Recalling that pairs of paths in the Bratteli diagram correspond to the seminormal basis vectors of CG , any set of linear transformations of CG that differentiates between these paths suffices to determine the coefficients.

3.3 Convolution Algorithms

Other algorithms have been used in the case of the Cooley-Tukey FFT to increase the efficiency of transforms on $\mathbb{Z}/p\mathbb{Z}$ for large prime p . These groups have no nontrivial subgroups, so they are susceptible to neither the decimation-in-time nor the decimation-in-frequency approaches described above. One useful algorithm in this setting is the Rader transform [5, 16], which relates the DFT on p points to a convolution on $(\mathbb{Z}/p\mathbb{Z})^\times$, which is a cyclic group of order $p-1$. If $p-1$ contains a number of small prime factors, these convolutions themselves are efficient by the usual Cooley-Tukey methods and in turn provide an FFT for these p points. Similarly, the chirp- z transform uses a different change of variables to relate the DFT to a convolution on a larger cyclic group. If this cyclic group has order equal to a power of 2, this convolution can again be performed efficiently by Cooley-Tukey.

4 The Symmetric Group

We now address the representations of the symmetric groups, S_n , and the FFTs these representations have afforded to date. In order to determine such FFTs, we must first

- characterize the irreducible representations of S_n ,
- identify a chain of subgroups of S_n such that the irreducible representations of S_n restrict to these subgroups conveniently, and
- determine a basis for $\mathbb{C}S_n$ that is adapted to this chain of subgroups.

Much of this has been done classically, although in recent years reformulations and generalizations of these classical approaches have emerged.

4.1 Representations of S_n

Much of the classical work on the representations of the symmetric group was performed by Alfred Young in the late 1920s [25]. James and Kerber [12] modernizes Young's approach significantly and remains a canonical reference on this classical characterization of these representations, and the following material draws heavily from their analysis. At the center of Young's formulation are partitions, Ferrers diagrams, and Young tableaux. A proper partition α of n , written $\alpha \vdash n$, is any nonincreasing sequence $\{\alpha_i\}$ of integers that sum to n ; for example, $(4, 2, 1, 1, 0, \dots)$ is a proper partition of 8. Then the Ferrers diagram corresponding to $\alpha \vdash n$ is a left-aligned arrangement of n boxes such that the i th row contains α_i boxes. Finally, a Young tableaux is formed from a Ferrers diagram by placing the numbers 1 through n in the boxes of the diagram. If the numbers increase left to right across rows and down columns, the tableaux is standard.

Each tableau λ determines a subgroup S_λ of S_n in the following way. Let S_{λ_i} be the subgroup of S_n that fixes everything except the $|\lambda_i|$ elements in the i th row of the tableau; this subgroup is isomorphic to $S_{|\lambda_i|}$. Then the Young subgroup S_λ is the direct product of all these S_{λ_i} s, which itself is isomorphic to a direct product of all the nontrivial S_{λ_i} s:

$$S_\lambda = \prod_{i=1}^{\infty} S_{\lambda_i} \cong S_{\lambda_1} \times \cdots \times S_{\lambda_k}.$$

Two Young subgroups with the same diagram (or shape) are then isomorphic.

The proper partitions of n determine the irreducible representations of S_n as follows. As a side note, any field is a splitting field for S_n [12, 19] so we need only consider $\mathbb{Q}S_n$ instead of $\mathbb{C}S_n$. Let α be a partition of n , and let α' be the complementary partition, such that the i th row of the Young diagram associated to α' contains the number of boxes in the i th column of α . We can construct such a partition graphically by taking the transpose of the Young diagram associated with α . Let IS_α denote the trivial representation of the Young subgroup S_α , and let $AS_{\alpha'}$ denote the alternating representation of $S_{\alpha'}$. Then

$$i(IS_\alpha \uparrow S_n, AS_{\alpha'} \uparrow S_n) = 1,$$

where i is the intertwining number, which measures the dimensionality of the space of $\mathbb{Q}S_n$ -homomorphisms from $IS_\alpha \uparrow S_n$ to $AS_{\alpha'} \uparrow S_n$ [1]. Since this quantity is one, these two representations share a single copy of an irreducible representation of S_n , which we represent by $[\alpha]$. These representations in fact determine all such irreducibles up to isomorphism. Furthermore, the dimensionality of $[\alpha]$ as a \mathbb{Q} -vector space is equal to the number f_α of standard tableaux in the shape α ; this is easily shown from the path-algebraic formulation given by Ram [23] or by Okounkov and Vershik [19], which we discuss below.

Finally, it is possible to construct matrix representations of these permutations that are adapted to the natural chain

$$1 < S_2 < S_3 < \cdots < S_{n-1} < S_n \tag{4.1}$$

of subgroups of S_n [12], although the details of the process are complicated and not especially pertinent here. The basis affording this matrix representation is called the Young seminormal basis,

and the matrix representation itself is then called the Young seminormal form. Typically, the matrix representations are constructed only for the transpositions $(i - 1 \ i)$, $1 < i \leq n$, as these suffice to generate S_n . This chain of subgroups and associated seminormal basis is precisely what is needed for the construction of FFTs on S_n .

More recently, alternate constructions of these representations have emerged. For example, Jucys and Murphy [17, 18] independently identified elements of QS_n (now called Jucys-Murphy elements in their honor [19, 23]) that generate these matrix representations in a far simpler fashion. These elements arise as the differences of the class sums of the transpositions in S_{i+1} and S_i . Thus, the first Jucys-Murphy element is $M_1 = (1 \ 2)$, the second $M_2 = (1 \ 3) + (2 \ 3)$, and so on. These elements have other desirable properties: for example, they generate a maximal commutative subalgebra of QS_n , and they are simultaneously diagonalizable in the Young seminormal basis.

In addition, Ram [23] and Okounkov and Vershik [19] present path-algebraic formulations of the seminormal bases for S_n . Their formulations rely on the property that S_n has a multiplicity-free character graph. In particular, if V^ρ is an irreducible representation of S_i , then its restriction to S_{i-1} yields a decomposition into irreducible representations of S_{i-1} that is multiplicity free, so that for each $\sigma \in \hat{S}_{i-1}$, there exists at most one isomorphic copy of V^σ in V^ρ . Thus, the Bratteli diagram for the chain of subgroups in Equation (4.1) contains only single edges between each level of nodes, so for each $\rho \in \hat{S}_n$, each path from \emptyset to ρ is uniquely determined by the list of representations it passes through in the graph. As discussed above, these paths correspond to basis vectors in the seminormal basis for the irreducible representations of S_n , so these basis vectors are also determined by such lists.

Ram [23] then constructs his path algebra from pairs of paths with common endpoints in \hat{S}_n , which as discussed above correspond to the seminormal basis vectors of the matrix representation of $\mathbb{C}[S_n]$. In particular, the path algebra has a basis $\{E_{LM}\}$, where L and M have common endpoints, with multiplication defined by

$$E_{ST}E_{PQ} = \delta_{TP}E_{SQ}. \quad (4.2)$$

Thus, the basis vector E_{ST} in the algebra acts as a linear operator that maps the vector v_T corresponding to T onto the vector v_S corresponding to path S . This relation then determines an isomorphism between the algebra and $\mathbb{C}S_n$ such that the E_{ST} s map to the elements of the seminormal basis for the matrix representation of $\mathbb{C}S_n$. This analysis rests only on the property that the group have a multiplicity-free character graph, in which case we call the group an *MC-group*. If the chain

$$1 = G_0 < G_1 < \dots < G_{n-1} < G_n = G$$

provides G with such a character graph, then $\mathbb{C}G$ also has a seminormal basis that can be indexed in this fashion. Such MC-groups include the Weyl groups WB_n , WF_4 , WE_6 , and WE_7 . The same holds for any semisimple algebra H exhibiting a similar chain of subalgebras that generates a multiplicity-free character graph. The Iwahori-Hecke algebras constitute one notable family of such algebras that is closely related to these Weyl groups.

To construct these representations for $\mathbb{C}G$ explicitly, Ram supposes the existence of central elements $z_{k,j}$ for each subalgebra $\mathbb{C}G_k$ arising from the chain of subgroups of G . Since each $z_{k,j}$ is

central in $\mathbb{C}G_k$, its action on an irreducible representation $\rho^{(k)}$ of G_k will be to scale it by some value $c_{k,j}(\rho^{(k)})$. These central elements then assign a set of weights $\{c_{k,j}(\rho^{(k)})\}$ to each node in each path in the Bratteli diagram, and hence to each vector in the seminormal basis. If this list of weights is distinct for distinct paths, then these lists of weights uniquely determine each seminormal basis vector.

Returning to the symmetric group, Ram identifies the class sum of transpositions

$$z_k = \sum_{1 \leq i < j \leq k} (i j) \quad (4.3)$$

as a central element in the algebra $\mathbb{C}S_k$, and shows that the eigenvalues of these elements at the irreducible representations of the S_i s suffice to distinguish paths in the diagram and hence seminormal basis vectors. Furthermore, it suffices to consider the differences $m_k = z_k - z_{k-1}$, which are precisely the Jucys-Murphy elements. In this way, the values of the Jucys-Murphy elements on the irreducible representations of the S_i s yield enough information to construct the Young seminormal basis for $\mathbb{C}S_n$. In addition, analogous elements for other groups or other algebras provide similar seminormal representations, which Ram details in subsequent sections of his paper [23].

Finally, this path algebra affords a generalization of the Young tableaux used in the construction of the symmetric group to other MC-groups. Ram [23] explicitly constructs such a system of tableaux for WB_n , while Clausen [4] explores this notion for supersolvable groups as well. In essence, moving from node σ at level i to node ρ at level $i + 1$ of the Bratteli diagram adds a box to the tableau of σ according to some rule. For the symmetric group, this rule is that the resulting tableau must represent a proper partition of $i + 1$, while for the Weyl group WB_n , a pair of partitions is constructed, each of which must remain proper. Furthermore, the paths from \emptyset to σ in level i correspond to standard tableaux for the shape determined by σ , constructed by placing k in the box added during the move from level $k - 1$ to level k . This correspondence also elegantly explains why the degree of each irreducible representation of S_n equals the number of standard tableaux in the associated shape.

4.2 FFTs on S_n

To date, significant work has been done on FFTs on the symmetric group from a decimation-in-time perspective. Clausen and Baum produced the first promising results in 1989 with a proof that the complexity of S_n is bounded above by $\frac{1}{2}(n^3 + n^2)n!$ operations [3] and in 1993 with a fairly explicit implementation of both a DFT and an inverse DFT for S_n , each requiring that number of operations [6]. Their results arise from a sparse factorization of the DFT matrix based on Young's seminormal form at each S_i in the chain specified in Equation (4.1) [6, 16].

Maslen's 1998 paper [14] improves upon this bound with a decimation-in-time algorithm yielding a DFT on S_n in fewer than $\frac{3}{4}n(n-1)n!$ operations. His method relies on a separation of variables at the scalar level, rather than the matrix separation that Clausen and Baum employ. The commutativity of these scalars allows more sophisticated rearrangement of the sums involved in constructing

the Fourier coefficients. This rearrangement entails a more complex indexing scheme based on the paths in the Bratteli diagram rather than only on the subgroups in the chain.

5 Applications

Applications for generalized Fourier transforms abound in engineering, mathematics, and the physical and social sciences. Fourier transforms on the symmetric group have natural applications to the spectral analysis of ranked data, for example. Each voter effectively creates a permutation in S_n by ranking their n candidates, so that the final tallies of votes yield a function on S_n which can be analyzed using the generalized Fourier transforms described above. Diaconis [7] identifies the decomposition of CS_n into its isotypic components as the key to understanding the effects of candidates on ranking preferences. Such transforms have been applied to partially ranked data as well [24].

The symmetric group is not the only finite group on which Fourier analysis presents applications. The group $SL_n(F_p)$ of two-by-two matrices with determinant one over the finite field F_p has applications in coding theory, particularly with respect to low-density parity check codes, and in graph theory [24]. Maslen, Orrison, and Rockmore [15] discuss applications of generalized Fourier analysis to the study of distance-transitive graphs; while their analysis includes examples that relate primarily to the symmetric group, other groups could also be used in this context. In addition, transforms on other finite groups may yield lossy data compression algorithms with better performance than such standards as JPEG, which is based on the Discrete Cosine Transform [5]. Finally, such transforms have applications to quantum mechanics and quantum computing. In particular, Shor's quantum factoring algorithm relies on transforms on the cyclic group $(\mathbb{Z}/n\mathbb{Z})^\times$, and it is conjectured that generalized FFTs may provide an efficient quantum algorithm for the graph isomorphism problem [24].

Fourier transforms on nonabelian compact groups also have significant applications. The spherical harmonics are orthogonal functions on the unit sphere S^2 that yield a series decomposition for functions on S^2 analogous to that provided by the Fourier transform on S^1 . Such decompositions have applications in physics, where they play a key role in describing the distributions of electrons in atomic orbitals [10, 28]. In addition, any frequency analysis of spherically distributed data rests on these spherical harmonic functions. Such analysis arises in global circulation modeling, control theory, and computer vision models, for example [16]. As mentioned above, Driscoll and Healy [8] present an efficient algorithm for the computation of spherical harmonics for band-limited functions on S^2 through the analysis of FFTs on the group $SO(3)$, which acts transitively on S^2 .

Applications of Fourier transforms on noncompact groups may even exist. Chirikjian and Kyatkin [2] present their analysis of transforms on the Euclidean motion group $SE(3)$ in order to describe the configuration space for certain robotic arms. Such transforms may also apply to the configuration space of proteins as they fold into their appropriate forms and hence may provide a convenient means of describing these folded states [24].

6 Open Questions

We conclude with a number of open questions and directions for future development in the fields of generalized FFTs and noncommutative harmonic analysis. Many of these derive from papers by Maslen and Rockmore [16, 24].

- Although certain classes of groups present $O(|G| \log |G|)$ or $O(|G| \log^2 |G|)$ FFTs algorithms, there exists no universal $O(|G| \log^c |G|)$ bound on the complexity of generalized FFTs for finite groups. One approach to this problem may involve the determination of FFTs for all the groups in the classification of finite simple groups. In particular, this goal requires better FFTs for finite groups of Lie type and for matrix groups.
- It remains to be seen if decimation-in-frequency algorithms can be generalized to match the level of progress that has been made with decimation-in-time algorithms. These decimation-in-frequency formulations are particularly appealing because their theory more closely reflects the module-theoretic underpinnings of group representation theory.
- Similarly, no noncommutative analogues of the important Rader and chirp- z transforms are currently known. If they exist, such analogues may relate transforms between groups that have no group-subgroup relationship.
- FFTs for groups seem to rest mainly on the semisimplicity of the group algebra $\mathbb{C}G$. Because of this result, it seems likely that there exist FFTs for band-limited functions on all semisimple Lie groups.
- Much of the theory of generalized Fourier transforms on noncompact groups such as $SE(n)$ is in its initial stages. Such transforms would require the development of suitable sampling algorithms for these groups as a first step.
- The recursive nature of many of the known FFTs algorithms suggests that there exist effective parallel implementations of these algorithms. Decimation-in-frequency algorithms in particular would seem to admit parallel implementations because of the explicit separation of frequency space they entail.

Part II
Current and Future Work

7 Introduction

We now describe an algorithm for the precomputation of a decimation-in-frequency fast Fourier transform on the symmetric group, S_n . Specifically, we construct one that delivers Young's seminormal matrix representations for S_n , as this representation is adapted to the chain $1 < S_2 < \dots < S_n$ of subgroups of S_n and as past decimation-in-time algorithms for S_n [3] have successfully employed this set of representations. We discuss properties of this representation in further detail in Section 8.

In essence, this algorithm computes a matrix factorization of the DFT matrix that takes a coordinate vector for the usual group element basis of $\mathbb{C}S_n$ to the coordinates of the usual matrix-element basis for Young's seminormal representation. Furthermore, each of the factors in the factorization corresponds to a separation of the frequency space. An additional permutation matrix and diagonal row-scaling matrix are required in the factorization to ensure that the resulting coefficients are scaled correctly and are in a convenient order.

8 Representation Theory of S_n

8.1 DFTs as a Change of Basis

We further develop the idea of the DFT as a change of basis on the group ring $\mathbb{C}G$ as a vector space over \mathbb{C} . Suppose G is a finite group with h classes of irreducible representations, so that by Wedderburn's Theorem there exists an isomorphism D from $\mathbb{C}G$ to $\bigoplus_{i=1}^h \mathbb{C}^{d_i \times d_i}$. Let $E_{k,ij}$ be the usual matrix basis element consisting of the matrix with a 1 in the i th row and j th column of the k th block in the matrix algebra, and let $u_{k,ij}$ be its preimage in $\mathbb{C}G$. Furthermore, let $U_{k,ij}$ be the space spanned by $u_{k,ij}$. Then by the isomorphism D afforded by Wedderburn's theorem,

$$\mathbb{C}G = \bigoplus_{k=1}^h \bigoplus_{j=1}^{d_k} \bigoplus_{i=1}^{d_k} U_{k,ij}.$$

D maps $U_{k,ij}$ onto the space spanned by $E_{k,ij}$, so each of these $U_{k,ij}$ s corresponds to one of these Fourier coefficient spaces in the matrix algebra. Thus, $\mathbb{C}G$ can be viewed both as a direct sum of the spaces spanned by the group elements of G and as a direct sum of these Fourier spaces $U_{k,ij}$; the former corresponds to the time domain of the classical DFT, while the latter corresponds to the frequency domain. The matrix representation of D with respect to the bases $\{g\}_{g \in G}$ of $\mathbb{C}G$ and $\{E_{k,ij}\}$ of the matrix algebra then transforms a coordinate vector for this standard group element basis into a coordinate vector for the basis $\{u_{k,ij}\}$ of $\mathbb{C}G$.

Example 8.1 We illustrate this decomposition explicitly for S_3 . We recall that S_3 has three conjugacy classes and thus three classes of irreducible representations, two of which are one-dimensional

and the third of which is two-dimensional. Thus,

$$\mathbb{C}S_3 \cong \mathbb{C}^{1 \times 1} \oplus \mathbb{C}^{2 \times 2} \oplus \mathbb{C}^{1 \times 1}.$$

We can then write $\mathbb{C}S_3$ as

$$\mathbb{C}S_3 = U_{1,11} \oplus U_{2,11} \oplus U_{2,12} \oplus U_{2,21} \oplus U_{2,22} \oplus U_{3,11},$$

where, for example, the spaces $U_{1,11}, U_{2,11} \oplus U_{2,21}, U_{2,12} \oplus U_{2,22}, U_{3,11}$ form irreducible left $\mathbb{C}S_3$ -modules, since they correspond to the four different columns of the matrix algebra. Computing these columns using the algorithm detailed below in Section 8.3, the matrix representation of the DFT that takes an element of $\mathbb{C}S_3$ into its seminormal matrix representation is then the change of basis matrix

$$D = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & -\frac{1}{2} \\ 0 & 0 & -\frac{3}{4} & \frac{3}{4} & \frac{3}{4} & -\frac{3}{4} \\ 0 & 0 & -1 & -1 & 1 & 1 \\ 1 & 1 & -\frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} \\ 1 & -1 & -1 & 1 & -1 & 1 \end{pmatrix} \quad (8.1)$$

Such matrices are precisely the ones we wish to factor in our algorithm. ■

8.2 Combinatorial Objects and Techniques

We now describe in more detail some of the combinatorial objects used in the construction of Young's seminormal matrix representations for S_n , some of which were introduced in Section 4.

Definition 8.2 Let $\alpha = (\alpha_1, \alpha_2, \dots)$ be a proper (decreasing) partition of n . Then the *Ferrers diagram* associated with α is the left-aligned arrangement of n boxes so that row k of the arrangement has α_k boxes. A *Young tableau* is a Ferrers diagram that has been filled with the numbers 1 through n . Such a tableau is *standard* if each row and each column of the tableau is in increasing order left to right and top to bottom, respectively. The number of standard tableaux of shape α is denoted f^α . ■

Example 8.3 We examine these objects in the case of $n = 3$. As $n = 3$ has three possible proper partitions, it also has three possible Ferrers diagrams:

$$(3) \mapsto \begin{array}{|c|c|c|} \hline \square & \square & \square \\ \hline \end{array} \quad (2, 1) \mapsto \begin{array}{|c|c|} \hline \square & \square \\ \hline \square & \\ \hline \end{array} \quad (1, 1, 1) \mapsto \begin{array}{|c|} \hline \square \\ \hline \square \\ \hline \square \\ \hline \end{array}$$

These Ferrers diagrams for $n = 3$ present the following four standard tableaux:

$$\begin{array}{|c|c|c|} \hline 1 & 2 & 3 \\ \hline \end{array} \quad \begin{array}{|c|c|} \hline 1 & 2 \\ \hline 3 & \\ \hline \end{array} \quad \begin{array}{|c|c|} \hline 1 & 3 \\ \hline 2 & \\ \hline \end{array} \quad \begin{array}{|c|} \hline 1 \\ \hline 2 \\ \hline 3 \\ \hline \end{array}$$

Thus, for $n = 3$, $f^{(3)} = f^{(1,1,1)} = 1$ and $f^{(2,1)} = 2$. ■

Each standard tableau specifies a way to construct a proper partition of n by adding boxes one at a time to the diagram with no boxes, in such a way that at each stage the resulting diagram represents a proper partition. Thus, we have a one-to-one correspondence between standard tableaux for n and sequences of proper partitions of k , $k = 1, \dots, n$, where the box containing k is added at the k th stage. As an example, consider that

$$\begin{array}{|c|c|c|} \hline 1 & 2 & 5 \\ \hline 3 & 4 & \\ \hline \end{array} \text{ is equivalent to } \emptyset \rightarrow \square \rightarrow \square \square \rightarrow \begin{array}{|c|c|} \hline \square & \square \\ \hline \square & \\ \hline \end{array} \rightarrow \begin{array}{|c|c|c|} \hline \square & \square & \square \\ \hline \square & \square & \\ \hline \end{array} \rightarrow \begin{array}{|c|c|c|c|} \hline \square & \square & \square & \square \\ \hline \square & \square & \square & \\ \hline \end{array}$$

We relate these tableaux to the seminormal basis for irreducible representations of S_n . As stated above, the irreducible representations of S_n are in bijective correspondence with the proper partitions of n and hence with Young diagrams for n . Furthermore, the standard tableaux correspond to chains of partitions of 1 through n constructed by adding blocks one at a time. Thus, each standard tableau for n corresponds to a path through the Bratteli diagram for S_n . Because this diagram is multiplicity-free, each path starting at \emptyset and ending at the irreducible representation ρ for S_n corresponds to a basis vector in the basis adapted to the subgroup chain $1 < S_2 < \dots < S_n$. We then have the following theorem:

Theorem 8.4 (Sagan [25, 2.5.2]) *If λ is a partition for n and S^λ the corresponding irreducible representation for S_n (i.e., simple $\mathbb{C}S_n$ -module), then the standard tableaux of shape λ correspond to a basis for S^λ , and $\dim_{\mathbb{C}} S^\lambda = f^\lambda$. ■*

Furthermore, there exists a order on standard tableaux based on the relative position of the letters in the boxes. Since this order derives from the placement of the letters in descending order, it is called the last-letter order on these tableaux [12]. As an example (taken from [6]), the five standard tableaux of shape $(3, 2)$ are ordered as follows:

$$\begin{array}{|c|c|c|} \hline 1 & 3 & 5 \\ \hline 2 & 4 & \\ \hline \end{array} < \begin{array}{|c|c|c|} \hline 1 & 2 & 5 \\ \hline 3 & 4 & \\ \hline \end{array} < \begin{array}{|c|c|c|} \hline 1 & 3 & 4 \\ \hline 2 & 5 & \\ \hline \end{array} < \begin{array}{|c|c|c|} \hline 1 & 2 & 4 \\ \hline 3 & 5 & \\ \hline \end{array} < \begin{array}{|c|c|c|} \hline 1 & 2 & 3 \\ \hline 4 & 5 & \\ \hline \end{array} \quad (8.2)$$

In particular, the tableaux are ordered as they are constructed last letter first: the “highest” three tableaux have 5 in a lower row than the “lowest” two tableaux, while among these three higher tableaux the tableau with its 4 in the second row is higher than the two that place the 4 in the first row, and so on. Because two standard tableaux always each contain the numbers 1 through n and because they will differ at some point, this procedure introduces a total order onto the tableaux of each shape. Thus, the last-letter order gives a natural order to the seminormal basis vectors of each irreducible representation of S_n , which we use in the construction of the matrix representations below.

We reformulate this last-letter order in terms of the *content* of boxes in the tableaux:

Definition 8.5 Let t be a tableau, and let b be a box in t in row i and column j . Then the content of b is defined to be $\text{ct}(b) = j - i$. ■

Proposition 8.6 *Let t and t' be two standard tableaux of the same shape α , where α is a proper partition of n . Let $c(t) = (ct(t(n)), ct(t(n-1)), \dots, ct(t(1)))$ and $c(t') = (ct(t'(n)), ct(t'(n-1)), \dots, ct(t'(1)))$, where $t(k)$ is the box of t that contains k . Then $t < t'$ in the last-letter order iff $c(t) > c(t')$ in the lexicographic order.*

Proof: This result follows from translating the description of the last-letter sequence given above into one that uses box content. Suppose $t < t'$, and let k be the largest integer such that k is placed in different boxes in t and t' . Since $t < t'$, k occurs in a higher row in t than it does in t' . Let s and s' be the Young tableaux obtained from t and t' by removing boxes $k + 1$ through n . Since these boxes occur in identical positions in t and t' , s and s' have the same shape, β . Furthermore, $t(k)$ and $t'(k)$ must be on outer corners of β since they are the last boxes to be added to form s and s' . Thus, since the row index of $t(k)$ is less than that of $t'(k)$, the column index of $t(k)$ is larger than that of $t'(k)$, and so $ct(t(k)) > ct(t'(k))$. Since $c(t)$ and $c(t')$ agree up to the $t(k)$ and $t'(k)$ entries, $c(t) > c(t')$ in the lexicographic order.

Similarly, if $c(t) > c(t')$, let k be the entry in the box corresponding to the first place at which these lists disagree. Then $ct(t(k)) > ct(t'(k))$, so by the same outer corner argument, the row index of $t(k)$ is less than that of $t'(k)$, and $t < t'$ in the last-letter order. ■

8.3 Young's Seminormal Matrix Representations

Using the above combinatorial concepts, we can describe Young's seminormal matrix representations for S_n in a relatively straightforward manner. Typically, the seminormal representation is described only for the elements $(i \ i + 1)$ of S_n , as these generate S_n . Throughout, we follow the explanations presented in Clausen and Baum [6] and in James and Kerber [12]. Let α be a partition of n and S^α the corresponding irreducible representation of S_n . We wish to construct the block in the seminormal matrix representation that corresponds to S^α ; we denote this block as σ^α , and note that it is a $f^\alpha \times f^\alpha$ matrix. Denote this block σ^α , with coefficients σ_{ij}^α . Let $E_{\alpha,ij}$ be the matrix with $\sigma_{ij}^\alpha = 1$ and all other coefficients 0. Then the action of $E_{\alpha,ij}$ on S^α maps the j th seminormal basis vector to the i th. Since the seminormal basis vectors are indexed in order by these standard tableaux of shape α , each coefficient in the seminormal representation is indexed by a pair of such tableaux.

Example 8.7 As an example, consider the 2-dimensional irreducible representation $S^{(2,1)}$ of S_3 , whose seminormal basis vectors are indexed by the two standard tableaux of shape $(2, 1)$. Then the matrix element

$$E_{122} = \begin{pmatrix} 0 & & & \\ & 0 & 1 & \\ & 0 & 0 & \\ & & & 0 \end{pmatrix} \text{ is indexed by the standard tableau pair } \left(\begin{array}{|c|c|} \hline 1 & 3 \\ \hline 2 & \\ \hline \end{array}, \begin{array}{|c|c|} \hline 1 & 2 \\ \hline 3 & \\ \hline \end{array} \right)$$

since this element maps the second vector of the seminormal basis for $S^{(2,1)}$ to the first vector in the basis. ■

Consider the set $\{t_i^\alpha\}_{i=1}^{f^\alpha}$ of standard tableaux for a shape α , ordered in the last letter order. The transposition $(k \ k + 1)$ acts on this set by exchanging boxes k and $k + 1$ in the tableau. This operation may or may not lead to a standard tableau: in particular, if k and $k + 1$ lie in the same row or same column, it will not, as either that row or that column will not be in ascending order after the action of the transposition. If that is not the case, however, the action will lead to a valid tableaux, as box $k + 1$ did not build onto box k in the construction of the original tableau.

We construct the seminormal representation of $(k \ k + 1)$ according to these three cases:

- If k and $k + 1$ lie in the same row of t_i^α , set $\sigma_{ii}^\alpha = 1$.
- If k and $k + 1$ lie in the same column of t_i^α , set $\sigma_{ii}^\alpha = -1$.
- Suppose the interchange of k and $k + 1$ in t_i^α generates t_j^α , and that $i < j$. Let $d = |\text{ct}(k + 1) - \text{ct}(k)|$, where as above $\text{ct}(k) = v - u$ if k is in the u th row and v th column of the tableau. Then set

$$\begin{pmatrix} \sigma_{ii}^\alpha & \sigma_{ij}^\alpha \\ \sigma_{ji}^\alpha & \sigma_{jj}^\alpha \end{pmatrix} = \begin{pmatrix} d^{-1} & 1 - d^{-2} \\ 1 & -d^{-1} \end{pmatrix}$$

We note that each element in the matrix is a rational number, so that this construction yields a representation over \mathbb{Q} . Consequently, we could formulate the DFT for S_n through the representation theory of $\mathbb{Q}S_n$. Despite this, we elect to continue working over $\mathbb{C}S_n$ in order to harness the full generality of the representation theory of complex group rings.

Example 8.8 We construct the block $\sigma^{(3,2)}$ of the seminormal matrix representations for the transpositions $(1 \ 2)$, $(2 \ 3)$, $(3 \ 4)$, and $(4 \ 5)$ using the standard tableaux for $\alpha = (3, 2)$ presented in Equation (8.2). Table 1 displays the effects of this action on these tableaux. Using the algorithm above, we have that

$$\begin{aligned} \sigma^{(3,2)}((1 \ 2)) &= \begin{pmatrix} -1 & & & & \\ & 1 & & & \\ & & -1 & & \\ & & & 1 & \\ & & & & 1 \end{pmatrix} & \sigma^{(3,2)}((2 \ 3)) &= \begin{pmatrix} \frac{1}{2} & \frac{3}{4} & & & \\ 1 & -\frac{1}{2} & & & \\ & & \frac{1}{2} & \frac{3}{4} & \\ & & 1 & -\frac{1}{2} & \\ & & & & 1 \end{pmatrix} \\ \sigma^{(3,2)}((3 \ 4)) &= \begin{pmatrix} -1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & \frac{1}{3} & \frac{8}{9} \\ & & & 1 & -\frac{1}{3} \end{pmatrix} & \sigma^{(3,2)}((4 \ 5)) &= \begin{pmatrix} \frac{1}{2} & \frac{3}{4} & & & \\ & \frac{1}{2} & \frac{3}{4} & \frac{3}{4} & \\ 1 & & -\frac{1}{2} & & \\ & 1 & & -\frac{1}{2} & \\ & & & & 1 \end{pmatrix} \end{aligned}$$

As noted above, these matrices allow the construction of the seminormal representation for any $\tau \in S_5$, as this map $\sigma^{(3,2)}$ is a ring homomorphism of $\mathbb{C}S_5$. ■

(3,2)-tableaux in last letter order

| | | | | | |
|----------|---|---|---|---|---|
| σ | $\begin{array}{ c c c } \hline 1 & 3 & 5 \\ \hline 2 & 4 & \\ \hline \end{array}$ | $\begin{array}{ c c c } \hline 1 & 2 & 5 \\ \hline 3 & 4 & \\ \hline \end{array}$ | $\begin{array}{ c c c } \hline 1 & 3 & 4 \\ \hline 2 & 5 & \\ \hline \end{array}$ | $\begin{array}{ c c c } \hline 1 & 2 & 4 \\ \hline 3 & 5 & \\ \hline \end{array}$ | $\begin{array}{ c c c } \hline 1 & 2 & 3 \\ \hline 4 & 5 & \\ \hline \end{array}$ |
| <hr/> | | | | | |
| (12) | C | R | C | R | R |
| (23) | $\begin{array}{ c c c } \hline 1 & 2 & 5 \\ \hline 3 & 4 & \\ \hline \end{array}$ | $\begin{array}{ c c c } \hline 1 & 3 & 5 \\ \hline 2 & 4 & \\ \hline \end{array}$ | $\begin{array}{ c c c } \hline 1 & 2 & 4 \\ \hline 3 & 5 & \\ \hline \end{array}$ | $\begin{array}{ c c c } \hline 1 & 3 & 4 \\ \hline 2 & 5 & \\ \hline \end{array}$ | R |
| (34) | C | R | R | $\begin{array}{ c c c } \hline 1 & 2 & 3 \\ \hline 4 & 5 & \\ \hline \end{array}$ | $\begin{array}{ c c c } \hline 1 & 2 & 4 \\ \hline 3 & 5 & \\ \hline \end{array}$ |
| (45) | $\begin{array}{ c c c } \hline 1 & 3 & 4 \\ \hline 2 & 5 & \\ \hline \end{array}$ | $\begin{array}{ c c c } \hline 1 & 2 & 4 \\ \hline 3 & 5 & \\ \hline \end{array}$ | $\begin{array}{ c c c } \hline 1 & 3 & 5 \\ \hline 2 & 4 & \\ \hline \end{array}$ | $\begin{array}{ c c c } \hline 1 & 2 & 5 \\ \hline 3 & 4 & \\ \hline \end{array}$ | R |

Table 1: Table of action of transpositions on the standard tableaux of shape (3, 2). An R denotes that the boxes to be interchanged lie in the same row, and a C that they lie in the same column.

8.4 Jucys-Murphy Elements and the Content of a Tableau

We now relate the notion of content to the class sums and Jucys-Murphy elements introduced in Section 4.

Definition 8.9 Define the element $z_k \in \mathbb{C}S_k$ by

$$z_k = \sum_{1 \leq i < j \leq k} (i \ j),$$

the class sum of transpositions from S_k . Define the *Jucys-Murphy element* $m_k \in \mathbb{C}S_n$ by $m_k = z_k - z_{k-1}$; then

$$m_k = \sum_{i=1}^{k-1} (i \ k) = (1 \ k) + \cdots + (k-1 \ k). \quad \blacksquare$$

These elements of $\mathbb{C}S_n$ gain their importance from modified versions of a theorem and a proposition from Ram [23]:

Theorem 8.10 (Ram [23, §3]) *Let k be such that $1 \leq k \leq n$, let α be a partition of k . Then $z_k = c_k(\alpha) \cdot \text{id}$ on the irreducible representation S^α of $\mathbb{C}S_k$, where*

$$c_k(\alpha) = \frac{\chi^\alpha(z_k)}{\chi^\alpha(1)} = \sum_{b \in \alpha} \text{ct}(b). \quad \blacksquare$$

Proposition 8.11 (Ram [23, §3]) *Each standard tableau t of shape α , where α is a partition of n , is determined by the list $c(t) = (\text{ct}(t(1)), \text{ct}(t(2)), \dots, \text{ct}(t(n)))$.*

Proof: Let $\alpha^{(k)}$ be the partition of k determined by the first k boxes of t . Since two boxes b and b' have the same content iff they lie on the same diagonal in the diagram, each of the boxes that can be added to $\alpha^{(k)}$ to obtain $\alpha^{(k+1)}$ has a different content value. Hence, the list of content values $(\text{ct}(t(1)), \dots, \text{ct}(t(k)))$ completely determines $\alpha^{(k)}$, so the full list $c(t)$ determines the partitions $\alpha^{(1)} \dots, \alpha^{(n)}$ and thus t . ■

Therefore, the seminormal basis vectors of representations of S_n are eigenvectors of Jucys-Murphy elements, and can be distinguished by the eigenvalues of the Jucys-Murphy elements.

Theorem 8.12 *Suppose α is a partition of n , and v is a seminormal basis vector of S^α associated with the standard tableau t . Then v is distinguished by the eigenvalues of the Jucys-Murphy elements m_2, \dots, m_n .*

Proof: As above, let $\alpha^{(k)}$ be the partition of k determined by the first k boxes of t . Then, for $2 \leq k \leq n$,

$$m_k v = z_k v - z_{k-1} v = c_k v - c_{k-1} v = \left(\sum_{b \in \alpha^{(k)}} \text{ct}(b) \right) v - \left(\sum_{b \in \alpha^{(k-1)}} \text{ct}(b) \right) v = \text{ct}(t(k)) v,$$

so v is an eigenvector of m_k with eigenvalue $\text{ct}(t(k))$. By Proposition 8.11, this list of eigenvalues suffices to determine t , and hence v . ■

To summarize the above results, we now have the following equivalent ways of identifying the space $U_{\alpha,ij}$:

- By the pair of paths through irreducible representations of S_1 through S_n ,
- By the pair of standard tableaux (t_i^α, t_j^α) , where i and j specify the index of the tableaux in the last-letter order,
- By the pair of lists of eigenvalues that correspond to the left and right actions of the Jucys-Murphy elements m_2, \dots, m_n on the space $U_{\alpha,ij}$.

We will interchange freely between these descriptions of the Fourier spaces.

9 Decimation-In-Frequency Algorithm Theory

We now describe how we employ this representation theory for S_n to compute a sparse factorization of the desired DFT matrix in stages corresponding to the subgroups in the chain $1 < S_2 < \dots < S_n$.

9.1 Choice of Basis Ordering

Because we must ultimately implement this algorithm in terms of linear algebraic computations, we must select bases for both the time and frequency representations of \mathbb{CS}_n . While the basis vectors are already specified (as the group elements for the time domain and the standard matrix basis element preimages for the frequency domain), we must still choose an ordering on each of these bases.

For the computation of the forward DFT factorization, the ordering of the time-domain basis is the more important of these two orderings, as it is this basis we start from in the decomposition. The frequency-domain basis is less important, since we do not work with the basis vectors themselves until we have separated out all of the one-dimensional frequency spaces. Ordering the group elements in the time-domain basis by left or by right cosets of S_k , where $k = 1, \dots, n-1$, confers several advantages. Each subgroup S_k in the chain $1 < S_2 < \dots < S_{n-1} < S_n$ partitions the subgroup S_{k+1} above it into $k+1$ cosets of S_k . Furthermore, the cosets of S_k have an easily stated set of coset representatives: the transpositions $(1 \ k+1)$ through $(k \ k+1)$, along with the identity. We can then construct S_{k+1} by taking S_k in a particular order and concatenating its cosets $S_k, (1 \ k+1)S_k, \dots, (k \ k+1)S_k$ (or $S_k, \dots, S_k(k \ k+1)$ in the right-coset order).

In addition, this coset order allows a convenient conversion between the index of the basis element in the order and its expression as a product of transpositions, as expressed in the following proposition. Figure 2 illustrates both orders for \mathbb{CS}_3 .

Proposition 9.1 *Suppose $\sigma \in S_n$ is the m th element in the right-coset ordering for S_n . The $\sigma = \tau_1 \cdots \tau_{n-1}$, where*

$$\tau_k = \begin{cases} (c_k \ k+1) & c_k > 0, \\ 1 & c_k = 0, \end{cases} \quad \text{and} \quad c_k = \left\lfloor \frac{m-1}{k!} \right\rfloor \bmod k,$$

and where $\lfloor \cdot \rfloor$ represents the floor or integer-part function. Furthermore, the c_k s are such that

$$m = c_{n-1}(n-1)! + c_{n-2}(n-2)! + \dots + c_1 \cdot 1! + 1.$$

Likewise, if σ has index m in the left coset order, then $\sigma = \tau_{n-1} \cdots \tau_2 \tau_1$.

Proof: This result follows directly from the recursive structure of the coset order and the Division Algorithm. Define $r_n = m-1$ and r_0, r_1, \dots, r_{n-1} recursively by the relation

$$r_k = c_{k-1}(k-1)! + r_{k-1},$$

where each r_k satisfies $0 \leq r_k < k!$. Applying these definitions successively yields

$$m-1 = c_{n-1}(n-1)! + c_{n-2}(n-2)! + \dots + c_1 \cdot 1! + 1.$$

Furthermore, we note that, for each k in $1, \dots, n-1$,

$$m-1 = c_{n-1}(n-1)! + c_{n-2}(n-2)! + \dots + c_k \cdot k! + r_k,$$

| Left Coset Order | | | | | | Right Coset Order | | | | | |
|------------------|-------|-------|----------|----------|-------------------------|-------------------|-------|-------|----------|----------|-------------------------|
| m | c_1 | c_2 | τ_2 | τ_1 | $\sigma = \tau_2\tau_1$ | m | c_1 | c_2 | τ_1 | τ_2 | $\sigma = \tau_1\tau_2$ |
| 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 |
| 2 | 1 | 0 | 1 | (12) | (12) | 2 | 1 | 0 | (12) | 1 | (12) |
| 3 | 0 | 1 | (13) | 1 | (13) | 3 | 0 | 1 | 1 | (13) | (13) |
| 4 | 1 | 1 | (13) | (12) | (123) | 4 | 1 | 1 | (12) | (13) | (132) |
| 5 | 0 | 2 | (23) | 1 | (23) | 5 | 0 | 2 | 1 | (23) | (23) |
| 6 | 1 | 2 | (23) | (12) | (132) | 6 | 1 | 2 | (12) | (23) | (123) |

Figure 2: Left and right coset orders for S_3 , along with decompositions of elements into transpositions according to Proposition 9.1.

so then

$$\left\lfloor \frac{m-1}{k!} \right\rfloor \bmod k = (c_{n-1}(n-1)(n-2)\cdots k + c_{n-2}(n-2)\cdots k + \cdots + c_k) \bmod k = c_k.$$

Then c_k specifies which coset of S_k σ belongs to and hence which coset representative S_k in S_{k+1} , by the order of the coset representatives $1, (1k) \dots (k-1k)$. Therefore, σ is simply a product of these coset representatives. ■

Another advantage conferred by this coset order is invariance under action by elements of S_k from one side: if $\sigma \in S_k$ and $\tau \in S_n$, then $\sigma\tau$ and τ belong to the same right coset of S_k . Furthermore, the right cosets of S_k are isomorphic under this left action of σ . Analogous results hold for σ acting on the right in the left-coset order on S_n .

A natural choice for the ordering of the seminormal basis vectors $u_{k,ij}$ of the frequency domain is lexicographically by k, i , and j , as then coefficient vectors with respect this basis contain the coefficients for the matrix algebra blocks in row-major order. Section 9.3 discusses how this ordering can be derived from the sequence of projection operators used in the decomposition.

9.2 Computation of Frequency Separation Factors

We now address the efficient separation of the frequency space into its one-dimensional Fourier spaces. Recalling that we can distinguish each vector in the seminormal basis by the two lists of representations it passes through in the Bratteli diagram, we can accomplish this separation gradually by building the associated lists of representations in stages corresponding to the left and right actions of $\mathbb{C}S_k$. Computationally, we perform this separation at the S_k stage by projecting frequency spaces separated up to the S_{k-1} paths onto the spaces corresponding to the S_k representations, each time effectively adding another representation to the pair of lists associated to each Fourier space.

We recall that, since $\mathbb{C}S_n$ is a $\mathbb{C}S_n$ -bimodule, we can represent the left and right actions of $r \in \mathbb{C}S_n$ on $\mathbb{C}S_n$ as \mathbb{C} -linear transformations on $\mathbb{C}S_n$. Thus, with a choice of \mathbb{C} -basis for $\mathbb{C}S_n$, these

actions correspond to $n! \times n!$ matrices acting from the left on coordinate vectors for $\mathbb{C}S_n$. In this fashion, we represent the (left *and* right) actions of the group ring elements described below as left matrix multiplication on coordinate vectors for $\mathbb{C}S_n$

9.2.1 Centrally Primitive Idempotents

Since we wish to distinguish the vectors by a pair of lists of representations of S_k for $2 \leq k \leq n$, a natural choice of elements of $\mathbb{C}S_n$ are the central primitive idempotents $e_1^{(k)}, e_2^{(k)}, \dots, e_{h_k}^{(k)}$ associated to each $\mathbb{C}S_k$, for $2 \leq k \leq n$. Suppose α is a partition of n and β a partition of k . Then the left action of $e_\beta^{(k)}$ identifies those $U_{\alpha,ij}$ such that the first k boxes of the tableau t_i^α are of shape β . Similarly, the right action of $e_\beta^{(k)}$ identifies those $U_{\alpha,ij}$ such that the first k boxes of the tableau t_j^α are of shape β .

Example 9.2 We decompose $\mathbb{C}S_3$ according to this set of operators. We first note that

$$W_1 = e_1^{(2)}\mathbb{C}S_3 = U_{1,11} \oplus U_{2,11} \oplus U_{2,12} \quad \text{and} \quad W_2 = e_2^{(2)}\mathbb{C}S_3 = U_{2,21} \oplus U_{2,22} \oplus U_{3,11},$$

so that $\mathbb{C}S_3 = W_1 \oplus W_2$, and the first set of idempotents separates the frequency space into two three-dimensional spaces. Applying the $\mathbb{C}S_2$ idempotents to these spaces on the right then yields

$$\begin{aligned} W_{1,1} &= W_1 e_1^{(2)} = U_{1,11} \oplus U_{2,11}, & W_{1,2} &= W_1 e_2^{(2)} = U_{2,12}, \\ W_{2,1} &= W_2 e_1^{(2)} = U_{2,21}, & W_{2,2} &= W_2 e_2^{(2)} = U_{2,22} \oplus U_{3,11}. \end{aligned}$$

Finally, the $\mathbb{C}S_3$ idempotents separate $W_{1,1}$ and $W_{2,2}$ into

$$\begin{aligned} W_{1,1,1} &= e_1^{(3)}W_{1,1} = U_{1,11}, & W_{1,1,2} &= e_2^{(3)}W_{1,1} = U_{2,11}, \\ W_{2,2,2} &= e_2^{(3)}W_{2,2} = U_{2,22}, & W_{2,2,3} &= e_3^{(3)}W_{2,2} = U_{3,11}. \end{aligned}$$

Thus, these idempotents separate $\mathbb{C}S_3$ out into these one-dimensional Fourier spaces. We note that since the final idempotents are central in $\mathbb{C}S_3$ itself, there is no need to apply them on both the right and the left. ■

While this approach is initially appealing in its simplicity, it faces a number of computational problems: the number of idempotents for S_n grows as the number of partitions of n , which grows exponentially for large n (for example, $n = 20$ has 627 partitions [27] and hence S_{20} has 627 central primitive idempotents). Furthermore, computation of the idempotents for S_n requires precomputation of the character tables of S_n .

9.2.2 Jucys-Murphy Eigenspace Projections

The Jucys-Murphy elements described above in Sections 4 and 8.4 provide an alternate means of specifying these representations and hence of separating these Fourier spaces. By Theorem 8.12, the eigenvalues of the Jucys-Murphy elements m_2, \dots, m_k suffice to distinguish seminormal basis vectors for representations of S_k . Hence, projecting the frequency spaces into the eigenspaces

associated with the left and right actions of these Jucys-Murphy elements accomplishes the same separation that we achieve with the idempotents.

Furthermore, the Jucys-Murphy elements make up for the deficiencies of these idempotents:

- Since the eigenvalues of m_k are the contents of boxes added to partitions of $k - 1$, they range from at least $-k + 1$ to at most $k + 1$. Thus, the number of projection operators at each S_k stage is bounded by $2k$.
- The matrix representations of these elements in the group element basis consist of the sum of $k - 1$ transposition matrices, which themselves consist of simple, symmetric permutation matrices. Thus, the Jucys-Murphy matrices are symmetric and easy to construct.
- These matrices can be stored efficiently in a sparse format, as each contains only $k \cdot n!$ 1s.

We now examine the application of these Jucys-Murphy elements and their eigenspace projection operators to the construction of this DFT matrix factorization.

9.2.3 Computation of Projection Matrix Factors

We first characterize the projection operators associated with the Jucys-Murphy eigenspaces.

Theorem 9.3 *Let m_k be the Jucys-Murphy element for S_k , $2 \leq k \leq n$, and let M_k be the matrix representation of its (left or right) action on $\mathbb{C}S_n$ in the standard basis for S_n . Then each eigenspace W_j has an orthonormal basis $B_j = \{u_j^i\}_{i=1}^{k_j}$, where $k_j = \dim W_j$, and none of the eigenspaces are deficient. Moreover,*

$$M_k = \sum_{j=-n+1}^{n-1} \sum_{i=1}^{k_j} j u_j^i (u_j^i)^T.$$

Proof: Since M_k is real and symmetric, as stated above, this result is a direct consequence of the spectral theorem. ■

Definition 9.4 *Let m_k be as above. Denote the matrix associated to the left action of m_k as M_{kL} , the eigenspaces of M_k as W_{kL}^λ , and the projection operators associated with those eigenspaces as P_{kL}^λ . Likewise, for the right action denote these as M_{kR} , W_{kR}^λ , and P_{kR}^λ , respectively.* ■

Since these projection operators act just as the idempotent elements do above in separating the frequency spaces at each pair of left and right actions, we are guaranteed that each projection matrix product

$$P_{nL}^\lambda \cdots P_{2R}^{\lambda'} P_{2L}^{\lambda''}$$

projects onto at most a one-dimensional subspace and hence has rank 1. As with the idempotents, we can disregard the right projection operators for the last Jucys-Murphy element: the sums of the

eigenvalues for the left and right actions must be equal, so we can recover the last right eigenvalue from all the others.

We can even compute these projection operators without orthogonal bases for the eigenspaces of M_k .

Proposition 9.5 *Suppose that W is an eigenspace associated with m_k and $B = \{u_i\}_{i=1}^k$ the orthonormal basis that is guaranteed to exist for W . Let $C = \{v_i\}_{i=1}^k$ be any basis for W , and let V be the $k \times n!$ matrix with rows consisting of the transposes of the v_i s. Then VV^T is invertible, and the projection matrix P for W is given by*

$$P = V^T(VV^T)^{-1}V.$$

Proof: Let U be the $k \times n!$ matrix with rows given by the transposes of the u_i s. Then, by the outer product definition of multiplication,

$$U^T U = \begin{pmatrix} u_1 & u_2 & \dots & u_k \end{pmatrix} \begin{pmatrix} u_1^T \\ u_2^T \\ \vdots \\ u_k^T \end{pmatrix} = \sum_{i=1}^k u_i u_i^T = P.$$

Furthermore, $UU^T = I_k$, the $k \times k$ identity matrix. Since the rows of V and the rows of U are both bases for W , there exists an invertible matrix R such that $V = RU$. Since the columns of V^T are the v_i s and hence are linearly independent, the rank of this $n! \times k$ matrix is k . Then $\text{rk } VV^T = \text{rk } V^T = k$, so the $k \times k$ matrix VV^T is fully ranked and hence invertible. Finally,

$$V^T(VV^T)^{-1}V = U^T R^T (R U U^T R^T)^{-1} R U = U^T R^T (R^T)^{-1} (U U^T)^{-1} R^{-1} R U = U^T U = P. \quad \blacksquare$$

In general, consider the stage s projection matrix $P = P_s^\lambda \cdots P_{2R}^{\lambda'} P_{2L}^{\lambda''}$. P projects $\mathbb{C}^{n!}$ onto a space W of dimension k . Then we can decompose P into $P = DC$, where C is a fully ranked $k \times n!$ matrix that maps $x \in \mathbb{C}^{n!}$ to its coordinates in a basis for W , and D is a fully ranked $n! \times k$ matrix that maps these coordinates back into the basis for $\mathbb{C}^{n!}$.

In fact, a similar decomposition holds for any $m \times n$ matrix A of rank r : $A = DC$, where C is $r \times n$ and D is $m \times r$. One way to compute this decomposition is by putting A into row-reduced echelon form to obtain C and recovering D from the sequence of row operations used to reduce A . In essence, C corresponds to a compression operator that gives the minimum amount of information necessary to store the projection, and D corresponds to a decompression operator that expands this information back into the full projection.

Consider the product $P_{nL} \cdots P_{2R} P_{2L}$, and suppose the intermediate product $T_s = P_s \cdots P_{2R} P_{2L}$ at each stage s projects onto a subspace of dimension d_s . Then $T_s = D_s C_s$, as above. Furthermore, reprojecting onto the same sequence of subspaces at each stage has no effect on the projection itself, so we can replace each P_s in the product with T_s and still have the same intermediate product at each stage. Hence,

$$P_{nL} \cdots P_{2R} P_{2L} = T_{nL} \cdots T_{2R} T_{2L} = D_{nL} C_{nL} \cdots C_{2R} D_{2R} C_{2R} D_{2L} C_{2L} = D_{nL} Q_{nL} \cdots Q_{3L} Q_{2R} Q_{2L}$$

where $Q_s = C_s D_{s-1}$ is then a $d_s \times d_{s-1}$ matrix, and $D_1 = I$. T_n projects into an at most one-dimensional subspace, so if $d_n = 1$, D_{nL} is simply a scalar multiple of the eigenvector corresponding to the one-dimensional Fourier frequency space of interest. Consequently, $Q_{nL} \cdots Q_{2R} Q_{2L}$ applied to $f \in \mathbb{C}^{n!}$ gives the coordinate of f in the one dimensional Fourier frequency space, which is the Fourier coefficient for f that we seek to compute, up to possibly some constant scaling factor (of which we explain the computation below in Section 9.3). Furthermore, we know the eigenvalue associated to each projection matrix P_s , so we can identify which space we are projecting into by Theorem 8.12.

With this decomposition in mind, we describe the algorithm that produces the k th factor in the DFT factorization. Suppose that at stage k in the factorization we have a set $\{D_i\}_{i=1}^m$ decomposition operators from the previous stage and a set $\{P_j\}_{j=1}^r$ projection matrices corresponding to the eigenspaces of the Jucys-Murphy element acting at this stage. Furthermore, suppose each P_j is factored in the form $V_j^T (V_j V_j^T)^{-1} V_j$ described in Proposition 9.5. Define $d_i = \text{rk } D_i$, $p_j = \text{rk } P_j$, and $d_{ij} = \text{rk } P_j D_i$. Then

$$D_i = \sum_{j=1}^r P_j D_i = \sum_{j=1}^r V_j^T (V_j V_j^T)^{-1} V_j D_i = \sum_{j=1}^r V_j^T (V_j V_j^T)^{-1} \tilde{D}_{ij} C_{ij} = \sum_{j=1}^r D_{ij} C_{ij},$$

where \tilde{D}_{ij} and C_{ij} are the $p_j \times d_{ij}$ and $d_{ij} \times d_i$ matrices that result from the decomposition of the product matrix $V_j D_i$, and where D_{ij} is the $n! \times d_{ij}$ matrix $V_j^T (V_j V_j^T)^{-1} \tilde{D}_{ij}$. Since $\sum_{j=1}^r d_{ij} = d_i$, we have

$$D_i = \sum_{j=1}^r D_{ij} C_{ij} = \begin{pmatrix} D_{i1} & D_{i2} & \cdots & D_{ir} \end{pmatrix} \begin{pmatrix} C_{i1} \\ C_{i2} \\ \vdots \\ C_{ir} \end{pmatrix} = \bar{D}_i \bar{C}_i.$$

Then the direct sum

$$F_s = \bar{C}_1 \oplus \bar{C}_2 \oplus \cdots \oplus \bar{C}_m = \begin{pmatrix} \bar{C}_1 & & & \\ & \bar{C}_2 & & \\ & & \ddots & \\ & & & \bar{C}_m \end{pmatrix}$$

forms the next factor in the DFT matrix factorization, and we pass the nonzero $\{D_{ij}\}_{i=1, j=1}^{m,r}$ on to the next stage. We write down these steps explicitly:

Algorithm 9.6 Given $\{D_i\}_{i=1}^m$ and $\{V_j\}_{j=1}^r$,

1. For each V_j , compute and store $(V_j V_j^T)^{-1}$.
2. For each i , $1 \leq i \leq m$,
 - 2.1. For each j , $1 \leq j \leq r$,
 - 2.1.1. Compute $V_j D_i$.

- 2.1.2. Decompose $V_j D_i$ into $\tilde{D}_{ij} C_{ij}$. This can be accomplished by row reduction to echelon form, among other means.
- 2.1.3. Compute $D_{ij} = V_j^T (V_j V_j^T)^{-1} \tilde{D}_{ij}$.
- 2.2. Concatenate the rows of the C_{ij} s in order to form \tilde{C}_i .
3. Set $F_s = \bigoplus_{i=1}^m \tilde{C}_i$ and add to list of matrix factors. ■

Along the way, to identify each eigenspace, we associate a list of Jucys-Murphy eigenvalues to each D_i matrix, at each stage forming the list for D_i by adding λ_j to the list for D_i . Consequently, after all the Jucys-Murphy stages, each row in the list of factors has an associated list of eigenvalues that specifies which Fourier space it corresponds to.

9.3 Computation of Permutation and Scaling Matrices

While this list of eigenvalues for each space specifies exactly which Fourier space it is, these spaces may not be in the desired order. For example, if the projection operators P_j are always applied in order of decreasing eigenvalue, then the resulting lists of eigenvalues will be in descending lexicographic order, which does not correspond to the last-letter order that is desired for the coefficients. We can achieve this order, however, from the list of eigenvalues, as Proposition 8.6 shows.

Taking every other element from the single list of eigenvalues generates the two lists of left and right eigenvalues. Since the full lists of eigenvalues sum to the same we can recover the n th right eigenvalue from the difference of the sums of the two lists. Also, the shape α associated to an irreducible representation of S_n is determined by the set of content values of the boxes in α , so we can group the rows by irreducible representation by grouping together those with identical lists of sorted left (or right) eigenvalues. Thus, we have the following procedure for computing the correct ordering of rows:

Algorithm 9.7 Given the list of Jucys-Murphy eigenvalue lists for the rows in the DFT matrix factorization,

1. Tag each list L of eigenvalues with the index of the row in the original order,
2. Separate each list L into left and right lists L_L and L_R , and add the last right eigenvalue to L_R ,
3. Group the rows by irreducible, determined from the sorted L_L list,
4. Concatenate L_R and L_L , reverse this list, and sort each group by descending lexicographic order.

This process puts the rows in row-major order by matrix block. We have then permuted the list of row indices, which we use to construct a permutation matrix P_{DFT} that permutes the coefficients into this order. ■

In particular, this row-major makes converting from the vector of seminormal coordinates to the seminormal matrix representation convenient. Furthermore, this irreducible-representation partitioning process gives the degrees of each of the irreducible representations, which we require to convert between the matrix and the vector forms.

Finally, we note that although we can now compute the coordinate of $f \in \mathbb{C}S_n$ with respect to a basis for $U_{k,ij}$, this basis may be some scalar multiple $\alpha_{k,ij}u_{k,ij}$, instead of $u_{k,ij}$. Consequently, we must determine these scalings for each row and add a final diagonal factor in the DFT matrix factorization that applies these scalings to the coefficients. These scaling factors can be determined by matching the images of the standard basis vectors to the seminormal representations of those elements that we can compute combinatorially as described in Section 8.3. Thus, if $\tau \in \mathbb{C}S_n$, then we can compute both the coordinates under $P_{\text{DFT}}F_{nL} \cdots F_{2L}$ and under the seminormal representation and determine the scalings $\alpha_{k,ij}$ for the rows that produces nonzero coefficients of τ . We can then continue matching the seminormal and partial factorization images of elements of $\mathbb{C}S_n$ until we have obtained all the $n!$ scaling factors.

We then have the following algorithm to generate this scaling matrix S_{DFT} .

Algorithm 9.8 Given the list of lists of left and right eigenvalues generated by Algorithm 9.7, and the partial matrix factorization $P_{\text{DFT}}, F_{nL}, \dots, F_{2L}$ of the DFT matrix,

1. Let L be the list of rows that we have yet to scale; initialize L to $1, \dots, n!$.
 2. Ensure that the scalings for the image of $1 \in \mathbb{C}S_n$ are correct.
 - 2.1. Since 1 maps to the identity in the block matrix algebra, its correct image contains 1s down the diagonal coordinates and 0s elsewhere. Hence, if the coordinate $c_{k,ij}$ is nonzero, it produces a scaling factor of $s_{k,ij} = 1/c_{k,ij}$.
 - 2.2. Remove the indices of nonzero coordinates from L .
 3. Construct the seminormal matrix representations of the generating transpositions $(1\ 2), (2\ 3), \dots, (n-1\ n)$ from the coefficients under the partial factorization that lie on the matrix diagonal, which we now can scale correctly because they are covered by the scalings for the identity. Thus, for each i such that $1 \leq i < n$,
 - 3.1. Rewrite the partial factorization image of $(i\ i+1)$ in block diagonal form to form the matrix C .
 - 3.2. For each nonzero element $c_{k,ij}$ in the i th row and j th column of the k th block of C ,
 - 3.2.1. Initialize a matrix σ to all zeros to store the correct seminormal representation of $(i\ i+1)$.
 - 3.2.2. If $i = j$, $c_{k,ij}$ is on the diagonal and is scaled correctly. Hence, $\sigma_{k,ij} = c_{k,ij}$.
 - 3.2.3. If $i > j$, $c_{k,ij}$ is below the diagonal, so $\sigma_{k,ij} = 1$. Set $s_{k,ij} = 1/c_{k,ij}$.
 - 3.2.4. If $i < j$, $c_{k,ij}$ is above the diagonal, so $\sigma_{k,ij} = 1 - c_{k,ii}^2$, and $s_{k,ij} = (1 - c_{k,ii}^2)/c_{k,ij}$.
- Store the $\sigma((i\ i+1))$ s and the computed scalings $s_{k,ij}$.

- 3.3. Remove the scaled row indices from L .
4. We now compute the remaining scaling coefficients. While L is nonempty,
 - 4.1. Let i be the first index remaining in L . Compute the i th row of the partial factorization product by $e_i^T \cdot P_{\text{DFT}} F_{nL} \cdots F_{2L}$, and let j be the index of the first nonzero entry in this row.
 - 4.2. Determine the permutation ρ_j associated with the index j in the right-coset order on S_n , and decompose ρ_j into transpositions according to Proposition 9.1.
 - 4.3. Compute the correct seminormal representation σ of ρ_j from products of the $\sigma((i+1))$ matrices.
 - 4.4. Compute the partial factorization image of ρ_j .
 - 4.5. For the nonzero coordinates $c_{k,ij}$ for rows that have not been scaled yet, set $s_{k,ij} = \sigma_{k,ij} / c_{k,ij}$.
 - 4.6. Remove the scaled row indices from L .

Construct S_{DFT} to be the matrix with the entries $s_{k,ij}$ along the diagonal. ■

We combine the above results into the following theorem:

Theorem 9.9 *Take $f \in \mathbb{C}S_n$, and let $[f]_S$ be its coordinates in the standard group-element basis for $\mathbb{C}S_n$ and $[f]_F$ its coordinates in the seminormal Fourier basis $u_{k,ij}$. Then*

$$[f]_F = S_{\text{DFT}} \cdot P_{\text{DFT}} \cdot F_{nL} \cdots F_{2R} \cdot F_{2L} \cdot [f]_S,$$

so that $S_{\text{DFT}} \cdot P_{\text{DFT}} \cdot F_{nL} \cdots F_{2R} \cdot F_{2L}$ provides a factorization of the DFT matrix that changes bases from S to T . ■

Thus, there exists a factorization of the DFT matrix for S_n that corresponds to a gradual separation of the frequency spaces. Based on examination of the number of nonzero entries in each of the factors and the total number of operations required for the evaluation of this factorization for $n = 3, \dots, 6$, we conjecture the following about the complexity of the FFT that this algorithm produces:

Conjecture 9.10 *The efficiency of the evaluation of the decimation-in-frequency FFT for S_n computed by Algorithms 9.6, 9.7 and 9.8 is $O(n^2 n!)$.* ■

The establishment of this result would place the complexity of this FFT in the same class as Maslen's decimation-in-time algorithm [14], which is to date the most efficient FFT known for S_n .

10 Initial Implementation and Results

10.1 *Mathematica* Implementation

For a prototype implementation of these algorithms, the symbolic computing program *Mathematica* 5.0 was used. This platform offers several advantages:

- The computational framework that *Mathematica* presented was more familiar than those of other symbolic computation programs such as Maple. *Mathematica* offers both procedural programming and functional programming, as well as sophisticated pattern matching. Furthermore, *Mathematica* is built around list-structures and hence handles list-processing algorithms well.
- *Mathematica* provides calculations in exact arithmetic, whereas Matlab natively supports only floating-point calculations.
- Like *Matlab*, *Mathematica* supports sparse representations of vectors and matrices and fast algorithms for computations involving sparse matrices.
- *Mathematica* features the *Combinatorica* package, which provides extensive functionality for working with permutations and permutations groups. Moreover, Pemmaraju and Skiena [21] provide extensive documentation of this package, and the . Since we are investigating the symmetric group S_n , this seemed a natural set of features to employ.
- *Mathematica* also provides excellent features for importing and exporting data and graphics conveniently.

The *Mathematica* code presented in Appendix A implements Algorithms 9.6, 9.7 and 9.8, as well as the left and right coset orders described in Section 9.1.

10.2 Operation Counts and Precomputation Times

Most of the computational testing of this implementation was performed on a 900 MHz Pentium III processor with 256 MB of RAM. To date, we have been able to compute the factorization of the DFT matrix for S_n for $n \leq 6$. Figure 3 illustrates the factors for $n = 3, 4$, and 5 , clearly showing how sparse the matrix factors are, especially when compared to the very dense original DFT matrices for S_n . Figure 4 presents the three projection factors in the matrix factorization of the S_3 DFT matrix.

Of principal interest is how efficiently the DFT can be evaluated on a generic coordinate vector for S_n , given the factorization produced by Algorithms 9.6, 9.7 and 9.8. We therefore determine the number of additions and multiplications needed to perform the series of operations associated with the evaluation of the DFT. We note in particular that a row with k nonzero entries, m of which differ from 1 and -1 , requires a total of $k - 1$ additions and m multiplications. In his analysis of his FFT for S_n , Maslen [14] defines an operation to be a complex addition and multiplication taken together; in this case, evaluation of the row takes $k - 1$ operations, unless all of the nonzero

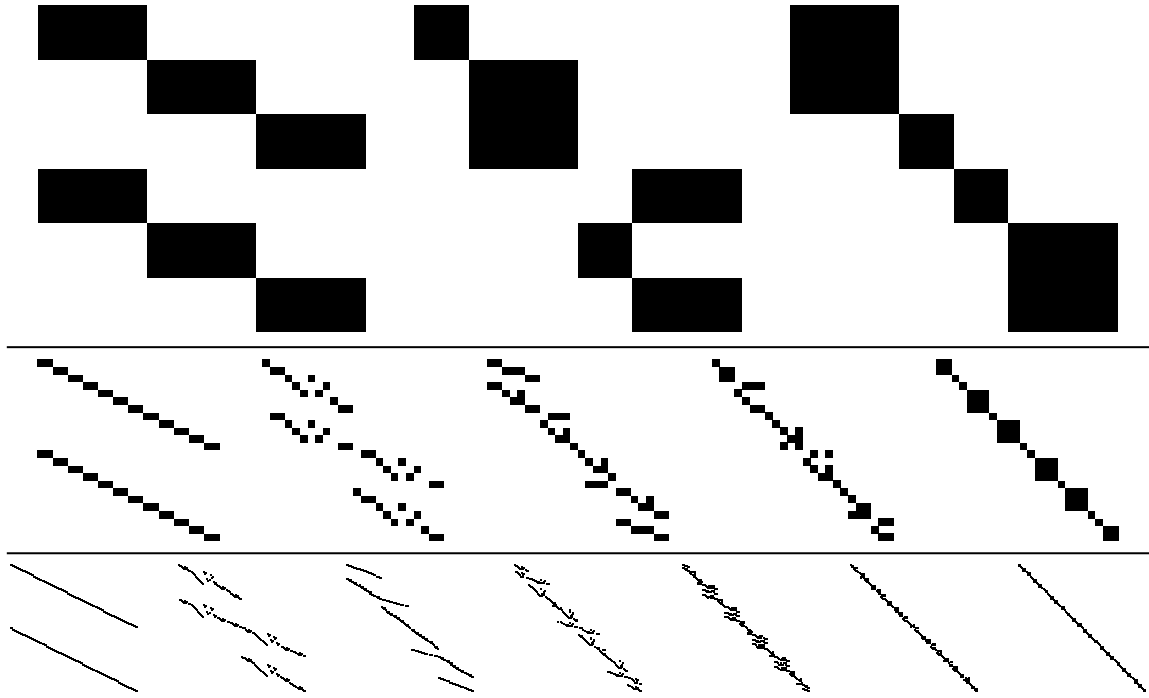


Figure 3: Plots of the projection factor matrices $F_{2L}, F_{2R}, \dots, F_{nL}$ in the factorizations of DFT matrices for $S_3, S_4,$ and S_5 . The filled-in squares represent nonzero elements in the matrices. Compared to the full DFT matrix, which is nearly filled with nonzero entries (see, for example, the DFT matrix for S_3 in Equation (8.1)), these matrix factors are quite sparse.

elements are also not ± 1 , in which case it takes k . The operation counts for these evaluations are tabulated in Table 2 and are compared with the operation counts for Maslen's decimation-in-time algorithm. We see that, according to Maslen's definition of an operation, the operation counts for the decimation-in-frequency algorithm are slightly less than those for Maslen's decimation-in-time algorithm.

The major limitations in computing transforms for $n = 7$ and above appear to be time and memory. While the FFT factorization takes 0.1 seconds to compute for S_3 , 0.8 seconds for S_4 , and 30 seconds for S_5 , it requires 5100 seconds of processing time for S_6 . This progression indicates that the precomputation algorithm is approximately $O((n!)^3)$. We discuss possible improvements to the algorithm implementation in Section 11.

$$\begin{pmatrix} 1 & 1 & & & & \\ 1 & -\frac{1}{2} & & & & \\ & & 1 & & & \\ & & & 1 & & \\ & & & & 1 & \frac{1}{2} \\ & & & & 1 & -1 \end{pmatrix} \begin{pmatrix} 1 & & & & & \\ & 1 & 1 & & & \\ & 1 & -1 & & & \\ & & & 1 & -1 & \\ & & & & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 & & & & \\ & & 1 & 1 & & \\ & & & & 1 & 1 \\ & & & & & 1 & 1 \\ & & & & 1 & -1 \\ & & & & & 1 & -1 \end{pmatrix}$$

Figure 4: The three factors $F_{3L}F_{2R}F_{2L}$ in the factorization of the S_3 DFT matrix. The block structure of the matrices clearly illustrates the decomposition of Example 9.2 into two 3-dimensional spaces, then into two 2-dimensional spaces and two 1-dimensional ones, and finally into the six 1-dimensional spaces associated to the Fourier coefficients.

| n | \oplus | \otimes | t_{DIF} | t_M | $\frac{1}{2}n(n-1)$ |
|-----|----------|-----------|------------------|-------|---------------------|
| 3 | 14 | 3 | 2.5 | 2.7 | 3 |
| 4 | 112 | 43 | 5.0 | 5.4 | 6 |
| 5 | 986 | 419 | 8.8 | 9.1 | 10 |
| 6 | 9696 | 4634 | 14.2 | 13.6 | 15 |

Table 2: Operation counts for the decimation-in-frequency FFT evaluations. In addition to the number of addition and multiplications \oplus and \otimes , respectively, we compare our complexity to that of Maslen [14] via the *reduced complexity*, or number of operations divided by $|S_n| = n!$. Here, t_{DIF} denotes the reduced complexity of our decimation-in-frequency algorithm, while t_M denotes Maslen’s reduced complexity. In each case, the reduced complexities are below the bound of $\frac{1}{2}n(n-1)$ that Maslen conjectures holds for his algorithm for all n .

11 Future Directions

With this prototype implementation of the decimation-in-frequency FFT precomputation complete, there are several significant areas of additional investigation to pursue, which we outline below.

11.1 Improvement of Algorithmic Efficiency

There are several procedures in the current Mathematica implementation that can be made more efficient. In particular, computing bases for the eigenspaces of the Jucys-Murphy elements and computing the inverses of the projection matrices are the processes that appear to make this algorithm $O((n!)^3)$. It should be possible to use the conjugated block-diagonal structure of the Jucys-Murphy matrices to increase the efficiency of these computations, however, and such investigations constitute the next phase of inquiry in this project.

11.2 Choice of Basis

As noted in Section 9.1, the selection of bases is of crucial importance to the form of the matrices in our factorization. We currently select bases for the decomposition into a compressor and a decompressor (described in Section 9.2.3) using row-reduction into echelon form, but it may ultimately be more advantageous to select different bases for these decompositions. For example, it may be that there is a choice of basis that is related to the seminormal representation itself. Similarly, while the right-coset basis for S_n appears to be well suited for this factorization, it may be that a different order on the standard basis yields better structure in the matrix factors. In particular, using the basis that Clausen and Baum [5] employ in their description of the representations of S_3 yields a factorization identical to that presented in Figure 4, except that the two 2×2 blocks in the last factor are identical.

11.3 SPIRAL Factorization

We ultimately seek to determine a highly structured factorization of the DFT matrix for S_n . Consequently, one area of investigation open to us is to process the factors in our factorization in the SPIRAL system [22], which automatically generates structured representations of matrices associated with linear digital signal processing transforms. These automated approaches may elucidate structure in these matrices that has heretofore escaped our notice, or may confirm that having two factors corresponding to the Jucys-Murphy element m_k acting on left and the right is an optimal decomposition for the action of S_k in this decomposition.

11.4 Matlab, GAP implementations

While Mathematica provides a good environment for prototyping these algorithms, eventually we wish to port these implementations to Matlab or to the GAP Groups, Algorithms, and Programming computational discrete algebra software system. Matlab is widely used in signal processing applications, and GAP is frequently used for computational research in group theory and representation theory. Since GAP works primarily with permutation representations of groups, it is potentially a good platform for these DFTs on S_n .

11.5 Recursive Structure of FFTs

Ideally, we would like to observe a direct relation of the DFT on S_n to n copies of the DFT on S_{n-1} , so that we can develop a recursive precomputation of the matrix factorization and so that we can establish concrete bounds on the complexity of this S_n FFT. In particular, such a recursive formulation would allow us to prove or disprove Conjecture 9.10. Maslen [14] claims to have related the Fourier transform on S_n to a collection of transforms on S_{n-1} in his decimation-in-time algorithm, so one approach we intend to follow is to examine his algorithm closely and to see if its recursive structure can be used in our decimation-in-frequency algorithm.

11.6 Computation of Inverse DFT

Finally, in order to compute products in $\mathbb{C}S_n$ efficiently by multiplication of their seminormal representations, we require not only an efficient DFT evaluation but also an efficient inverse DFT evaluation. Therefore, we plan to examine how to adapt the current DFT precomputation algorithm to one that produces a factorization of the inverse DFT matrix as well.

A Mathematica Code: FFT Generation Algorithm

The following pages present the Mathematica code that constitutes the initial implementation of the decimation-in-frequency FFT precomputation algorithm specified in Algorithms 9.6, 9.7 and 9.8. Sparse matrices are used throughout to save efficiency in both memory and computation time.

```

<< DiscreteMath`Combinatorica`
<< LinearAlgebra`MatrixManipulation`
<< LinearAlgebra`Orthogonalization`

(* Smart matrix operations that ignore {{}} *)
SmartDot[A_?MatrixQ, B_?MatrixQ] := If[A = {{}} || B = {{}}, {{}}, A.B];
SameColumnSize[l_List] := (SameQ@@ (Dimensions[#][[2]] & /@ l));
MatrixNullQ[x_] = x = {{}};
ListNullQ[x_] = x = {};
SmartColumnAppend[x_?MatrixQ] :=
  Join@@ DeleteCases[x, _?MatrixNullQ] /; SameColumnSize[DeleteCases[x, _?MatrixNullQ]]

(* Matrix Direct Sum *)
MatrixDirectSumSparse[mat : {_?MatrixQ..}] :=
  MatrixDirectSumSparseInner[DeleteCases[mat, _?MatrixNullQ]];
MatrixDirectSumSparseInner[mat : {_?MatrixQ..}] :=
  SparseArray[Flatten[MapThread[Array[List, #1, #2 + 1] &,
    {#, Most[FoldList[Plus, {0, 0}, #]}] & [Dimensions/@mat]], 2] → Flatten[mat]]

(* Permutation matrices: moves row  $\sigma[i]$  to row  $i$  *)
PermutationMatrix[ $\sigma$ ?PermutationQ] :=
  SparseArray[MapThread[{#1, #2} &, {Range[1, Length[ $\sigma$ ],  $\sigma$ ]} → Table[1, {Length[ $\sigma$ ]}]];

(* JM ELEMENT CONSTRUCTION *)
(* JMTransposition produces the permutation in  $S_n$  that swaps  $i$  and  $j$ . *)
JMTransposition[n_Integer, i_Integer, j_Integer] :=
  Table[k + KroneckerDelta[i, k] (j - i) - KroneckerDelta[j, k] (j - i), {k, 1, n}];

(* JMTranspositionList[n,k] corresponds to Jucys-Murphy element  $m_k$  for  $S_n$ . *)
JMTranspositionList[n_Integer, k_Integer] /; n > 1 && k > 1 && k ≤ n :=
  Table[JMTransposition[n, j, k], {j, 1, k - 1}];

(* CONSTRUCTION OF COSET-ORDERED SYMMETRIC GROUP *)
(* SymmetricGroupCoset[A,  $\sigma$ , 0] =  $\sigma$  A;
  SymmetricGroupCoset[A,  $\sigma$ , 1] = A  $\sigma$ . *)
SymmetricGroupCoset[subset : {_?PermutationQ..},  $\sigma$ ?PermutationQ, 0] :=
  Map[Permute[ $\sigma$ , #] &, subset];
SymmetricGroupCoset[subset : {_?PermutationQ..},  $\sigma$ ?PermutationQ, 1] :=
  Map[Permute[#,  $\sigma$ ] &, subset];

(* SymmetricGroupByCosets[n, k, 0] constructs  $S_n$  ordered by left cosets.
  SymmetricGroupByCosets[n, k, 1] constructs  $S_n$  ordered by right cosets. *)
SymmetricGroupByCosets[n_Integer, 1, side_Integer] /; n > 0 := {Range[1, n]};
SymmetricGroupByCosets[n_Integer, k_Integer, side_Integer] /;
  n > 0 && k > 1 && n ≥ k && (side = 0 || side = 1) := Module[{SSubGroup},
  SSubGroup = SymmetricGroupByCosets[n, k - 1, side];
  Flatten[
    Join[{SSubGroup}, Map[SymmetricGroupCoset[SSubGroup, #, side] &, JMTranspositionList[n, k]]], 1
  ];
];

```

```

(* CONSTRUCTION OF PERMUTATION RANKING BY ORDER IN COSET *)
(* ConstructSymmetricGroupLookups[n, 0/1] constructs
the lookup tables for  $S_n$  in the left/right coset order. *)
ConstructSymmetricGroupLookups[n_Integer, side_Integer] := (SymmetricGroupInverseLookup[n, side] =
  (RankPermutation[#] + 1) & /@ SymmetricGroupByCosets[n, n, side];
  SymmetricGroupLookup[n, side] = InversePermutation[SymmetricGroupInverseLookup[n, side]]);
(* RankPermutationByCoset[ $\sigma$ , 0/1] generates the rank of  $\sigma$  in the left/right coset order for  $S_n$ ,
where  $n = \text{Length}[\sigma]$ .
  UnrankPermutationByCoset[ $\iota$ , n, 0/1] generates  $\sigma$  from
  its rank  $\iota$  in the left/right coset order for  $S_n$ , where  $n$ . *)
RankPermutationByCoset[ $\sigma$ ?PermutationQ, side_Integer] :=
  SymmetricGroupLookup[Length[ $\sigma$ ], side][[RankPermutation[ $\sigma$ ] + 1]];
UnrankPermutationByCoset[index_Integer, n_Integer, side_Integer] :=
  UnrankPermutation[SymmetricGroupInverseLookup[n, side][[index]] - 1, n];
(* Default permutation ranking is by right coset order *)
RankPermutationByCoset[ $\sigma$ ?PermutationQ] = RankPermutationByCoset[ $\sigma$ , 1];

(* JM MATRIX GENERATION *)
(* JMPRep[ $\sigma$ ,0/1] gives the left/right regular permutation
repn of  $S_n$  with respect to Mathematica's lex. basis ordering.*)
JMPRep[ $\sigma$ ?PermutationQ, 0] := SparseArray[
  Map[{RankPermutation[Permute[ $\sigma$ , #]] + 1, RankPermutation[#] + 1} &, SymmetricGroup[Length[ $\sigma$ ]]] ->
  Table[1, {i, 1, Factorial[Length[ $\sigma$ ]}]];
JMPRep[ $\sigma$ ?PermutationQ, 1] := SparseArray[
  Map[{RankPermutation[Permute[#,  $\sigma$ ]] + 1, RankPermutation[#] + 1} &, SymmetricGroup[Length[ $\sigma$ ]]] ->
  Table[1, {i, 1, Factorial[Length[ $\sigma$ ]}]];

(* JMPRepCS[ $\sigma$ , 0/1, 0/1] gives the left/right regular
perm repn of  $S_n$  with respect to the left/right coset ordering. *)
JMPRepCS[ $\sigma$ ?PermutationQ, 0, side_Integer] :=
  SparseArray[Map[{RankPermutationByCoset[Permute[ $\sigma$ , #], side], RankPermutationByCoset[#, side]} &,
  SymmetricGroupByCosets[Length[ $\sigma$ ], Length[ $\sigma$ ], 0]] -> Table[1, {i, 1, Factorial[Length[ $\sigma$ ]}]];
JMPRepCS[ $\sigma$ ?PermutationQ, 1, side_Integer] := SparseArray[
  Map[{RankPermutationByCoset[Permute[#,  $\sigma$ ], side], RankPermutationByCoset[#, side]} &,
  SymmetricGroupByCosets[Length[ $\sigma$ ], Length[ $\sigma$ ], 0]] -> Table[1, {i, 1, Factorial[Length[ $\sigma$ ]}]];

(* JMMatrix[n, k, 0/1] gives the left/right regular
perm repn of  $M_k$  on  $S_n$  with respect to the right coset basis. *)
JMMatrix[n_Integer, k_Integer, side_Integer] /; n > 1 && k > 1 && k ≤ n :=
  Plus@@Map[JMPRepCS[#, side, 1] &, JMTranspositionList[n, k]];

(* UTILITIES FOR FFT *)
(* splitList splits a list into two lists,
the first containing the elements at odd indices and the second those at even ones. *)
splitList[x?ListQ] := Reap[Map[Sow[x][[#]], Mod[#, 2]] &, Range[1, Length[x]]][[2]];

(* SparseIdentityMatrix generates the sparse identity matrix of size n. *)
SparseIdentityMatrix[n_Integer] /; n > 0 := SparseArray[{{i_, i_} -> 1}, {n, n}];

(* SparseDiagonalMatrix generates a sparse diagonal matrix from a list of values *)
SparseDiagonalMatrix[values?ListQ] :=

```

```

SparseArray[Table[{i, i}, {i, 1, Length[values]}] → values, {Length[values], Length[values]};

(* ColumnSelectionMatrix generates a
matrix that picks out the specified columns from a matrix *)
ColumnSelectionMatrix[l_List, rows_Integer] :=
  SparseArray[MapThread[List, {l, Range[1, Length[l]]}] → Table[1, {Length[l]}, {rows, Length[l]}];

(* Convolution operations *)
VectorToBlocks[v_?VectorQ, blocksizes_?(VectorQ[#, IntegerQ] &)] /;
  Length[v] = Plus@@ (#^2 & /@blocksizes) := Module[{vblocks, blocksquares},
  blocksquares = Map[Plus@@Function[{x}, x^2] /@blocksizes[Range[1, #]]] &,
  Range[1, Length[blocksizes]]];
  vblocks = MapThread[v[Range[#1 + 1, #2]]] &,
  {Join[{0}, blocksquares[Range[1, Length[blocksizes] - 1]]], blocksquares};
  vblocks = MapThread[Partition[#1, #2] &, {vblocks, blocksizes}]
];

ConvolveInFourier[x_?VectorQ, y_?VectorQ, blocksizes_?(VectorQ[#, IntegerQ] &),
  changeofbasis: {_?MatrixQ..}] := Module[{xhat, yhat, product},
  xhat = Fold[Dot[#1, Transpose[#2]] &, {x}, changeofbasis][[1]];
  yhat = Fold[Dot[#1, Transpose[#2]] &, {y}, changeofbasis][[1]];
  product = MapThread[Dot, {VectorToBlocks[xhat, blocksizes], VectorToBlocks[yhat, blocksizes]}]
];

ConvolveInStandard[x_?VectorQ, y_?VectorQ, blocksizes_?(VectorQ[#, IntegerQ] &),
  changeofbasis: {_?MatrixQ..}] := Module[{xhat, yhat, product},
  xhat = Fold[Dot[#1, Transpose[#2]] &, {x}, changeofbasis][[1]];
  yhat = Fold[Dot[#1, Transpose[#2]] &, {y}, changeofbasis][[1]];
  product = MapThread[Dot, {VectorToBlocks[xhat, blocksizes], VectorToBlocks[yhat, blocksizes]}];
  product =
  Fold[Dot[#1, Transpose[Inverse[#2]]] &, {Flatten[product]}, Reverse@changeofbasis][[1]]
];

ConvolveByPermutations[x_?VectorQ, y_?VectorQ, n_Integer] := Module[{xf, yf, products},
  xf = PadRight[x, Factorial[n]][[Range[1, Factorial[n]]]];
  yf = PadRight[y, Factorial[n]][[Range[1, Factorial[n]]]];
  products =
  Sort[Flatten[Outer[{#1[[1]] #2[[1]], RankPermutationByCoset[Permute[UnrankPermutationByCoset[
  #1[[2]], n, 1], UnrankPermutationByCoset[#2[[2]], n, 1], 1]} &,
  MapThread[{#1, #2} &, {xf, Range[1, Factorial[n]]}], MapThread[{#1, #2} &,
  {yf, Range[1, Factorial[n]]}], 1], 1], OrderedQ[{#1[[2]], #2[[2]]}] &];
  products = Partition[products, Factorial[n]];
  Map[Apply[Plus, #] &, Map[Map[First, #] &, products]]
];

(* RREF MATRIX DECOMPOSITION *)
(* Decomposes an m-by-n matrix A of rank k into an m-
by-k matrix D and a k-by-n matrix C such that CD = A *)
CD[A_?MatrixQ] := Module[{B, rank, Am, An, C, D},
  rank = MatrixRank[A]; If[rank ≤ 0, {{}}, {{}}],
  {Am, An} = Dimensions[A]; (* Number of rows, columns in A *)

```

```

B = RowReduce[Transpose[Join[Transpose[A], IdentityMatrix[Am]]]];
C = B[[Range[1, rank], Range[1, An]]];
D = Inverse[
  B[[All, Range[An + 1, An + Am]]]
][[All, Range[1, rank]]];
{C, D}
];

(* FFT FACTORIZATION GENERATION *)
(* JMEigensystem generates bases and eigenvalues for the eigenspaces of the Jucys-
Murphy element  $R_k$  for  $S_n$  acting on the left/right. *)
JMEigensystem[n_Integer, k_Integer, side_Integer] := Module[{jm, sparseidentity, evecs},
  jm = JMMatrix[n, k, side];
  sparseIdentity = SparseIdentityMatrix[Factorial[n]];
  evecs =
  Reverse@DeleteCases[Map[NullSpace[jm - # sparseIdentity] &, Range[-k + 1, k - 1]], _?ListNullQ];
  {evecs, Map[{#[[1]]}.jm.#[[1]][[1]] / Norm[#[[1]]]^2 &, evecs]}
];

(* MakeFFTFactor takes the eigenbases and eigenvalues
for the JM element and a set of decompression matrices and returns a
new set of decompression matrices and the next factor in the FFT *)
MakeFFTFactor[evecs_, evals_, decompresslist_] := Module[{vecdecompresslist,
  clist, csublist, dnewlist = {}, devals = {}, drank, projranks, crank, C, D, i, j},
  Print[Timing[vecdecompresslist = Map[Transpose[#].Inverse[#.Transpose[#]] &, evecs];][[1]]];
  Print["Finished inverses."];
  projranks = MatrixRank/@evecs;
  clist = {};
  For[i = 1, i ≤ Length[decompresslist], i++,
    csublist = {};
    drank = MatrixRank[decompresslist[[i]]];
    For[j = 1, j ≤ Length[evecs] && drank > 0, j++,
      {C, D} = CD[evecs[[j]].decompresslist[[i]]];
      crank = MatrixRank[C];
      If[crank > 0,
        AppendTo[csublist, C];
        AppendTo[dnewlist, SmartDot[vecdecompresslist[[j]], D]];
        AppendTo[devals, evals[[j]]];
        drank = drank - crank;
        projRanks = ReplacePart[projranks, projranks[[j]] - crank, j];,];
  ];

(* cull out projection matrices that have already been completely projected *)
(* use Position instead? try that next *)
For[j = 1, j ≤ Length[projranks], j++,
  If[projranks[[j]] = 0,
    projranks = Delete[projranks, j];
    evecs = Delete[evecs, j];
    vecdecompresslist = Delete[vecdecompresslist, j];
    evals = Delete[evals, j];
    j--;

```

```

];
];
AppendTo[clist, SmartColumnAppend@@ csublist];
];
devals = Flatten[MapThread[Table[#1, {#2}] &, {devals, MatrixRank/@dnewlist}]];
dnewlist = DeleteCases[dnewlist, _?MatrixNullQ];
{dnewlist, devals, MatrixDirectSumSparse[clist]}
];

ModifyFFTEigenvalueList[devallist_] := Module[{devalsplitlist, devallastrow, lastletterordersort},
  (* split lists of eigenvalues into lists for left, right actions *)
  devalsplitlist = splitList@devallist;

  (* compute list of eigenvalues from last right-
     action based on differences of previous eigenvalues; add to master list*)
  devallastrow = Plus@@(devalsplitlist[[1]]) - Plus@@(devalsplitlist[[2]]);
  AppendTo[devalsplitlist[[2]], devallastrow];

  (* transpose eigenvalue list to create left/right lists corresponding to each row *)
  devalsplitlist = Transpose/@devalsplitlist;

  (* add row number, sorted list of eigenvalues to each entry in eigenvalue master list;
     sorted list of eigenvalues determines which  $S_n$  irreducible row belongs to *)
  devalsplitlist = MapThread[Append[#1, #2] &,
    {MapThread[{#1, #2, Sort[#1]} &, devalsplitlist], Range[1, Length[devalsplitlist[[1]]]}]];

  (* sort master list first based on sorted eval list (in order to sort by irreducible) *)
  devalsplitlist = Sort[devalsplitlist, OrderedQ[{#2[[3]], #1[[3]]}] &];
  (* split by irreducible *)
  devalsplitlist = Split[devalsplitlist, SameQ[#1[[3]], #2[[3]]] &];

  (* now sort entries in each irreducible into last-
     letter order (which is equivalent to lex. order on reversed eigenvalue list) *)
  lastletterordersort[x_] := Sort[x, OrderedQ[
    {Reverse[Join[#2[[2]], #2[[1]]], Reverse[Join[#1[[2]], #1[[1]]]}] &];
  devalsplitlist = Map[lastletterordersort, devalsplitlist];

  (* return modified master list *)
  devalsplitlist
];

(* MakeFFTScalingMatrix makes the row-scaling matrix for the FFT factorization
   from the previous factors and from the sizes of the irreducible repns *)
MakeFFTScalingMatrix[n_Integer, fftfactors_List, irredrepsizes_List] :=
Module[{matrixsize, scalingentries, remainingrows, nextrow, currentscalingmatrix,
  fftvalues, snvalues, nonzerolocs, newscalings, genmatrices, i},
  matrixsize = Factorial[n];
  scalingentries = Table[1, {matrixsize}];
  remainingrows = Range[1, matrixsize];
  currentscalingmatrix = SparseDiagonalMatrix[scalingentries];

```

```

(* make sure all entries for identity are 1 *)
(* note: _should_ get column selection numbers by generating permutation by right-
  coset order, then ranking permutation by FFT order;
  will revise once we start testing other bases *)
fftvalues = Flatten[Dot@@Reverse@fftfactors.ColumnSelectionMatrix[{1}, matrixsize]];
nonzerolocs =
  Reap[MapThread[(If[#1 == 0, Sow[#2]]; #1) &, {fftvalues, Range[1, matrixsize]}][[2]][[1]];
nonzerolocs = Intersection[nonzerolocs, remainingrows];
scalingentries[nonzerolocs] =
  MapThread[#1/#2 &, {Table[1, {Length[nonzerolocs]}], fftvalues[nonzerolocs]}];
remainingrows = Complement[remainingrows, nonzerolocs];
currentscalingmatrix = SparseDiagonalMatrix[scalingentries];

(* get scalings for (i i+1) transpositions *)
genmatrices = GenSNMatrices[n, fftfactors, currentscalingmatrix, irredrepsizes];

For[i = 1, i <= Length[genmatrices], i++,
  fftvalues = Flatten[currentscalingmatrix.
    (Dot@@Reverse@fftfactors.ColumnSelectionMatrix[{i Factorial[i] + 1}, matrixsize])];
  snvalues = Flatten[genmatrices[[i]]];
  nonzerolocs =
    Reap[MapThread[(If[#1 == 0, Sow[#2]]; #1) &, {fftvalues, Range[1, matrixsize]}][[2]][[1]];
  nonzerolocs = Intersection[nonzerolocs, remainingrows];
  Print[nonzerolocs];
  Print[Normal[fftvalues[nonzerolocs]]];
  newscalings = MapThread[#1/#2 &, {snvalues[nonzerolocs], fftvalues[nonzerolocs]}];
  Print[newscalings];
  scalingentries[nonzerolocs] = newscalings;
  remainingrows = Complement[remainingrows, nonzerolocs];
  Print[remainingrows];
];

(* compute scalings for remaining entries *)
While[Length[remainingrows] > 0,
  nextrow = First[remainingrows];
  (* select next row from FFT matrix *)
  rowvalues =
    Flatten[Transpose[ColumnSelectionMatrix[{nextrow}, matrixsize].(Dot@@Reverse@fftfactors)]];
  nextcolumn = 1;
  While[rowvalues[[nextcolumn]] == 0, nextcolumn++];
  Print[nextcolumn];
  fftvalues = Flatten[currentscalingmatrix.
    (Dot@@Reverse@fftfactors.ColumnSelectionMatrix[{nextcolumn}, matrixsize])];
  snvalues = Flatten[SNMatricesFromGens[n, nextcolumn, genmatrices]];
  nonzerolocs =
    Reap[MapThread[(If[#1 == 0, Sow[#2]]; #1) &, {fftvalues, Range[1, matrixsize]}][[2]][[1]];
  nonzerolocs = Intersection[nonzerolocs, remainingrows];
  Print[nonzerolocs];
  Print[Normal[fftvalues[nonzerolocs]]];
  newscalings = MapThread[#1/#2 &, {snvalues[nonzerolocs], fftvalues[nonzerolocs]}];
  Print[newscalings];
];

```

```

    scalingentries[[nonzerolocs]] = newscalings;
    remainingrows = Complement[remainingrows, nonzerolocs];
    Print[remainingrows];
];

SparseDiagonalMatrix[scalingentries]
];

(* GenSNMatrices creates the lists of Young's seminormal repn matrices for the
generating transpositions (i i+1) from the FFT matrices and the irred sizes *)
GenSNMatrices[n_Integer, fftfactors_, currentscalingmatrix_, irredrepsizes_] :=
Module[{genindices, gencolumns, genmatrices, snmatrices, matrixsize},
    matrixsize = Factorial[n];
    genindices = Table[i Factorial[i] + 1, {i, 1, n - 1}];
    gencolumns = Transpose[currentscalingmatrix.
        (Dot@@Reverse@fftfactors.ColumnSelectionMatrix[genindices, matrixsize])];
    genmatrices = VectorToBlocks[#, irredrepsizes] & /@gencolumns;
    Print[Map[MatrixForm, genmatrices, {2}]];

    snmatrices = Map[IrredSNMatrix, genmatrices, {2}];
    Print[Map[MatrixForm, snmatrices, {2}]];
    snmatrices
];

IrredSNMatrix[matrix_] := Module[{nonzerolist, diagelems, sntentries, dim},
    dim = Dimensions[matrix][[1]];
    diagelems = Map[matrix[#, #] &, Range[1, dim]];
    nonzerolist = Reap[MapThread[If[#1 = 0, , Sow[#2]] &,
        {matrix, Table[{i, j}, {i, 1, dim}, {j, 1, dim}], 2}][[2]][[1]];
    sntentries = Map[
        If[#[[1]] > #[[2]], 1, If[#[[1]] < #[[2]], 1 - diagelems[[#[[1]]]]^2, diagelems[[#[[1]]]]]] &,
        nonzerolist, {1}];
    SparseArray[nonzerolist -> sntentries, {dim, dim}]
    (* If[#[[1]] > #[[2]], 1, If[#[[1]] < #[[2]], 1 - diagelems[[#[[1]]]]^2, diagelems[[#[[1]]]]] *)
];

SNMatricesFromGens[n_Integer, rank_Integer, genmatrices_] := Module[{decomp, snmatrices},
    decomp = Map[{Mod[Floor[(rank - 1) / Factorial[#]], # + 1], # + 1} &, Range[1, n - 1]];
    decomp = DeleteCases[decomp, {0, _}];
    (* start with identity matrix *)
    snmatrices = MapThread[Dot, {genmatrices[[1]], genmatrices[[1]]}];
    If[Length[decomp] > 0,
        snmatrices = MapThread[Dot, Map[SNTransposMatrices[#[[1]], #[[2]], genmatrices] &, decomp, 1]];
        snmatrices = MapThread[Dot, {genmatrices[[1]], genmatrices[[1]]}];
    snmatrices
];

SNTransposMatrices[i_Integer, j_Integer, genmatrices_] /; i < j := Module[{indices, snmatrices},
    indices = Join[Range[i, j - 1]];
    indices = Join[indices, Reverse[Delete[indices, -1]]];
    MapThread[Dot, Map[genmatrices[#[#]] &, indices]]
];

```

```

];

(* FFTFactorizationToStage[n, s] generates the DFT matrix decompositon to stage s. *)
FFTFactorizationToStage[n_Integer, stage_Integer] /; n ≥ 2 && stage > 0 && stage ≤ 2 n - 3 :=
Module[{factorlist = {}, dlist, devallist = {}, devalfulllist = {},
  irreddims, jmevecs, jmevals, dnevevals, newfactor, s, k, side},
  sparseIdentity = SparseIdentityMatrix[Factorial[n]];
  dlist = {sparseIdentity};
  For[s = 1, s ≤ stage, s++,
    k = Floor[(s + 3) / 2];
    side = Mod[s + 1, 2];
    Print["Stage ", k, If[side = 0, "L", "R"]];

    {jmevecs, jmevals} = JMEigensystem[n, k, side];
    Print["Generated sets of eigenvectors."];

    {dlist, dnevevals, newfactor} = MakeFFTFactor[jmevecs, jmevals, dlist];
    AppendTo[devallist, dnevevals];

    Print["Number of Decompressors: ", Length[dlist]];
    AppendTo[factorlist, newfactor];
  ];

  (* generate permutation matrix, scaling matrix, convolution operator from eigenvalue lists *)
  devalfulllist = ModifyFFTEigenvalueList[devallist];

  (* create permutation matrix from row tags in full eigenvalue list *)
  AppendTo[factorlist, PermutationMatrix[Flatten[Map#[[4] &, devalfulllist, {2}]]]];

  (* get dimensions of each irreducible representation
  from number of distinct eigenvalue lists in each irred repn *)
  irreddims = Length/@Union/@Sort/@Map[Map[First, #] &, devalfulllist];

  (* calculate scaling matrix, add to list*)
  AppendTo[factorlist, MakeFFTScalingMatrix[n, factorlist, irreddims]];

  {devalfulllist, factorlist, irreddims}
  (* Return full eigenvalue info, list of matrices, lengths of irreducibles *)
];

(* FFTFactorization[n] generates the DFT matrix factorization for Sn. *)
FFTFactorization[n_Integer] /; n > 1 := FFTFactorizationToStage[n, 2 n - 3];

```

References

- [1] William A. Adkins and Steven H. Weintraub. *Algebra: An Approach via Module Theory*. Number 136 in Graduate Texts in Mathematics. Springer, New York, 1992.

ANNOTATION: This text presents the material for a first-year graduate course in algebra from a module-theoretic viewpoint. Covers groups, rings, and fields, modules and their structure theorems, and some linear algebra, as well as additional topics in module theory and the representation theory of finite groups. Especially useful as a reference on module theory and module-theoretic formulations of representation theory.

- [2] Gregory S. Chirikjian and Alexander B. Kyatkin. *Engineering Applications of Noncommutative Harmonic Analysis: With Emphasis on Rotation and Motion Groups*. CRC Press, Boca Raton, FL, 2001.

- [3] Michael Clausen. Fast generalized fourier transforms. *Theoretical Computer Science*, 67:55–63, 1989.

ANNOTATION: Clausen improves the upper bound for the linear complexity of FFTs on a general group G using bases adapted to chains of subgroups. The results also apply to inverse FFTs. He then derives an $O(n^3|S_n|)$ upper bound for FFTs on the symmetric group S_n .

- [4] Michael Clausen. Elements of a general algebraic theory of standard tableaux. In A. Betten, A. Kohnert, R. Laue, and A. Wassermann, editors, *Algebraic Combinatorics and Applications*, pages 67–78, Berlin, 2001. Springer-Verlag.

ANNOTATION: Clausen generalizes the standard Young tableaux used to determine the irreducible representations of the symmetric group to tableaux that apply to groups that have multiplicity-free character graphs (MC-groups). This framework is applied to supersolvable groups to yield an efficient determination of their irreducible representations. Such a decomposition is a first step towards determining FFTs on such MC-groups.

- [5] Michael Clausen and Ulrich Baum. *Fast Fourier Transforms*. BI-Wissenschaftsverlag, Mannheim, Germany, 1993.

ANNOTATION: This text provides an excellent introduction to the theory of generalized FFTs as of 1993. It discusses the necessary background in group algebras and their representations and in linear complexity theory and then provides FFTs for abelian groups, solvable groups, supersolvable groups, and the symmetric groups. The symmetric group algorithms it provides are those found in Clausen and Baum's 1993 article [6].

- [6] Michael Clausen and Ulrich Baum. Fast fourier transforms for symmetric groups: Theory and implementation. *Mathematics of Computation*, 61(204):833–847, October 1993.

ANNOTATION: Clausen and Baum present an implementable algorithm for an FFT and an inverse FFT on the symmetric group S_n that requires $\frac{1}{2}(n^3 + n^2)n!$ operations. This implementation realizes the upper bound on linear complexity that Clausen specified in his 1989 article [3]. The algorithm relies on a left-coset factorization of S_n .

- [7] Persi Diaconis. A generalization of spectral analysis with application to ranked data. *The Annals of Statistics*, 17(3):949–979, 1989.

ANNOTATION: Diaconis presents a means of analyzing both fully and partially ranked data using the representation theory of the symmetric group, and in particular to identify higher-order effects in ranking preferences. He applies these techniques to voting data from the APA presidential elections as well as from other sources. Finally, he investigates the statistical significance of the analysis he presents. Also includes an appendix pertaining to the decomposition of symmetry spaces into isotypics.

- [8] James R. Driscoll and Dennis M. Healy, Jr. Computing fourier transforms and convolutions on the 2-sphere. *Advances in Applied Mathematics*, 15:202–250, 1994.

ANNOTATION: Discusses computationally efficient approaches to Fourier transforms (spherical harmonic expansions) on the unit two-sphere. In particular, a fast discrete Fourier transform on the two-sphere, a fast inverse transform, and a fast convolution algorithm are all presented. Theoretical and experimental effects of finite precision computations are also considered.

- [9] David Dummit and Richard Foote. *Abstract Algebra*. John Wiley and Sons, New York, 1999.

ANNOTATION: Standard advanced undergraduate or first-year graduate student text on algebra, covering the basic theory of groups, rings, and fields along with additional advanced topics. Most useful for the material on module theory (Chs. 10–12) and representation theory (Chs. 18–19).

- [10] Robert Eisberg and Robert Resnick. *Quantum Physics of Atoms, Molecules, Solids, Nuclei, and Particles*. John Wiley and Sons, New York, 1985.

- [11] Kenneth I. Gross. On the evolution of noncommutative harmonic analysis. *Am. Math. Monthly*, 85:525–548, August 1978.

ANNOTATION: Gross provides an overview of the development of noncommutative harmonic analysis. He covers its beginnings in classical Fourier series, its generalizations to functions on finite groups, the Peter-Weyl Theorem and harmonic

analysis on compact Lie groups, the special case of spherical harmonics, classical harmonic analysis on noncompact domains, and applications to quantum mechanics.

- [12] Gordon James and Adalbert Kerber. *The Representation Theory of the Symmetric Group*. Addison-Wesley, Reading, MA, 1981.

ANNOTATION: This text discusses the classic representations of S_n as determined by Alfred Young. Of particular interest to this project is Ch. 3, which derives the ordinary irreducible matrix representations of S_n from the decomposition of its group algebra through the seminormal basis of the algebra. Also potentially applicable is the discussion of the representation of wreath products of S_n , some of which correspond to the Weyl groups.

- [13] J. David Logan. *Applied Partial Differential Equations*. Undergraduate Texts in Mathematics. Springer-Verlag, New York, 1998.

- [14] David K. Maslen. The efficient computation of fourier transforms on the symmetric group. *Mathematics of Computation*, 67(223):1121–1147, July 1998.

ANNOTATION: Maslen presents an algorithm for an $O(n^2|S_n|)$ FFT algorithm for the symmetric groups S_n , which improves upon the results Clausen and Baum give in their 1993 article [6]. He accomplishes this by summing over the Young tableaux associated with the representations of S_n ; these sums over these combinatorial objects render the algorithms difficult to implement, however.

- [15] David K. Maslen, Michael E. Orrison, and Daniel N. Rockmore. Computing isotypic projections with the Lanczos iteration. *SIAM J. Matrix Anal. Appl.*, 25(3):784–803, 2004.

ANNOTATION: Presents material on isotypic projections and their applications to decimation-in-frequency FFTs similar to that found in [20].

- [16] David K. Maslen and Daniel N. Rockmore. Generalized FFTs - a survey of some recent results. *DIMACS Series in Discrete Mathematics and Computer Science*, 28:183–237, 1997.

ANNOTATION: Surveys results pertaining to the generalization of FFTs in a group-theoretic context. Discusses FFTs on abelian groups, on non-abelian finite groups, and on band-limited functions on compact Lie groups. Also covers results on fast discrete polynomial transforms. Closes with a set of open questions in the field.

- [17] G. E. Murphy. A new construction of young's seminormal representation of the symmetric groups. *Journal of Algebra*, 69:287–297, 1981.

ANNOTATION: Murphy introduces a simple set of elements L_u of KS_n that he uses to construct Young's seminormal representation of the symmetric group. These elements arise as differences of class sums of transpositions and hence are simultaneously diagonalizable. Murphy also uses these elements to reformulate proof of other relations pertaining to the representations of the symmetric group.

- [18] G. E. Murphy. The idempotents of the symmetric group and nakayama's conjecture. *Journal of Algebra*, 81:58–265, 1983.

ANNOTATION: Murphy uses the elements L_u from his 1981 paper [17] to construct the primitive idempotents of KS_n for a field K of arbitrary characteristic and to demonstrate that the ring of symmetric function in L_u is equal to the center of KS_n . He further uses this framework to give a simple proof of Nakayama's Conjecture.

- [19] Andrei Okounkov and Anatoly Vershik. A new approach to representation theory of symmetric groups. *Selecta Mathematica*, 2(4):581–605, 1996.

ANNOTATION: Okounkov and Vershik use Gelfand-Tsetlin bases and Jucys-Murphy elements to construct the Young tableaux for the symmetric group more naturally. They generalize these techniques in the construction of similar characterizations for other Coxeter groups.

- [20] Michael E. Orrison. *An eigenspace approach to decomposing representations of finite groups*. PhD thesis, Dartmouth College, 2001.

ANNOTATION: This document provides many of the basic theoretical techniques for determining decimation-in-frequency generalized FFTs, including the concept of projecting into isotypic subspaces with primitive idempotents. Discusses the computation of such projections through the Lanczos iteration. Gives several useful example involving the symmetric group, the hyperoctahedral group, and the finite general linear and symplectic groups.

- [21] Sriram Pemmaraju and Steven Skiena. *Computational Discrete Mathematics: Combinatorics and Graph Theory with Mathematica*. Cambridge University Press, Cambridge, 2003.

ANNOTATION: Provides extensive documentation on the *Combinatorica* package for *Mathematica*. Material on algebraic combinatorics, permutation manipulations, and Young tableau and partitions appears to be the most relevant to this project.

- [22] Markus Püschel, José M. F. Moura, Jeremy Johnson, David Padua, Manuela Veloso, Bryan W. Singer, Jianxin Xiong, Franz Franchetti, Aca Gačić, Yevgen Voronenko, Kang Chen, Robert W. Johnson, and Nick Rizzolo. SPIRAL: Code generation for DSP transforms. *Proceedings of the IEEE, special issue on "Program Generation, Optimization, and Adaptation"*, 93(2), 2005. to appear.

ANNOTATION: Provides an overview of the SPIRAL system, which generates structured representations of linear transformations used in signal processing automatically.

- [23] Arun Ram. Seminormal representations of Weyl groups and Iwahori-Hecke algebras. *Proc. London Math. Soc.*, 73:99–133, 1997.

ANNOTATION: Ram presents a method of determining representations for Weyl groups and Iwahori-Hecke algebras through a generalization of Young’s seminormal representations for S_n . He accomplishes this by constructing Jucys-Murphy elements for these groups and algebras. Such representations are key for determining FFTs on general Weyl groups.

- [24] Daniel N. Rockmore. Recent progress and applications in group FFTs. NATO Advanced Study Institute on Computational Noncommutative Algebra and Applications, July 2003.

ANNOTATION: Provides an accessible introduction to the state of generalized FFTs circa 2003. Includes good descriptions of both the Cooley-Tukey decimation-in-time FFT and the Gentleman-Sande decimation-in-frequency FFT as well as their generalizations to nonabelian finite, compact, and noncompact groups.

- [25] Bruce E. Sagan. *The Symmetric Group: Representations, Combinatorial Algorithms, and Symmetric Functions*. Wadsworth & Brooks/Cole, Pacific Grove, CA, 1991.

ANNOTATION: Provides an accessible survey of group representation theory, the representations of the symmetric group, combinatorial algorithms associated with these representations, and results and techniques involving symmetric functions. Of particular interest here is the representation theory of S_n , which Sagan approaches through an analysis of Specht modules.

- [26] Jean-Pierre Serre. *Linear Representations of Finite Groups*. Number 42 in Graduate Texts in Mathematics. Springer, New York, 1977.

ANNOTATION: This is a classic text on representation theory and will be useful primarily as a reference. Many articles on the theoretical framework of generalized FFTs refer to this text for important theorems in representation theory.

- [27] N. J. A. Sloane. Sequence A000041 in *The Online Encyclopedia of Integer Sequences*. Found at <http://www.research.att.com/~njas/sequences/>.

- [28] John S. Townsend. *A Modern Approach to Quantum Mechanics*. University Science Books, Sausalito, CA, 2000.