Chapter 1

DFT and FFT: An Algebraic View

In infinite, or non-periodic, discrete-time signal processing, there is a strong connection between the z-transform, Laurent series, convolution, and the discrete-time Fourier transform (DTFT) [10]. As one may expect, a similar connection exists for the DFT but bears surprises. Namely, it turns out that the proper framework for the DFT requires modulo operations of polynomials, which means working with so-called polynomial algebras [6]. Associated with polynomial algebras is the Chinese remainder theorem, which describes the DFT algebraically and can be used as a tool to concisely derive various FFTs as well as convolution algorithms [9, 20, 21, 1] (see also Chapter **??**). The polynomial algebra framework was fully developed for signal processing as part of the algebraic signal processing theory. It identifies the structure underlying many transforms used in signal processing, provides deep insight into their properties, and enables the derivation of their fast algorithms [14, 12, 11, 13]. Here we focus on the algebraic description of the DFT and on the algebraic derivation of the general-radix Cooley-Tukey FFT from Chapter **??**. We start with motivating the appearance of modulo operations.

The z-transform associates with infinite discrete signals $X = (\dots, x(-1), x(0), x(1), \dots)$, a Laurent series:

$$X \mapsto X(s) = \sum_{n \in \mathbb{Z}} x(n) s^n. \tag{1.1}$$

Here we used $s = z^{-1}$ to simplify the notation in the following. The DTFT of X is the evaluation of $X(s)$ on the unit circle

$$X(e^{-j\omega}), \quad -\pi < \omega \leq \pi. \tag{1.2}$$

Finally, filtering or (linear) convolution is simply the multiplication of Laurent series,

$$H * X \leftrightarrow H(s)X(s). \tag{1.3}$$

For finite signals $X = (x(0), \dots, x(N-1))$ one expects that the equivalent of

Concept	Infinite time	Finite time
Signal	$X(s) = \sum_{n\in\mathbb{Z}} x(n)s^n$	$\sum_{n=0}^{N-1} x(n)s^n$
Filter	$H(s) = \sum_{n\in\mathbb{Z}} h(n)s^n$	$\sum_{n=0}^{N-1} h(n)s^n$
Convolution	Linear: $H(s)X(s)$	Circular: $H(s)X(s)$ mod $(s^n - 1)$
Fourier transform	DTFT: $X(e^{-j\omega}), \quad -\pi < \omega \leq \pi$	DFT: $X(e^{-j\frac{2\pi k}{n}}), \quad 0 \leq k < n$

Table 1.1: Infinite and finite discrete time signal processing.

(1.1) becomes a mapping to polynomials of degree $N-1$,

$$X \mapsto X(s) = \sum_{n=0}^{N-1} x(n)s^n, \tag{1.4}$$

and that the DFT is an evaluation of these polynomials. Indeed, the definition of the DFT in (**??**) shows that

$$C(k) = X(W_N^k) = X(e^{-j\frac{2\pi k}{N}}), \quad 0 \leq k < N, \tag{1.5}$$

i.e., the DFT computes the evaluations of the polynomial $X(s)$ at the n nth roots of unity.

The problem arises with the equivalent of (1.3), since the multiplication $H(s)X(s)$ of two polynomials of degree $N-1$ yields one of degree $2N-2$. Also, it does not coincide with the circular convolution known to be associated with the DFT. The solution to both problems is to reduce the product modulo $s^n - 1$:

$$H *_{\text{circ}} X \leftrightarrow H(s)X(s) \text{ mod } (s^n - 1). \tag{1.6}$$

The resulting polynomial then has again degree $N-1$ and this form of convolution becomes equivalent to circular convolution of the polynomial coefficients. We also observe that the evaluation points in (1.5) are precisely the roots of $s^n - 1$. This connection will become clear in this chapter.

The discussion is summarized in Table 1.1.

The proper framework to describe the multiplication of polynomials modulo a fixed polynomial are polynomial algebras. Together with the Chinese remainder theorem, they provide the theoretical underpinning for the DFT and the Cooley-Tukey FFT.

In this chapter, the DFT will naturally arise as a linear mapping with respect to chosen bases, i.e., as a matrix. Indeed, the definition shows that if all input and outputs are collected into vectors $X = (X(0), \dots, X(N-1))$ and $C = (C(0), \dots C(N-1))$, then (**??**) is equivalent to

$$C = \text{DFT}_N\, X, \tag{1.7}$$

where

$$\text{DFT}_N = [W_N^{kn}]_{0\leq k,n<N}. \tag{1.8}$$

The matrix point of view is adopted in the FFT books [18, 17].

1.1 Polynomial Algebras and the DFT

In this section we introduce polynomial algebras and explain how they are associated to transforms. Then we identify this connection for the DFT. Later we use polynomial algebras to derive the Cooley-Tukey FFT.

For further background on the mathematics in this section and polynomial algebras in particular, we refer to [6].

1.1.1 Polynomial Algebra

An algebra $\mathcal{A}$ is a vector space that also provides a multiplication of its elements such that the distributivity law holds (see [6] for a complete definition). Examples include the sets of complex or real numbers $\mathbb{C}$ or $\mathbb{R}$, and the sets of complex or real polynomials in the variable s: $\mathbb{C}[s]$ or $\mathbb{R}[s]$.

The key player in this chapter is the *polynomial algebra*. Given a fixed polynomial $P(s)$ of degree $\deg(P) = N$, we define a polynomial algebra as the set

$$\mathbb{C}[s]/P(s) = \{X(s) \mid \deg(X) < \deg(P)\}$$

of polynomials of degree smaller than N with addition and multiplication modulo P. Viewed as a vector space, $\mathbb{C}[s]/P(s)$ hence has dimension N.

Every polynomial $X(s) \in \mathbb{C}[s]$ is reduced to a unique polynomial $R(s)$ modulo $P(s)$ of degree smaller than N. $R(s)$ is computed using division with rest, namely

$$X(s) = Q(s)P(s) + R(s), \quad \deg(R) < \deg(P).$$

Regarding this equation modulo P, $P(s)$ becomes zero, and we get

$$X(s) \equiv R(s) \text{ mod } P(s).$$

We read this equation as "$X(s)$ is congruent (or equal) $R(s)$ modulo $P(s)$." We will also write $X(s)$ mod $P(s)$ to denote that $X(s)$ is reduced modulo $P(s)$. Obviously,

$$P(s) \equiv 0 \text{ mod } P(s).$$

As a simple example we consider $\mathcal{A} = \mathbb{C}[s]/(s^2 - 1)$, which has dimension 2. A possible basis is $b = (1, s)$. In $\mathcal{A}$, for example, $s \cdot (s+1) = s^2 + s \equiv s + 1$ mod $(s^2 - 1)$, obtained through division with rest

$$s^2 + s = 1 \cdot (s^2 - 1) + (s + 1)$$

or simply by replacing s^2 with 1 (since $s^2 - 1 = 0$ implies $s^2 = 1$).

1.1.2 Chinese Remainder Theorem (CRT)

Assume $P(s) = Q(s)R(s)$ factors into two coprime (no common factors) polynomials Q and R. Then the Chinese remainder theorem (CRT) for polynomials is the linear mapping[1]

$$\begin{array}{rcl}\Delta:\ \mathbb{C}[s]/P(s) & \to & \mathbb{C}[s]/Q(s)\oplus\mathbb{C}[s]/R(s),\\ X(s) & \mapsto & (X(s)\bmod Q(s), X(s)\bmod R(s)).\end{array}$$

Here, $\oplus$ is the Cartesian product of vector spaces with elementwise operation (also called outer direct sum). In words, the CRT asserts that computing (addition, multiplication, scalar multiplication) in $\mathbb{C}[s]/P(s)$ is equivalent to computing in parallel in $\mathbb{C}[s]/Q(s)$ and $\mathbb{C}[s]/R(s)$.

If we choose bases b, c, d in the three polynomial algebras, then Δ can be expressed as a matrix. As usual with linear mappings, this matrix is obtained by mapping every element of b with Δ, expressing it in the concatenation $c \cup d$ of the bases c and d, and writing the results into the columns of the matrix.

As an example, we consider again the polynomial $P(s) = s^2 - 1 = (s-1)(s+1)$ and the CRT decomposition

$$\Delta:\ \mathbb{C}[s]/(s^2-1) \to \mathbb{C}[s]/(x-1)\oplus\mathbb{C}[s]/(x+1).$$

As bases, we choose $b = (1, x),\ c = (1),\ d = (1)$. $\Delta(1) = (1, 1)$ with the same coordinate vector in $c \cup d = (1, 1)$. Further, because of $x \equiv 1 \bmod (x-1)$ and $x \equiv -1 \bmod (x+1)$, $\Delta(x) = (x, x) \equiv (1, -1)$ with the same coordinate vector. Thus, Δ in matrix form is the so-called butterfly matrix, which is a DFT of size 2: $\mathrm{DFT}_2 = \left[\begin{smallmatrix}1 & 1\\ 1 & -1\end{smallmatrix}\right]$.

1.1.3 Polynomial Transforms

Assume $P(s) \in \mathbb{C}[s]$ has pairwise distinct zeros $\alpha = (\alpha_0, \ldots, \alpha_{N-1})$. Then the CRT can be used to completely decompose $\mathbb{C}[s]/P(s)$ into its *spectrum*:

$$\begin{aligned}\Delta:\ \mathbb{C}[s]/P(s) &\to \mathbb{C}[s]/(s-\alpha_0)\oplus\ldots\oplus\mathbb{C}[s]/(s-\alpha_{N-1}),\\ X(s) &\mapsto (X(s)\bmod (s-\alpha_0), \ldots, X(s)\bmod (s-\alpha_{N-1}))\\ &= (s(\alpha_0), \ldots, s(\alpha_{N-1})).\end{aligned}\tag{1.9}$$

If we choose a basis $b = (P_0(s), \ldots, P_{N-1}(s))$ in $\mathbb{C}[s]/P(s)$ and bases $b_i = (1)$ in each $\mathbb{C}[s]/(s-\alpha_i)$, then Δ, as a linear mapping, is represented by a matrix. The matrix is obtained by mapping every basis element P_n, $0 \le n < N$, and collecting the results in the columns of the matrix. The result is

$$\mathcal{P}_{b,\alpha} = [P_n(\alpha_k)]_{0\le k,n<N}$$

and is called the *polynomial transform* for $\mathcal{A} = \mathbb{C}[s]/P(s)$ with basis b.

If, in general, we choose $b_i = (\beta_i)$ as spectral basis, then the matrix corresponding to the decomposition (1.9) is the *scaled polynomial transform*

$$\mathrm{diag}_{0\le k<N}(1/\beta_n)\mathcal{P}_{b,\alpha},$$

where $\mathrm{diag}_{0\le n<N}(\gamma_n)$ denotes a diagonal matrix with diagonal entries γ_n.

We jointly refer to polynomial transforms, scaled or not, as Fourier transforms.

[1] More precisely, isomorphism of algebras or isomorphism of $\mathcal{A}$-modules.

1.1.4 DFT as a Polynomial Transform

We show that the DFT_N is a polynomial transform for $\mathcal{A} = \mathbb{C}[s]/(s^N - 1)$ with basis $b = (1, s, \dots, s^{N-1})$. Namely,

$$s^N - 1 = \prod_{0 \le k < N} (x - W_N^k),$$

which means that Δ takes the form

$$\begin{aligned} \Delta:\ \mathbb{C}[s]/(s^N - 1) &\to \mathbb{C}[s]/(s - W_N^0) \oplus \dots \oplus \mathbb{C}[s]/(s - W_N^{N-1}), \\ X(s) &\mapsto (X(s) \bmod (s - W_N^0), \dots, X(s) \bmod (s - W_N^{N-1})) \\ &= (X(W_N^0), \dots, X(W_N^{N-1})). \end{aligned} \tag{1.10}$$

The associated polynomial transform hence becomes

$$\mathcal{P}_{b,\alpha} = [W_N^{kn}]_{0 \le k,n < N} = \mathrm{DFT}_N\,.$$

This interpretation of the DFT has been known at least since [20, 9] and clarifies the connection between the evaluation points in (1.5) and the circular convolution in (1.6).

In [3], DFTs of types 1–4 are defined, with type 1 being the standard DFT. In the algebraic framework, type 3 is obtained by choosing $\mathcal{A} = \mathbb{C}[s]/(s^N + 1)$ as algebra with the same basis as before:

$$\mathcal{P}_{b,\alpha} = [W_N^{(k+1/2)n}]_{0 \le k,n < N} = \text{DFT-3}_N, \tag{1.11}$$

The DFTs of type 2 and 4 are scaled polynomial transforms [14].

1.2 Algebraic Derivation of the Cooley-Tukey FFT

Knowing the polynomial algebra underlying the DFT enables us to derive the Cooley-Tukey FFT *algebraically*. This means that instead of manipulating the DFT definition, we manipulate the polynomial algebra $\mathbb{C}[s]/(s^N - 1)$. The basic idea is intuitive. We showed that the DFT is the matrix representation of the complete decomposition (1.10). The Cooley-Tukey FFT is now derived be performing this decomposition *in steps* as shown in Fig. (1.1). Each step yields a sparse matrix; hence, the DFT_N is factorized into a product of sparse matrices, which will be the matrix representation of the Cooley-Tukey FFT.

This stepwise decomposition can be formulated generically for polynomial transforms [15, 13]. Here, we consider only the DFT.

We first introduce the matrix notation we will use and in particular the Kronecker product formalism that became mainstream for FFTs in in [18, 17].

Then we first derive the radix-2 FFT using a *factorization* of $s^N - 1$. Subsequently, we obtain the general-radix FFT using a *decomposition* of $s^N - 1$.

$$\begin{tikzcd}
\mathbb{C}[s]/P(s) \arrow[dd, "\text{Fourier transform}"'] \arrow[dr] & \\
& \text{partial decomposition} \arrow[dl] \\
\bigoplus_{0\leq k<N} \mathbb{C}[s]/(s-\alpha_k) &
\end{tikzcd}$$

Figure 1.1: Basic idea behind the algebraic derivation of Cooley-Tukey type algorithms for a Fourier transform.

1.2.1 Matrix Notation

We denote the $N \times N$ identity matrix with I_N, and diagonal matrices with

$$\operatorname{diag}_{0\leq k<N}(\gamma_k) = \begin{bmatrix} \gamma_0 & & \\ & \ddots & \\ & & \gamma_{N-1} \end{bmatrix}.$$

The $N \times N$ *stride permutation* matrix is defined for $N = KM$ by the permutation

$$L^N_M : \ iK + j \mapsto jM + i \tag{1.12}$$

for $0 \leq i < K,\ 0 \leq j < M$. This definition shows that L^N_M transposes a $K \times M$ matrix stored in row-major order. Alternatively, we can write

$$\begin{aligned} L^N_M : \qquad i &\mapsto iM \bmod N-1, \quad \text{for } 0 \leq i < N-1, \\ N-1 &\mapsto N-1. \end{aligned}$$

For example ($\cdot$ means 0),

$$L^6_2 = \begin{bmatrix} 1 & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & 1 & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & 1 & \cdot \\ \cdot & 1 & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & 1 & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & 1 \end{bmatrix}.$$

$L^N_{N/2}$ is sometimes called the perfect shuffle.

Further, we use matrix operators; namely the direct sum

$$A \oplus B = \begin{bmatrix} A & \\ & B \end{bmatrix}$$

and the Kronecker or tensor product

$$A \otimes B = [a_{k,\ell} B]_{k,\ell}, \quad \text{for } A = [a_{k,\ell}].$$

In particular,

$$I_n \otimes A = A \oplus \ldots \oplus A = \begin{bmatrix} A & & \\ & \ddots & \\ & & A \end{bmatrix}$$

is block-diagonal.

We may also construct a larger matrix as a matrix of matrices, e.g.,

$$\begin{bmatrix} A & B \\ B & A \end{bmatrix}.$$

If an algorithm for a transform is given as a product of sparse matrices built from the constructs above, then an algorithm for the transpose or inverse of the transform can be readily derived using mathematical properties including

$$\begin{aligned} (AB)^T &= B^T A^T, & (AB)^{-1} &= B^{-1} A^{-1}, \\ (A \oplus B)^T &= A^T \oplus B^T, & (A \oplus B)^{-1} &= A^{-1} \oplus B^{-1}, \\ (A \otimes B)^T &= A^T \otimes B^T, & (A \otimes B)^{-1} &= A^{-1} \otimes B^{-1}. \end{aligned} \tag{1.13}$$

Permutation matrices are orthogonal, i.e., $P^T = P^{-1}$. The transposition or inversion of diagonal matrices is obvious.

1.2.2 Radix-2 FFT

The DFT decomposes $\mathcal{A} = \mathbb{C}[s]/(s^N - 1)$ with basis $b = (1, s, \ldots, s^{N-1})$ as shown in (1.10). We assume $N = 2M$. Then

$$s^{2M} - 1 = (s^M - 1)(s^M + 1)$$

factors and we can apply the CRT in the following steps:

$$\begin{aligned} & \mathbb{C}[s]/(s^N - 1) \\ \rightarrow\ & \mathbb{C}[s]/(s^M - 1) \oplus \mathbb{C}[s]/(s^M + 1) & (1.14) \\ \rightarrow\ & \bigoplus_{0 \le i < M} \mathbb{C}[s]/(x - W_N^{2i}) \oplus \bigoplus_{0 \le i < M} \mathbb{C}[s]/(x - W_M^{2i+1}) & (1.15) \\ \rightarrow\ & \bigoplus_{0 \le i < N} \mathbb{C}[s]/(x - W_N^{i}). & (1.16) \end{aligned}$$

As bases in the smaller algebras $\mathbb{C}[s]/(s^M - 1)$ and $\mathbb{C}[s]/(s^M + 1)$, we choose $c = d = (1, s, \ldots, s^{M-1})$. The derivation of an algorithm for DFT_N based on (1.14)-(1.16) is now completely mechanical by reading off the matrix for each of the three decomposition steps. The product of these matrices is equal to the DFT_N.

First, we derive the base change matrix B corresponding to (1.14). To do so, we have to express the base elements $s^n \in b$ in the basis $c \cup d$; the coordinate vectors are the columns of B. For $0 \leq n < M$, s^n is actually contained in c and d, so the first M columns of B are

$$B = \begin{bmatrix} I_M & * \\ I_M & * \end{bmatrix},$$

where the entries $*$ are determined next. For the base elements s^{M+n}, $0 \leq n < M$, we have

$$\begin{aligned} s^{M+n} &\equiv s^n \bmod (s^M - 1), \\ s^{M+n} &\equiv -s^n \bmod (s^M + 1), \end{aligned}$$

which yields the final result

$$B = \begin{bmatrix} I_M & I_M \\ I_M & -I_M \end{bmatrix} = \mathrm{DFT}_2 \otimes I_M.$$

Next, we consider step (1.15). $\mathbb{C}[s]/(s^M - 1)$ is decomposed by DFT_M and $\mathbb{C}[s]/(s^M + 1)$ by DFT-3_M in (1.11).

Finally, the permutation in step (1.16) is the perfect shuffle L^N_M, which interleaves the even and odd spectral components (even and odd exponents of W_N).

The final algorithm obtained is

$$\mathrm{DFT}_{2M} = L^N_M(\mathrm{DFT}_M \oplus \text{DFT-3}_M)(\mathrm{DFT}_2 \otimes I_M).$$

To obtain a better known form, we use $\text{DFT-3}_M = \mathrm{DFT}_M\, D_M$, with $D_M = \mathrm{diag}_{0 \leq i < M}(W_N^i)$, which is evident from (1.11). It yields

$$\begin{aligned} \mathrm{DFT}_{2M} &= L^N_M(\mathrm{DFT}_M \oplus \mathrm{DFT}_M\, D_M)(\mathrm{DFT}_2 \otimes I_M) \\ &= L^N_M(I_2 \otimes \mathrm{DFT}_M)(I_M \oplus D_M)(\mathrm{DFT}_2 \otimes I_M). \end{aligned}$$

The last expression is the radix-2 decimation-in-frequency Cooley-Tukey FFT. The corresponding decimation-in-time version is obtained by transposition using (1.13) and the symmetry of the DFT:

$$\mathrm{DFT}_{2M} = (\mathrm{DFT}_2 \otimes I_M)(I_M \oplus D_M)(I_2 \otimes \mathrm{DFT}_M)L^N_2.$$

The entries of the diagonal matrix $I_M \oplus D_M$ are commonly called *twiddle factors*.

The above method for deriving DFT algorithms is used extensively in [9].

1.2.3 General-radix FFT

To algebraically derive the general-radix FFT, we use the *decomposition property* of $s^N - 1$. Namely, if $N = KM$ then

$$s^N - 1 = (s^M)^K - 1.$$

Decomposition means that the polynomial is written as the composition of two polynomials: here, s^M is inserted into $s^K - 1$. Note that this is a special property: most polynomials do not decompose.

Based on this polynomial decomposition, we obtain the following stepwise decomposition of $\mathbb{C}[s]/(s^N - 1)$, which is more general than the previous one in (1.14)–(1.16). The basic idea is to first decompose with respect to the outer polynomial $t^K - 1$, $t = s^M$, and then completely [15]:

$$\begin{aligned} & \mathbb{C}[s]/(s^N - 1) = \mathbb{C}[x]/((s^M)^K - 1) \\ \to \quad & \bigoplus_{0\le i<K} \mathbb{C}[s]/(s^M - W_K^i) && (1.17) \\ \to \quad & \bigoplus_{0\le i<K} \bigoplus_{0\le j<M} \mathbb{C}[s]/(x - W_N^{jK+i}) && (1.18) \\ \to \quad & \bigoplus_{0\le i<N} \mathbb{C}[s]/(x - W_N^i). && (1.19) \end{aligned}$$

As bases in the smaller algebras $\mathbb{C}[s]/(s^M - W_K^i)$ we choose $c_i = (1, s, \dots, s^{M-1})$. As before, the derivation is completely mechanical from here: only the three matrices corresponding to (1.17)–(1.19) have to be read off.

The first decomposition step requires us to compute s^n mod $(s^M - W_K^i)$, $0 \le n < N$. To do so, we decompose the index n as $n = \ell M + m$ and compute

$$s^n = s^{\ell M + m} = (s^M)^\ell s^m \equiv W_k^{\ell m} s^m \text{ mod } (s^M - W_K^i).$$

This shows that the matrix for (1.17) is given by $\mathrm{DFT}_K \otimes I_M$.

In step (1.18), each $\mathbb{C}[s]/(s^M - W_K^i)$ is completely decomposed by its polynomial transform

$$\mathrm{DFT}_M(i, K) = \mathrm{DFT}_M \cdot \mathrm{diag}_{0\le i<M}(W_N^{ij}).$$

At this point, $\mathbb{C}[s]/(s^N - 1)$ is completely decomposed, but the spectrum is ordered according to $jK + i$, $0 \le i < M$, $0 \le j < K$ (j runs faster). The desired order is $iM + j$.

Thus, in step (1.19), we need to apply the permutation $jK + i \mapsto iM + j$, which is exactly the stride permutation L_M^N in (1.12).

In summary, we obtain the Cooley-Tukey decimation-in-frequency FFT with arbitrary radix:

$$\begin{aligned} & L_M^N \left(\bigoplus_{0\le i<K} \mathrm{DFT}_M \cdot \mathrm{diag}_{j=0}^{M-1}(W_N^{ij}) \right) (\mathrm{DFT}_k \otimes I_M) \\ = \quad & L_M^N (I_K \otimes \mathrm{DFT}_M) T_M^N (\mathrm{DFT}_k \otimes I_M). && (1.20) \end{aligned}$$

The matrix T_M^N is diagonal and usually called the *twiddle matrix*. Transposition using (1.13) yields the corresponding decimation-in-time version:

$$(\mathrm{DFT}_k \otimes I_M) T_M^N (I_K \otimes \mathrm{DFT}_M) L_K^N.$$

1.3 Discussion and Further Reading

This chapter only scratches the surface of the connection between algebra and the DFT or signal processing in general. We provide a few references for further reading.

1.3.1 Algebraic Derivation of Transform Algorithms

As mentioned before, the use of polynomial algebras and the CRT underlies much of the early work on FFTs and convolution algorithms [20, 9, 1]. For example, Winograd's work on FFTs minimizes the number of non-rational multiplications. This and his work on complexity theory in general makes heavy use of polynomial algebras [20, 21, 22] (see Chapter **??** for more information and references). See [4] for a broad treatment of algebraic complexity theory.

Since $\mathbb{C}[x]/(s^N - 1) = \mathbb{C}[C_N]$ can be viewed a group algebra for the cyclic group, the methods shown in this chapter can be translated into the context of group representation theory. For example, [8] derives the general-radix FFT using group theory and also uses already the Kronecker product formalism. So does Beth and started the area of FFTs for more general groups [2, 7]. However, Fourier transforms for groups have found only sporadic applications [16]. Along a related line of work, [5] shows that using group theory it is possible that to discover and generate certain algorithms for trigonometric transforms, such as discrete cosine transforms (DCTs), automatically using a computer program.

More recently, the polynomial algebra framework was extended to include most trigonometric transforms used in signal processing [12, 14]. It turns out that the same techniques shown in this chapter can then be applied to derive, explain, and classify most of the known algorithms for these transforms and even obtain a large class of new algorithms including general-radix algorithms for the discrete cosine and sine transforms (DCTs/DSTs) [15, 13, 19].

This latter line of work is part of the algebraic signal processing theory briefly discussed next.

1.3.2 Algebraic Signal Processing Theory

The algebraic properties of transforms used in the above work on algorithm derivation hints at a connection between algebra and (linear) signal processing itself. This is indeed the case and was fully developed in a recent body of work called algebraic signal processing theory (ASP) [14, 12, 11].

ASP first identifies the algebraic structure of (linear) signal processing: the common assumptions on available operations for filters and signals make the set of filters an *algebra* $\mathcal{A}$ and the set of signals an associated $\mathcal{A}$*-module* $\mathcal{M}$. ASP then builds a signal processing theory formally from the axiomatic definition of a *signal model*: a triple $(\mathcal{A}, \mathcal{M}, \Phi)$, where Φ generalizes the idea of the z-transform to mappings from vector spaces of signal values to $\mathcal{M}$. If a signal model is given, other concepts, such as spectrum, Fourier transform, frequency response are automatically defined but take different forms for different models. For example, infinite and finite time as discussed

Signal model	Infinite time	Finite time
$\mathcal{A}$	$\left\{\sum_{n\in\mathbb{Z}} H(n)s^n \mid (\dots,H(-1),H(0),H(1),\dots)\in\ell^1(\mathbb{Z})\right\}$	$\mathbb{C}[x]/(s^n-1)$
$\mathcal{M}$	$\left\{\sum_{n\in\mathbb{Z}} X(n)s^n \mid (\dots,X(-1),X(0),X(1),\dots)\in\ell^2(\mathbb{Z})\right\}$	$\mathbb{C}[s]/(s^n-1)$
Φ	$\Phi:\ \ell^2(\mathbb{Z})\to\mathcal{M}$ defined in (1.1)	$\Phi:\ \mathbb{C}^n\to\mathcal{M}$ defined in (1.4)

Table 1.2: Infinite and finite time models as defined in ASP. Φ is the z-transform and finite z-transform, respectively.

in Table 1.1 are two examples of signal models. Their complete definition is provided in Table 1.2 and identifies the proper notion of a finite z-transform as a mapping $\mathbb{C}^n \to \mathbb{C}[s]/(s^n-1)$.

ASP shows that many signal models are in principle possible, each with its own notion of filtering and Fourier transform. Those that support shift-invariance have commutative algebras. Since finite-dimensional commutative algebras are precisely polynomial algebras, their appearance in signal processing is explained. For example, ASP identifies the polynomial algebras underlying the DCTs and DSTs, which hence become Fourier transforms in the ASP sense. The signal models are called finite *space* models since they support signal processing based on an undirected shift operator, different from the directed time shift. Many more insights are provided by ASP including the need for and choices in choosing boundary conditions, properties of transforms, techniques for deriving new signal models, and the concise derivation of algorithms mentioned before.

Bibliography

[1] L. Auslander, E. Feig, and S. Winograd. Abelian semi-simple algebras and algorithms for the discrete Fourier transform. *Advances in Applied Mathematics*, 5:31–55, 1984.

[2] Th. Beth. *Verfahren der Schnellen Fouriertransformation [Fast Fourier Transform Methods]*. Teubner, 1984.

[3] V. Britanak and K. R. Rao. The fast generalized discrete Fourier transforms: A unified approach to the discrete sinusoidal transforms computation. *Signal Processing*, 79:135–150, 1999.

[4] P. Bürgisser, M. Clausen, and M. A. Shokrollahi. *Algebraic Complexity Theory*. Springer, 1997.

[5] S. Egner and M. Püschel. Automatic generation of fast discrete signal transforms. *IEEE Trans. on Signal Processing*, 49(9):1992–2002, 2001.

[6] Paul A. Fuhrman. *A Polynomial Approach to Linear Algebra*. Springer Verlag, New York, 1996.

[7] D. Maslen and D. Rockmore. Generalized FFTs – a survey of some recent results. In *Proceedings of IMACS Workshop in Groups and Computation*, volume 28, pages 182–238, 1995.

[8] P. J. Nicholson. Algebraic theory of finite Fourier transforms. *Journal of Computer and System Sciences*, 5:524–547, 1971.

[9] H. J. Nussbaumer. *Fast Fourier Transformation and Convolution Algorithms*. Springer, 2nd edition, 1982.

[10] A. V. Oppenheim, R. W. Schafer, and J. R. Buck. *Discrete-Time Signal Processing*. Prentice Hall, 2nd edition, 1999.

[11] M. Püschel and J. M. F. Moura. Algebraic signal processing theory. available at http://arxiv.org/abs/cs.IT/0612077, parts of this manuscript are submitted as [14] and [12].

[12] M. Püschel and J. M. F. Moura. Algebraic signal processing theory: 1-D space. submitted for publication, part of [11].

[13] M. Püschel and J. M. F. Moura. Algebraic signal processing theory: Cooley-Tukey type algorithms for DCTs and DSTs. *IEEE Transactions on Signal Processing*. to appear; a longer version is available at http://arxiv.org/abs/cs.IT/0702025.

[14] M. Püschel and J. M. F. Moura. Algebraic signal processing theory: Foundation and 1-D time. submitted for publication, part of [11].

[15] M. Püschel and J. M. F. Moura. The algebraic approach to the discrete cosine and sine transforms and their fast algorithms. *SIAM Journal of Computing*, 32(5):1280–1316, 2003.

[16] D. Rockmore. Some applications of generalized FFT's. In *Proceedings of DIMACS Workshop in Groups and Computation*, volume 28, pages 329–370, 1995.

[17] R. Tolimieri, M. An, and C. Lu. *Algorithms for Discrete Fourier Transforms and Convolution*. Springer, 2nd edition, 1997.

[18] C. Van Loan. *Computational Framework of the Fast Fourier Transform*. Siam, 1992.

[19] Y. Voronenko and M. Püschel. Algebraic derivation of general radix Cooley-Tukey algorithms for the real discrete Fourier transform. In *Proc. International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2006.

[20] S. Winograd. On computing the discrete Fourier transform. *Mathematics of Computation*, 32:175–199, 1978.

[21] S. Winograd. On the multiplicative complexity of the discrete Fourier transform. *Advances in Mathematics*, 32:83–117, 1979.

[22] S. Winograd. *Arithmetic Complexity of Computation*. Siam, 1980.