Review: Fast Fourier Methods in Computational Complex Analysis

Jason Waataja

June 10, 2019

Abstract

The discrete Fourier transform is an operation with uses in almost every area of science and it comes up in countless branches of mathematics. In particular, it has many applications in computational complex analysis. The fast Fourier transform algorithm is ubiquitous in its computation. Peter Henrici covers many of these applications in his article "Fast Fourier Methods in Computational Complex Analysis." This paper will cover the discrete Fourier transform, the fast Fourier transform algorithm, and a selection of applications from Henrici's paper, as well as extend what he did in the multi-dimensional case.

Contents

1 Introduction

In “Fast Fourier Methods in Computational Analysis,” Peter Henrici outlines the discrete Fourier Transform, a fast algorithm for its implementation called the *fast Fourier transform* and explains many of its applications in complex analysis and related fields [3]. When the article was written, Fourier analysis was already one of the most pervasive tools in applied mathematics, and its prominence has only grown. In particular, Fourier analysis is invaluable in phenomena that are periodic in time or space and anywhere circular geometry arises. The article serves as a survey of discrete Fourier analysis, informing its audience of the basics of its theory and more importantly its applications.

Henrici begins by introducing the *discrete Fourier operator* which is a transform on bilateral infinite sequences. Several basic facts are established such as its linearity and invertibility. From its definition, computing the transform is computationally expensive, leading to the introduction of the *fast Fourier transform* family of algorithms. Henrici also describes the multi-dimensional case. The bulk of the paper then summarizes applications of the discrete Fourier transform and instances where a computationally efficient method to compute it proves useful. Highlights include approximating the coefficients of Fourier and Laurent series, convolutions and their applications, and algorithms for manipulation of formal power series.

In this paper I will give the definition of the transform and the fast Fourier transform algorithm. I will then go over the applications relevant to an introductory course on complex analysis and finally I will explain the multi-dimensional transform, expanding upon what Henrici did and filling in missing details.

2 The Transform

2.1 Infinite Sequences

The space of the discrete Fourier transform is that of *periodic bilateral infinite sequences.*

Definition 2.1. We denote Π_n to be space of sequences

$$\mathbf{x} = \{x_k\}_{k=-\infty}^{\infty}$$

where each x_k is a complex number and where the sequence satisfies

$$x_{k+n} = x_k.$$

That is, the sequence is periodic in n. This becomes a vector space when we define addition and scalar multiplication as

$$(\mathbf{x}+\mathbf{y})_k = x_k + y_k$$
$$(c\mathbf{x})_k = cx_k.$$

and we define the zero element to be $\mathbf{0}$ to be 0 in every term. This space admits an n-dimensional basis where for $m = 0, 1, \ldots, n-1$ we define $\mathbf{e}^{(m)} = \{e_k^m\}$ where

$$e_k^m = \begin{cases} 1, & k \equiv m \pmod{n}, \\ 0, & k \not\equiv m \pmod{n}. \end{cases}$$

2.2 Trigonometric Series

In his paper, Henrici begins by defining the transform and then its inverse. I will take the opposite approach and develop the theory similarly to how it may have been done in a first introduction to Fourier series. For a function f we start by assuming f has a representation as a trigonometric series and then determine the formula for the coefficients, which will be the transform. I assume the reader is familiar with Fourier series, which are periodic functions with a representation of the form

$$f(\theta) = \sum_{-\infty}^{\infty} c_n e^{in\theta}$$

where the c_n are complex constant coefficients. Understanding of Fourier series will assist in understanding the discrete Fourier transform. It will also turn out that the discrete Fourier transform can approximate these coefficients.

We now attempt to do this with discrete sequences. To begin, we define the *principle root of unity.*

Definition 2.2. The *principle roof of unity* w_n is defined as

$$w_n = e^{2\pi i/n}.$$

For $0 \leq k \leq n-1$ the numbers w_n^k are the nth roots of unity. We recall three facts from complex analysis,

$$\begin{aligned} w_n^n &= 1 \\ w_n^{k+n} &= w_n^k \\ \sum_{k=0}^{n-1} w_n^{kr} &= \begin{cases} n & \text{if } r \equiv 0 \pmod n \\ 0 & \text{if } r \not\equiv 0 \pmod n. \end{cases} \end{aligned} \tag{2.1}$$

The second of these follows from the first and the third follows from the formula for the sum of a finite geometric series.

Given $\mathbf{x} \in \Pi_n$, suppose we wish to represent $\mathbf{x}$ as a sum of such terms. That is, we want to represent x_r by a sum of terms of the form $a_k w_n^{rk}$ where k ranges over some set of integers. Such a sequence would be determined by the coefficients a_k. Since $\mathbf{x}$ is completely determined by n pieces of information, say x_0 through x_{n-1}, it is reasonable to assume we could form this representation using n numbers without any loss of information, say $k = 0$ through $k = n-1$. Indeed, by (2.1), we have $w_n^{r(k+n)} = w_n^{rk}$ so all terms except aside from $k = 0$ through $k = n-1$ would be redundant.

Such a series would take the form

$$x_r = \sum_{k=0}^{n-1} a_k w_n^{rk}. \tag{2.2}$$

In view of $w_n^{(r+n)(k)} = w_n^{rk}$, which also follows from (2.1), we see that this is indeed periodic in n. Since there are n such coefficients a_k, we can represent them as a sequence $\mathbf{y} \in \Pi_n$. The question arises of how we compute the terms in $\mathbf{y}$. Suppose $\mathbf{x}$ has such a representation. Say we wish to compute the mth term. Similarly to how in the case of regular Fourier series we integrated our function multiplied by $e^{-in\theta}$, here we will sum with multiplication by w_n^{-kr}. This gives,

$$\begin{aligned}\sum_{j=0}^{n-1} w_n^{-rj} x_j &= \sum_{j=0}^{n-1} w_n^{-rj} \sum_{k=0}^{n-1} y_k w_n^{jk} \\ &= \sum_{j=0}^{n-1} \sum_{k=0}^{n-1} y_k w_n^{(k-r)j}.\end{aligned}$$

Interchanging the order of summations we have,

$$\begin{aligned}&\sum_{k=0}^{n-1} \sum_{j=0}^{n-1} y_k w_n^{(k-r)j} \\ &= \sum_{k=0}^{n-1} y_k \sum_{j=0}^{n-1} w_n^{(k-r)j} \\ &= n y_r.\end{aligned}$$

In the last step we have used (2.1) which says the inner sum is n if $k - r \equiv 0 \pmod{n}$ and 0 otherwise. As k ranges from 0 to $n-1$, this will happen exactly once when $k = r$ and so all terms will be filtered except the desired case where $r \equiv k \pmod{n}$. Correcting for the factor of n we come to the what Henrici calls the *discrete Fourier operator*.

Definition 2.3. We define the *discrete Fourier operator* $\mathscr{F}_n$ on a sequence $\mathbf{x} \in \Pi_n$ as $\mathbf{y} = \mathscr{F}_n \mathbf{x}$ where

$$y_m = \frac{1}{n} \sum_{k=0}^{n-1} w_n^{-mk} x_k. \tag{2.3}$$

This operator defines a map from Π_n to Π_n as we can see from (2.1) that (2.3) satisfies $y_{m+n} = y_m$. One can also easily verify the map is *linear* which follows from the formula itself.

2.3 Existence of the Transform and its Inverse

When does a sequence $\mathbf{x}$ admit a representation of the form (2.2)? In

$$y_m = \frac{1}{n} \sum_{k=0}^{n-1} w_n^{-mk} x_k$$

the coefficients w_n^{-mk} are constants and we see that $\mathscr{F}_n$ is actually equivalent to multiplication by the matrix

$$\frac{1}{n} \begin{pmatrix} (w_n^{-0})^0 & (w_n^{-0})^1 & \cdots & (w_n^{-0})^{n-1} \\ (w_n^{-1})^0 & (w_n^{-1})^1 & \cdots & (w_n^{-1})^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ (w_n^{-(n-1)})^0 & (w_n^{-(n-1)})^1 & \cdots & (w_n^{-(n-1)})^{n-1} \end{pmatrix}$$

which is the Vandermonde matrix for $w_n^0, w_n^{-1}, \ldots, w_n^{-(n-1)}$. These are actually the nth roots of unity and hence distinct. It is known from linear algebra that if the inputs to the Vandermonde matrix are distinct that the determinant of the Vandermonde matrix is non-zero, hence invertible. See, for example, page 2 of [5]. This shows the linear transformation given by $\mathscr{F}_n$ is a bijective map from Π_n to Π_n. The important result is that the inverse operator $\mathscr{F}_n^{-1}$ is well defined and that every $\mathbf{y} \in \Pi_n$ has a corresponding $\mathbf{x}$ such that $\mathscr{F}_n\mathbf{x} = \mathbf{y}$.

What is this inverse? We have already seen it in (2.2).

Definition 2.4. If we are given a sequence $\mathbf{y} \in \Pi_n$ then we define the *inverse Fourier operator* $\mathscr{F}_n^{-1}$ as $\mathbf{x} = \mathscr{F}_n\mathbf{y}$ where

$$x_m = \sum_{k=0}^{n-1} a_k w_n^{mk}. \tag{2.4}$$

We already showed in Section 2.2 that in the case a sequence $\mathbf{x}$ has a representation of the form (2.4) then $\mathscr{F}_n\mathscr{F}_n^{-1}\mathbf{x} = \mathbf{x}$ and so $\mathscr{F}_n$ composed with $\mathscr{F}_n^{-1}$ is the identity map. This shows that each indeed gives the inverse of the other. However, we needed their bijectivity to know that every $\mathbf{x}$ did have a representation of this form.

2.4 Implementation and the Fast Fourier Transform

Suppose we wish to compute $\mathscr{F}_n\mathbf{x}$. How fast can we implement this operation. A naïve implementation based on (2.3) would require $n-1$ multiplications if we assume all powers of w_n are precomputed and we exclude the trivial case where the exponent of w_n is 0. To completely determine $\mathscr{F}_n\mathbf{x}$ requires knowing a full period of n coefficients, so the full calculations takes $n(n-1)$ total multiplications. A similar bound would hold for the number of additions and so the time it would take to compute $\mathscr{F}_n$ using this algorithm would be $O(n^2)$. This would be infeasible for very large n and so a faster algorithm is required to make the discrete Fourier operator computationally efficient.

Enter the *fast Fourier transform.* It is the existence of this algorithm that makes the applications in Henrici's article feasible. It is an instance of the *divide and conquer* family of algorithms where we break the calculation into subproblems and combine the result. Consider the case where $n = pq$ for integers p and q. For $0 \le j \le p-1$ we can form the subsequence $\mathbf{x}^{(j)}$ given by taking every pth element of $\mathbf{x}$ starting at j of which there will be q unique values. That is, $\mathbf{x}^{(j)} \in \Pi_q$ and

$$\mathbf{x}^{(j)} = \{x_k^{(j)}\} = \{x_{j+pk}\}.$$

We can also think of this as selecting the subsequence $\{x_k\}$ where $k \equiv j \pmod p$. The crux of algorithm is breaking the computation of $\mathscr{F}_n\mathbf{x}$ into p subproblems of size q where we compute the p values of $\mathscr{F}_q\mathbf{x}^{(j)}$ with $j = 0$ through $j = p-1$ and then combine them for each x_n with with one summation of size p.

Indeed, we can rewrite the summation formula for y_m as p groups of q multiplications as follows

$$\begin{aligned} y_m &= \frac{1}{n}\sum_{k=0}^{n-1} w_n^{-mk}x_k \\ &= \frac{1}{p}\sum_{j=0}^{p-1}\frac{1}{q}\sum_{h=0}^{q-1} w_n^{-m(j+ph)}x_{j+ph} \\ &= \frac{1}{p}\sum_{j=0}^{p-1}\frac{1}{q}\sum_{h=0}^{q-1} w_n^{-mj}w_n^{-pmh}x_{j+ph} \\ &= \frac{1}{p}\sum_{j=0}^{p-1} w_n^{-mj}\frac{1}{q}\sum_{h=0}^{q-1} w_n^{-pmh}x_{j+ph}. \end{aligned}$$

We now note that

$$w_n^p = e^{2\pi i p/n} = e^{2\pi i/q} = w_q$$

which gives

$$\frac{1}{p}\sum_{j=0}^{p-1} w_n^{-mj}\frac{1}{q}\sum_{h=0}^{q-1} w_q^{-mh}x_{j+ph}.$$

But, the inner sum

$$\frac{1}{q}\sum_{h=0}^{q-1} w_q^{-mh}x_{j+ph}$$

is exactly $\mathscr{F}_q\mathbf{x}^{(j)}$ which is $y_m^{(j)}$. Thus, what we've shown is

$$y_m = \frac{1}{p}\sum_{j=0}^{p-1} w_n^{-mj}y_m^{(j)}. \tag{2.5}$$

The property of this relation that allows us to compute the transform quickly is that the same sequences $\mathbf{y}^{(j)}$ come up in the computation of each y_m without having to be recalculated. So, we can recursively compute the transform of each $y^{(j)}$ and use those to compute each y_m. This completes the specification of the algorithm.

To analyze the running time, suppose n can be factorized as $n = n_1n_2\cdots n_k$ where we do not require that the n_j are prime. If we let $p = n_1$ and $q = n_2\cdots n_k$. Assuming that the n_1 transforms of size $n_2\cdots n_k$ have already been computed, for each y_m we do $n(n_1-1)$ multiplications. To compute the n_1 transforms at the next level down, for each we must do $\frac{n}{n_1}(n_2-1)$ multiplications n_1 times yielding $n_1\frac{n}{n_1}(n_2-1) = n(n_2-1)$ multiplications. We can repeat this, finding the number of multiplication required at the jth level of recursion where there are $n_1n_2\cdots n_j$ transforms and in each we must do $\frac{n}{n_1n_2\cdots n_j}(n_j-1)$ multiplications for a total of $n_1n_2\cdots n_j\frac{n}{n_1n_2\cdots n_j}(n_j-1) = n(n_j-1)$. Thus, the overall work done by the whole algorithm is

$$n(n_1-1)+n(n_2-1)+\cdots+n(n_k-1) = n\sum_{j=1}^{k}(n_j-1). \tag{2.6}$$

Consider the special case where $n = 2^k$ and there are simply k factors where $n_1 = n_2 = \cdots = n_k = 2$. In this case we would require

$$n \sum_{j-1}^{k} 1 = nk$$

multiplications. Since $k = \log_2 n$ we see that the overall work done in this case is $O(n \log n)$. This represents a huge improvement from the previous $O(n^2)$ algorithm. The speed at which we will be able to carry out this algorithm will be vital to everything that comes later.

3 Approximating Fourier Coefficients and Applications

In the next few sections I will summarize particular results that Henrici develops with a focus on applications to the complex analysis we have seen in class.

3.1 Fourier Coefficients

Suppose we wish to calculate the Fourier coefficients of a function f. Also, to make the connection with the discrete Fourier transform easier, suppose instead of working with 2π periodic functions we are working with functions periodic in 1 where we compute the coefficients $\{a_m\}$ with the formula

$$a_m = \int_0^1 f(t) e^{-2\pi i m t} \, dt. \tag{3.1}$$

We denote the space of 1-periodic functions as Π. In calculating (3.1) the above integral may be hard to evaluate in closed form. Also, it will not always be that we are working a function f specified by a formula. It may be that we are given n data points sampled at equal times t_k given by the sequence $x_k = f(t_k)$. If the sampling points are given by

$$t_k = \frac{k}{n}$$

by the periodicity of f we can extend $\{x_k\}$ infinitely with period n to get $\mathbf{x} \in \Pi_n$ where $\mathbf{x} = \{x_k\}_{k=-\infty}^{\infty}$.

It turns out that the discrete Fourier transform aligns with the Riemann sum for (3.1) with equally spaced intervals where we take the value of f at the left hand endpoint of each interval. That is, we let

$$\hat{a}_m = \frac{1}{n} \sum_{k=0}^{n-1} x_k w_n^{-mk} \tag{3.2}$$

and then declare

$$\hat{\mathbf{a}} = \{\hat{a}_m\} = \mathscr{F}_n \mathbf{x}.$$

There are formulas available to estimate the error of this summation interpreted as a general Riemann sum, but in the case of Fourier series we get a much better result.

Indeed, suppose the Fourier series for $x = f(t)$ is absolutely convergent, then we can write

$$x_k = f(t_k) = \sum_{m=\infty}^{\infty} a_m e^{2\pi i t_k} = \sum_{m=\infty}^{\infty} a_m e^{2\pi i k/n} = \sum_{m=\infty}^{\infty} a_m w_n^{km}.$$

Substituting this into (3.2) we see

$$\begin{aligned}
\hat{a}_m &= \frac{1}{n}\sum_{k=0}^{n-1} w^{-mk} x_k \\
&= \frac{1}{n}\sum_{k=0}^{n-1} w^{-mk} \sum_{h=\infty}^{\infty} a_m w_n^{kh} \\
&= \frac{1}{n}\sum_{k=0}^{n-1} \sum_{h=\infty}^{\infty} a_m w_n^{k(h-m)}.
\end{aligned}$$

Because this is a sum of convergent series we may interchange the order of summations which gives,

$$\begin{aligned}
&\frac{1}{n}\sum_{h=\infty}^{\infty} \sum_{k=0}^{n-1} a_m w_n^{k(h-m)} \\
&= \frac{1}{n}\sum_{h=\infty}^{\infty} a_m \sum_{k=0}^{n-1} w_n^{k(h-m)}.
\end{aligned}$$

By (2.1) the inner sum vanishes unless $h \equiv m \pmod{n}$ and thus all terms except these are filtered out. For those that don't vanish we have the inner sum is n and so the factors of n will cancel out. This gives the important estimate

$$\hat{a}_m - a_m = \sum_{k\neq 0} a_{m+kn} \tag{3.3}$$

We can go further with this. We define

$$g(z) = f\left(\frac{1}{2\pi i}\operatorname{Log} z\right).$$

If f is analytic on the strip $-\eta \leq \operatorname{Im} z \leq \eta$ then g is analytic on the annulus $e^{-2\pi\eta} \leq |z| \leq e^{2\pi\eta}$. Consider the Laurent coefficients $\{a_m\}$ of g computed with unit circle. These are

$$\begin{aligned}
a_m &= \frac{1}{2\pi i}\int_{|z|=1} \frac{f\left(\frac{1}{2\pi i}\operatorname{Log} z\right)}{z^{m+1}}\,dz \\
&= \frac{1}{2\pi i}\int_0^{2\pi} \frac{f\left(\frac{1}{2\pi i}\operatorname{Log} e^{i\theta}\right)}{e^{i\theta(m+1)}} ie^{i\theta}\,d\theta \\
&= \frac{1}{2\pi}\int_0^{2\pi} f(\theta/2\pi)e^{-im\theta}\,d\theta \\
&= \int_0^1 f(u)e^{-2\pi imu}\,du
\end{aligned}$$

where in the last line we have made the substitution $u = \frac{\theta}{2\pi}$. This shows the Fourier coefficients of f are exactly the Laurent coefficients of g. Since g is analytic on the annulus, we can use the Cauchy estimates to get

$$|a_m| \leq Me^{-2\pi\eta|m|}$$

where M is the maximum modulus of g on the annulus.

If we let $|m| < n$ and use this estimate with (3.3) we get

$$\begin{aligned}
|\hat{a}_m - a_m| &= \left| \sum_{k \neq 0} a_{m+kn} \right| \\
&\leq \sum_{k=1}^{\infty} M e^{-2\pi\eta|m+kn|} + \sum_{k=1}^{\infty} M e^{-2\pi\eta|m-kn|} \\
&= \sum_{k=1}^{\infty} M e^{-2\pi\eta(m+kn)} + \sum_{k=1}^{\infty} M e^{-2\pi\eta(kn-m)} \\
&= M e^{-2\pi\eta m} \sum_{k=1}^{\infty} e^{-2\pi\eta kn} + M e^{2\pi\eta m} \sum_{k=1}^{\infty} e^{-2\pi\eta kn}.
\end{aligned}$$

Because $e^{-2\pi\eta n} < 1$ we may use the formula for the sum of a geometric series which gives,

$$\begin{aligned}
&M\left(e^{-2\pi\eta m} + e^{2\pi\eta m}\right) \frac{e^{-2\pi\eta n}}{1 - e^{-2\pi\eta n}} \\
&= 2M \cosh(2\pi m\eta) \frac{e^{-2\pi\eta n}}{1 - e^{-2\pi\eta n}}
\end{aligned}$$

We arrive at our convergence theorem.

Theorem 3.1. *If $x = f(z)$ is analytic for $|\operatorname{Im} z| \leq \eta$, then in approximating a_n with $\hat{a}_m$ defined as in* (3.2)*, the error does not exceed*

$$|\hat{a}_m - a_m| \leq 2M \cosh(2\pi m\eta) \frac{e^{-2\pi\eta n}}{1 - e^{-2\pi\eta n}}. \tag{3.4}$$

As a consequence, the error $\hat{a}_m - a_m$ tends to 0 like $e^{-2\pi\eta n}$ as $n \to \infty$ which is geometric decay.

3.2 Trigonometric Interpolation

As one might want to interpolate a function with a polynomial, so too might we wish to interpolate a periodic function with a trigonometric polynomial of the form

$$p(t) = \sum_{m=-n/2}^{-n/2} a_m e^{2\pi i m t}.$$

Suppose $f \in \Pi$ and we are given n equally spaced data points $x_k = f(t_k)$. The obvious strategy would be to use f's Fourier series and to truncate the terms. The fast Fourier transform would produce the coefficients. This seems as if it would be a crude approximation, but it turns out that by construction the approximation matches exactly with f at the sampling points. Let

$$\hat{a}_m = \frac{1}{n} \sum_{k=0}^{n-1} x_k w_n^{-km}.$$

and if n is even then we set

$$p(t) = \sum_{m=-n/2}^{-n/2}{}' \hat{a}_m e^{2\pi imt}.$$

Note the prime on the summation which indicates that for $m = \pm\frac{n}{2}$ we multiply the term by $\frac{1}{2}$, analogous to how we take $\frac{a_0}{2}$ in regular Fourier series. We will show the following.

Theorem 3.2. *The trigonometric polynomial approximating a function using the coefficients from the discrete Fourier transform is exactly exactly that function at the sampling points. That is,*

$$p(t_k) = x_k$$

for all k.

Indeed, at t_k we have $e^{2\pi imk/n} = w_n^{mk}$ so

$$p(t_k) = \sum_{m=-n/2}^{-n/2}{}' \hat{a}_m w_n^{km}.$$

This has $n+1$ terms. By (2.1), the expression we are summing is periodic in n. Thus, we can combine the first and last terms where m differs by n and since each was multiplied by $\frac{1}{2}$ it will be as if they were a single term. Secondly, periodicity in n allows us to shift indices right by $\frac{n}{2}$ and recollect the terms to get

$$p(t_k) = \sum_{m=0}^{n} \hat{a}_m w_n^{km}$$

In view of (2.4), this is exactly x_k, completing the proof.

Basically, it turned out that the process we would use to find the best possible trigonometric interpolation is exactly the same as what we did to derive $\mathscr{F}_n$. We actually constructed $\mathscr{F}_n$ as a way to generate a trigonometric polynomial with a certain set of values, and so it turns out that this is the same problem as trigonometric interpolation.

Henrici gives a lower a bound on the error for points other than t_k. It turns out that

$$|p(t) - f(t)| \leq |a_{n/2}| + |a_{-n/2}| + 2 \sum_{|k|>n/2} |a_k| \tag{3.5}$$

where $\{a_k\}$ are the regular Fourier coefficients of f. For his proof, see page 496 of [3]. This will hold if the Fourier series converges absolutely and show that the error approaches 0 as $n \to \infty$.

3.3 Computation of Harmonic Conjugates

Fourier series also aid in understanding harmonic conjugates. Suppose we are given a harmonic function u on the unit disk that extends continuously to its boundary. Henrici gives a method for approximating u's harmonic conjugate v.

Consider u on the boundary so that $u(e^{i\theta})$ is a 2π-periodic function of θ. Suppose it's given by an absolutely convergent Fourier series

$$u(\theta) = \sum_{m=-\infty}^{\infty} a_m e^{im\theta}.$$

It's easy to verify that if u is real then $a_{-m} = \overline{a_m}$ so that this can be rewritten as

$$u(\theta)a_0 + 2\sum_{m=1}^{\infty}(a_m + \overline{a_m})e^{im\theta}.$$

Consider the function

$$f(z) = a_0 + 2\sum_{m=1}^{\infty} a_m z^m.$$

In view of the absolute convergence of the Fourier series for u on the boundary, f will converge uniformly and absolutely on the whole closure of the unit disk by the Weierstrass M-test, and moreover it will be to an analytic function. Taking the real part of f given by $\frac{1}{2}(f + \overline{f})$ we get exactly our formula for $u(\theta)$. Since the real part of f is harmonic and agrees with u on the boundary, it will agree on the interior as well. We also get a formula for the series of u's harmonic conjugate v, given by

$$\operatorname{Im} f = \frac{1}{i}(f - \overline{f}) = \frac{2}{i}\sum_{m=1}^{\infty}(a_m - \overline{a_m})z^m = \sum_{m=1}^{\infty}(-ia_m + ia_{-m})z^n.$$

Thus,

$$v(\theta) = \sum_{m=-\infty}^{\infty} b_m e^{im\theta}$$

where

$$b_m = \begin{cases} -ia_m & m > 0 \\ 0 & m = 0 \\ ia_m & m < 0. \end{cases} \tag{3.6}$$

We can use the fast Fourier transform for approximating these coefficients. First, sample u at n equally spaced intervals and compute $\hat{a}_m$ exactly as in the previous section. Use (3.6) to calculate $\hat{b}_m$ and approximate v with

$$\hat{v}(\theta) = \sum_{|m|\le n/2}{}' \hat{b}_m e^{im\theta}.$$

Formula (3.5) gives a bound on the error between $\hat{v}$ and v.

3.4 Computation of Laurent Coefficients

Similarly, we can use the fast Fourier transform to estimate Laurent coefficients. Consider a function f with a Laurent series that converges on an annulus $A = \{\rho_1 < z < \rho_2\}$ where $\rho_1 < 1 < \rho_2$. We have

$$f(z) = \sum_{m=-\infty}^{\infty} a_m z^m.$$

The coefficients are given by

$$a_m = \frac{1}{2\pi i}\int_{|z|=1} z^{-m-1} f(z)\, dz.$$

Parameterizing this with $z = e^{i\theta}$ and in view of $dz = iz\,d\theta$ we get

$$a_m = \frac{1}{2\pi i}\int_0^{2\pi} z^m f(z) i\,d\theta = \frac{1}{2\pi}\int_0^{2\pi} e^{im\theta} f(e^{i\theta})\,d\theta.$$

This is exactly the mth Fourier coefficients of the function obtained by evaluating f along the unit circle, which will be 2π periodic in the angle.

To approximate these, evaluate f at n equally spaced points around the unit circle to get a sequence $\mathbf{f} \in \Pi_n$. Compute $\hat{a}_n = \mathscr{F}_n\mathbf{f}$. The inequality (3.4) gives an estimate on the error between $\hat{a_m}$ and a_m.

3.5 Numeric Differentiation

Suppose we are given a function f analytic on a disk $\{|z - z_0| < \sigma\}$. Choose any $0 < \rho < \delta$ and define $g(z) = f(z_0 + \rho z)$. Note, that g is analytic on a disk containing the unit circle and we also have $f^{(m)}(z_0) = \frac{1}{\rho^m} g^{(m)}(0)$. The function g is given by

$$g(z) = \sum_0^\infty a_m z^m.$$

We showed in the previous section that $\{a_m\}$ can be approximated with the fast Fourier transform. Moreover, since

$$g^{(m)}(0) = m! a_m$$

we have

$$f^{(m)}(z_0) = \frac{m!}{\rho^m} a_m.$$

Estimating $f^{(m)}(z_0) \approx \frac{m!}{\rho^m}\hat{a}_m$ we once again have by (3.4) that for fixed m that the error goes to 0 geometrically as $n \to \infty$. Using the fast Fourier transform to compute $\hat{a}_m$ gives an efficient method of numeric differentiation.

4 Multiplication of Polynomials and Power Series

A wide swath of the applications in Henrici's paper make use of the *convolution* operation. In particular, algorithms for formal multiplications are made much faster by use of the fast Fourier transform.

4.1 The Conjugate, Reversion, and Convolution Operators

We now introduce several new operators.

Definition 4.1. The *conjugate discrete Fourier operator* that takes Π_n to Π_n is defined as

$$(\overline{\mathscr{F}}_n\mathbf{x})_m = \frac{1}{n}\sum_{k=0}^{n-1} w_n^{mk} x_k. \tag{4.1}$$

Compare this with (2.3), the definition of $\mathscr{F}_n$. The formula is the same except w_n^{-mk} is replaced by w_n^{mk}. Also compare this with the definition of $\mathscr{F}_n^{-1}$ given by (2.4). We see that

$$\mathscr{F}_n^{-1} = n\overline{\mathscr{F}}_n. \tag{4.2}$$

Definition 4.2. We define the *reversion* operator on $\mathbf{x} \in \Pi_n$ as

$$(R\mathscr{F}_n\mathbf{x})_m = x_{-m}. \tag{4.3}$$

We have

$$(R\mathscr{F}_n\mathbf{x})_m = \frac{1}{n}\sum_{k=0}^{n-1} w_n^{mk} x_k = (\overline{\mathscr{F}}_n\mathbf{x})_m$$

and

$$\begin{aligned}(\mathscr{F}_n R\mathbf{x})_m &= \frac{1}{n}\sum_{k=0}^{n-1} w_n^{-mk} x_{-k} \\ &= \frac{1}{n}\sum_{k=0}^{n-1} w_n^{mk} x_k \\ &= (\overline{\mathscr{F}}_n\mathbf{x})_m.\end{aligned}$$

Combining these we get

$$\mathscr{F}_n R = R\mathscr{F}_n = \overline{\mathscr{F}}_n.$$

To summarize, we have

Theorem 4.1.

$$\mathscr{F}_n^{-1} = n\overline{\mathscr{F}}_n = n\mathscr{F}_n R = nR\mathscr{F}_n. \tag{4.4}$$

Note that (4.4) allow us to compute $\mathscr{F}_n^{-1}$ or $\overline{\mathscr{F}}_n$ with the same algorithm as for $\mathscr{F}_n$, the fast Fourier transform. This shows that they can both be calculated with $O(n \log n)$ complexity.

Definition 4.3. The *Hadamard product* of $\mathbf{x}$ and $\mathbf{y}$ in Π_n is given by

$$(\mathbf{x} \cdot \mathbf{y})_m = x_m y_m \tag{4.5}$$

This is just term-wise multiplication.

Definition 4.4. Given $\mathbf{u}$ and $\mathbf{v}$ in Π_n we define their *convolution* as

$$(\mathbf{u} * \mathbf{v})_m = \sum_{k=0}^{m} u_k v_{m-k} = \sum_{k=0}^{m} u_{m-k} v_k. \tag{4.6}$$

It's easy to see these two sums have the same terms and are therefore the same.

We now relate this to the discrete Fourier transform. Given any $\mathbf{u}$ and $\mathbf{v}$, we can associate unique sequences $\mathbf{x} = \mathscr{F}_n^{-1}u$ and $\mathbf{y} = \mathscr{F}_n^{-1}v$. Equivalently, $\mathbf{u} = \mathscr{F}_n\mathbf{x}$ and $\mathbf{v} = \mathscr{F}_n\mathbf{y}$. We can write their convolution as

$$\begin{aligned}(\mathbf{u}*\mathbf{v})_m &= \sum_{k=0}^{n-1} u_{m-k}v_k \\ &= \sum_{k=0}^{n-1} \frac{1}{n}\left(\sum_{j=0}^{n-1} x_j w_n^{-(m-k)j}\right) v_k \\ &= \frac{1}{n}\sum_{k=0}^{n-1}\sum_{j=0}^{n-1} v_k x_j w_n^{-(m-k)j}.\end{aligned}$$

Interchanging the order of summations we have

$$\begin{aligned}&\frac{1}{n}\sum_{j=0}^{n-1}\sum_{k=0}^{n-1} v_k x_j w_n^{-(m-k)j} \\ &= \frac{1}{n}\sum_{j=0}^{n-1} x_j w_n^{-mj}\sum_{k=0}^{n-1} v_k w_n^{jk}.\end{aligned}$$

Using the formula for $\mathscr{F}_n^{-1}$ we get,

$$\begin{aligned}&\frac{1}{n}\sum_{j=0}^{n-1} x_j w_n^{-mj}(y_j) \\ &= \frac{1}{n}\sum_{j=0}^{n-1}(x_j y_j) w_n^{-mj} \\ &= (\mathscr{F}_n(\mathbf{x}\cdot\mathbf{y}))_m.\end{aligned}$$

What we've shown is that

$$\mathscr{F}_n(\mathbf{x}\cdot\mathbf{y}) = \mathscr{F}_n\mathbf{x} * \mathscr{F}_n\mathbf{y}$$

We can also rewrite this as

$$\mathbf{u}*\mathbf{v} = \mathscr{F}_n(\mathscr{F}_n^{-1}\mathbf{u}\cdot\mathscr{F}_n^{-1}\mathbf{v}).$$

Using the identities in (4.4) we have

$$\begin{aligned}\mathbf{u}*\mathbf{v} &= \mathscr{F}_n(\mathscr{F}_n^{-1}\mathbf{u}\cdot\mathscr{F}_n^{-1}\mathbf{v}) \\ &= \mathscr{F}_n(nR\mathscr{F}_n\mathbf{u}\cdot nR\mathscr{F}_n\mathbf{v}) \\ &= n^2\mathscr{F}_n(R\mathscr{F}_n\mathbf{u}\cdot R\mathscr{F}_n\mathbf{v})\end{aligned}$$

Clearly, the order of termwise multiplication and reversion can be done in either order, so

$$n^2\mathscr{F}_n(R\mathscr{F}_n\mathbf{u}\cdot R\mathscr{F}_n\mathbf{v}) = n^2\mathscr{F}_nR(\mathscr{F}_n\mathbf{u}\cdot\mathscr{F}_n\mathbf{v})$$

Also by (4.4) we have $\mathscr{F}_n^{-1} = n\mathscr{F}_nR$ and so

$$n^2\mathscr{F}_nR(\mathscr{F}_n\mathbf{u}\cdot\mathscr{F}_n\mathbf{v}) = n\mathscr{F}_n^{-1}(\mathscr{F}_n\mathbf{u}\cdot\mathscr{F}_n\mathbf{v})$$

and hence

$$\mathbf{u} * \mathbf{v} = n\mathscr{F}_n^{-1}(\mathscr{F}_n\mathbf{u} \cdot \mathscr{F}_n\mathbf{v})$$

Taking $\mathscr{F}_n$ of both sides we get

$$\mathscr{F}_n(\mathbf{u} * \mathbf{v}) = n\mathscr{F}_n\mathbf{u} \cdot \mathscr{F}_n\mathbf{v}$$

In short, we've shown the *convolution theorem.*

Theorem 4.2. *For sequences* $\mathbf{x}$ *and* $\mathbf{y}$ *in* Π_n*,*

$$\begin{aligned} \mathscr{F}_n(\mathbf{x} \cdot \mathbf{y}) &= \mathscr{F}_n\mathbf{x} * \mathscr{F}_n\mathbf{y} \\ \mathscr{F}_n(\mathbf{x} * \mathbf{y}) &= n\mathscr{F}_n\mathbf{x} \cdot \mathscr{F}_n\mathbf{y} \end{aligned} \tag{4.7}$$

Because the convolution is equivalent to three Fourier transforms and a single Hadamard product, it may be computed in $O(n \log n)$ time. More specifically, if n is a power of two, it will take no more than

$$\frac{3}{2}n\log_2(2n)$$

complex multiplications. This is a large asymptotic improvement when compared to the naïve $O(n^2)$ algorithm for computing a convolution from the definition.

4.2 Polynomial Multiplication

Suppose we are given polynomials p and q of degree $n-1$ or less given by

$$\begin{aligned} p(z) &= p_0 + p_1 z + \cdots + p_{n-1}z^{n-1} \\ q(z) &= q_0 + q_1 z + \cdots + q_{n-1}z^{n-1} \end{aligned}$$

Consider their product $r(z) = p(z)q(z)$ which is a polynomial of degree at most $2n-2$. The naïve algorithm for multiplication requires n^2 complex multiplications.

Let $\mathbf{p}$ and $\mathbf{q}$ be two sequences in Π_{2n-2} define as

$$\begin{aligned} \mathbf{p} &= \{p_0, p_1, \ldots, p_{n-1}, 0, \ldots, 0\} \\ \mathbf{q} &= \{q_0, q_1, \ldots, q_{n-1}, 0, \ldots, 0\} \end{aligned}$$

What is the kth coefficient of r? This is the sum of every possible combination of coefficients p_j and q_h such that

$$j + h = k$$

and $0 \le j \le n-1$ and $0 \le h \le n-1$. Examining the definition of convolution in (4.6) we see that

$$\mathbf{r} = \mathbf{p} * \mathbf{q}$$

where $\mathbf{r}$ is the corresponding sequence for r in Π_{2n-2}. Thus, with the fast Fourier transform in hand, we may improve the $O(n^2)$ naïve algorithm to $O(n \log n)$, a rather astounding result.

Note, also that the same idea may be used for an $O(n \log n)$ algorithm for the multiplication of large integers where n is the number of digits. This is a huge improvement over the $O(n^2)$ algorithm taught in school and most often used by humans. For more information, see [4].

4.3 Formal Power Series Multiplication

In this section we work with infinite power series without considering convergence. We think of them as objects on which we can do formal manipulation. For a power series $P = a_0 + a_1 z + \cdots$, we define P_n to be the partial sum containing the first n terms. That is,

$$P_n = a_0 + a_1 z + \cdots a_{n-1} z^{n-1}$$

Given another power series Q, we can compute the nth coefficient in a similar matter. Indeed, the nth of PQ is only affected by the first n terms of P and Q, so we may reduce it to the finite case.

So, to compute the first n coefficients of PQ, we instead simply compute $P_n Q_n$. What we have is that

$$(PQ)_n = (P_n Q_n)_n.$$

Using the algorithm in the previous section for polynomial multiplication, computing the first n terms requires $O(n \log n)$ complex multiplications. This is another big improvement over the $O(n^2)$ naïve algorithm.

5 Additional Details on the Multi-Dimensional Transform

In the paper, Henrici gives a definition of the multi-dimensional discrete Fourier transform but omits several details and proofs. Here I will define the multi-dimensional transform and describe in detail how to derive its properties.

I will not give applications of the multi-dimensional transformation here, but Henrici gives several, see [3]. For example, the multi-dimensional convolution theorem may be used to create fast Poisson solvers.

5.1 The Multi-Dimensional Transform

We define a new space $\Pi_n^{(d)}$ of d-dimensional bilateral infinite sequences periodic in n for each dimension. Such an object $\mathbf{x} \in \Pi_n^{(d)}$ is denoted

$$\mathbf{x} = \{x_{k_1 k_2 \cdots k_d}\}_{k_i=\infty}^{\infty}$$

where $\mathbf{x}$ satisfies

$$x_{k_1 k_2 \cdots (k_j+n) \cdots k_d} = x_{k_1 k_2 \cdots k_j \cdots k_d}$$

for any j and k_j.

Definition 5.1. The *d-dimensional discrete Fourier transform* is defined as

$$\mathbf{y} = \mathscr{F}_n^{(d)} \mathbf{x}$$

where

$$y_{m_1 m_2 \cdots m_d} = \frac{1}{n^d} \sum_{k_1=0}^{n-1} \cdots \sum_{k_d=0}^{n-1} w_n^{-k_1 m_1 - \cdots - k_d m_d} x_{k_1 k_2 \cdots k_d}. \tag{5.1}$$

For the same reason as in the one-dimensional case, $\mathscr{F}_n^{(d)}$ is linear. Bijectivity carries over as well. Henrici does not give a proof of the bijectivity of the multi-dimensional transform, so I will give one here. Something similar in the one-dimensional case may be found in [1], but my version will generalize to higher dimensions.

First we define two more operators similarly to the one-dimensional case.

Definition 5.2. For $\mathbf{x} \in \Pi_n^{(d)}$, the *multi-dimensional conjugate Fourier operator* $\mathbf{y} = \overline{\mathscr{F}}_n^{(d)}\mathbf{x}$ is defined as

$$y_{m_1 m_2 \cdots m_d} = \frac{1}{n^d} \sum_{k_1=0}^{n-1} \cdots \sum_{k_d=0}^{n-1} w_n^{k_1 m_1 + \cdots + k_d m_d} x_{k_1 k_2 \cdots k_d}. \tag{5.2}$$

The *multi-dimensional inverse Fourier operator* is defined as

$$\mathscr{F}_n^{(d)^{-1}}\mathbf{x} = n^d \overline{\mathscr{F}}_n^{(d)}\mathbf{x}. \tag{5.3}$$

Because $\mathscr{F}_n^{(d)}$ is a linear operator on a n^d-dimensional space, we may associate to it a matrix A. Similarly, we see from the definition that $\mathscr{F}_n^{(d)^{-1}}$ is linear and has a matrix A^{-1}. It will suffice to show that $AA^{-1} = I$.

To proceed, it is convenient to introduce vector notation for the multi-dimensional transform. If $\mathbf{k}$ is an d-dimensional vector of integers and $\mathbf{x} \in \Pi_n^{(d)}$, then we define

$$x_{\mathbf{k}} = x_{k_1 k_2 \cdots k_d}.$$

We define Q^n to be the d-dimensional unit lattice which is the set

$$Q_n = \{\mathbf{k} : 0 \leq k_j \leq n-1 \, \forall j\}$$

With this notation, (5.1) becomes

$$y_{\mathbf{m}} = \frac{1}{n^d} \sum_{\mathbf{k} \in Q_n} w_n^{-\mathbf{k}\cdot\mathbf{m}} x_{\mathbf{k}}$$

where here the dot indicates the *dot product.*

To each possible index vector $\mathbf{k}$ we associate unique integer k in the range $0 \leq k \leq n^d - 1$. Such a mapping may be given by $k = k_1 + nk_2 + \cdots + k_n n^{d-1}$. Regardless, all that matters is that for all $\mathbf{k} \in Q^n$ we may associate a *row* of the matrix A, or a column. That is, given integers j and k that lie between 1 and $n^d - 1$, the entry A_{jk} is associated with two index vectors $\mathbf{j}$ and $\mathbf{k}$. We may now describe what A looks like. Examining where each basis vector in $\Pi_n^{(d)}$ is sent we see that

$$A_{jk} = \frac{1}{n^d} w_n^{-\mathbf{j}\cdot\mathbf{k}} \tag{5.4}$$

where here we once again mean the dot product and $\mathbf{j}$ and $\mathbf{k}$ are the associated index vectors to the integers j and k. Similarly, from the definition we see

$$A_{jk}^{-1} = w_n^{\mathbf{j}\cdot\mathbf{k}}.$$

These matrices are more complicated than simply Vandermonde matrices.

Let

$$\delta_{ij} = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$$

which is the Kronecker delta function. Showing $AA^{-1} = I$ is the same as showing $(AA^{-1})_{jk} = \delta_{jk}$. This entry is given by

$$\begin{aligned}(AA^{-1})_{jk} &= \sum_{\mathbf{r}\in Q^n} \frac{1}{n^d} w_n^{-\mathbf{j}\cdot\mathbf{r}} w_n^{\mathbf{r}\cdot\mathbf{k}} \\ &= \frac{1}{n^d} \sum_{\mathbf{r}\in Q^n} w_n^{\mathbf{r}\cdot(\mathbf{k}-\mathbf{j})}.\end{aligned}$$

Unrolling the summation we get

$$\begin{aligned}&\frac{1}{n}\sum_{r_1}^{n-1} \frac{1}{n}\sum_{r_2}^{n-1} \cdots \frac{1}{n}\sum_{r_d=0}^{n-1} w_n^{r_1(k_1-j_1)+\cdots+r_d(k_d-j_d)} \\ &= \frac{1}{n}\sum_{r_1}^{n-1} w_n^{r_1(k_1-j_1)} \frac{1}{n}\sum_{r_2}^{n-1} w_n^{r_2(k_2-j_2)} \cdots \frac{1}{n}\sum_{r_d=0}^{n-1} w_n^{r_d(k_d-j_d)}.\end{aligned}$$

Now, examine the innermost sum. By (2.1), this is 0 if $k_d \neq j_d$. This would make each outer sum 0 as well. On the other hand, if $k_d = j_d$ then $\frac{1}{n} = \sum_{r_d=0}^{n-1} w_n^{r_d(k_d-j_d)} = 1$ and we have

$$\frac{1}{n}\sum_{r_1}^{n-1} w_n^{r_1(k_1-j_1)} \frac{1}{n}\sum_{r_2}^{n-1} w_n^{r_2(k_2-j_2)} \cdots \frac{1}{n}\sum_{r_{d-1}=0}^{n-1} w_n^{r_{d-1}(k_{d-1}-j_{d-1})}$$

We may repeat this process as many times as desired. If it turns that $k_\alpha = j_\alpha$ for all possible α then we end up with 1. On the other hand, if $k_\alpha \neq j_\alpha$ at any point then one of these sums becomes 0 and all outer sums vanish. Hence, we've shown that $(AA^{-1})_{jk} = 1$ if $\mathbf{j} = \mathbf{k}$ and $(AA^{-1})_{jk} = 0$ if $\mathbf{j} \neq \mathbf{k}$. This says exactly that $(AA^{-1})_{jk} = \delta_{jk}$, hence $AA^{-1} = I$, completing the proof.

What we've shown is that because $\mathscr{F}_n^{(d)}$ and $\mathscr{F}_n^{(d)^{-1}}$ are linear operators with invertible matrices that give the identity matrix when multiplied. This means both are bijective operators where each is the inverse of the other.

5.2 Computation of Multi-Dimensional Transform

The question arises of how to compute multi-dimensional transforms efficiently. Doing this directly from (5.1) requires n^d multiplications for each of the n^d components. This is a total of $(n^d)^2$ multiplications which requires quadratic time in the input size. That is, if we are given $\mathbf{x} \in \Pi_n^{(d)}$ then that is n^d total numbers and so the input size is n^d.

Thankfully, no new algorithms are required. We may write (5.1) as

$$y_{m_1 m_2 \cdots m_d} = \frac{1}{n}\sum_{k_1=0}^{n-1} w_n^{-k_1 m_1} \frac{1}{n}\sum_{k_2=0}^{n-1} w_n^{-k_2 m_2} \cdots \frac{1}{n}\sum_{k_d=0}^{n-1} w_n^{-k_d m_d} x_{k_1 k_2 \cdots k_d}.$$

This shows that we may think of a d-dimensional transform as the combination of d transforms over n^d possible combinations. Indeed, imagine for each of the n^{d-1} combinations of k_1, k_2, $\ldots k_{d-1}$ that we compute the inner transform

$$\frac{1}{n}\sum_{k_d=0}^{n-1} w_n^{-k_d m_d} x_{k_1 k_2 \cdots k_d}.$$

Then, for each of the n^{d-1} combinations of x_1, x_2, $\ldots$, x_{d-2}, x_d we compute the next inner-most transform using the values of the innermost transform we just computed. Repeating this d times we may compute each $y_{m_1 m_2 \cdots m_d}$ when we reach the outermost level. This shows the total number of multiplications is dn^{d-1} times the number required for a single transform. If $n = n_1 n_2 \cdots n_k$ then from (2.6) the running time is

$$dn^d \sum_{j=1}^{k} (n_j - 1).$$

In the case that n is a power of two then we get

$$dn^d \log n = n^d \log n^d$$

What this last equality shows is that if the input size is n^d then the running time is the same $O(m \log m)$ as in the one-dimensional case where here we take m to be the input size n^d.

There exist algorithms outside of simple iterated one-dimensional transforms. Henrici gives an example that gives better results for large values of d, see [3].

5.3 Multi-Dimensional Convolution Theorem

The convolution theorem generalizes to higher dimensions. As in the one-dimensional case, we define the *Hadamard product* $\mathbf{x} \cdot \mathbf{y}$ for $\mathbf{x}, \mathbf{y} \in \Pi_n^{(d)}$ to be their entry-wise product. For the multi-dimensional convolution we have

Definition 5.3. For vectors $\mathbf{x}$ and $\mathbf{y}$ in $\Pi_n^{(d)}$ we define $\mathbf{z} = \mathbf{x} * \mathbf{y}$ to be

$$z_{\mathbf{k}} = \sum_{\mathbf{m} \in Q_n} x_{\mathbf{m}} y_{\mathbf{k}-\mathbf{m}} \tag{5.5}$$

This leads us to the multi-dimensional convolution theorem.

Theorem 5.1. *For any sequences* $\mathbf{x}, \mathbf{y} \in \Pi_n^{(d)}$ *we have*

$$\begin{aligned} \mathscr{F}_n^{(d)}(\mathbf{x} \cdot \mathbf{y}) &= \mathscr{F}_n^{(d)}\mathbf{x} * \mathscr{F}_n^{(d)}\mathbf{y} \\ \mathscr{F}_n^{(d)}(\mathbf{x} * \mathbf{y}) &= \mathscr{F}_n^{(d)}\mathbf{x} \cdot \mathscr{F}_n^{(d)}\mathbf{y} \end{aligned} \tag{5.6}$$

Henrici omits a proof of this theorem. For a proof in the two-dimensional case, see [2]. Here I will give a proof of the general case, similar to what we did in one dimension. Given sequences $\mathbf{x}$ and $\mathbf{y}$ and let $\mathbf{u} = \mathscr{F}_n^{(d)}\mathbf{x}$ and $\mathbf{v} = \mathscr{F}_n^{(d)}\mathbf{y}$. We have,

$$\begin{aligned} \mathscr{F}_n^{(d)}\mathbf{x} * \mathscr{F}_n^{(d)}\mathbf{y} &= (\mathbf{u} * \mathbf{v})_{\mathbf{m}} \\ &= \sum_{\mathbf{k} \in Q_n} u_{\mathbf{m}-\mathbf{k}} v_{\mathbf{k}} \end{aligned}$$

We can use the formula for the inverse transform to substitute for $u_{\mathbf{m}-\mathbf{k}}$.

$$\begin{aligned}
&\sum_{\mathbf{k}\in Q_n} \frac{1}{n^d}\left(\sum_{\mathbf{j}\in Q_n} x_{\mathbf{j}} w_n^{-(\mathbf{m}-\mathbf{k})\cdot\mathbf{j}}\right) v_{\mathbf{k}} \\
&= \frac{1}{n^d}\sum_{\mathbf{k}\in Q_n}\sum_{\mathbf{j}\in Q_n} v_{\mathbf{k}} x_{\mathbf{j}} w_n^{-(\mathbf{m}-\mathbf{k})\cdot\mathbf{j}}.
\end{aligned}$$

Interchanging the order of summations we have

$$\frac{1}{n^d}\sum_{\mathbf{j}\in Q_n}\sum_{\mathbf{k}\in Q_n} v_{\mathbf{k}} x_{\mathbf{j}} w_n^{-(\mathbf{m}-\mathbf{k})\cdot\mathbf{j}}$$

which we can rewrite as

$$\begin{aligned}
&\frac{1}{n^d}\sum_{\mathbf{j}\in Q_n} x_{\mathbf{j}} w_n^{-\mathbf{m}\cdot\mathbf{j}} \sum_{\mathbf{k}\in Q_n} v_{\mathbf{k}} w_n^{\mathbf{j}\cdot\mathbf{k}} \\
&= \frac{1}{n^d}\sum_{\mathbf{j}\in Q_n} x_{\mathbf{j}} w_n^{-\mathbf{m}\cdot\mathbf{j}}(y_{\mathbf{j}}) && \text{Definition of inverse} \\
&= \frac{1}{n^d}\sum_{\mathbf{j}\in Q_n} (x_{\mathbf{j}} y_{\mathbf{j}}) w_n^{-\mathbf{m}\cdot\mathbf{j}} \\
&= (\mathscr{F}_n^{(d)}(\mathbf{x}\cdot\mathbf{y}))_{\mathbf{m}}.
\end{aligned}$$

Thus,

$$\mathscr{F}_n^{(d)}(\mathbf{x} * \mathbf{y}) = \mathscr{F}_n^{(d)}\mathbf{x}\cdot\mathscr{F}_n^{(d)}\mathbf{y}$$

The other half of (5.6) can be derived from this exactly as in the justification of (4.7). This completes the proof.

6 Conclusion

This paper has introduced the very basics of the fast Fourier transform. I invite the interested reader to examine Henrici's whole article "Fast Fourier Methods in Computational Complex Analysis." Topics not covered here include time series analysis, numerical solutions to certain equations such as Symm's equation and Theodorsen's integral equation. Approximating the Fourier coefficients of the solution to the Dirichlet problem on doubly connected regions is of particular interest for for complex analysis. Henrici also covers a lot more on power series.

The applications of the fast Fourier transform include almost every area of science as well as countless branches of mathematics. This paper has hopefully served as a gentle and interesting introduction to the topic. However, the applications here and in Henrici's paper are only the beginning. Fourier analysis will surely turn up in many more places to come and whenever that happens fast Fourier methods will almost surely play a role.

References

[1] A. V. Aho, *The design and analysis of computer algorithms*, ser. Addison-Wesley series in computer science and information processing. Reading, Mass.: Addison-Wesley Pub. Co., 1974, ISBN: 0201000296.

[2] D. E. Dudgeon, *Multidimensional digital signal processing*, ser. Prentice-Hall signal processing series. Englewood Cliffs, NJ: Prentice-Hall, 1984, pp. 70–71, ISBN: 0136049591.

[3] P. Henrici, "Fast fourier methods in computational complex analysis," eng, *SIAM Review*, vol. 21, no. 4, pp. 481–527, 1979, ISSN: 00361445.

[4] A. Schönhage and V. Strassen, "Fast multiplication of large numbers," *Computing*, vol. 7, no. 3, pp. 281–292, 1971, ISSN: 0010-485X.

[5] L. R. Turner, "Inverse of the vandermonde matrix with applications - nasa-tn-d-3547," Tech. Rep., 1966. [Online]. Available: `http://hdl.handle.net/2060/19660023042`.