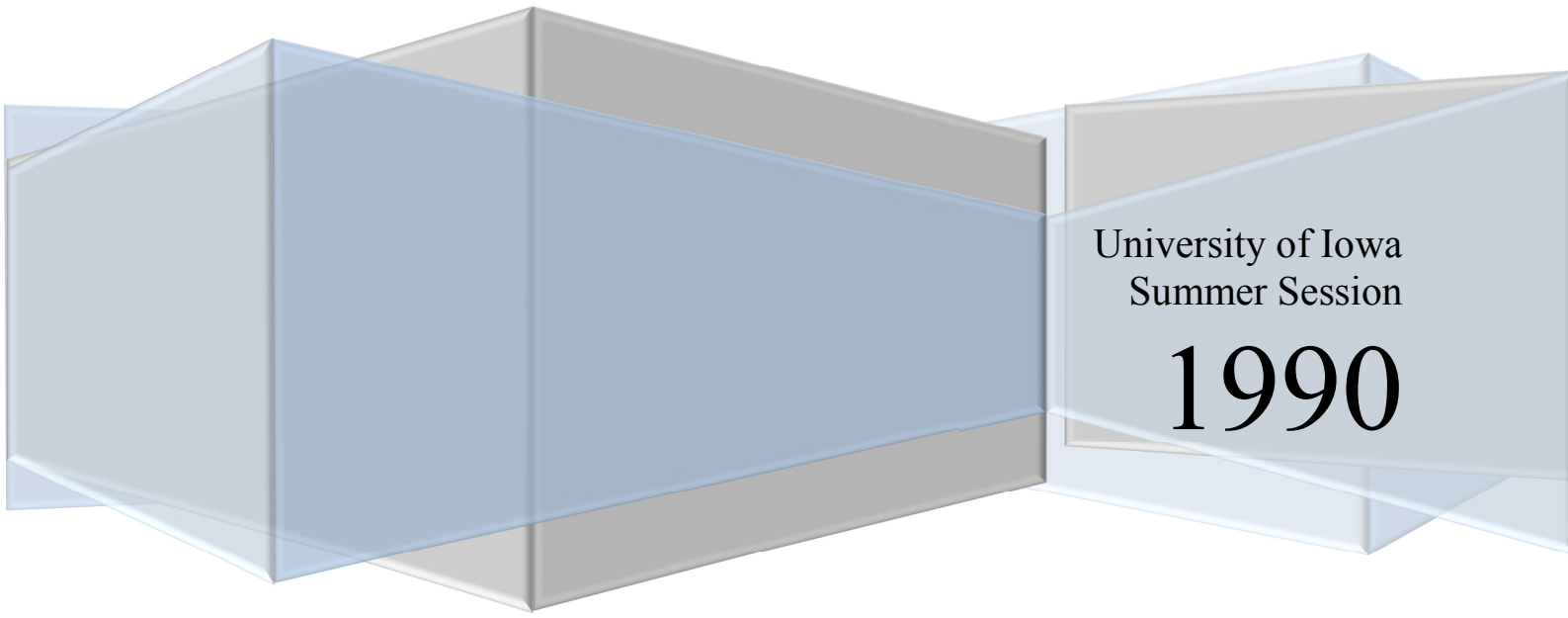
55:198 Individual Investigations: Electrical and Computer Engineering.
University of Iowa, Iowa City, IA 52242

On Computing the Discrete Fourier Transform

Advisor: Prof. John P. Robinson

Alastair Roxburgh



Newly Revised & Updated

Copyright notice:

On computing the discrete Fourier transform

Abstract: The development of time-efficient small-*N* discrete Fourier transform (DFT) algorithms has received a lot of attention due to the ease with which they combine, "building block" style, to yield time-efficient large transforms. This paper reports on the discovery that efficient computational algorithms for small-*N* DFT developed during the 19th century bear more than a passing resemblance to similar-sized modern-day algorithms, including the same nested $(+)(\times)(+)$ structure, similar flow graphs, and a comparable number of arithmetic operations. This suggests that despite the formal sophistication of more recent approaches to the development of efficient small-*N* DFT algorithms, the key underlying principles are still the symmetry and periodicity properties of the sine and cosine basis functions of the Fourier transform. While the earlier methods explicitly manipulated the DFT operator on the level of these properties, the present-day methods (typically based on the cyclic convolution properties of the DFT operator) tend to hide this more basic level of reality from view. All reduced-arithmetic DFT algorithms take advantage of how easy it is to factor the DFT operator. From the matrix point of view, an efficient DFT algorithm results when we factor the DFT operator into a product of sparse matrices containing mostly ones and zeros. Given that there are innumerable factorizations, it is interesting that modern-day algorithms developed using number-theoretic techniques quite removed from the trigonometric identities and simple algebraic techniques used by the pioneers of discrete signal analysis, should be so similar in form to the early algorithms.

© 1990–2013 by Alastair J. Roxburgh. All rights reserved.

Publication data:

Version 2.1.3, December 9, 2013

Send error notifications and update enquiries to: aroxburgh@ieee.org

ON COMPUTING THE DISCRETE FOURIER TRANSFORM

by Alastair Roxburgh

Abstract

*The development of time-efficient small-*N *discrete Fourier transform (DFT) algorithms has received a lot of attention due to the ease with which they combine, "building block" style, to yield time-efficient large transforms. This paper reports on the discovery that efficient computational algorithms for small-*N *DFT developed during the 19th century bear more than a passing resemblance to similar-sized modern-day algorithms, including the same nested* $(+)(\times)(+)$ *structure, similar flow graphs, and a comparable number of arithmetic operations. This suggests that despite the formal sophistication of more recent approaches to the development of efficient small-*N *DFT algorithms, the key underlying principles are still the symmetry and periodicity properties of the sine and cosine basis functions of the Fourier transform. While the earlier methods explicitly manipulated the DFT operator on the level of these properties, the present-day methods (typically based on the cyclic convolution properties of the DFT operator) tend to hide this more basic level of reality from view. All reduced-arithmetic DFT algorithms take advantage of how easy it is to factor the DFT operator. From the matrix point of view, an efficient DFT algorithm results when we factor the DFT operator into a product of sparse matrices containing mostly ones and zeros. Given that there are innumerable factorizations, it is interesting that modern-day algorithms developed using number-theoretic techniques quite removed from the trigonometric identities and simple algebraic techniques used by the pioneers of discrete signal analysis, should be so similar in form to the early algorithms.*

Table of Contents

List of Tables vi

List of Figures vi

1. Prelude 1

2. The early developmental period 5

3. The modern developmental period 19

4. Efficient small-*N* DFT algorithms 23

5. Large transforms from small ones 31

6. Summary and conclusions 39

Appendix A: Runge *N* = 12 DFT algorithm for real data 40

Appendix B: Hann, Brooks and Carruthers *N* = 12 DFT algorithm for real data. 43

Appendix C: Winograd *N* = 8 DFT algorithm 46

References 47

List of Tables

Table 1. Real-Input DFT Algorithm for $N = 24$, Kämtz 15

Table 2. Real-Input DFT Algorithm for $N = 12$, Hann, Brooks & Carruthers 16

Table 3. Real-Input DFT Algorithm for $N = 12$, Runge 17

Table 4. Real-Input DFT Algorithm for $N = 8$ using Circular Convolution, Winograd 21

Table 5. Conjectured Classification of $N = 4m$ DFT Algorithms 29

Table 6. Input and Output Index Calculations for the $N = 3 \times 4$ WFTA algorithm 34

Table 7. Number of arithmetic operations for modern-day small-N DFT having (+)(×)(+) nested arithmetic structure 36

List of Figures

Figure 1. Sande-Tukey $N = 4$ radix-2 DIF FFT. Input and output in natural order. Arithmetic: 8 additions. 24

Figure 2. Small-N DFT with nested (+)(×)(+) arithmetic structure showing the expansion caused by more than one multiply per data point. 26

Figure 3. Winograd small-N DFT matrix factorization for $N = 4$. Inputs and outputs in natural order. Arithmetic: 8+ 26

Figure 4. Winograd small-N DFT matrix factorization for $N = 3$. Inputs and outputs in natural order. Arithmetic: 6+, 2×. 27

Figure 5. Elliot and Rao small-N DFT factorization for $N = 3$. Inputs and outputs in natural order. Arithmetic: 6+, 1×, 1 shift. 27

Figure 6. Runge's $N = 24$ hybrid FFT algorithm for real-data. 38

ON COMPUTING THE DISCRETE FOURIER TRANSFORM

Alastair Roxburgh

Report submitted in partial fulfillment of the requirements for
55:198, *Individual Investigations: ECE*.

Revised and updated by Alastair Roxburgh

Advisor: Prof. John P. Robinson

Electronics and Computer Engineering Department,
University of Iowa
Iowa City, IA 52242, USA

Version Log:

Version #	*Date*	*Details*
2.0.1	12/31/2010	Corrected pagination.
2.0.2	1/1/2012	Added missing R.W. Hamming reference.
2.0.4	1/14/2012	Expanded discussion of WFTA canonical forms; clarified discussion of WFTA input and output indices ordered according to Chinese Remainder Theorem and Second Integer Representation theorem.
2.0.16	7/18/2012	Least-squares derivation of DFT improved. Added relationship between FS and DFT.
2.0.19	1/24/2013	Rewritten introduction spun off as new chapter: Prelude.
2.1.1	12/4/2013	Proofing.
2.1.3	12/9/2013	Final proofing and pdf rendering.

1. Prelude

Joseph Fourier's famous memoir, *Théorie de la propagation de la chaleur dans les solides* (Fourier, 1807), an extract of which was read before the First Class of *l'Académie des Sciences de l'Institut de France*, 21st of December 1807, contained the extraordinary claim that an arbitrary function[1] defined on a finite domain can be represented analytically by means of an infinite trigonometric series. Not one person in the distinguished audience, that cold and foggy evening in Paris, just four days before Christmas, realized that they had just witnessed one of the key events in the history of mathematics.

Fourier's presentation consisted of a carefully contrived mix of theoretical development and the results of physical experiments. This put him in an unassailable, although not necessarily popular position. If the presentation had a weakness, it certainly did not lie in Fourier's oratory skills, which were renowned; instead, it lay in his lack of a *complete* formal mathematical proof. Past the initial surprise and incredulity regarding Fourier's use of infinite trigonometric series, the mathematicians in the audience left the meeting with the troublesome realization that some of their cherished 18th century notions of mathematical functions were possibly wrong, or at best incomplete. If truth be known, some of these notions had been experiencing increasing (although supposedly minor) difficulties for several decades, and some of Fourier's ideas had been suggested previously by others, albeit unsuccessfully. Lacking Fourier's scientific vision and mathematical virtuosity, these predecessors had been unable to determine the correct linear partial differential equation for heat flow in solids, let alone generate physically verifiable analytical solutions and prove their uniqueness. Mathematical proofs aside, Fourier's presentation that evening was certainly a *tour de force.* Not only did he derive the correct heat flow equation, but also showed how to resolve arbitrary initial temperature profiles into easily-solvable spatially sinusoidal components, using a particular type of infinite trigonometric series (now known as a Fourier series). In each case, he neatly supported his theoretical calculations with the results of carefully conducted laboratory heat experiments.

At the time of Fourier's presentation, it was common knowledge that infinite trigonometric series sometimes converged, and other times did not. As a result, they were widely regarded as being unreliable and untrustworthy. Senior academician Joseph-Louis Lagrange, who years before had gone out of his way to discredit such series, was particularly shocked that one of his former star pupils and colleague from *l'École Polytechnique*, should attempt to present them as a reliable solution to anything. The horns of the dilemma were that if Fourier's theory was wrong, why did his experimental work, which consisted of measuring temperature gradients in heated metal shapes, corroborate it? On the other hand, if Fourier were right, the ramifications would extend far outside Fourier's heat laboratory. The elderly

[1] Seventeen years later, when Fourier published his theory of heat (*Théorie Analytique de la Chaleur*, 1824; Eng. trans. 1878), his ideas had developed to where he was explicitly stating that the arbitrary function must also be integrable. Thus Fourier presaged the first definitive set of conditions for the existence of a Fourier series, which were published five years later by (Johann Peter) Gustav Lejeune Dirichlet (1829).

Lagrange, who was the ranking scientific referee for Fourier's presentation (the other referees were Pierre-Simon Laplace, Gaspard Monge, and Sylvestre François Lacroix), issued a summary dismissal of the memoir, and flatly refused to discuss publication because it disagreed with his own investigation of trigonometric expansions. In a more normal course of events, timely publication in the *Mémoires de l'Académie des Sciences*[2] would have been assured.

Fortunately for Fourier, Siméon Denis Poisson who was not yet a member of the Academy, and therefore still somewhat of a free agent, deserves particular credit for publishing a short account of Fourier's presentation. In the absence of any official publication of Fourier's memoir, Poisson's article (1808), firmly established Fourier's scientific priority in the subject material. In utter contrast to Lagrange's reaction to Fourier's presentation, Poisson's report ends with Poisson barely able to contain his excitement: « La plus remarquable est celle qui est relative au refroidissement d'un anneau métallique… »

> "The most remarkable [experiment performed by Fourier to verify the results of his analysis,] is the one relating to the cooling of a metal ring: …[irrespective of the initial distribution of heat] the ring soon reaches a state in which the sum of the temperatures of the two points at the ends of the same diameter, is the same for all diameters, and that once this state is reached, it is maintained until full cooling…and on this point the experiment was found to agree with his analysis that had led to the same result."

Given Poisson's obvious interest in Fourier's results, it is not surprising that Poisson later became a rival. Also not surprising, given the incomplete state of the mathematics of infinite series at that time, is that Fourier's heat propagation memoir was the beginning of many years of difficulties for Fourier. Having simply searched for an unsolved physics problem, and chosen the flow of heat in solid bodies, Fourier had no idea that in solving this problem he would inadvertently stir up a controversy that would consume the energies of at least five generations of mathematicians. Moreover, research on the question of just how-arbitrary a function can be, and still have a convergent Fourier series, continues unabated even today. Recent research by Lennart Carleson, Yitzhak Katznelson, Jean-Pierre Kahane, and others, suggests that although Fourier, from a strictly analytical point of view, was wrong about arbitrary signals, he was far more right than he knew. However, in the case of practical real-world (causal) signals (which all have Fourier series), Fourier was completely right.

Twenty-five years after Fourier presented his (then infamous and now famous) memoir, a ground-breaking proof of the conditions for convergence of Fourier series was published by a bright young German mathematician, (Johann Peter) Gustav Lejeune Dirichlet (1829). At long last the dust stirred up by Fourier's memoir began to settle, and those parts of mathematics concerned with the convergence of infinite series, limits, continuity, functions, derivatives and integrals, finally gained a firm analytical footing. Over the following seventy years or so, the deep original physical and mathematical insights of Fourier and Lejeune Dirichlet would grow

---

[2] In publication since 1666, *Mémoires de l'Académie des Sciences* was renamed in 1835 as *Comptes rendus hebdomadaires des séances de l'Académie des Sciences* (or simply *Comptes rendus; English: Proceedings of the Academy of Sciences*).

into the unified analytical framework upon which our modern age of science and technology has flourished.



The Institut Nationale de France buildings, where Fourier read his famous memoir on 21st December 1807, are situated on the left bank of the Seine, across from the Palais des Arts (now the Louvre).



A modern-day view of the Académie des sciences - Institut de France buildings, 23, quai de Conti, 75006, Paris, France, as seen from Pont des Arts across the Seine.
Photo by Benh Lieu Song, Sept 2007. Licensed under Creative Commons.

2. The early developmental period

Moving away from the traditional preoccupation with astronomy and the concept of perfect celestial order, the first decades of the 19th century saw a new generation of physical scientists begin to subject the least regarded celestial realm, the planet Earth itself, to increasing scientific scrutiny. Armed with tools provided by the new mathematics, no problem seemed out of reach. Their tool of choice was the very same one perfected by Fourier: the modeling of natural phenomena as a boundary value problem using time-dependent partial differential equations. Unprecedented levels of success resulted, in fields as diverse as electromagnetism, fluid dynamics, and quantum theory, not to mention heat flow. This lent a new air of objectivity to science, and no problem seemed out of reach.

Building on earlier work by Alexis Claude Clairaut (1754) on finite cosine series as a means of modeling planetary orbits, and by Joseph-Louis Lagrange (1759, p. 79, art. 23) on finite sine series as part of his study of the propagation of sound, many histories recount that Fourier's infinite sine and cosine series were recast by astronomer and applied mathematician, (Friedrich) Wilhelm Bessel, into a form suitable for uniformly sampled empirical data. Known initially as "Bessel's formula," this finite approximation to the Fourier series, which in turn can be considered as a special case of the Fourier integral (or transform), later came to be known as the discrete Fourier transform (DFT),[3] and quickly became an essential item in every Earth scientist's toolkit.

It would be remiss to suggest that the history of Fourier analysis (of which the DFT is but one aspect) is as simple and straightforward as the previous paragraph suggests. Twenty years before Fourier was born, trigonometric series had been at the center of another, related mathematical controversy, known as *the vibrating string problem*. This earlier debate ran its course, ending when the greatest mathematician of that time, Leonhard Euler, rejected infinite trigonometric series as a general solution, a view endorsed by a capable new mathematician, Joseph-Louis Lagrange. Thus, when Fourier read before the French Science Academy in 1807, the uncertain status of trigonometric series was still very alive in the mind of the now elderly Lagrange. Lagrange was the sole remaining combatant from the vibrating string controversy, and now in his final years had earned the stature of most senior and respected mathematician at the Academy.

---

[3] It is not clear when the terms "Fourier transform" and "discrete Fourier transform" came into general use. Probably the former term is older than the latter, which probably arrived with the electronic computer age in the 1940s. The terms "Fourier transform," "Fourier integral," and "Fourier integral transform" are interchangeable. Numerical integration of the Fourier integral leads to the "finite Fourier transform" of which the "discrete Fourier transform" is a modified form with the origin moved left and the right-hand end-point deleted. The discrete Fourier transform, or DFT, can also be viewed as an approximation to the Fourier series, which itself can be derived as a special case of the Fourier integral for a periodic function. The terms "Fourier transform" and "discrete Fourier transform" often mean the transformation operation or the result of the transformation.

The earlier controversy surrounding the vibrating string concerned the proper solution to the wave equation (Wheeler & Crummett, 1987). In 1747, French mathematician Jean le Rond d'Alembert had developed a partial differential equation that described the transverse vibration of a taut string of length L, fixed at both ends (such as used in musical instrument design from time immemorial). Known as the wave equation in one-dimension, d'Alembert's equation, written as $u_{tt} = u_{xx}$, looks innocuous but in its brevity hides a lot of subtlety. Attempts at a solution by d'Alembert and Leonhard Euler were seriously hampered by the limited notion of a function at the time, and even after a young and very able Lagrange got involved, there was no real progress. Euler, who was probably already familiar with the equation for the vibration's fundamental envelope,

$$y(x) = A \sin \frac{\pi x}{L}$$

devised years before by English mathematician Brook Taylor,[4] obtained additional solutions from this by superposition, but although a valiant effort, was far from a complete solution. It was only when physicist Daniel Bernoulli took a different and in fact very modern approach, treating the vibrating string as a physics problem rather than one of strict mathematical analysis, did the necessary breakthrough occur. The year was 1753, fifteen years before Fourier was born, and more than half a century before his heat experiments.

Unlike the other participants in the controversy, Bernoulli actually *listened* to a string (he also exhorted his readers to do the same), and in doing so he noticed that in addition to the fundamental vibration, there were overtones or harmonics of the fundamental. His proposed solution to the d'Alembert wave equation was as radical as it was synthetic. He argued that in order for the boundary conditions to be satisfied, the sum had to be infinite, and that this is the general solution. Expressing an arbitrary transverse vibration of an ideal elastic string as an infinite sum of harmonically related sinusoids (nowadays known as a linear combination of normal vibration modes), Bernoulli's solution is

$$y(x,t) = A \sin \frac{\pi x}{L} \cos \frac{\pi t}{L} + B \sin \frac{2\pi x}{L} \cos \frac{2\pi t}{L} + \ldots,$$

where function $y(x, t)$ is the displacement of the string at spatial coordinate x and time t. This solution differs from a Fourier series only in the $\sin n\pi x$ term, the so-called *shape factor*, which is a function of x alone, and required due to the boundary condition,

$$y(0,t) = y(L,t) = 0, \forall t.$$

Even though Bernoulli did not give a calculation for the harmonic amplitudes, A, B,…, he claimed his solution to be the most general. However, presaging Fourier's mixed reception

---

[4] Famous for Taylor series and integration by parts, Taylor also invented the *calculus of finite differences*, used to construct difference equations of Taylor series coefficients important in the numerical solution of differential equations

by the scientific community some fifty years later (where Fourier first presented *his* version of this series), Euler and Lagrange both objected to Bernoulli's claim of generality on the grounds that acceptance of it would lead to the doubtful conclusion that Bernoulli's trigonometric series could represent an arbitrary function. What they did not realize was that Bernoulli was *correct* if we define the problem domain to be finite, in this case L, and that it does not matter to the problem how the series behaves elsewhere.

Thus, in 1804–05, when two applied mathematicians, one German and the other French, embraced trigonometric series whole-heartedly as a natural, and indeed the simplest solution to a number of problems, most people still regarded trigonometric series as risky, and wanted to run the other way at their mere mention. These two pioneers, respectively, were Carl (Friedrich) Gauss (who used finite trigonometric series in his search for a more efficient interpolation method for asteroidal orbits), and (Jean Baptiste) Joseph Fourier (whose theory of heat diffusion in solid bodies relied on infinite trigonometric series, but who also used finite trigonometric series in his experimental verification of his theory).

Heinrich (Friedrich Karl Ludwig) Burkhardt (1904, p. 650; Fr. trans. 1912, p. 91) in his review of trigonometric interpolation methods, mentions that both Gauss and Fourier obtained a DFT-like formula (a trigonometric series of harmonically-related terms in which the number of equations is equal to the number of unknowns). Although these efforts were successful, no proof was given that the resulting trigonometric coefficients were in any way optimal. It was Gauss's student, (Friedrich) Wilhelm Bessel, who in his desire to interpolate empirical data gleaned from equally-spaced telescopic observations of periodic astronomical phenomena, first treated the completely general problem of determining the coefficients in the trigonometric interpolation formula, equivalent to the modern-day DFT.

Prior to Bessel's analysis, workers in this field had assumed that one could just make use of a truncated (finite) Fourier series and use some sort of numerical approximation to the coefficient integrals. The problem with this approach is that whereas the Fourier series uses continuous time, and gives exact frequencies, the DFT is discrete in frequency and time, and generally gives only approximate frequencies (exact frequencies requires the sample spacing to be commensurate with the period). The DFT and the finite Fourier series may both give errors in amplitude due to the finite number of trigonometric terms in the former case, and truncation of the Fourier series in the latter. It is, however, one of those neat tie-ins between different areas of mathematics that the "best" values of the coefficients in trigonometric interpolation lead exactly to the DFT, and to the conclusion that all Fourier methods are optimal in a true statistical sense. It is to Bessel's enduring credit, that he was able to show that this is so. His proof relies on the principle of least squares, which was the statistical method of choice way back then, just as it is today.

Bessel's formula

Bessel, in a preface to his first volume of astronomical measurements taken at the Royal University Observatory in Königsberg (Bessel 1815, IX-X; *also see* 1816, VIII-IX), spent several pages applying the principle of least squares to optimize a trigonometric interpolation calculation. Bessel aimed to minimize errors in his interpolation calculations by determining which values of the unknowns "are the most probable" (*sind die wahrscheinlichsten*). The

unknowns referred to are the weights of the harmonic terms in the trigonometric polynomial that interpolates the data. Effectively, Bessel was computing a finite Fourier series of discrete data points, using a calculation identical to the modern discrete Fourier transform. The interesting result is that the trigonometric weights derived by Bessel using the statistical approach of the least squares method, are essentially identical to those assumed by Joseph Fourier for his infinite Fourier series, using a methodology that was a lot more arbitrary. Most likely Bessel proceeded with this work on the encouragement of his mentor, Carl (Friedrich) Gauss, who had invented the least squares method possibly as early as 1795, using it to calculate the orbit of asteroid Ceres in 1801.

Bessel's investigation proceeded more or less as follows: As was already well known, a suitably well-behaved function $y(t)$, with fundamental period T, such that $y(t) = y(t+T)\ \forall t$, can be expanded as an infinite Fourier series,

$$y(t) = \frac{A_0}{2} + \sum_{n=1}^{\infty}\left(A_n \cos\frac{2\pi nt}{T} + B_n \sin\frac{2\pi nt}{T}\right), \tag{2.1}$$

where the Fourier coefficients, A_n, B_n are given by the well-known integrals. Bessel wished to find a finite approximation $\hat{y}(t)$ which gives the best possible fit to $y(t)$,

$$\hat{y}(t) = \frac{a_0}{2} + \sum_{n=1}^{m}\left(a_n \cos\frac{2\pi nt}{T} + b_n \sin\frac{2\pi nt}{T}\right) + \varepsilon_m(t). \tag{2.2}$$

where the error term $\varepsilon_m(t)$ is the difference between $y(t)$, and $\hat{y}(t)$. Note the change of case since we have not yet established the degree to which Bessel's coefficients approximate Fourier's.

Bessel then sampled $\hat{y}(t)$ by dividing its period into N equal parts,[5] such that $T = N\Delta t$, where Δt is the grid spacing, and $t = k\Delta t$. This gives a system of N equations with $2m+1$ unknowns[6], such that

$$\hat{y}(k\Delta t) = \frac{a_0}{2} + \sum_{n=1}^{m}\left(a_n \cos\frac{2\pi nk}{N} + b_n \sin\frac{2\pi nk}{N}\right) + \varepsilon_m(k\Delta t),\quad k = 0,1,2,\cdots,N-1 \tag{2.3}$$

---

[5] In this treatment, N is odd. The even case is slightly more complicated, and will not be discussed here.

[6] Although Gauss and Fourier only considered the case where the number of equations is *equal* to the number of unknowns (Burkhardt 1904, 650; 1912, 91), Bessel also discussed the case where the number of equations is greater than the number of unknowns (Bessel 1815, p. X). Based on his application of the method of least squares to finding the most probable values of the coefficients, Bessel clearly understood that the number of equations must be *equal to or larger* than the number of unknowns; the larger the better. An elegant paper by Charles H. Lees (1914) uses the least squares method to show that if the errors of observation of a periodic function are normally distributed, then in the limit as the number of observations becomes very large, the discrete Fourier transform (DFT) of the function becomes identical with the Fourier series representation.

where n is called the harmonic number. For a given function $\hat{y}(t)$, and set of coefficients $\{a_0, a_1, \cdots, a_m; b_1, \cdots, b_m\}$, the accuracy of the interpolation depends only on N and m, in other words on the grid spacing (smaller is better, corresponding to higher N), and on the number of harmonics to be used in the approximation (higher is better). Rearranging (2.3), we obtain the error term as

$$\varepsilon_m(k\Delta t) = \hat{y}(k\Delta t) - \left[\frac{a_0}{2} + \sum_{n=1}^{m}\left(a_n \cos\frac{2\pi nk}{N} + b_n \sin\frac{2\pi nk}{N}\right)\right]. \tag{2.4}$$

Bessel's goal was to minimize the discrete least squares error through suitable choice of coefficients $a_0, a_n,$ and b_n.

Squaring equation (2.4) and summing over the fundamental period of $\hat{y}(k\Delta t)$, we obtain an expression for the *discrete square error* of the finite Fourier series approximation,

$$E_m = \sum_{k=0}^{N-1}\left[\hat{y}_k - \frac{a_0}{2} - \sum_{n=1}^{m}\left(a_n \cos\frac{2\pi nk}{N} + b_n \sin\frac{2\pi nk}{N}\right)\right]^2, \tag{2.5}$$

where the sampled function $\hat{y}(k\Delta t)$ is written as the discrete sequence $\{\hat{y}_k\}$. Applying the least-squares criterion of minimizing the sum of the squares of the differences, if the errors in the values of $\hat{y}_k$ are normally distributed, this will yield the most probable values of the coefficients a_0, a_n, and b_n. Setting each of the first partial derivatives of E_m with respect to each of the coefficients to zero,

$$\left.\begin{aligned} &\frac{\partial E_m}{\partial a_n} = 0 \qquad n = 0,1,2,\cdots,m, \\ \text{and} \quad &\frac{\partial E_m}{\partial b_n} = 0 \qquad n = 1,2,\cdots,m, \end{aligned}\right\} \tag{2.6}$$

we can interchange the order of differentiation and summation, to obtain a system of $2m+1 = N$ linear equations, known as the *normal equations*, which are to be solved for N unknowns, the coefficients a_0, a_n, and b_n, as follows:

$$\left.\begin{aligned} &\frac{\partial E_m}{\partial a_0} = 0 = -2\sum_{k=0}^{N-1}\left\{y_k - \frac{a_0}{2}\right\}\cos\frac{2\pi nk}{N} && n = 0 \\ &\frac{\partial E_m}{\partial a_n} = 0 = -2\sum_{k=0}^{N-1}\left\{y_k - \left(a_n \cos\frac{2\pi nk}{N} + b_n \sin\frac{2\pi nk}{N}\right)\right\}\cos\frac{2\pi nk}{N} && n = 1,2,\cdots,m \\ \text{and} \quad &\frac{\partial E_m}{\partial b_n} = 0 = -2\sum_{k=0}^{N-1}\left\{y_k - \left(a_n \cos\frac{2\pi nk}{N} + b_n \sin\frac{2\pi nk}{N}\right)\right\}\sin\frac{2\pi nk}{N} && n = 1,2,\cdots,m \end{aligned}\right\} \tag{2.7}$$

Applying the orthogonality properties of sine and cosine, these summations greatly simplify, as indicated,

$$\left.\begin{aligned}
\underbrace{2\sum_{k=0}^{N-1}\frac{a_0}{2}}_{=a_0N} &= 2\sum_{k=0}^{N-1} y_k \cos\frac{2\pi nk}{N} && n=0\\
\underbrace{2\sum_{k=0}^{N-1} a_n \cos^2\frac{2\pi nk}{N}}_{=a_nN} + \underbrace{2\sum_{k=0}^{N-1} b_n \sin\frac{2\pi nk}{N}\cos\frac{2\pi nk}{N}}_{=0} &= 2\sum_{k=0}^{N-1} y_k \cos\frac{2\pi nk}{N} && n=1,2,\cdots,m\\
\underbrace{2\sum_{k=0}^{N-1} a_n \cos\frac{2\pi nk}{N}\sin\frac{2\pi nk}{N}}_{=0} + \underbrace{2\sum_{k=0}^{N-1} b_n \sin^2\frac{2\pi nk}{N}}_{=b_nN} &= 2\sum_{k=0}^{N-1} y_k \sin\frac{2\pi nk}{N} && n=1,2,\cdots,m
\end{aligned}\right\} \quad (2.8)$$

finally yielding Bessel's trigonometric interpolation formulae,

$$\left.\begin{aligned}
a_n &= \frac{2}{N}\sum_{k=0}^{N-1} y_k \cos\frac{2\pi nk}{N} && n=0,1,2,\cdots,m\\
\text{and}\qquad b_n &= \frac{2}{N}\sum_{k-0}^{N-1} y_k \sin\frac{2\pi nk}{N} && n=1,2,\cdots,m
\end{aligned}\right\} \quad (2.9)$$

where $m = N/2$. Unlike Fourier coefficients $A_n, B_n,\ n=0,1,2,\cdots,$ Bessel's coefficients, $a_n, b_n,$ repeat (with a change of sign for the $b_n,$ for $n > m$. Note that coefficients $a_n, b_n,$ are independent of *m,* depending only on the sampling grid and *N*. This is a very important result. Therefore, for a given *N*, we select from the same set of coefficients irrespective of whether we wish to calculate 3 DFT terms or 33.

Aside from the 2/*N* scaling factor, the equations in (2.9) exactly define the complex DFT sequence $\{Y_n = c_n = \frac{1}{2}(a_n + jb_n) : n = 0,\ldots,N-1\}$ of a length-*N,* real data sequence $\{y_k\}$. If the y_k are equidistant samples of $y(t)$, a suitably band-limited periodic function (no harmonic periods less than twice the sampling interval), Bessel's interpolation formula provides a useful approximation to the Fourier series, and therefore to the Fourier transform itself. Bessel's approximation becomes exact for the special case of a sampled data sequence length commensurate with the natural period of the phenomenon being analyzed. However, since the natural period of the function is often known beforehand, it is easy to arrange for this latter condition. The limited bandwidth requirement is not so easily met, and to the degree that $y(t)$ is *not* bandlimited, produces aliasing errors in the DFT.

Relationship between the DFT and Fourier series

A suitably well-behaved periodic function $y(t)$, with fundamental period *T*, has a Fourier series expansion,

$$y(t) = \sum_{n=-\infty}^{\infty} c_n e^{j2\pi nt/T}, \tag{2.10}$$

where $\{c_n\}$ is an infinite set of complex Fourier coefficients given by

$$c_n = \frac{1}{T}\int_T y(t)\, e^{-j2\pi nt/T} dt \qquad n = 0, \pm 1, \pm 2, \pm 3, \cdots. \tag{2.11}$$

Starting at $t = 0$, if we sample $y(t)$ using a grid spacing of $\Delta t = T / N$, this gives N equally spaced sample points in $[0, T)$. Denoting the k^{th} sample $y(k\Delta t)$, or simply y_k, equation (2.10) becomes

$$\begin{aligned} y_k &= \sum_{n=-\infty}^{\infty} c_n e^{j2\pi nk/N} \\ &= \sum_{m=-\infty}^{\infty} \sum_{n=mN}^{mN+N-1} c_n e^{j2\pi nk/N} \\ &= \sum_{m=-\infty}^{\infty} \sum_{n=0}^{N-1} c_{n+mN} e^{j2\pi k(n+mN)/N} \\ &= \sum_{n=0}^{N-1} \left[\sum_{m=-\infty}^{\infty} c_{n+mN} \right] e^{j2\pi nk/N} \\ &= \sum_{n=0}^{N-1} \tilde{c}_n e^{j2\pi nk/N} \qquad k = 0, 1, \ldots, N-1, \end{aligned} \tag{2.12}$$

where

$$\tilde{c}_n = \sum_{m=-\infty}^{\infty} c_{n+mN}\,. \tag{2.13}$$

Moreover,

$$\tilde{c}_{N-n} = \sum_{m=-\infty}^{\infty} c_{-n+mN}\,. \tag{2.14}$$

Apart from a scaling factor of N, equation (2.12) is in the form of an inverse discrete Fourier transform (IDFT). It follows that sequences $\{y_k : k = 0, 1, \ldots, N-1\}$ and $\{N\tilde{c}_n : n = 0, 1, \ldots, N-1\}$ are a discrete Fourier transform pair.

In practical situations, the $\tilde{c}_n$ will differ from the ideal Fourier series coefficients due to *aliasing* error, as a result of analyzing a sampled version of $y(t)$, rather than $y(t)$ itself. Aliasing occurs whenever there are significant contributions to the sums in (2.13) and (2.14) from terms with $m \neq 0$, due to image band overlap.

Essentially, therefore, discrete Fourier transforms are Fourier series with aliasing. Moreover, if we choose N so that $c_n \doteq 0$ (i.e., c_n nearly equal to 0) when $|n| \geq N/2$, it follows from equations (2.12) through (2.14) that

$$\tilde{c}_n \doteq c_n \quad \text{and} \quad \tilde{c}_{N-n} \doteq c_{-n} \qquad n = 0,1,\cdots,N/2.$$

By way of expanding on this topic a little further, considering that the Fourier series expansion of a suitably behaved periodic function, $y(t) = y(t+T),\ \forall t$, is an infinite harmonic sum, it has no defined upper frequency limit. Although the DFT sequence of the same function is also the result of a harmonic sum, albeit finite, because of discrete sampling in both the time and frequency domains, unless we take sufficient care, DFTs will often behave quite differently from Fourier series. Generally a result of *aliasing* error, this will always occur unless $y(t)$ is band-limited prior to sampling. The precise statement is that aliasing will occur unless $Y(f) = 0 : f > f_s/2$, where f_s is the sampling frequency. The equivalent statement in Fourier series terminology is that series coefficients c_n must be zero, or practically zero, for $n > N/2$. In the context of baseband analog signals, this is achieved with an *anti-aliasing lowpass filter,* applied prior to sampling (or following analog reconstruction). In practical filters, the choice of f_s must be balanced against the complexity of the filter structure (which governs roll-off rate), and the available sample-processing speed.

Some other DFT errors (none of which will be discussed here) are *leakage* (a type of aliasing that occurs when the data period is not commensurate with the analysis period); the *picket fence effect* (due to the frequency response of the individual DFT filters, noticed when the DFT frequency grid does not line up with the harmonic components of the data); and a type of high frequency roll off called $\sin x / x$ aperture error, which convolves the rectangular zero-order hold function with the analog input and output signal in sampled data systems.

Some early efforts at improving DFT efficiency

Even though it was several decades before Fourier's original 1807 thesis regarding arbitrary functions (Fourier, 1807) was accepted as an established mathematical fact, Fourier's empirical investigations into the nature of heat conduction (Fourier, 1822; Eng. trans. 1878) supported his mathematics and helped establish the validity of infinite trigonometric series as an analytical tool. Thus, the 19th century saw a period of intense research into climatic cycles, terrestrial magnetism, and the prediction of ocean tides. Due to the large number calculations $O(N^2)$ required in a typical harmonic analysis, algorithmic efficiency was a large concern. An early example of an improved algorithm for computing the DFT was published in a manual of meteorology written by Ludwig Friedrich Kämtz (1831). Kämtz's DFT algorithm computes the mean and three harmonic terms for a real data sequence of length $N = 24$, and is given in Table 1. Kämtz's algorithm gains efficiency through a process of thrice folding the data (dashed lines), followed by taking sums and differences at each stage. Kämtz's work sheet looked like this:

Data: $\{\ x_0\ x_1\ x_2\ x_3\ \cdots\ x_9\ x_{10}\ x_{11}\ |\ x_{12}\ x_{13}\ x_{14}\ x_{15}\ \cdots\ x_{23}$

1st fold: $\left\{\begin{array}{ccccccc|ccccccc} x_0 & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 & x_9 & x_{10} & x_{11} & \\ & x_{23} & x_{22} & x_{21} & x_{20} & x_{19} & x_{18} & x_{17} & x_{16} & x_{15} & x_{14} & x_{13} & x_{12} \end{array}\right.$

2nd fold: $\left\{\begin{array}{cccc|ccc} x_0 & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\ & x_{11} & x_{10} & x_9 & x_8 & x_7 & \\ x_{12} & x_{13} & x_{14} & x_{15} & x_{16} & x_{17} & \\ & x_{23} & x_{22} & x_{21} & x_{20} & x_{19} & x_{18} \end{array}\right.$

3rd fold: $\left\{\begin{array}{ccc|c} x_0 & x_1 & x_2 & x_3 \\ x_6 & x_5 & x_4 & \\ & x_7 & x_8 & \\ & x_{11} & x_{10} & x_9 \\ x_{12} & x_{13} & x_{14} & x_{15} \\ & x_{17} & x_{16} & \\ x_{18} & x_{19} & x_{20} & \\ & x_{23} & x_{22} & x_{21} \end{array}\right.$

The resulting expressions in Kämtz's DFT algorithm such as $(x_1 - x_{11} - x_{13} + x_{23}).\cos u$ and $(x_2 - x_4 - x_8 + x_{10} + x_{14} - x_{16} - x_{20} + x_{22}).\cos 4u$ (marked by pairs of solid vertical lines) significantly reduce the number of multiplications. In all, if we ignore the 1/N scaling factor, Kämtz's method has only 16 multiplications. This compares to 3×12 = 36 multiplications that would be required if we did not fold the data, and instead computed the same first three terms of the DFT, the mean and two harmonic terms (X_0, X_1, X_3), using a straightforward sum of products. Although Kämtz's method requires 137 additions (he did not factor out redundant additions), the more time-consuming part of the computation, namely multiplication, is reduced by 55%. The Kämtz DFT algorithm is discussed further in chapter 3.

By the start of the 20th century, reduced arithmetic DFT algorithms that use data folding to exploit the symmetry properties of the sin and cosine basis functions in (2.9), reached a plateau of perfection, as exemplified by the $N = 12$ and $N = 24$ algorithms given by German meteorologist Julius von Hann (1901), and the $N = 4m$ algorithms derived by German mathematician Carl (David Tolmé) Runge (1903). These algorithms saved arithmetic by being much more highly factored than Kämtz's. For example, Runge's $N = 12$ algorithm required only half the number of multiplications and a quarter of the additions of the Kämtz algorithm,

yet computed 2½ times as many harmonic terms. Interestingly, Hann's $N = 12$ algorithm, which computed two harmonic terms, (albeit in a slightly expanded form to compute several more harmonic terms), remained part of the meteorologist's toolbox until the advent of high-speed electronic computers in the early 1950s (*see* Brooks and Carruthers 1953, p. 344). These latter algorithms are presented in algebraic form in Tables 2 and 3, and in matrix form in appendices A and B.

The above-mentioned DFT algorithms represent only a small sampling of the work done by many generations of applied mathematicians and scientists throughout the 1800s and the early 1900s. Burkhardt's trigonometric interpolation review article in *Encyclopäedie der Mathematischen Wissenschaften* (1904 pp. 685-693; updated in the Fr. Trans. 1912, pp. 142-153) lists more than 70 algorithms (for $N = 2$, 4, 6, 8, 9, 10, 12, 15, 16, 18, 24, 30, 32, 36, 40, 52, 64, 72, 73, $4m$) and 40 authors more or less evenly spread over the years 1828 to 1911.[7] The large body of work cited by Burkhardt includes several early fast Fourier transform (FFT) algorithms, and as we will see in the following chapters, remarkably even includes transforms that are structurally similar to many of today's small-N DFTs and the Winograd Fourier transform algorithm (WFTA).

An often-heard modern opinion is that efficient DFT factorizations for $N > 3$ are hard to find without a systematic method (*see*, for example, Elliot and Rao, 1982). Therefore, it is remarkable that before the modern developmental period (which is characterized by algorithms, such as the WFTA, that leverage advanced number theoretic concepts), more than a few small-N (and not-so-small-N) reduced-arithmetic DFT algorithms having a similar structure were developed using nothing more than a few trigonometric identities and simple algebra. That the these algorithms, old and new, have more similarities than differences, is testament to the fact that all are just factorizations of the DFT operator, and do the same job, often in similar way, irrespective of the method of derivation.

[7] Burkhardt also lists a number of graphical methods, and several machines including the very famous Michelson-Stratton harmonic analyzer

Table 1
Real-Input DFT Algorithm for $N = 24$
Kämtz (1831)

$$X_n = \frac{1}{24}\sum_{k=0}^{23} x_k W_{24}^{nk} \quad n = 0,1,2,3 \qquad \text{where } W_{24} = e^{-j\frac{2\pi}{24}}$$

$$X_0 = (x_0 + x_1 + \ldots + x_{23})/24 \qquad u = \frac{2\pi}{24}$$

$$X_1 = (2/24)\left\{\begin{array}{r}[\ (x_1 - x_{11} - x_{13} + x_{23}).\cos u \\ +(x_2 - x_{10} - x_{14} + x_{22}).\cos 2u \\ +(x_3 - x_9 - x_{15} + x_{21}).\cos 3u \\ +(x_4 - x_8 - x_{16} + x_{20}).\cos 4u \\ +(x_5 - x_7 - x_{17} + x_{19}).\cos 5u \\ +(x_0 - x_{12})] \\ +j.[\ (x_1 + x_{11} - x_{13} - x_{23}).\sin u \\ +(x_2 + x_{10} - x_{14} - x_{22}).\sin 2u \\ +(x_3 + x_9 - x_{15} - x_{21}).\sin 3u \\ +(x_4 + x_8 - x_{16} - x_{20}).\sin 4u \\ +(x_5 + x_7 - x_{17} - x_{19}).\sin 5u \\ +(x_6 - x_{18})]\end{array}\right\}$$

$$X_2 = (2/24)\left\{\begin{array}{r}[(x_1 - x_5 - x_7 + x_{11} + x_{13} - x_{17} - x_{19} + x_{23}).\cos 2u \\ +(x_2 - x_4 - x_8 + x_{10} + x_{14} - x_{16} - x_{20} + x_{22}).\cos 4u \\ +(x_0 - x_6 + x_{12} - x_{18})] \\ +j.[(x_1 + x_5 - x_7 - x_{11} + x_{13} + x_{17} - x_{19} - x_{23}).\sin 2u \\ +(x_2 + x_4 - x_8 - x_{10} + x_{14} + x_{16} - x_{20} - x_{22}).\sin 4u \\ +(x_3 - x_9 + x_{15} - x_{21})]\end{array}\right\}$$

$$X_3 = (2/24)\left\{\begin{array}{r}[(x_1 - x_3 - x_5 + x_7 + x_9 - x_{11} - x_{13} + x_{15} + x_{17} - x_{19} - x_{21} + x_{23}).\cos 3u \\ +(x_0 - x_4 + x_8 - x_{12} + x_{16} - x_{20})] \\ +j.[(x_1 + x_3 - x_5 - x_7 + x_9 + x_{11} - x_{13} - x_{15} + x_{17} + x_{19} - x_{21} - x_{23}).\sin 3u \\ +(x_2 - x_6 + x_{10} - x_{14} + x_{18} - x_{22})]\end{array}\right\}$$

16 Multiplications (omitting scaling factors), 137 Additions

Table 2
Real-Input DFT Algorithm for $N = 12$
Hann (1901), Brooks and Carruthers (1953)

Hann: X_0, X_1, X_2 (X_0 by mean of data)
Brookes & Carruthers: X_1, X_2, X_3, X_4, X_5

$$X_n = \frac{1}{12}\sum_{k=0}^{11} x_k W_{12}^{nk} \qquad n = 0,1,2,3,4,5 \qquad \text{where } W_{12} = e^{-j\frac{2\pi}{12}}$$

$s_1 = x_0+x_6$ $s_2 = x_0-x_6$ $s_3 = x_1+x_7$ $s_{22} = s_1+s_7$
$s_4 = x_1-x_7$ $s_5 = x_2+x_8$ $s_6 = x_2-x_8$ $s_{23} = s_3+s_9$
$s_7 = x_3+x_9$ $s_8 = x_3-x_9$ $s_9 = x_4+x_{10}$ $s_{24} = s_5+s_{11}$
$s_{10} = x_4-x_{10}$ $s_{11} = x_5+x_{11}$ $s_{12} = x_5-x_{11}$ $s_{25} = s_{23}+s_{24}$
$s_{13} = s_4+s_{12}$ $s_{14} = s_4-s_{12}$ $s_{15} = s_6+s_{10}$ $s_{26} = s_{23}-s_{24}$
$s_{16} = s_6-s_{10}$ $s_{17} = s_1-s_7$ $s_{18} = s_3-s_9$ $s_{27} = s_2-s_{16}$
$s_{19} = s_5-s_{11}$ $s_{20} = s_{18}+s_{19}$ $s_{21} = s_{18}-s_{19}$ $s_{28} = s_{13}-s_8$

$m_1 = j.s_8$ $m_2 = \frac{\sqrt{3}}{2}.s_{14}$ $m_3 = j\frac{\sqrt{3}}{2}.s_{15}$

$m_4 = j\frac{\sqrt{3}}{2}.s_{20}$ $m_5 = ½.s_{16}$ $m_8 = j.s_{28}$

$m_6 = j\ ½.s_{13}$ $m_7 = ½.s_{21}$ $m_9 = j\frac{\sqrt{3}}{2}.s_{26}$ $m_{10} = ½.s_{25}$

$s_{29} = s_2+m_5$ $s_{30} = m_1+m_6$ $s_{36} = s_{22}-m_{10}$ $s_{37} = s_{29}-m_2$
$s_{31} = s_{17}+m_7$ $s_{32} = s_{29}+m_2$ $s_{38} = s_{30}-m_3$ $s_{39} = s_{27}+m_8$
$s_{33} = s_{30}+m_3$ $s_{34} = s_{32}+s_{33}$ $s_{40} = s_{36}+m_9$ $s_{41} = s_{37}+s_{38}$
$s_{35} = s_{31}+m_4$

$X_0 = (x_0 + x_1 + \ldots + x_{11})/12$
$X_1 = s_{34}/6$ $X_2 = s_{35}/6$ $X_3 = s_{39}/6$ $X_4 = s_{40}/6$ $X_5 = s_{41}/6$

Hann (1901) X_0, X_1, X_2: 3 Multiplications, 36 Additions, 3 Shifts
Brooks and Carruthers (1953): 4 Multiplications, 47 Additions, 4 Shifts

Note: Sums $s_{34}, s_{35}, s_{39}, s_{40}, s_{41}$ are not included in the additions total because in each case the sum is composed of a real term and a pure imaginary term. Multiplications by 1 or j are not included in the multiplication total. Multiplication by ½ is counted as an arithmetic right shift. To simplify comparison with modern DFT algorithms, scale factors are also not included in the multiplication total. Even though Hann and Brooks & Carruthers could have halved the number of additions in calculating X_0 (using $12X_0 = s_1 + s_3 + s_5 + s_9 + s_7 + s_{11}$), published examples show that they preferred to simply sum the raw data.

Table 3
Real-Input DFT Algorithm for $N = 12$
Runge (1903)

$$X_n = \sum_{k=0}^{11} x_k W_{12}^{nk} \qquad n = 0,1,2,3,4,5,6 \qquad \text{where } W_{12} = e^{-j\frac{2\pi}{12}}$$

$$\begin{array}{llll}
s_1 = x_0+x_6 & s_2 = x_0-x_6 & s_3 = x_1+x_{11} & s_4 = x_1-x_{11} \\
s_5 = x_2+x_{10} & s_6 = x_2-x_{10} & s_7 = x_3+x_9 & s_8 = x_3-x_9 \\
s_9 = x_4+x_8 & s_{10} = x_4-x_8 & s_{11} = x_5+x_7 & s_{12} = x_5-x_7 \\
s_{13} = s_3+s_{11} & s_{14} = s_3-s_{11} & s_{15} = s_4+s_{12} & s_{16} = s_4-s_{12} \\
s_{17} = s_5+s_9 & s_{18} = s_5-s_9 & s_{19} = s_6+s_{10} & s_{20} = s_6-s_{16} \\
s_{21} = s_1+s_{17} & s_{22} = s_2-s_{18} & s_{23} = s_{13}+s_7 & s_{24} = s_{15}-s_8
\end{array}$$

$$\begin{array}{lllll}
m_1 = j\,½.s_{15} & m_2 = j\frac{\sqrt{3}}{2}.s_{19} & m_3 = j.s_8 & m_4 = j\frac{\sqrt{3}}{2}.s_{16} & m_5 = j\frac{\sqrt{3}}{2}.s_{20} \\
m_6 = j.s_{24} & m_7 = ½\,.s_{18} & m_8 = \frac{\sqrt{3}}{2}.s_{14} & m_{10} = ½\,.s_{17} & m_{11} = ½\,.s_{13}
\end{array}$$

$$\begin{array}{llll}
s_{25} = m_1+m_3 & s_{26} = m_7+s_2 & s_{27} = s_1-m_{10} & s_{28} = m_{11}-s_7 \\
s_{29} = s_{25}+m_2 & s_{30} = s_{25}-m_2 & s_{31} = m_4+m_5 & s_{32} = m_4-m_5 \\
s_{33} = s_{26}+m_8 & s_{34} = s_{26}-m_8 & s_{35} = s_{27}+s_{28} & s_{36} = s_{27}-s_{28} \\
s_{37} = s_{21}+s_{23} & s_{38} = s_{21}-s_{23} & s_{39} = s_{33}+s_{29} & s_{40} = s_{35}+s_{31} \\
s_{41} = s_{22}+m_6 & s_{42} = s_{36}+s_{32} & s_{43} = s_{34}+s_{30} &
\end{array}$$

$$\begin{array}{llll}
X_0 = s_{37} & X_1 = s_{39} & X_2 = s_{40} & X_3 = s_{41} \\
X_4 = s_{42} & X_5 = s_{43} & X_6 = s_{38} &
\end{array}$$

4 Multiplications, 38 Additions, 4 Shifts.

Note: The last five sums, s_{39} through s_{43}, are not included in the additions total because in each case the sum is composed of a real term and a pure imaginary term. Multiplications by 1 or j are not included in the multiplication total. Multiplication by ½ is counted as an arithmetic right shift.

3. The modern developmental period

Small-N DFT algorithms became the topic of intense research in the 1970s. The stimulus was an epoch-making paper by C.M. Rader (1968), which showed how a DFT computation can be changed into cyclic convolution when N is prime. For example, consider the 7-point DFT,

$$X_n = \sum_{k=0}^{6} x_k W_7^{nk}, \quad n = 0,1,2,\cdots,6 \tag{3.1}$$

where $W_7 = e^{-j2\pi/7}$ is the reciprocal of a primitive 7th root of unity, and the generator of a cyclic group, and where a scaling factor 1/7 has been ignored for convenience. This allows equation (3.1) to be expressed in matrix form, $\mathbf{X} = \mathbf{W}\,\mathbf{x}$, where $\mathbf{W} = \left[w_{n,k}\right]_{N\times N} = W_N^{nk \bmod(N)}$,

$$\begin{pmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \\ X_6 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & W^1 & W^2 & W^3 & W^4 & W^5 & W^6 \\ 1 & W^2 & W^4 & W^6 & W^1 & W^3 & W^5 \\ 1 & W^3 & W^6 & W^2 & W^5 & W^1 & W^4 \\ 1 & W^4 & W^1 & W^5 & W^2 & W^6 & W^3 \\ 1 & W^5 & W^3 & W^1 & W^6 & W^4 & W^2 \\ 1 & W^6 & W^5 & W^4 & W^3 & W^2 & W^1 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{pmatrix} \tag{3.2}$$

To convert this equation into cyclic convolution X_0 must be calculated separately, as $X_0 = x_0 + x_1 + x_2 + x_3 + x_4 + x_5 + x_6$. We then apply a suitable permutation to the remaining indices, using elementary row and column operations. Exchanging row 2 with row 3, row 6 with rows 4 and 5, column 2 with column 3, and column 6 with columns 4 and 5, equation (3.2) can be rewritten as,

$$\begin{pmatrix} X_1 \\ X_3 \\ X_2 \\ X_6 \\ X_4 \\ X_5 \end{pmatrix} = \begin{pmatrix} X_0 \\ X_0 \\ X_0 \\ X_0 \\ X_0 \\ X_0 \end{pmatrix} + \begin{pmatrix} W^1 & W^3 & W^2 & W^6 & W^4 & W^5 \\ W^3 & W^2 & W^6 & W^4 & W^5 & W^1 \\ W^2 & W^6 & W^4 & W^5 & W^1 & W^3 \\ W^6 & W^4 & W^5 & W^1 & W^3 & W^2 \\ W^4 & W^5 & W^1 & W^3 & W^2 & W^6 \\ W^5 & W^1 & W^3 & W^2 & W^6 & W^4 \end{pmatrix} \begin{pmatrix} x_1 \\ x_3 \\ x_2 \\ x_6 \\ x_4 \\ x_5 \end{pmatrix}, \tag{3.3}$$

which apart from addition by the $\mathbf{X}_0$ column vector, is length-6 cyclic convolution. Winograd (1976, 1978) extended Rader's index permutation method for prime-length DFTs to prime-power lengths, and used computational complexity theory to show that the minimum number of multiplications for N-point cyclic convolution is $2N - K$, where K is the number of irreducible factors of N. Agawal and Cooley (1977), and Winograd (1978) give several convolution algorithms that achieve or come close to achieving this minimum for small N.

Various methods for synthesizing such algorithms for small N are reviewed by McClellan and Rader (1979, p. 61-71).

Small-N DFT algorithms based on minimum multiplication cyclic convolution are given by Winograd (1978), McClellan and Rader (1979), Elliot and Rao (1982), and Morgera and Krishna (1989). A characteristic feature of these DFT algorithms is a nested (+)(×)(+) arithmetic structure (*see*, for example, Table 4 which shows Winograd's algorithm[8] for $N = 8$). However, many if not most of the early algorithms, including those described by Hann (1901) and Runge (1903) (*see* Tables 2 and 3), have this same nested structure, and have similar flow graphs and matrix representations. On the other hand, Kämtz's DFT algorithm (Kämtz , 1831) published 70 years earlier (*see* Table 1), is not as completely factored, as suggested by its (+)(×) structure, which lacks the final (+) stage.

These ideas suggest that it should be possible to place much of the earlier work, and the more recent cyclic convolution approach, into the same theoretical context, perhaps shedding a bit more light on the discrete Fourier transformation in the process. Although the nested (+)(×)(+) arithmetic structure is widely associated with the small-N fast cyclic convolution DFT algorithms of Winograd, this structure is now seen to be far more basic than the particular formalism used to derive an efficient DFT algorithm in the first place. Clearly, this nested DFT structure predates the modern period.

It is worth noting that the Winograd Fourier transform algorithm (WFTA), which represents a generalization of small-N DFT algorithms to larger N, also has this nested (+)(×)(+) structure. The WFTA is restricted to N prime or a prime power, including transforms "built-up" from smaller prime and prime-power transforms, but this is a consequence of the method of derivation and does not affect the structural properties of the DFT itself.

[8] This and the other Winograd DFT algorithms presented here have j replaced by $-j$ in $W_N = e^{-j2\pi/N}$, which is more standard, especially in signal processing.

Table 4
Real-Input DFT Algorithm for $N = 8$ using Circular Convolution
Winograd (1978)

$$X_n = \sum_{k=0}^{8} x_k W_8^{nk} \qquad n = 0,1,2,\cdots,8 \qquad \text{where } W_8 = e^{-j\frac{2\pi}{8}}$$

$s_1 = x_0+x_4$ $s_2 = x_0-x_4$ $s_3 = x_2+x_6$ $s_4 = x_2-x_6$

$s_5 = x_1+x_5$ $s_6 = x_1-x_5$ $s_7 = x_3+x_7$ $s_8 = x_3-x_7$

$s_9 = s_1+s_3$ $s_{10} = s_1-s_3$ $s_{11} = s_5+s_7$ $s_{12} = s_5-s_7$

$s_{13} = s_9+s_{11}$ $s_{14} = s_9-s_{11}$ $s_{15} = s_6+s_8$ $s_{16} = s_6-s_8$

$m_1 = 1.s_{13}$ $m_2 = 1.s_{14}$ $m_3 = 1.s_{10}$ $m_4 = j \sin 2u.s_{12}$ $u = \frac{2\pi}{8}$

$m_5 = 1.s_2$ $m_6 = j \sin 2u.s_4$ $m_7 = j \sin u.s_{15}$ $m_4 = \cos u.s_{16}$

$s_{17} = m_3+m_4$ $s_{18} = m_3-m_4$ $s_{19} = m_5+m_8$ $s_{20} = m_5-m_8$

$s_{21} = m_6+m_7$ $s_{22} = m_6-m_7$ $s_{23} = s_{19}+s_{21}$ $s_{24} = s_{19}-s_{21}$

$s_{25} = s_{20}+s_{22}$ $s_{26} = s_{20}-s_{22}$

$X_0 = m_1$ $X_1 = s_{23}$ $X_2 = s_{17}$ $X_3 = s_{26}$

$X_4 = m_2$ $X_5 = s_{25}$ $X_6 = s_{18}$ $X_7 = s_{24}$

2 Multiplications, 26 Additions

Note: Multiplications by 1 or j are not included in the multiplication total.

4. Efficient small-*N* DFT algorithms

The DFT of a length-*N* data sequence $\{x_k : k = 0,1,\cdots,N-1\}$, is another length-*N* sequence $\{X_n : n = 0,1,\cdots,N-1\}$, defined by

$$X_n = \sum_{k=0}^{N-1} x_k W_N^{nk} \qquad n = 0,1,\cdots,N-1, \tag{4.1}$$

Where, as before, $W_N = e^{-j2\pi/N}$ is the reciprocal of an N^{th} primitive root of unity and a scaling factor of $1/N$ has been ignored for convenience. Sequence $\{x_n\}$ may be real or complex, whereas $\{X_k\}$ is generally always complex. Equation (4.1) may be expressed in matrix form as,

$$\mathbf{X} = \mathbf{W}\,\mathbf{x} \tag{4.2}$$

where $\mathbf{W}$ is the $N \times N$ DFT operator matrix defined by $\mathbf{W} = \left[w_{n,k}\right]_{N\times N} = W_N^{nk \bmod N}$, and column vector $\mathbf{X} = (X_0, X_1, \cdots, X_{N-1})^T$ is the DFT of column vector $\mathbf{x} = (x_0, x_1, \cdots, x_{N-1})^T$. The superscript *T* denotes the transpose. If *N* is composite with *m* factors, i.e.,

$$N = r_1 r_2 \cdots r_m = \prod_{i=1}^{m} r_i, \tag{4.3}$$

the DFT operator can be expressed as the product of *m*+1 sparse $N \times N$ matrices,

$$\mathbf{W} = \mathbf{W}_m\ \mathbf{W}_{m-1} \cdots \mathbf{W}_2\ \mathbf{W}_1\ \mathbf{P}^T \tag{4.4}$$

where matrix $\mathbf{W}_i$ corresponds to factor r_i and $\mathbf{P}^{\mathrm{T}}$ is a permutation matrix. Thus, (4.2) becomes,

(Cooley-Tukey DIT FFT) $$\boxed{\mathbf{X} = \mathbf{W}_m\ \mathbf{W}_{m-1} \cdots \mathbf{W}_2\ \mathbf{W}_1\ \mathbf{P}^T\ \mathbf{x}}, \tag{4.5}$$

which is called the Cooley-Tukey, or decimation in time (DIT), fast Fourier transform (FFT) algorithm (Cooley & Tukey, 1965). The computation begins with a permutation $\mathbf{P}^T$ applied to **x**, and ends with a combine stage, $\mathbf{W}_m$. It is called DIT because the permutation re-orders the input data, splitting it into r_1 interleaved sets, each with effectively r_1 times the sample spacing or $1/r_1$ times the sampling rate. For example, if $r_1 = 2$, the input data is split into two sets, even and odd, each with effectively half the sampling rate. The input and output data are both in natural order.

Since the DFT operator matrix **W** is symmetric, we can use the transpose operation to derive a canonical variant called the Sande-Tukey, or decimation in frequency (DIF), FFT algorithm (Gentleman & Sande, 1966). In this re-arrangement of the DFT factorization, the

$\mathbf{W}_m$ combine stage appears first, and the **P** re-ordering (permutation) stage appears last in the computation. This version of the FFT is called DIF because the DFT sequence is computed as r_1 interleaved sets, each with effectively r_1 times the frequency sample spacing, or $1/r_1$ times the frequency resolution. For example, if $r_1 = 2$, the DFT terms are computed in two interleaved sets, even and odd, each with effectively half of the frequency resolution. The input and output data are both in natural order. Note that the permutation $\mathbf{P} = (\mathbf{P}^{\mathrm{T}})^{\mathrm{T}}$ used by the DIF algorithm is the transpose of the permutation used by the DIT algorithm. Thus, for the DIF case, equation (4.4) becomes,

$$\begin{aligned} \mathbf{W}^T &= \left(\mathbf{W}_m\, \mathbf{W}_{m-1} \cdots \mathbf{W}_2\, \mathbf{W}_1\, \mathbf{P}^T\right)^T \\ &= \mathbf{P}\, \mathbf{W}_1^T\, \mathbf{W}_2^T \cdots \mathbf{W}_{m-1}^T\, \mathbf{W}_m^T \end{aligned} \tag{4.6}$$

and likewise equation (4.2) becomes,

$$\text{(Sande-Tukey DIF FFT)} \qquad \boxed{\mathbf{X} = \mathbf{P}\, \mathbf{W}_1^T\, \mathbf{W}_2^T \cdots \mathbf{W}_{m-1}^T\, \mathbf{W}_m^T\, \mathbf{x}}. \tag{4.7}$$

Other canonical forms exist, but will not be described here. All have an equivalent amount of arithmetic, but may offer advantages depending on properties of the data or the machine architecture (Brigham 1974, 177).

By skipping multiplication by 0 or 1 in the above FFT algorithms, computational savings result. The most important special case occurs when $r_1 = r_2 = \ldots = r_m = 2$. A DFT algorithm with identical factors, *r*, is called a radix-*r* FFT, and an algorithm with different factors is called a mixed-radix FFT. An example of an $N = 4$ radix-2 FFT is given in Figure 1. The divider lines shown in the figure give a hint that this DFT factorization may be arrived at by building up from smaller transforms of size 2.

$$\mathbf{W}_{(N=4)} = \mathbf{P}\, \mathbf{W}_1^T\, \mathbf{W}_2^T = \left(\begin{array}{cccc} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{array}\right) \left(\begin{array}{cc|cc} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ \hline 0 & 0 & 1 & -j \\ 0 & 0 & 1 & j \end{array}\right) \left(\begin{array}{cc|cc} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ \hline 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{array}\right)$$

Figure 1. Sande-Tukey $N = 4$ radix-2 DIF FFT.
Input and output in natural order. Arithmetic: 8 additions.

As interest in FFT algorithms peaked in the latter part of the 20th century, practitioners in the art were surprised to uncover a lineage that went back 160 years, to the inventive mind of German mathematician, and human computer extraordinaire, Carl (Friedrich) Gauss. Although Gauss's interest was trigonometric interpolation of asteroid orbital data, rather than harmonic analysis as such, over several months in 1805 (*see* time-line in Heideman *et al.* 1984), Gauss derived the DFT (also inventing the least squares approach to determining the series coefficients), ten years before Bessel's DFT formula was published. However, Gauss did not stop with the DFT. Ever the perfectionist, just a few pages later in his notebook he also invented the decimation in time FFT algorithm as a way of computing the DFT more

efficiently.[9] His method, as in modern FFT practice, uses a phase correction factor that allows the results of several smaller interleaved DFT calculations from the same data sequence to be combined into a larger transform, the so-called *twiddle factor*.[10] Gauss's notes are replete with examples, including a radix-6 FFT $(N = r_1 \times r_2 = 6 \times 6)$, and a mixed-radix FFT done two different ways as a check: $(N = r_1 \times r_2 = 4 \times 3)$ and $(N = r_1 \times r_2 = 3 \times 4)$. He also stated that if the factors of N are themselves composite, his FFT algorithm can be applied recursively (Gauss 1866, articles 27–41; Goldstine 1977, 249; Heideman et al. 1984; Rabiner et al. 1972). Despite careful documentation of this work in his lab notebooks, Gauss unfortunately chose not to publish, and moved on to other interests. Even following publication in his collected works, Gauss's FFT achievement mostly escaped notice, exacerbated by his use of an obscure dialect called neo Latin![11]

Despite the popularity of the Gauss FFT factorization method, when N is small (less than 24 or so) useful algorithms result from a rather different factorization of the DFT matrix (Kolba and Parks, 1977),

$$\mathbf{W} = \mathbf{S}\,\mathbf{C}\,\mathbf{T} \tag{4.8}$$

where real matrices **T** and **S** perform the additions, and complex matrix **C** performs all of the multiplications. The **S** and **T** matrices can usually be factored further,

$$\mathbf{W} = \cdots \mathbf{S}_2\,\mathbf{S}_1\,\mathbf{C} \cdots \mathbf{T}_2\,\mathbf{T}_1 \tag{4.9}$$

into a set of sparse matrices with non-zero elements all ± 1. The **T** matrices perform the input additions, often called the *pre-weave*, while the **S** matrices perform the output additions or *post-weave*. The important feature of this DFT factorization is its (+)(×)(+) nested arithmetic structure (*see* Figure 2). The **C** matrix is diagonal with the numbers along the diagonal either real or pure imaginary. Winograd (1976, 1978) established that this property of the diagonal elements is general, at least for those cases when N is *prime* or a *prime-power*, or is "built up" out of relatively prime factors.

---

[9] Heideman et al. (1984) incorrectly identify Gauss's algorithm as decimation in frequency.

[10] Coined by Gentleman & Sande (1966), to give name and form to complex sinusoidal phase corrections required between FFT stages, *twiddle factor* has become one of the more durable entries in the signal-processing lexicon. In an increasingly common usage, it may also refer to any data-independent complex trigonometric phase rotation coefficients in an FFT or DFT computation.

[11] Burkhardt (1904, p. 686 footnote 169; Fr. trans. 1912, p. 143 footnote 188) in his otherwise quite extensive review of trigonometric interpolation says, "The method given by Gauss for the decomposition into groups, in the case where N is a composite number, seems little known and is rarely used in practice." This degree of understatement is on a par with Ford Prefect's revised entry for Earth in Douglas Adams' *Hitchhiker's Guide to the Galaxy*: "Mostly harmless." (This was not Ford's submitted text, but is all that remained after his editors had done with it.) Clearly, Burkhardt was not a practitioner, or he would have found for himself the truth in Gauss's words, that "…the [FFT] method greatly reduces the tediousness of [DFT] calculations, and success will teach the one who tries it." If Burkhardt could have foreseen the future, we might today be calling the FFT the fast Gauss transform (FGT), or the Gauss-Fourier-Burkhardt transform! The FFT is not, however the main topic of this paper, so little more will be said about it.

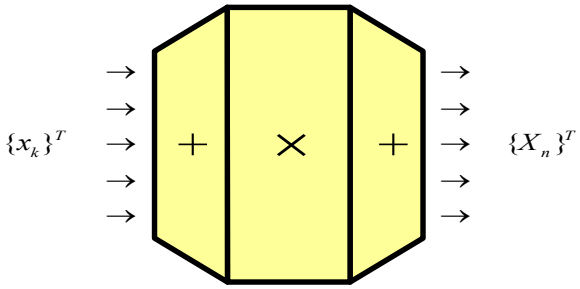


Figure 2. Small-*N* DFT with nested (+)(×)(+) arithmetic structure showing the expansion caused by more than one multiply per data point.

By enumerating primes, prime-powers, and products of relatively prime factors, it is easy to show that DFT algorithms of this type exist for all *N* up to our arbitrarily selected useful upper limit of 24.[12] With this type of algorithm, the number of multiplications is generally greater than *N*, as suggested by the expanded center section in Figure 2. The number of multiplications is the same as the order of matrix **C**. However, since matrix **C** is diagonal with elements that are either real or pure imaginary, each multiplication is either one real multiplication (real input data) or two real multiplications (complex input data). Of course, trivial multiplications by ± 1 and $\pm j$ may be omitted, and multiplication by ½ implemented using an arithmetic right shift (assuming binary arithmetic).

Winograd's $N = 4$ DFT algorithm is shown in matrix form in Figure 3. It is interesting to compare this algorithm with the Sande-Tukey radix-2 DFT factorization given above, in Figure1. Although different factorizations generally require different amounts of arithmetic, in this case the amounts are the same.

$$\mathbf{W}_{(N=4)} = \underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{pmatrix}}_{\mathbf{S}\ \text{Output Additions (+)}} \underbrace{\left(\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & -j \end{array}\right)}_{\mathbf{C}\ \text{Multiplications (×)}} \underbrace{\begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{\mathbf{T}_2} \underbrace{\begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{pmatrix}}_{\mathbf{T}_1\ \text{Input Additions (+)}}$$

Figure 3. Winograd (1978, 193) small-*N* DFT matrix factorization for $N = 4$. Scale factor of 1/4 ignored. Inputs and outputs in natural order. Arithmetic: 8+.

As mentioned above, the number of multiplications in small-*N* DFT algorithms having the (+)(×)(+) nested structure is equal to the number of diagonal elements in matrix **C**. With

[12] Imposed due to Winograd's statement (1976) that all known algorithms for computing cyclic convolution in the minimum number of multiplications require a large number of additions when polynomial $z^N - 1$ has large irreducible factors.

reference to Figure 3, since the only multiplications are by 1 or -*j* they may all be skipped, bringing the practical number of multiplications in the Winograd $N = 4$ algorithm to zero.

The number of additions in these DFT matrix factorizations is also readily determined. Assuming that the DFT is fully factored, i.e., no more than two 1s per row, the number of additions is equal to the number of matrix rows (in the input and output addition matrices) that contain two 1s. With reference to Figure 3, by inspection we see that matrix **S** contributes two additions: one from the 1,1 in row two, and the other from the 1,-1 in row four. Matrix $\mathbf{T_2}$ likewise contributes a further two additions, and $\mathbf{T_1}$ contributes another four, for a grand total of eight additions. For complex input data, the number of addition operations is doubled.

Winograd's $N = 3$ DFT algorithm is given in matrix form in Figure 4. A similar, but slightly more advantageously factored algorithm by Elliot and Rao (1982) is given in Figure 5 (instead of Winograd's multiplication by -3/2, at minimum requiring a shift and add, Elliot and Rao have multiplication by -1/2, which can be implemented as a simple shift).

$$\mathbf{W}_{(N=3)} = \underbrace{\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & -1 \end{pmatrix}}_{\mathbf{S_2}} \underbrace{\begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}}_{\mathbf{S_1}} \underbrace{\left(\begin{array}{cc|c} 1 & 0 & 0 \\ 0 & -\frac{3}{2} & 0 \\ \hline 0 & 0 & -j\frac{\sqrt{3}}{2} \end{array}\right)}_{\mathbf{C}} \underbrace{\begin{pmatrix} 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}}_{\mathbf{T_2}} \underbrace{\begin{pmatrix} 0 & 1 & 1 \\ 0 & 1 & -1 \\ 1 & 0 & 0 \end{pmatrix}}_{\mathbf{T_1}}$$

Output Additions (+) — Multiplications (×) — Input Additions (+)

Figure 4. Winograd (1978, 193) small-*N* DFT matrix factorization for $N = 3$. Scale factor of 1/3 ignored. Inputs and outputs in natural order. Arithmetic: 6+, 2×.

$$\mathbf{W}_{(N=3)} = \underbrace{\begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{pmatrix}}_{\mathbf{S_2}} \underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{\mathbf{S_1}} \underbrace{\left(\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -\frac{1}{2} & 0 \\ \hline 0 & 0 & 0 & -j\frac{\sqrt{3}}{2} \end{array}\right)}_{\mathbf{C}} \underbrace{\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}}_{\mathbf{T_2}} \underbrace{\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & -1 \end{pmatrix}}_{\mathbf{T_1}}$$

Output Additions (+) — Multiplications (×) — Input Additions (+)

Figure 5. A small-N DFT factorization for N = 3 (Elliot & Rao 1982, 127–132). Scale factor of 1/3 ignored. Inputs and outputs in natural order. Arithmetic: 6+, 1×, 1 shift.

In these latter two examples, there is no corresponding Sande-Tukey or Cooley-Tukey algorithm since the transform length is prime.

The above examples are too small to provide an accurate estimate of the amount of arithmetic for larger *N*. Winograd's small-*N* DFT algorithm for $N = 8$ (given in algebraic form in Table 4 and in matrix form in appendix C) requires 8 multiplications and 26 additions.

Omitting multiplies by 1 and j, the number of multiplies for real input data reduces to two. The corresponding Cooley-Tukey radix-2 case requires $\frac{1}{2}N\log_2 N = 12$ complex multiplies and $N\log_2 N = 24$ complex additions. If we perform complex multiplication in three real multiplies and omit multiplication by ± 1 and $\pm j$, a more realistic estimate for the Cooley-Tukey FFT (Kolba and Parks 1977) is $3(\frac{1}{2}N\log_2 N - \frac{3}{2}N + 2) = 6$ multiplies and $2N\log_2 N + \frac{5}{3}(\#\text{ of multiplies}) = 58$ additions. Thus, even in the worst case (complex input data), Winograd's $N = 8$ DFT requires only 4/6 = 67% of the multiplies and 16/58 = 28% of the additions compared to the $N = 8$ Cooley-Tukey FFT case. For real input data these percentages halve.

Note that due to the large number of zero elements in these matrices, it is inefficient to store the matrices themselves. Instead, algebraic equations that define the *non-zero* entries are stored. For example, Winograd's small-N DFT for $N = 8$ has 384 matrix elements, of which only 74 or 19% are non-zero. The matrix representation is, however, most useful for derivation, understanding, and documenting various DFT algorithms.

The implicit necessity of the (+)(×)(+) nested arithmetic structure is suggested by its presence in the majority of modern-day small-N DFT algorithms, most notably the prime and prime-power length high-speed convolution algorithms given by Winograd (1978), Kolba and Parks (1977), Elliot and Rao (1982), and others. It is further suggested by its presence in the DFT algorithms described by Hann (1901), Runge (1903), Brooks and Carruthers (1953), and Kämtz (1831), all described in chapter 1.

As briefly mentioned at the end of chapter 2, Winograd also generalized the (+)(×)(+) nested structure to large-N DFT algorithms that are "built up" from relatively prime-length small-N algorithms (these large-N algorithms are discussed in the next section). *This same nested arithmetic structure is common to most of the early DFT algorithms, suggesting that they too are algorithms of this type, despite their completely different derivation.*

The fact that the reduced arithmetic DFT algorithm given by noted German meteorologist, Ludwig Friedrich Kämtz (1831), described in chapter 1, and shown in Table 1, is missing the final (+) stage, is simply because Kämtz stopped short of complete factorization of the DFT operator. Nevertheless, Kämtz's algorithm, which is relatively efficient, compared to naive DFT computation, is one of the earliest examples of this type. It computes X_0, X_1, X_2, X_3 from 24 evenly spaced data points, $\{x_k : k = 0,1,\cdots,23\}$, and was created to analyze the daily and annual cycles of temperature, barometric pressure, and humidity.

DFT algorithms with the (+)(×)(+) nested arithmetic structure (including Kämtz's) exploit the symmetry of the sine and cosine functions in the four quadrants of the circle. Since $\sin x = \cos(x - \pi/2)$, and $\cos x = -\cos(\pi - x) = -\cos(\pi + x) = \cos(2\pi - x)$, it is evident that for $N = 4m$,[13] a considerable number of multiplications can be eliminated by combining the x_k

[13] $4m$ was a popular data sequence length presumably because it guaranteed that the sequence could be twice-folded, or broken into four equal parts.

(in a pre-weave module) before forming the products. Twice folding the input data sequence eliminates approximately 1- ¼^2 = 15/16 or 94% of the multiplications required by straightforward (sum-of-products) evaluation of the DFT. Two types of folding are possible: ordinary folding about the center of the sequence, and superposition of one-half of the sequence on the other. For example, Kämtz's (1831) $N = 24$ algorithm and Runge's (1903) $N = 12$ DFT algorithm both use ordinary folds, while Hann's (1901) $N = 12$ algorithm and Winograd's (1978) $N = 16$ small-N high-speed convolution DFT algorithm both use superposition followed by either a superposition or a fold. It is easily shown that two folds are the same as superposition followed by a fold.

Since $N = 4m$ can be expressed as a prime-power or as the product of relatively prime factors for all multiples of 4 up to at least $N = 64$, it is conjectured that there is a direct correspondence between the early DFT algorithms and recent high-speed convolution DFT algorithms, as shown in Table 5. Note that Runge gave a general method for deriving algorithms for any $N = 4m$.

Table 5
Conjectured Classification of $N = 4m$ DFT Algorithms

N	Classification	Author(s)
4	prime power, 2^2	
8	prime power, 2^3	
12	rel. prime factors, 3 × 4	Hann (1901), Runge (1903)
16	prime power, 2^4	Danielson and Lanczos (1942)
20	rel. prime factors, 4 × 5	
24	rel. prime factors, 3 × 8	Hann (1901)
28	rel. prime factors, 4 × 7	
32	prime power, 2^5	
36	rel. prime factors, 4 × 9	Runge (1903)

The point of these examples is to illustrate the fact that *all* reduced arithmetic DFT algorithms achieve their computational savings in fundamentally the same way and ultimately through factorization of the DFT operator. Irrespective of the method used to derive a particular factorization, the underlying theoretical principles are *always* the symmetry and/or periodicity properties of the orthogonal set of sine and cosine basis functions used in discrete Fourier analysis. However, the similarities between early DFT algorithms, and modern-day algorithms based on high-speed convolution, only show that the convolution property of the DFT is also quite fundamental, and similar algorithms are obtained despite differences in the formal methods used to derive them.

Whereas 180 years ago various trial-and-error algebraic methods were used to do the factorization, now a variety of algorithmic procedures are available based on the Cook-Toom

algorithm, the polynomial version of the Chinese Remainder Theorem[14] (CRT), and various other number-theoretic approaches. Moreover, when used in combination with the Kronecker product, these methodologies allow efficient small-*N* DFT algorithms to be combined, "building block" style, to yield time-efficient large transforms.

As shown by Charles Van Loan (1992) in his *tour de force* of DFT matrix/vector mathematics, *Computational frameworks for fast Fourier transform*, the Kronecker product is fundamental to the structure of the DFT matrix, and simplifies the search for efficient factorizations, whether the structure be radix-2, general radix (radix-4, radix-8, mixed- or split-radix), prime factor, or nested; single- or multi-dimensional.

The underlying principal is, however, that irrespective of the methodology used to derive a particular DFT algorithm, the same algorithm could (given enough time and patience, or monkeys and typewriters, or all four options) be arrived at, through trial and error, by directly manipulating the DFT operator into various factored representations.

[14] Modern-day number theory is much more ancient than even the DFT. A cornerstone is the Chinese remainder theorem, which extends at least as far back as the 3rd century A.D., to the work of Chinese mathematician Sun Tzu (or Sun Zi), about who little is known, but who developed a method of measuring plots of land using simultaneous congruences of number residues, today known as the Chinese remainder theorem, which resulted from a clever use of distance measuring wheels having relatively prime circumferences.

5. Large transforms from small ones

The (+)(×)(+) nested small-N DFT structure described above is extendable to large N by combining relatively prime length small-N DFTs in a way that retains the (+)(×)(+) nested arithmetic structure. This generalization is known as the Winograd Fourier transform algorithm, or WFTA, after its originator Shmuel Winograd (1976; 1978). It is also known as the nested algorithm (Kolba and Parks 1977), although this name is less suitable because it fails to distinguish between the small-N DFTs, which make up the WFTA, and the WFTA itself, both of which have the same nested structure.

We combine L relatively prime length small-N DFT operator matrices according to

$$\mathbf{W} = \mathbf{W}_L \otimes \cdots \otimes \mathbf{W}_2 \otimes \mathbf{W}_1 \tag{5.1}$$

where the dimension of $M \times M$ matrix $\mathbf{W}$ is the product of the dimensions of the individual matrices, $M = N_L N_{L-1} \cdots N_2 N_1$, and $\otimes$ is the Kronecker product (a special case of the tensor product, also known as the direct product). The resulting mixed-radix length-M DFT has L factors, and the inputs and outputs are in permuted order.

If each of the $\mathbf{W}_i : i = 1, 2, \cdots, L$ matrices in (5.1) are factored according to the Winograd (+)(×)(+) nested arithmetic structure, $\mathbf{W}_i = \mathbf{S}_i \mathbf{C}_i \mathbf{T}_i$, we can write

$$\mathbf{W} = (\mathbf{S}_L \mathbf{C}_L \mathbf{T}_L) \otimes \cdots \otimes (\mathbf{S}_2 \mathbf{C}_2 \mathbf{T}_2) \otimes (\mathbf{S}_1 \mathbf{C}_1 \mathbf{T}_1). \tag{5.2}$$

Using the identity,

$$(\mathbf{AB}) \otimes (\mathbf{C}\,\mathbf{D}) = (\mathbf{A} \otimes \mathbf{C})(\mathbf{B} \otimes \mathbf{D}), \tag{5.3}$$

where **A**, **B**, **C**, and **D** are matrices with dimensions $a \times b$, $b \times c$, and $e \times f$, $f \times g$, respectively, we finally get

$$\mathbf{W} = \overbrace{(\mathbf{S}_L \otimes \cdots \otimes \mathbf{S}_2 \otimes \mathbf{S}_1)}^{\text{output additions}} \overbrace{(\mathbf{C}_L \otimes \cdots \otimes \mathbf{C}_2 \otimes \mathbf{C}_1)}^{\text{products}} \overbrace{(\mathbf{T}_L \otimes \cdots \otimes \mathbf{T}_2 \otimes \mathbf{T}_1)}^{\text{input additions}} \tag{5.4}$$

which is has the same (+)(×)(+) nested structure as the individual small-N Winograd DFT algorithms we started with, giving us a way of systematically constructing WFTAs for larger values of N. As before, the $\mathbf{S}_i$ and $\mathbf{T}_i$ matrices are sparse, with non-zero entries of ±1, which therefore specify additions. The center term nests all of the multiplications inside the additions. Note that the inputs and outputs are in permuted order; inputs according to the Chinese Remainder Theorem (CRT), and outputs according to the Second Integer Representation (SIR) theorem, or vice versa (Kolba and Parks, 1977).

As an example of the above procedure, consider the building-up of an $N = 12$ WFTA from two small-N algorithms given earlier: Winograd $N = 4$ (*see* Figure 3) and Elliot and Rao

$N = 3$ (*see* Figure 5). Since the lengths are relatively prime (i.e., gcd(3,4) = 1) we can write the $N = 12$ DFT operator matrix as the Kronecker matrix product,

$$\begin{aligned}
\mathbf{W}_{N=12} &= \mathbf{W}_{N=4} \otimes \mathbf{W}_{N=3} \\
&= \mathbf{S}'' \, \mathbf{C}'' \, \mathbf{T}'' \otimes \mathbf{S}' \, \mathbf{C}' \, \mathbf{T}' \\
&= \left(\mathbf{S}'' \, \mathbf{C}'' \, \mathbf{T}_2'' \mathbf{T}_1''\right) \otimes \left(\mathbf{S}_2' \, \mathbf{S}_1' \mathbf{C}' \, \mathbf{T}_2' \, \mathbf{T}_1'\right) \\
&= \left(\left(\mathbf{S}''\right)\left(\mathbf{C}'' \, \mathbf{T}_2'' \mathbf{T}_1''\right)\right) \otimes \left(\left(\mathbf{S}_2' \, \mathbf{S}_1'\right)\left(\mathbf{C}' \, \mathbf{T}_2' \, \mathbf{T}_1'\right)\right) \\
&= \left(\mathbf{S}'' \otimes \left(\mathbf{S}_2' \, \mathbf{S}_1'\right)\right)\left(\left(\mathbf{C}''\left(\mathbf{T}_2'' \mathbf{T}_1''\right)\right) \otimes \left(\mathbf{C}'\left(\mathbf{T}_2' \, \mathbf{T}_1'\right)\right)\right) \\
&= \left(\left(\mathbf{S}'' \, \mathbf{I}_4\right) \otimes \left(\mathbf{S}_2' \, \mathbf{S}_1'\right)\right)\left(\mathbf{C}'' \otimes \mathbf{C}'\right)\left(\left(\mathbf{T}_2'' \mathbf{T}_1''\right) \otimes \left(\mathbf{T}_2' \, \mathbf{T}_1'\right)\right) \\
&= \left(\mathbf{S}'' \otimes \mathbf{S}_2'\right)\left(\mathbf{I}_4 \otimes \mathbf{S}_1'\right)\left(\mathbf{C}'' \otimes \mathbf{C}'\right)\left(\mathbf{T}_2'' \otimes \mathbf{T}_2'\right)\left(\mathbf{T}_1'' \otimes \mathbf{T}_1'\right) \\
&= \mathbf{S}_2 \, \mathbf{S}_1 \, \mathbf{C} \, \mathbf{T}_2 \, \mathbf{T}_1 \\
&= \mathbf{S} \, \mathbf{C} \, \mathbf{T},
\end{aligned} \tag{5.5}$$

where double primes denote the $N_1 = 4$ transform, single primes the $N_2 = 3$ transform, and $\mathbf{I}_4$ is the 4th-order identity matrix. The factors of the new $N = 12$ DFT operator matrix are therefore

$$\begin{aligned}
\mathbf{S} &= \left(\mathbf{S}'' \otimes \mathbf{S}_2'\right)\left(\mathbf{I}_4 \otimes \mathbf{S}_1'\right), \\
\mathbf{C} &= \mathbf{C}'' \otimes \mathbf{C}', \\
\text{and} \quad \mathbf{T} &= \left(\mathbf{T}_2'' \otimes \mathbf{T}_2'\right)\left(\mathbf{T}_1'' \otimes \mathbf{T}_1'\right).
\end{aligned}$$

The remarkable thing about (5.5) is that the WFTA has the same nested structure as the individual small-N DFT algorithms that it is built-up from. The resulting diagonal matrix $\mathbf{C}$ is composed of real or purely imaginary components,

$$\mathbf{C} = \operatorname{diag}\left(1, 1, -\tfrac{1}{2}, -j\tfrac{\sqrt{3}}{2}, 1, 1, -\tfrac{1}{2}, -j\tfrac{\sqrt{3}}{2}, 1, 1, -\tfrac{1}{2}, -j\tfrac{\sqrt{3}}{2}, -j, -j, j\tfrac{1}{2}, -\tfrac{\sqrt{3}}{2}\right). \tag{5.6}$$

The above example, a two-factor WFTA, is one of two possible canonical forms generated by exchanging the order of the $\mathbf{W}_N$ matrices in the Kronecker product. If there are no repeated factors (as is the case for algorithms having relatively prime factors, such as the WFTA), an L-factor DFT algorithm generated according to (5.1) has $L!$ possible canonical forms. With just two factors, as in this example, there are two such forms. With 3, 4, and 5 factors, there are 6, 24, and 120 canonical forms, respectively. As discussed by Winograd (1978), all such equivalent forms have the same number of multiplications, but will differ in the number of additions.

By way of an illustration, we will examine the effect of reversing the order of the Kronecker product in equation (5.5). Using the same two small-N algorithms given earlier, in Figure 3 (Winograd $N = 4$) and Figure 5 (Elliot and Rao $N = 3$),

$$\begin{aligned}
\mathbf{W}_{N=12} &= \mathbf{W}_{N=3} \otimes \mathbf{W}_{N=4} \\
&= \mathbf{S}'\,\mathbf{C}'\,\mathbf{T}' \otimes \mathbf{S}''\,\mathbf{C}''\,\mathbf{T}'' \\
&= (\mathbf{S}'_2\,\mathbf{S}'_1\,\mathbf{C}'\,\mathbf{T}'_2\,\mathbf{T}'_1) \otimes (\mathbf{S}''\,\mathbf{C}''\,\mathbf{T}''_2\,\mathbf{T}''_1) \\
&= \left((\mathbf{S}'_2\,\mathbf{S}'_1)(\mathbf{C}'\,\mathbf{T}'_2\,\mathbf{T}'_1)\right) \otimes \left((\mathbf{S}'')(\mathbf{C}''\,\mathbf{T}''_2\,\mathbf{T}''_1)\right) \\
&= \left((\mathbf{S}'_2\,\mathbf{S}'_1) \otimes \mathbf{S}''\right)\left((\mathbf{C}'(\mathbf{T}'_2\,\mathbf{T}'_1)) \otimes (\mathbf{C}''(\mathbf{T}''_2\,\mathbf{T}''_1))\right) \\
&= \left((\mathbf{S}'_2\,\mathbf{S}'_1) \otimes (\mathbf{S}''\,\mathbf{I}_4)\right)(\mathbf{C}' \otimes \mathbf{C}'')\left((\mathbf{T}'_2\,\mathbf{T}'_1) \otimes (\mathbf{T}''_2\,\mathbf{T}''_1)\right) \\
&= (\mathbf{S}'_2 \otimes \mathbf{S}'')(\mathbf{S}'_1 \otimes \mathbf{I}_4)(\mathbf{C}' \otimes \mathbf{C}'')(\mathbf{T}'_2 \otimes \mathbf{T}''_2)(\mathbf{T}'_1 \otimes \mathbf{T}''_1) \\
&= \mathbf{S}_2\,\mathbf{S}_1\,\mathbf{C}\,\mathbf{T}_2\,\mathbf{T}_1 \\
&= \mathbf{S}\,\mathbf{C}\,\mathbf{T},
\end{aligned} \tag{5.7}$$

where, as before, single and double primes denote the $N_1 = 3$ and $N_2 = 4$ transforms, respectively, and $\mathbf{I}_4$ is the order-4 identity matrix. Similarly,

$$\begin{aligned}
&\mathbf{S} = (\mathbf{S}'_2 \otimes \mathbf{S}'')(\mathbf{S}'_1 \otimes \mathbf{I}_4), \\
&\mathbf{C} = \mathbf{C}' \otimes \mathbf{C}'', \\
\text{and} \quad &\mathbf{T} = (\mathbf{T}'_2 \otimes \mathbf{T}''_2)(\mathbf{T}'_1 \otimes \mathbf{T}''_1).
\end{aligned}$$

The resulting diagonal matrix $\mathbf{C}$ is composed of real or purely imaginary components,

$$\mathbf{C} = \mathrm{diag}\left(1,1,1,-j,1,1,1,-j,-\tfrac{1}{2},-\tfrac{1}{2},-\tfrac{1}{2},j\tfrac{1}{2},-j\tfrac{\sqrt{3}}{2},-j\tfrac{\sqrt{3}}{2},-j\tfrac{\sqrt{3}}{2},-\tfrac{\sqrt{3}}{2}\right). \tag{5.8}$$

In this latter example, the input and output ordered is according to the CRT and the SIR, respectively.

Input Indexing

Building-up a WFTA from two relatively prime length small-N DFTs, essentially maps a one-dimensional calculation into two-dimensions. In this two-factor case, the CRT provides a 1-to-1 mapping between the one-dimensional input index, k, and the two-dimensional internal time indices k_1 and k_2 (Elliot and Rao, 1982):

$$k = \left[\left(\frac{N}{N_2}\right)^2 k_2 + \left(\frac{N}{N_1}\right) k_1\right] \bmod N. \tag{5.9}$$

For $N = 12$, $N_1 = 3$, and $N_2 = 4$, this reduces to

$$k = [9k_2 + 4k_1] \bmod 12 \tag{5.10}$$

Output Indexing

The two-dimensional internal frequency indices, n_1 and n_2, are likewise mapped 1-to-1 to the one-dimensional output index, n, by the SIR theorem (Elliot and Rao, 1982):

$$n = \left[\left(\frac{N}{N_2}\right)n_2 + \left(\frac{N}{N_1}\right)n_1\right] \bmod N \tag{5.11}$$

In other words,

$$n = \left[3n_2 + 4n_1\right] \bmod 12 \tag{5.12}$$

Placing the respective 2-dimensional indices in (5.10) and (5.12) in lexicographical order, we get, respectively, an input index order (by CRT) of 0, 9, 6, 3, 4, 1, 10, 7, 8, 5, 2, 11, and we get an output index order (by SIR) of 0, 3, 6, 9, 4, 7, 10, 1, 8, 11, 2, 5. Table 6 shows these calculations in more detail.

Table 6
Input and Output Index Calculations for the $N = 3 \times 4$ WFTA algorithm discussed in the text

CRT mapping $k \mapsto k_1, k_2$			SIR mapping $n_1, n_2 \mapsto n$		
k_1	k_2	$k \pmod{12}$	n_1	n_2	$n \pmod{12}$
0	0	0	0	0	0
0	1	9	0	1	3
0	2	6	0	2	6
0	3	3	0	3	9
1	0	4	1	0	4
1	1	1	1	1	7
1	2	10	1	2	10
1	3	7	1	3	1
2	0	8	2	0	8
2	1	5	2	1	11
2	2	2	2	2	2
2	3	11	2	3	5

Thus, the discrete Fourier transform defined by $\mathbf{W}_{N=12} = \mathbf{W}_{N=4} \otimes \mathbf{W}_{N=3}$ can be written,

$$\begin{pmatrix} X_0 \\ X_3 \\ X_6 \\ X_9 \\ X_4 \\ X_7 \\ X_{10} \\ X_1 \\ X_8 \\ X_{11} \\ X_2 \\ X_5 \end{pmatrix} = \left(\begin{array}{cccc|cccc|cccc} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -j & -1 & j & 1 & -j & -1 & j & 1 & -j & -1 & j \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & j & -1 & -j & 1 & j & -1 & -j & 1 & j & -1 & -j \\ \hline 1 & 1 & 1 & 1 & W^1 & W^1 & W^1 & W^1 & W^{-1} & W^{-1} & W^{-1} & W^{-1} \\ 1 & -j & -1 & j & W^1 & -jW^1 & -W^1 & jW^1 & W^{-1} & -jW^{-1} & -W^{-1} & jW^{-1} \\ 1 & -1 & 1 & -1 & W^1 & -W^1 & W^1 & -W^1 & W^{-1} & -W^{-1} & W^{-1} & -W^{-1} \\ 1 & j & -1 & -j & W^1 & jW^1 & -W^1 & -jW^1 & W^{-1} & jW^{-1} & -W^{-1} & -jW^{-1} \\ \hline 1 & 1 & 1 & 1 & W^{-1} & W^{-1} & W^{-1} & W^{-1} & W^1 & W^1 & W^1 & W^1 \\ 1 & -j & -1 & j & W^{-1} & -jW^{-1} & -W^{-1} & jW^{-1} & W^1 & -jW^1 & -W^1 & jW^1 \\ 1 & -1 & 1 & -1 & W^{-1} & -W^{-1} & W^{-1} & -W^{-1} & W^1 & -W^1 & W^1 & -W^1 \\ 1 & j & -1 & -j & W^{-1} & jW^{-1} & -W^{-1} & -jW^{-1} & W^1 & jW^1 & -W^1 & -jW^1 \end{array}\right) \begin{pmatrix} x_0 \\ x_9 \\ x_6 \\ x_3 \\ x_4 \\ x_1 \\ x_{10} \\ x_7 \\ x_8 \\ x_5 \\ x_2 \\ x_{11} \end{pmatrix} \tag{5.13}$$

As mentioned previously, the input and output vectors of a DFT built-up using the Kronecker product in the method shown, are scrambled. Therefore, the DFT equation (4.2) becomes,

$$\mathbf{\Gamma X} = (\mathbf{SCT})\,\mathbf{\Theta x},$$

where $\mathbf{\Gamma}$ and $\mathbf{\Theta}$ are permutation matrices, and $\mathbf{x}$ and $\mathbf{X}$ are in natural order. However, since the inverse of a permutation matrix is simply its transpose, we can write this as

$$\mathbf{X} = \mathbf{\Gamma}^T (\mathbf{SCT})\,\mathbf{\Theta x}. \tag{5.14}$$

Comparing (5.6) and (5.8) with the **C** matrix in Runge's (1903) $N = 12$ DFT (given in algebraic form in Table 3, and in matrix form in appendix A), strongly suggests that the $N = 12$ WFTA algorithm (in either of its canonical forms), and Runge's algorithm are isomorphic. A similar conclusion applies to the Hann (1901) and Brooks and Carruthers (1953) algorithms (given in Table 2, and in appendix B). All of these algorithms have same nested structure and the same **C** matrix (apart from a reordering of the elements, which could be adjusted by elementary row and column operations on the **S** and **T** matrices). The major difference between these and the $N = 12$ WFTA derived here is the reordering of the input and output indices according to the CRT. However, we may restore natural order is restored using $\mathbf{\Gamma}^T$ and $\mathbf{\Theta}$ permutation matrices, which could be combined with the **S** and **T** matrices, respectively, as $\mathbf{S}_P = \mathbf{\Gamma}^T\mathbf{S}$ and $\mathbf{T}_P = \mathbf{T\Theta}$, if desired.

Arithmetic Operations

Using formulae given by Kolba and Parks (1977) and Winograd (1978) for the number of arithmetic operations, we must count all multiplies by ± 1 and $\pm j$ that were previously omitted. Given $N = N_1 N_2 = 3 \times 4 = 12,$ where respectively, the number of adds is a_1 and a_2, and the number of multiplies is m_1 and m_2, we get,

$$\#\text{multiplies} = m_1 m_2 = 4 \times 4 = 16 \text{ (reduces to 4×, 4 shifts)} \quad \text{(Runge: 4×, 4 shifts)}$$
$$\#\text{adds} = n_2 a_1 + m_1 a_2 = 3 \times 8 + 4 \times 6 = 48 \quad \text{(Runge: 38+)}$$

Thus, even though the $N = 12$ WFTA algorithm is isomorphic with Runge's $N = 12$ algorithm, these arithmetic results imply that, in respect of the **S** and **T** matrices, it is not quite as highly factored. Keep in mind, however, that the WFTA algorithm derived here can process complex input, whereas Runge's inputs are restricted to real. Table 7 presents similar arithmetic complexity data for various sizes of small-N DFT algorithms. The performance data in the table is based on Winograd (1978), Kolba and Parks (1977), and Burrus and Parks (1985). Most of the algorithms included in the table achieve the theoretical minimum number of multiplications, or else the smallest number of multiplications that does not require a very large number of additions.

Table 7

Number of arithmetic operations for modern-day small-*N* DFT having (+)(×)(+) nested arithmetic structure. Numbers are for real data (double for complex data).

N	# Mults, excl. W^0	# Mults by W^0	# Adds
2	0	2	2
3	2	1	6
4	0	4	8
5	5	1	17
7	8	1	36
8	2	6	26
9	10	2(1)	49(45)[42]
11	20	?	84
13	20	?	94
16	10	8	74
17	35	?	157
19	38	?	186
25	66	?	210

Note: Numbers are from Kolba and Parks (1977), Winograd (1978), and Burrus & Parks (1985). The numbers in parentheses indicate Winograd, and in square brackets indicate Burrus & Parks, where they differ from those of Kolba and Parks. The numbers for WFTA are mostly identical to equivalent-sized small-N DFTs (*see* table 2-7 in Burrus & Parks, 1985).

Having identified Hann's (1901) DFT and Runge's (1903) $N = 12$ DFT as WFTAs, in other words, members of the class of high-speed convolution DFT algorithms having a (+)(×)(+) nested arithmetic structure, it is now possible to make another identification. In a later paper Runge (1905) used an FFT doubling procedure (radix-2) to extend his previously published $N = 12$ DFT algorithm to $N = 24$, His method separated the input data into even and odd set, applied a 12-point DFT to each, and applied phase correction (twiddle factor) equal to the 1-sample time difference to the odd transform before adding the results together in the usual way (Runge, 1905). Hence Runge's efficient $N = 24$ DFT algorithm can be identified as a

hybrid *WFTA and radix-2 DIT FFT* (*see* Figure 6). Although Runge did not build up the WFTA part of the algorithm out of smaller relatively prime small-*N*, his results are structurally similar and functionally equivalent in terms of the amount of arithmetic required.

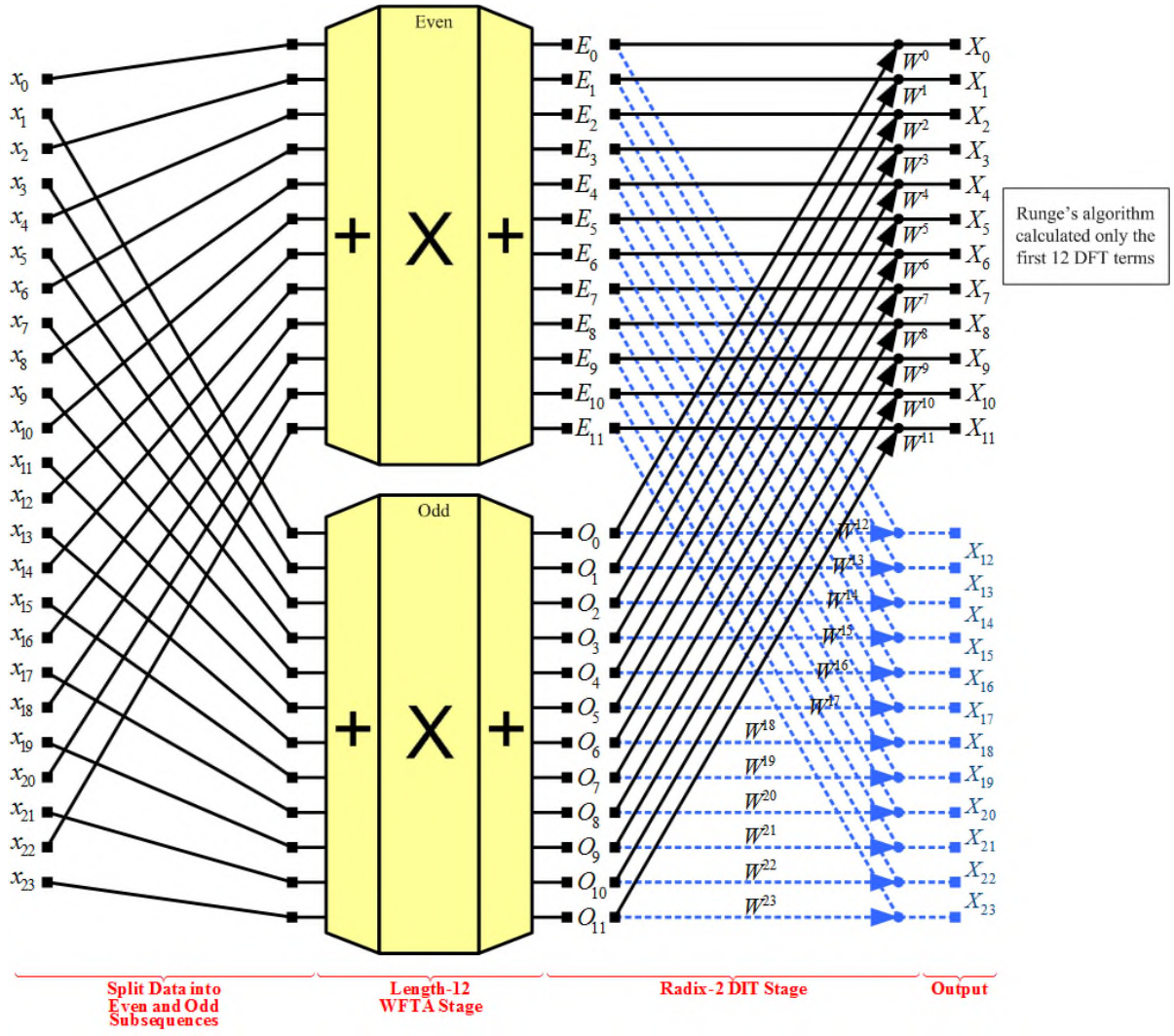


Figure 6. Runge's $N = 24$ hybrid FFT algorithm for real-data (Runge, 1905) . The input data are "decimated in time", into two interleaved sets, even $\{x_0, x_2, \cdots\}$ and odd $\{x_1, x_3, \cdots\}$, and a 12-point DFT (similar to a WFTA) is computed for each. As is appropriate for real data, Runge pruned the radix-2 output stage to compute only the first twelve DFT terms, combining the first stage results in twelve "half -butterflies." These consist of i) twiddle factors (i.e., phase rotators, applied to the odd transform output to adjust for the one-sample time difference between the even and odd data sets, indicated by "➔"), and ii) addition (indicated by "•"). The half -butterfly is $X_n = E_n + W_N^n O_n$, where $W_N = e^{-j2\pi/N}$ is the twiddle factor. For example, $X_3 = E_3 + W_{24}^3 O_3$, where the twiddle factor represents a 3 × 360°/24 = 3 × 15° phase rotation in complex space. The twiddle factor exponent adjusts the phase shift vs. frequency index *n,* to give a constant one-sample time correction irrespective of frequency.

6. Summary and conclusions

Thus it is clear that the WFTA and small-N high-speed convolution algorithms are almost as old as the DFT itself, in a rudimentary form dating back to 1831 (with the work of Kämtz) and possibly earlier, although probably a lot later than Gauss's invention of the mixed-radix and common-radix decimation-in-time FFT for real data sequences. That Gauss's worked examples of mixed-radix FFTs used relatively prime factors is just a happenstance related to his choice of N, and of no consequence for his algorithms. He first used $N = N_1N_2 = 12$, where $N_1 = 3$ and $N_2 = 4$, and then repeated the calculation with $N_1 = 4$ and $N_2 = 3$, to check the method and the correctness of the results.

Kämtz's DFT algorithm (1831) does not completely exhibit the (+)(×)(+) nested arithmetic structure, only having the *pre-weave* and multiply (+)(×), with no *post-weave*, suggesting that it is an algorithm of the same general type, and just not as completely factored.

Among the first complete versions of WFTA type algorithms for real data were those published by Julius von Hann (1901) and Carl Runge (1903). Thus, when Richard W. Hamming (1973, p. 543) presented an $N = 12$ DFT similar to Runge's algorithm and stated that it was closely related to the FFT, he was intuitively correct. Hann and Runge simply folded the data twice to reduce the number of multiplications by taking advantage of the symmetry properties of the sine and cosine functions. As has been demonstrated here, in doing so, Hann and Runge both derived a DFT algorithm isomorphic with the WFTA for relatively prime factors 3 and 4. On the other hand, Runge's $N = 24$ algorithm (Runge, 1905), by taking advantage of the periodicity properties of the sine and cosine functions, is more advanced. In its second stage, it uses a radix-2 FFT to combine the two length-12 WFTA first stages. For this reason, Runge's $N = 24$ DFT algorithm is classifiable as a hybrid WFTA and radix-2 FFT.

Finally, the design of DFT algorithms seems to have many parallels with bridge design. All bridges share the same set of structural components: beams, arches, trusses and suspensions (think sine and cosine basis functions). Since time immemorial, various combinations of these technologies have allowed for numerous bridge designs, ranging from arch bridges and simple beam bridges, to truss bridges, to gigantic suspension bridges with spans longer than 1 km not uncommon. And, just as today's efficient DFT algorithms have an ancient history, even the latest highly-efficient bridge designs such as side-spar cable-stayed are based on suspension principles first suggested some three centuries ago.

Appendix A

Runge $N = 12$ DFT algorithm for real data

$$\mathbf{X} = \mathbf{S}_3\,\mathbf{S}_2\,\mathbf{S}_1\,\mathbf{C}\,\mathbf{T}_3\,\mathbf{T}_2\,\mathbf{T}_1\,\mathbf{x}$$

$\mathbf{S}_3$ $\qquad$ $\mathbf{S}_2$

$$\begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \\ X_6 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$\mathbf{S}_1$

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\mathbf{C}$$

$$\left[\begin{array}{cccccccccc|cccccc}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{\sqrt{3}}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & j & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & j & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{j}{2} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{j\sqrt{3}}{2} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{j\sqrt{3}}{2} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{j\sqrt{3}}{2}
\end{array}\right]$$

$$\mathbf{C} = \operatorname{diag}\left(1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad \frac{1}{2} \quad \frac{1}{2} \quad \frac{1}{2} \quad \frac{\sqrt{3}}{2} \quad j \quad j \quad \frac{j}{2} \quad \frac{j\sqrt{3}}{2} \quad \frac{j\sqrt{3}}{2} \quad \frac{j\sqrt{3}}{2} \right)$$

$$\mathbf{T}_3$$

$$\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0
\end{bmatrix}$$

$$\mathbf{T}_2$$

$$\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1
\end{bmatrix}$$

$$\mathbf{T}_1$$

$$\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \\ x_{11}
\end{bmatrix}$$

Appendix B

Hann, Brooks and Carruthers *N* = 12 DFT algorithm for real data.

$$\mathbf{X} = \mathbf{S}_3\,\mathbf{S}_2\,\mathbf{S}_1\,\mathbf{C}\,\mathbf{T}_3\,\mathbf{T}_2\,\mathbf{T}_1\,\mathbf{x}$$

$$\begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \\ X_6 \end{bmatrix} = \underset{\mathbf{S}_3}{\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}} \underset{\mathbf{S}_2}{\begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}}$$

$$\underset{\mathbf{S}_1}{\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}}$$

$$\mathbf{C}$$

$$\left[\begin{array}{cccccccc|cccccc}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{\sqrt{3}}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\
\hline
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & j & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & j & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{j}{2} & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{j\sqrt{3}}{2} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{j\sqrt{3}}{2} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{j\sqrt{3}}{2}
\end{array}\right]$$

$$\mathbf{C} = \operatorname{diag}\left(1 \quad 1 \quad 1 \quad 1 \quad \frac{1}{2} \quad \frac{1}{2} \quad \frac{1}{2} \quad \frac{\sqrt{3}}{2} \quad j \quad j \quad \frac{j}{2} \quad \frac{j\sqrt{3}}{2} \quad \frac{j\sqrt{3}}{2} \quad \frac{j\sqrt{3}}{2} \right)$$

$$\mathbf{T}_3$$

$$\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

$$\mathbf{T}_2$$

$$\begin{bmatrix}
1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0
\end{bmatrix}$$

$$\mathbf{T}_1$$

$$\left[\begin{array}{cccccc|cccccc}
1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\
\hline
1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1
\end{array}\right]
\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \\ x_{11} \end{bmatrix}$$

Appendix C

Winograd $N = 8$ DFT algorithm

$$\mathbf{X} = \mathbf{S}_2\,\mathbf{S}_1\,\mathbf{C}\,\mathbf{T}_3\,\mathbf{T}_2\,\mathbf{T}_1\,\mathbf{x}$$

$$\begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \\ X_6 \\ X_7 \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 \end{bmatrix}}_{\mathbf{S}_2} \underbrace{\begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}}_{\mathbf{S}_1}$$

$$\underbrace{\left[\begin{array}{ccccc|ccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{\sqrt{2}} & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & j & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & j & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{j}{\sqrt{2}} \end{array}\right]}_{\mathbf{C}} \underbrace{\begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}}_{\mathbf{T}_3}$$

$$\mathbf{C} = \operatorname{diag}\left(1 \quad 1 \quad 1 \quad 1 \quad \frac{1}{\sqrt{2}} \quad j \quad j \quad \frac{j}{\sqrt{2}}\right)$$

$$\underbrace{\begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}}_{\mathbf{T}_2} \underbrace{\left[\begin{array}{cccc|cccc} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ \hline 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \end{array}\right]}_{\mathbf{T}_1} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix}$$

References

Agarwal, Ramesh C., and Cooley, James W., [1977] "New algorithms for digital convolution," *IEEE Transactions on Acoustics, Speech, and Signal Processing,* **25**:2, 392–410.

Bessel, Friedrich Wilhelm, [1815] "Astronomische Beobachtungen auf der Königlichen Universitäts-Sternwarte in Königsberg" [Astronomical observations at the Royal University Observatory in Königsberg]. Part 1, November 12, 1813 to December 31, 1814, pp. IX–X. Königsberg: Friedrich Nicolovius.

Bessel, Friedrich Wilhelm, [1816] "Astronomische Beobachtungen auf der Königlichen Universitäts-Sternwarte in Königsberg" [Astronomical observations at the Royal University Observatory in Königsberg]. Part 2, January 1 to December 31, 1815, pp. VIII–IX. Königsberg: Friedrich Nicolovius.

Brigham, E. Oran, [1974] *The fast Fourier transform*, Englewood Cliffs, NJ: Prentice-Hall.

Brooks, Charles E. P., and Carruthers, N., [1953] *Handbook of statistical methods in meteorology,* Meteorological Office, M.O. 538, London: Her Majesty's Stationery Office.

Burkhardt, H., [1904] "Trigonometrische interpolation (mathematische Behandlung periodischer Naturerscheinungen)," chapter 9 in *Encyklopädie der mathematischen Wissenchaften,* **II**:1, 1st half, pp. 642–693, Leipzig: B. G. Teubner. Translated into French with additional notes by E. Esclangon as « Interpolation trigonométrique, » chapter 27 in *Encyclopédie des sciences mathématiques,* **II, 5**:1, pp. 82–153, Paris: Gauthier-Villars, 1912.

Burrus, C. Sydney, and Parks, Dean P., [1985] *DFT/FFT and Convolution Algorithms*. New York, NY: Wiley-Intersicence.

Clairaut, Alexis Claude, [1754] « Sur l'orbite apparente du Soleil autour de la terre, en ayant égard aux perturbations produites par les actions de la lune & des planètes principales », Mémoires (Histoire) de l'Académie des Sciences, Paris, pp. 521–564. *See* esp. Article 4: « De la manière de convertir une fonction quelconque T de t en une série, telle que $A + B$ cos.$t + C$ cos.$2t + D$ cos.$3t$ + etc. », pp. 544–564.

Cooley, James W. & Tukey, John W. [1965] "An algorithm for the machine calculation of complex Fourier series," *Math. Comput.* **19**, 297–301.

Elliot, Douglas F., and Rao K. Ramamohan, [1982] *Fast transforms: algorithms, analyses, applications,* Orlando, FL : Academic Press.

Fourier, Jean-Baptiste Joseph, [1807] « Théorie de la propagation de la chaleur dans les solides », In *Joseph Fourier, 1768-1830; a survey of his life and work, based on a critical edition of his monograph on the propagation of heat, presented to the Institut de France i 1807.*, by Ivor Grattan-Guinness, & Jerome R Ravetz, 30-440. Cambridge, MA: The MIT Press, 1972.

Fourier, Jean-Baptiste Joseph, [1822] *Théorie Analytique de la Chaleur,* Paris : Firmin Didot.

Fourier, Jean-Baptiste Joseph, [1878] *The Analytical Theory of Heat*. Translated, with notes by Alexander Freeman. London, UK: Cambridge University Press.

Gauss, Carl Friedrich, [1866] "Nachlass: Theoria interpolationis methodo nova tractata," pp. 265–327, in *Carl Friedrich Gauss, Werke,* ***Band 3*** Königlichen Gesellschaft der Wissenschaften, Göttingen.

Gentleman, W. Morven, and Sande, Gordon, [1966] "Fast Fourier transforms—for fun and profit," Fall Joint Computer Conf., AFIPS, Proc., **29**, pp. 563–578.

Goldstine, Herman H., [1977] *A history of numerical analysis from the 16th through the 19th century,* New York, NY: Springer-Verlag.

Grattan-Guinness, Ivor, and Jerome R. Ravetz, [1972] *Joseph Fourier 1768–1830: a survey of his life and work,* Cambridge, MA: MIT Press.

Hamming, Richard W., [1973] *Numerical Methods for Scientists and Engineers,* New York, NY: McGraw-Hill.

Hann, Julius von, [1901] *Lehrbuch de Meteorologie,* 1st ed., Leipzig: C. H. Tauchnitz.

Heideman, Michael T., Johnson, Don H., and Burrus, C. Sydney [1984] "Gauss and the history of the fast Fourier transform," *IEEE ASSP Magazine,* October 1984, pp. 14–21.

Kämtz, L. F., [1831] *Lehrbuch der Meteorologie,* vol. 1, Halle: Gebauerachen Buchhandlung.

Kolba, Dean P., and Parks, Thomas W., [1977] "A prime factor FFT algorithm using high-speed convolution," *IEEE Transactions on Acoustics, Speech, and Signal Processing,* **25**:4, pp. 281–294.

Lagrange, Joseph-Louis, [1759] « Recherches sur la nature et la propagation du son », *Misc. Taurinensia*, **I** (Reprinted: *Œuvres de Lagrange,* **I**, ed. J. A. Serret, pp. 39–148, Paris: Gauthier-Villars, 1867).

Lees, Charles H. [1914] "Note on the connection between the method of least squares and the Fourier method of calculating the coefficients of trigonometrical series to represent a given series of observations of a periodic quantity," *Proc. Physical Society London* **XXVI**, article XXIX, December 1913–August 1914, pp. 275–278.

Lejeune Dirichlet, J. P. Gustav. « Sur la convergence des séries trigonométriques qui servent à représenter une fonction arbitraire entre deux limites données », *Journal für die reine und angewandte Mathematik* 4 (1829): 157–169.

McClellan, J. H., and Rader, C. M., [1979] *Number theory in digital processing,* Englewood Cliffs, NJ: Prentice-Hall.

Morgera, Salvatore D., and Krishna, Hari, [1989] *Digital signal processing,* Boston, MA: Academic Press.

Poisson, Siméon Denis, [1808] « Mémoire sur la propagation de la chaleur dans les corps solides; par M. Fourier. Présenté le 21 décembre 1807 à l'Institut national », [Summary & Review]. Nouveau bulletin des sciences, par la Société philomathique de Paris, **No. 6**, March 1808: 112-116.

Rabiner, Lawrence R., et al., [1972] "Terminology in digital signal processing," *IEEE Trans. Audio and Electroacoustics,* **AU–20**:5, pp. 322–337.

Rader, C. M., [1968] "Discrete Fourier transforms when the number of data samples is prime," *Proceedings of the IEEE* (Letters)*,* **56**, pp. 1107–1108.

Runge, C., [1903] "Über die Zerlegung empirisch gegebener periodischer Funktionen in Sinuswellen," *Zeitschrift für Mathematik und Physik,* **48**, pp. 443–456.

Runge, C., [1905] "Über die Zerlegung einer empirisch Funktionen in Sinuswellen," *Zeitschrift für Mathematik und Physik,* **52**, pp. 117–123.

Van Loan, Charles, [1992] *Computational frameworks for the fast Fourier transform*, Philadelphia, PA: Society for Industrial and Applied Mathematics (SIAM).

Wheeler, Gerald F., and Crummett, William P., [1987] "The vibrating string controversy," Am. J. Physics, 55(1) January 1987.

Winograd, Shmuel, [1976] "On computing the discrete Fourier transform," *Proceedings National Academy of Sciences,* **73**:4, pp. 1005–1006.

Winograd, Shmuel, [1978] "On computing the discrete Fourier transform," *Mathematics of computation,* **32**:141, pp. 175–199.

Figure 1 check: (Sande-Tukey N = 4):
{{1, 0, 0, 0}, {0, 0, 1, 0}, {0, 1, 0, 0}, {0, 0, 0, 1}}.
{{1, 1, 0, 0}, {1, -1, 0, 0}, {0, 0, 1, -I}, {0, 0, 1, I}}.
{{1, 0, 1, 0}, {0, 1, 0, 1}, {1, 0, -1, 0}, {0, 1, 0, -1}} // MatrixForm

```
{{1.,    1.,         1.,    1.       },
 {1.,    0. -1. i, -1.,    0. +1. i},
 {1.,   -1.,         1.,   -1.       },
 {1.,    0. +1. i, -1.,    0. -1. i}}
```

Crosscheck against naïve *N* = 4 DFT:
{{1, 1, 1, 1},
 {1, Exp[-1 2 Pi I/4], Exp[-2 2 Pi I/4], Exp[-3 2 Pi I/4]},
 {1, Exp[-2 2 Pi I/4], Exp[-4 2 Pi I/4], Exp[-6 2 Pi I/4]},
 {1, Exp[-3 2 Pi I/4], Exp[-6 2 Pi I/4], Exp[-9 2 Pi I/4]}} // MatrixForm

```
{{1.,    1.,         1.,    1.       },
 {1.,    0. -1. i, -1.,    0. +1. i},
 {1.,   -1.,         1.,   -1.       },
 {1.,    0. +1. i, -1.,    0. -1. i}}
```

Figure 3 check: (Winograd *N* = 4, exp-j2Pi/N, i.e., negative exponent, opposite to Winograd):
{{1, 0, 0, 0}, {0, 0, 1, 1}, {0, 1, 0, 0}, {0, 0, 1, -1}}.
{{1, 0, 0, 0}, {0, 1, 0, 0}, {0, 0, 1, 0}, {0, 0, 0, -I}}.
{{1, 1, 0, 0}, {1, -1, 0, 0}, {0, 0, 1, 0}, {0, 0, 0, 1}}.
{{1, 0, 1, 0}, {0, 1, 0, 1}, {1, 0, -1, 0}, {0, 1, 0, -1}} // MatrixForm

```
{{1.,    1.,         1.,     1.       },
 {1.,    0. -1. i, -1.,     0. +1. i},
 {1.,   -1.,         1.,    -1.       },
 {1.,    0. +1. i, -1.,     0. -1. i}}
```

Figure 4 check: (Winograd *N* = 3, exp-j2Pi/N, i.e., negative exponent, opposite to Winograd):
{{0, 1, 0}, {1, 0, 1}, {1, 0, -1}}.
{{1, 1, 0}, {1, 0, 0}, {0, 0, 1}}.
{{1, 0, 0}, {0, -3/2, 0}, {0, 0, -I Sqrt[3]/2}}.
{{1, 0, 1}, {1, 0, 0}, {0, 1, 0}}.
{{0, 1, 1}, {0, 1, -1}, {1, 0, 0}} // MatrixForm

```
{{1.,     1.,               1.             },
 {1.,    -0.5-0.866025 i, -0.5+0.866025 i},
 {1.,    -0.5+0.866025 i, -0.5-0.866025 i}}
```

Figure 5 check: (Elliot & Rao *N* = 3):
{{1, 1, 0, 0}, {0, 0, 1, 1}, {0, 0, 1, -1}}.

{{1, 0, 0, 0}, {0, 1, 0, 0}, {1, 0, 1, 0}, {0, 0, 0, 1}}.
{{1, 0, 0, 0}, {0, 1, 0, 0}, {0, 0, -1/2, 0}, {0, 0, 0, -I Sqrt[3]/2}}.
{{1, 0, 0}, {0, 1, 0}, {0, 1, 0}, {0, 0, 1}}.
{{1, 0, 0}, {0, 1, 1}, {0, 1, -1}} // MatrixForm

```
{{1.,      1.,                1.             },
 {1.,     -0.5-0.866025 i,  -0.5+0.866025 i},
 {1.,     -0.5+0.866025 i,  -0.5-0.866025 i}}
```

Crosscheck against naïve $N = 3$ DFT:
{{1, 1, 1}, {1, Exp[-2 Pi I/3], Exp[-2 2 Pi I/3]}, {1, Exp[-2 2 Pi I/3], Exp[-4 2 Pi I/3]}} // MatrixForm

```
{{1.,         1.,                1.             },
 {1.,        -0.5-0.866025 i,  -0.5+0.866025 i},
 {1.,        -0.5+0.866025 i,  -0.5-0.866025 i}}
```

Table 1 (Check Kämtz's DFT algorithm using his data):

x0=16.17	x4=16.27	x8=13.68	x12=12.19	x16=11.17	x20=12.99
x1=16.56	x5=15.61	x9=13.12	x13=11.94	x17=11.1	x21=14.09
x2=16.79	x6=14.86	x10=12.78	x14=11.66	x18=11.48	x22=14.93
x3=16.75	x7=14.19	x11=12.48	x15=11.39	x19=12.12	x23=15.59

y0=(x0+x1+x2+x3+x4+x5+x6+x7+x8+x9+x10+x11+x12+x13+x14+x15+x16+x17+x18+x19+x20+x21+x22+x23)/24
Out = 13.7462 (Kämtz 13.7463)

u=2 Pi/24	v2=Cos[2 u]	w3=Sin[3u]	v5=Cos[5u]
v=Cos[u]	w2=Sin[2u]	v4=Cos[4 u]	w5=Sin[5u]
w=Sin[u]	v3=Cos[3 u]	w4=Sin[4 u]	

y1=(((x1-x11-x13+x23)v+(x2-x10-x14+x22)v2+(x3-x9-x15+x21)v3+(x4-x8-x16+x20)v4+(x5-x7-x17+x19)v5+(x0-x12))+ I ((x1+x11-x13-x23)w+(x2+x10-x14-x22)w2+(x3+x9-x15-x21)w3+(x4+x8-x16-x20)w4+(x5+x7-x17-x19)w5+(x6-x18))/12
Out = 2.08865+1.64459 I (Kämtz 2.0886+1.6446 I)

y2=(((x1-x5-x7+x11+x13-x17-x19+x23)v2+(x2-x4-x8+x10+x14-x16-x20+x22)v4+(x0-x6+x12-x18))+
I ((x1+x5-x7-x11+x13+x17-x19-x23)w2+(x2+x4-x8-x10+x14+x16-x20-x22)w4+(x3-x9+x15-x21)))/12
Out = 0.509949+0.221058 I (Kämtz 0.5099+0.2211 I)

y3=(((x1-x3-x5+x7+x9-x11-x13+x15+x17-x19-x21+x23)v3+(x0-x4+x8-x12+x16-x20))+
I ((x1+x3-x5-x7+x9+x11-x13-x15+x17+x19-x21-x23)w3+(x2-x6+x10-x14+x18-x22)))/12
Out = -0.0971159-0.0734027 I (Kämtz -0.0971-0.0731 I)

Identities:

Symmetry: $$W_N^{n+N/2} = -W_N^n$$

Periodicity: $$W_N^{n+N} = W_N^n$$

$$W_N^{nk} = W_N^{nk \bmod N}$$

Kronecker product: $$(\mathbf{A} \otimes \mathbf{B})(\mathbf{C} \otimes \mathbf{D}) = \mathbf{AC} \otimes \mathbf{BD}$$