THE FFT: AN ALGORITHM THE WHOLE FAMILY CAN USE

The fast Fourier transform is one of the fundamental algorithm families in digital information processing. The author discusses its past, present, and future, along with its important role in our current digital revolution.

> A paper by Cooley and Tukey described a recipe for computing Fourier coefficients of a time series that used many fewer machine operations than did the straightforward procedure ... What lies over the horizon in digital signal processing is anyone's guess, but I think it will surprise us all. – Bruce P. Bogert, *IEEE Trans. Audio Electronics*, AU-15, No. 2, 1967, p. 43.

These days, it is almost beyond belief that there was a time before digital technology. It seems almost everyone realizes that the data whizzing over the Internet, bustling through our modems, or crashing into our cell phones is ultimately just a sequence of 0's and 1's—a digital sequence—that magically makes the world the convenient, high-speed place it is today. Much of this magic is due to a family of algorithms that collectively go by the name the *fast Fourier transform*. Indeed, the FFT is perhaps the most ubiquitous algorithm used today to analyze and manipulate digital or discrete data.

My own research experience with various flavors of the FFT is evidence of its wide range of applicability: electroacoustic music and audio-signal processing, medical imaging, image processing, pattern recognition, computational chemistry, error-correcting codes, spectral methods for partial differential equations, and last but not least, mathematics. Of course, I could list many more applications, notably in radar and communications, but space and time restrict. E. Oran Brigham's book is an excellent place to start, especially pages two and three, which contain a (nonexhaustive) list of 77 applications![1]

History

We can trace the FFT's first appearance, like so much of mathematics, back to Gauss.[2] His interests were in certain astronomical calculations (a recurrent area of FFT application) that dealt with the interpolation of asteroidal orbits from a finite set of equally spaced observations. Undoubtedly, the prospect of a huge, laborious hand calculation provided good motivation to develop

1521-9615/00/$10.00 © 2000 IEEE

DANIEL N. ROCKMORE
Dartmouth College

Discrete Fourier transforms

The fast Fourier transform efficiently computes the discrete Fourier transform. Recall that the DFT of a complex input vector of length N, $X = (X(0) \ldots, X(N-1))$, denoted $\hat{X}$, is another vector of length N given by the collection of sums

$$\hat{X}(k) = \sum_{j=0}^{N-1} X(j) W_N^{jk} \tag{1}$$

where $W_N = \exp\left(2\pi\sqrt{-1}/N\right)$. Equivalently, we can view this as the matrix-vector product $F_N \cdot X$, where

$$F_N = \left(\left(W_N^{jk}\right)\right)$$

is the so-called *Fourier matrix*. The DFT is an invertible transform with inverse given by

$$X(j) = \frac{1}{N} \sum_{k=0}^{N-1} \hat{X}(k)_j W_N^{-jk} . \tag{2}$$

Thus, if computed directly, the DFT would require N^2 operations. Instead, the FFT is an algorithm for computing the DFT in $O(N \log N)$ operations. Note that we can view the inverse as the DFT of the function

$$\frac{1}{N}\hat{X}(-k),$$

so that we can also use the FFT to invert the DFT.

One of the DFT's most useful properties is that it converts circular or cyclic convolution into pointwise multiplication, for example,

$$\widehat{X * Y}(k) = \hat{X}(k) \hat{Y}(k) \tag{3}$$

where

$$X * Y(j) = \sum_{l=0}^{n} X(l) Y(j-l) . \tag{4}$$

Consequently, the FFT gives an $O(N \log N)$ (instead of an N^2) algorithm for computing convolutions: First compute the DFTs of both X and Y, then compute the inverse DFT of the sequence obtained by multiplying pointwise $\hat{X}$ and $\hat{Y}$.

In retrospect, the idea underlying the Cooley-Tukey FFT is quite simple. If $N = N_1 N_2$, then we can turn the 1D equation (Equation 1) into a 2D equation with the change of variables

$$\begin{aligned} j &= j(a,b) = aN_1 + b, \; 0 \le a < N_2, \; 0 \le b < N_1 \\ k &= k(c,d) = cN_2 + d, \; 0 \le c < N_1, \; 0 \le d < N_2 \end{aligned} \tag{5}$$

Using the fact $W_N^{m+n} = W_N^m W_N^n$, it follows quickly from Equation 5 that we can rewrite Equation 1 as

$$\hat{X}(c,d) = \sum_{b=0}^{N_1-1} W_N^{b(cN_2+d)} \sum_{a=0}^{N_2-1} X(a,b) W_{N_2}^{ad} \tag{6}$$

The computation is now performed in two steps. First, compute for each b the inner sums (for all d)

$$\tilde{X}(b,d) = \sum_{a=0}^{N_2-1} X(a,b) W_{N_2}^{ad} \tag{7}$$

which is now interpreted as a subsampled DFT of length N_2. Even if computed directly, at most $N_1 N_2^2$ arithmetic operations are required to compute all of the $\tilde{X}(b,d)$. Finally, we compute $N_1 N_2$ transforms of length N_1:

$$\sum_{b=0}^{N_1-1} W^{b(cN_2+d)} \tilde{X}(b,d) \tag{8}$$

which requires at most an additional $N_1 N_1^2$ operations. Thus, instead of $(N_1 N_2)^2$ operations, this two-step approach uses at most $(N_1 N_2)(N_1 + N_2)$ operations. If we had more factors in Equation 6, then this approach would work even better, giving Cooley and Tukey's result. The main idea is that we have converted a 1D algorithm, in terms of indexing, into a 2D algorithm. Furthermore, this algorithm has the advantage of an in-place implementation, and when accomplished this way, concludes with data reorganized according to the well-known bit-reversal shuffle.

This "decimation in time" approach is one of a variety of FFT techniques. Also notable is the dual approach of "decimation in frequency" developed simultaneously by Gordon Sande, whose paper with W. Morven Gentleman also contains an interesting discussion on memory consideration as it relates to implementational issues.[1] Charles Van Loan's book discusses some of the other variations and contains an extensive bibliography.[2] Many of these algorithms rely on the ability to factor N. When N is prime, we can use a different idea in which the DFT is effectively reduced to a cyclic convolution instead.[3]

References

1. W.M. Gentleman and G. Sande, "Fast Fourier Transforms—For Fun and Profit," *Proc. Fall Joint Computer Conf. AFIPS*, Vol. 29, Spartan, Washington, D.C., 1966, pp. 563–578.
2. C. Van Loan, *Computational Framework for the Fast Fourier Transform*, SIAM, Philadelphia, 1992.
3. C.M. Rader, "Discrete Fourier Transforms When the Number of Data Points is Prime," *Proc. IEEE*, IEEE Press, Piscataway, N.J., Vol. 56, 1968, pp. 1107–1108.

a fast algorithm. Fewer calculations also imply less opportunity for error and therefore lead to numerical stability. Gauss observed that he could break a Fourier series of bandwidth $N = N_1 N_2$ into a computation of N_2 subsampled discrete Fourier transforms of length N_1, which are combined as N_1 DFTs of length N_2. (See the "Discrete Fourier transforms" sidebar for detailed information.) Gauss's algorithm was never published outside of his collected works.

The statistician Frank Yates published a less general but still important version of the FFT in 1932, which we can use to efficiently compute the Hadamard and Walsh transforms.[3] Yates's "interaction algorithm" is a fast technique designed to compute the analysis of variance for a 2^n-factorial design and is described in almost any text on statistical design and analysis of experiments.

Another important predecessor is the work of G.C. Danielson and Cornelius Lanczos, performed in the service of x-ray crystallography, another area for applying FFT technology.[4] Their "doubling trick" showed how to reduce a DFT on $2N$ points to two DFTs on N points using only N extra operations. Today, it's amusing to note their problem sizes and timings: "Adopting these improvements, the approximate times for Fourier analysis are 10 minutes for 8 coefficients, 25 minutes for 16 coefficients, 60 minutes for 32 coefficients, and 140 minutes for 64 coefficients."[4] This indicates a running time of about .37 $N \log N$ minutes for an N-point DFT!

Despite these early discoveries of an FFT, it wasn't until James W. Cooley and John W. Tukey's article that the algorithm gained any notice. The story of their collaboration is an interesting one. Tukey arrived at the basic reduction while in a meeting of President Kennedy's Science Advisory Committee. Among the topics discussed were techniques for offshore detection of nuclear tests in the Soviet Union. Ratification of a proposed United States–Soviet Union nuclear test ban depended on the development of a method to detect the tests without actually visiting Soviet nuclear facilities. One idea was to analyze seismological time-series data obtained from offshore seismometers, the length and number of which would require fast algorithms to compute the DFT. Other possible applications to national security included the long-range acoustic detection of nuclear submarines.

Richard Garwin of IBM was another participant at this meeting, and when Tukey showed him the idea, he immediately saw a wide range of potential applicability and quickly set to getting the algorithm implemented. He was directed to Cooley, and, needing to hide the national security issues, told Cooley that he wanted the code for another problem of interest: the determination of the spin-orientation periodicities in a 3D crystal of He^3. Cooley was involved with other projects, and sat down to program the Cooley-Tukey FFT only after much prodding. In short order, he and Tukey prepared a paper which, for a mathematics or computer science paper, was published almost instantaneously (in six months).[5] This publication, as well as Garwin's fervent proselytizing, did a lot to publicize the existence of this (apparently) new fast algorithm.[6]

The timing of the announcement was such that usage spread quickly. The roughly simultaneous development of analog-to-digital converters capable of producing digitized samples of a time-varying voltage at rates of 300,000 samples per second had already initiated something of a digital revolution. This development also provided scientists with heretofore unimagined quantities of digital data to analyze and manipulate (just as is the case today). The "standard" applications of FFT as an analysis tool for waveforms or for solving PDEs generated a tremendous interest in the algorithm a priori. But moreover, the ability to do this analysis quickly let scientists from new areas try the algorithm without having to invest too much time and energy.

Its effect

It's difficult for me to overstate FFT's importance. Much of its central place in digital signal and image processing is due to the fact that it made working in the frequency domain equally computationally feasible as working in the temporal or spatial domain. By providing a fast algorithm for convolution, the FFT enabled fast, large-integer and polynomial multiplication, as well as efficient matrix-vector multiplication for Toeplitz, circulant, and other kinds of structured matrices. More generally, it plays a key role in most efficient sorts of filtering algorithms. Modifications of the FFT are one approach to fast algorithms for discrete cosine or sine transforms, as well as Chebyshev transforms. In particular, the discrete cosine transform is at the heart of MP3 encoding, which gives life to real-time audio streaming. Last but not least, it's also one of the few algorithms to make it into the movies—I can still recall the scene in *No Way Out* where the image-processing guru declares that he will need to "Fourier transform the image" to help Kevin Costner see the detail in a photograph!

Even beyond these direct technological applications, the FFT influenced the direction of academic research, too. The FFT was one of the first instances of a less-than-straightforward algorithm with a high payoff in efficiency used to compute something important. Furthermore, it raised the natural question, "Could an even faster algorithm be found for the DFT?" (the answer is no[7]), thereby raising awareness of and heightening interest in the subject of lower bounds and the analysis and development of efficient algorithms in general. With respect to Shmuel Winograd's

lower-bound analysis, Cooley writes in the discussion of the 1968 Arden House Workshop on FFT, "These are the beginnings, I believe, of a branch of computer science which will probably uncover and evaluate other algorithms for high speed computers."[8]

Ironically, the FFT's prominence might have slowed progress in other research areas. It provided scientists with a big analytic hammer, and, for many, the world suddenly looked as though it were full of nails—even if this wasn't always so. Researchers sometimes massaged problems that might have benefited from other, more appropriate techniques into a DFT framework, simply because the FFT was so efficient. One example that comes to mind is some of the early spectral-methods work to solve PDEs in spherical geometry. In this case, the spherical harmonics are a natural set of basis functions. Discretization for numerical solutions implies the computation of discrete Legendre transforms (as well as FFTs). Many of the early computational approaches tried instead to approximate these expansions completely in terms of Fourier series, rather than address the development of an efficient Legendre transform.

Even now there are still lessons to learn from the FFT's development. In this day and age, where any new technological idea seems fodder for Internet venture capitalists and patent lawyers, it is natural to ask, "Why didn't IBM patent the FFT?" Cooley explained that because Tukey wasn't an IBM employee, IBM worried that it might not be able to gain a patent. Consequently, IBM had a great interest in putting the algorithm in the public domain. The effect was that then nobody else could patent it either. This did not seem like such a great loss because at the time, the prevailing attitude was that a company made money in hardware, not software. In fact, the FFT was designed as a tool to analyze huge time series, in theory something only supercomputers tackled. So, by placing in the public domain an algorithm that would make time-series analysis feasible, more big companies might have an interest in buying supercomputers (like IBM mainframes) to do their work.

Whether having the FFT in the public domain had the effect IBM hoped for is moot, but it certainly provided many scientists with applications on which to apply the algorithm. The breadth of scientific interests at the Arden workshop (held only two years after the paper's publication) is truly impressive. In fact, the rapid pace of today's technological developments is in many ways a testament to this open development's advantage. This is a cautionary tale in today's arena of proprietary research, and we can only wonder which of the many recent private technological discoveries might have prospered from a similar announcement.

The future FFT

As torrents of digital data continue to stream into our computers, it seems that the FFT will continue to play a prominent role in our analysis and understanding of this river of data. What follows is a brief discussion of future FFT challenges, as well as a few new directions of related research.

Even bigger FFTs

Astronomy continues to be a chief consumer of large FFT technology. The needs of projects like MAP (Microwave Anisotropy Project) or LIGO (Laser InterFerometer Gravitational-Wave Observatory) require FFTs of several (even tens of) gigapoints. FFTs of this size do not fit in the main memory of most machines, and these so-called *out-of-core* FFTs are an active area of research.[9]

As computing technology evolves, undoubtedly, versions of the FFT will evolve to keep pace and take advantage of it. Different kinds of memory hierarchies and architectures present new challenges and opportunities.

Approximate and nonuniform FFTs

For a variety of applications (such as fast MRI), we need to compute DFTs for nonuniformly spaced grid points and frequencies. Multipole-based approaches efficiently compute these quantities in such a way that the running time increases by a factor of

$$\log\left(\frac{1}{\epsilon}\right)$$

where ϵ denotes the approximation's precision.[10] Algebraic approaches based on efficient polynomial evaluation are also possible.[11]

Group FFTs

The FFT might also be explained and interpreted using the language of group representation theory—working along these lines raises some interesting avenues for generalization. One approach is to view a 1D DFT of length N as computing the expansion of a function defined on C_N, the cyclic group of length N (the group of integers mod N) in terms of the basis of irreducible matrix elements of C_N, which are precisely the familiar sampled exponentials: $e_k(m) = \exp(2\pi\sqrt{-1}km/N)$. The FFT is a highly efficient algorithm for computing the expansion in this basis. More generally, a function on

any compact group (cyclic or not) has an expansion in terms of a basis of irreducible matrix elements (which generalize the exponentials from the point of view of group invariance). It's natural to wonder if efficient algorithms for performing this change of basis exist. For example, the problem of efficiently computing spherical harmonic expansions falls into this framework.

The first FFT for a noncommutative finite group seems to have been developed by Alan Willsky in the context of analyzing certain Markov processes.[12] To date, fast algorithms exist for many classes of compact groups.[11] Areas of applications of this work include signal processing, data analysis, and robotics.[13]

Quantum FFTs

One of the first great triumphs of the quantum-computing model is Peter Shor's fast algorithm for integer factorization on a quantum computer.[14] At the heart of Shor's algorithm is a subroutine that computes (on a quantum computer) the DFT of a binary vector representing an integer. The implementation of this transform as a sequence of one- and two-bit quantum gates, now called the quantum FFT, is effectively the Cooley-Tukey FFT realized as a particular factorization of the Fourier matrix into a product of matrices composed as certain tensor products of two-by-two unitary matrices, each of which is a so-called local unitary transform. Similarly, the quantum solution to the Modified Deutsch-Josza problem uses the matrix factorization arising from Yates's algorithm.[15] Extensions of these ideas to the more general group transforms mentioned earlier are currently being explored.

That's the FFT—both parent and child of the digital revolution, a computational technique at the nexus of the worlds of business and entertainment, national security and public communication. Although it's anyone's guess as to what lies over the next horizon in digital signal processing, the FFT will most likely be in the thick of it. CiSE

Acknowledgment

Special thanks to Jim Cooley, Shmuel Winograd, and Mark Taylor for helpful conversations. The Santa Fe Institute provided partial support and a very friendly and stimulating environment in which to write this paper. NSF Presidential Faculty Fellowship DMS-9553134 supported part of this work.

References

1. E.O. Brigham, *The Fast Fourier Transform and Its Applications*, Prentice Hall Signal Processing Series, Englewood Cliffs, N.J., 1988.
2. M.T. Heideman, D.H. Johnson, and C.S. Burrus, "Gauss and the History of the Fast Fourier Transform," *Archive for History of Exact Sciences*, Vol. 34, No. 3, 1985, pp. 265–277.
3. F. Yates, "The Design and Analysis of Factorial Experiments," *Imperial Bureau of Soil Sciences Tech. Comm.*, Vol. 35, 1937.
4. G.C. Danielson and C. Lanczos, "Some Improvements in Practical Fourier Analysis and Their Application to X-Ray Scattering from Liquids," *J. Franklin Inst.*, Vol. 233, Nos. 4 and 5, 1942, pp. 365–380 and 432–452.
5. J.W. Cooley and J.W. Tukey, "An Algorithm for Machine Calculation of Complex Fourier Series," *Mathematics of Computation*, Vol. 19, Apr., 1965, pp. 297–301.
6. J.W. Cooley, "The Re-Discovery of the Fast Fourier Transform Algorithm," *Mikrochimica Acta*, Vol. 3, 1987, pp. 33–45.
7. S. Winograd, "Arithmetic Complexity of Computations," *CBMS-NSF Regional Conf. Series in Applied Mathematics, Vol. 33*, SIAM, Philadelphia, 1980.
8. "Special Issue on Fast Fourier Transform and Its Application to Digital Filtering and Spectral Analysis," *IEEE Trans. Audio Electronics*, AU-15, No. 2, 1969.
9. T.H. Cormen and D.M. Nicol, "Performing Out-of-Core FFTs on Parallel Disk Systems," *Parallel Computing*, Vol. 24, No. 1, 1998, pp. 5–20.
10. A. Dutt and V. Rokhlin, "Fast Fourier Transforms for Nonequispaced Data," *SIAM J. Scientific Computing*, Vol. 14, No. 6, 1993, pp. 1368–1393; continued in *Applied and Computational Harmonic Analysis,* Vol. 2, No. 1, 1995, pp. 85–100.
11. D.K. Maslen and D.N. Rockmore, "Generalized FFTs—A Survey of Some Recent Results," *Groups and Computation, II, DIMACS Ser. Discrete Math. Theoret. Comput. Sci., Vol. 28*, Amer. Math. Soc., Providence, R.I., 1997, pp. 183–237.
12. A.S. Willsky, "On the Algebraic Structure of Certain Partially Observable Finite-State Markov Processes," *Information and Control,* Vol. 38, 1978, pp. 179–212.
13. D.N. Rockmore, "Some Applications of Generalized FFTs (An Appendix with D. Healy)," *Groups and Computation II*, *DIMACS Series on Discrete Math. Theoret. Comput. Sci.*, Vol. 28, American Mathematical Society, Providence, R.I., 1997, pp. 329–369.
14. P.W. Shor, "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer," *SIAM J. Computing*, Vol. 26, No. 5, 1997, pp. 1484–1509.
15. D. Simon, "On the Power of Quantum Computation," *Proc. 35th Annual ACM Symp. on Foundations of Computer Science*, ACM Press, New York, 1994, pp. 116–123.

Daniel N. Rockmore is an associate professor of mathematics and computer science at Dartmouth College, where he also serves as vice chair of the Department of Mathematics. His general research interests are in the theory and application of computational aspects of group representations, particularly to FFT generalizations. He received his BA and PhD in mathematics from Princeton University and Harvard University, respectively. In 1995, he was one of 15 scientists to receive a five-year NSF Presidential Faculty Fellowship from the White House. He is a member of the American Mathematical Society, the IEEE, and SIAM. Contact him at the Dept. of Mathematics, Bradley Hall, Dartmouth College, Hanover, NH 03755; rockmore@cs.dartmouth.edu; www.cs.darthmouth.edu/~rockmore.