The Fast Fourier Transform and Its Applications

JAMES W. COOLEY, PETER A. W. LEWIS, AND PETER D. WELCH

Abstract—**The advent of the fast Fourier transform method has greatly extended our ability to implement Fourier methods on digital computers. A description of the alogorithm and its programming is given here and followed by a theorem relating its operands, the finite sample sequences, to the continuous functions they often are intended to approximate. An analysis of the error due to discrete sampling over finite ranges is given in terms of aliasing. Procedures for computing Fourier integrals, convolutions and lagged products are outlined.**

Historical Background

THE FAST Fourier transform algorithm has an interesting history which has been described in [3]. Time does not permit repeating this history here in detail. The essentials are, however, that until the recent publication of fast Fourier transform methods, computer programs were using up hundreds of hours of computer time with procedures requiring something proportional to N^2 operations to compute Fourier transforms of N data points. It is not surprising then that the "new" methods requiring a number of operations proportional to $N \log N$ received considerable attention and led to revisions in computer programs and in problem-solving techniques using Fourier methods. It was discovered later that the base 2 form of the fast Fourier transform algorithm had been published many years ago by Runge and König [10] and by Stumpff [12], [13]. These are authors whose works are widely read and their papers certainly were used by those computing Fourier series. How then could these important algorithms have gone unnoticed? The answer is that the papers of Runge, König, and Stumpff described primarily how one could use *symmetries* of the sine–cosine functions to reduce the amount of computation by factors of 4, 8, or even more. Relatively small portions of these papers mentioned the *successive doubling algorithm* which permitted one to take two Fourier analyses of N-point samples of data and combine them in N operations to obtain an analysis of a $2N$-point sampling of the same data. Successive application of this algorithm obviously yields an N-point Fourier analysis in $\log_2 N$ doublings, and therefore, takes $N \log_2 N$ operations. Thus, while the computational method using *symmetries* reduced the proportionality factor in the KN^2 operations required to transform an N-point sequence, the method based on the doubling algorithm took a number of operations proportional to $N \log_2 N$.

Manuscript received August 16, 1968.
The authors are with the IBM Watson Research Center, Yorktown Heights, N.Y.

It is most likely that with the relatively small values of N used in preelectronic computer days, the former methods were easier to use and took fewer operations. Consequently, the methods requiring $N \log N$ operations were neglected. With the arrival of electronic computers capable of doing calculations of Fourier transforms with large values of N, the $N \log N$ methods were overlooked and the well-known hand calculator methods were programmed for the computers. Perhaps there is something to be learned from this experience, namely, that there may exist numerical methods in the older literature which should be reappraised whenever computing devices undergo radical changes.

Fast Fourier Algorithms

In the interest of coherent presentation, the definitions and procedures utilized herein will be developed in blocks and shown as figures. Thus, the discrete Fourier series is defined in Fig. 1 with $A(n)$ being a sequence which gives the complex Fourier amplitudes as a function of frequency n. The $X(j)$, $j=0, 1, \cdots, N-1$ are regarded here as a complex sequence, and in a problem may represent a sampling of a signal at N sampling points. Also,

$$W_N = \exp\left\{\frac{2\pi i}{N}\right\} = \cos\frac{2\pi}{N} + i\sin\frac{2\pi}{N}$$

is the principle Nth root of unity and if we substitute the expression for W_N in terms of sines and cosines, we obtain the perhaps more familiar sine–cosine Fourier series. Complex Fourier series is employed for ease of notation and derivation of formulas. One should note next the inversion formula in Fig. 1, giving the $A(n)$'s in terms of the $X(j)$'s. Since $A(n)$ is also a Fourier series, an algorithm or a program for computing the $A(n)$'s from the $X(j)$'s can be used to compute the $X(j)$'s from the $A(n)$'s. Since $W_N^N=1$, the exponent of W_N is to be interpreted modulo N. This leads to an essential property of the sequences $X(j)$ and $A(n)$, i.e., that they are periodic functions of j and n, respectively, with period N.

It is shown in Fig. 2 that when N is a product, $N=rs$, the Fourier series can be calculated in a two-stage process. This is done just as though the sequences $A(n)$ and $X(j)$ were defined on two-dimensional $r \times s$ arrays with the array indices (j_1, j_0) and (n_1, n_0) being defined as shown. When we substitute for j and n in W_N^{jn}, and jn is reduced modulo N, it is found that the series can be

Discrete Fourier Series:

$$X(j) = \sum_{n=0}^{N-1} A(n) W_N^{jn}$$

where $W_N = \exp(2\pi i/N)$

Inversion Formula:

$$A(n) = \frac{1}{N} \sum_{j=0}^{N-1} X(j) W_N^{-jn}$$

Let us write: $X(j) \longleftrightarrow A(n)$

Periodicity:

$$W_N^N = 1, \quad W_N^{j+N} = W_N^j$$

$$W_N^j = W_N^{j \bmod N} \qquad 0 \leq j < N$$

$$X(j) = X(j \bmod N)$$

$$A(n) = A(n \bmod N)$$

Orthogonality:

$$\sum_{j=0}^{N-1} W_N^{nj} W_N^{-mj} = \begin{cases} N & \text{if } n = m \bmod N \\ 0 & \text{otherwise.} \end{cases}$$

Fig. 1. Definition of discrete Fourier transform.

Mapping of Indices:

Assume $N = r \times s$,

$$j \rightarrow (j_1, j_0)$$

$$j = j_1 r + j_0 \qquad j_0 = 0, 1, \ldots, r-1$$
$$\qquad j_1 = 0, 1, \ldots, s-1$$

$$n \rightarrow (n_1, n_0)$$

$$n = n_1 s + n_0 \qquad n_0 = 0, 1, \ldots, s-1$$
$$\qquad n_1 = 0, 1, \ldots, r-1$$

$$W^{jn} = W_N^{j_1 n_1 rs} \times W_N^{j_0 n_1 s} \times W_N^{j_1 n_0 r} \times W_N^{j_0 n_0}$$
$$= 1 \times W_r^{j_0 n_1} \times W_s^{j_1 n_0} \times W_N^{j_0 n_0}$$

$$X(j_1, j_0) = \sum_{n_0=0}^{s-1} \underbrace{\left\{ \sum_{n_1=0}^{r-1} A(n_1 n_0) W_r^{j_0 n_1} \right\} W_N^{j_0 n_0}}_{A_1(j_0, n_0)} W_s^{j_1 n_0}$$

$$A_1(j_0 n_0) = W_N^{j_0 n_0} \sum_{n_1=0}^{r-1} A(n_1, n_0) W_r^{j_0 n_1}$$

$$X(j_1, j_0) = \sum_{n_0=0}^{s-1} A_1(j_0, n_0) W_s^{j_1 n_0}$$

Fig. 2. Factorization into subseries.

Number of operations (multiply-adds) for N-point Fourier analysis is:

Number of operations $= N^2$

By factoring, $N = r \times s$:

Number of operations $= r^2 s + s^2 r = N \times (r+s)$.

By factoring $N = r_1 \times r_2 \times \ldots \times r_m$:

Number of operations $= N \times (r_1 + r_2 + \ldots + r_m)$.

For minimum, $r_1 = \ldots = r_m = r$, $N = r^m$, i.e., $m = \log_r N$:

Number of operations $= N \cdot r \cdot \log_r N = N \cdot \log_2 N \cdot \frac{r}{\log_2 r}$

r	r/log₂r
2	2.00
3	1.88
4	2.00
5	2.15
6	2.31
7	2.49
8	2.67
9	2.82
10	3.01

Fig. 3. Minimization of number of operations.

Let:

$$X_j \longleftrightarrow A_n \qquad j, n = 0, 1, \ldots, 2N-1$$

$$\left.\begin{array}{l} X_{2j} \longleftrightarrow A'_n \\ X_{2j+1} \longleftrightarrow A''_n \end{array}\right\} \quad j, n = 0, 1, \ldots, N-1$$

$$A_n = \frac{1}{2N} \sum_{j=0}^{2N-1} X_j W_{2N}^{-jn}$$

$$A_n = \frac{1}{2N} \left\{ \sum_{j'=0}^{N-1} X_{2j'} W_{2N}^{-2j'n} + X_{2j'+1} W_{2N}^{-(2j'+1)n} \right\}$$

But, $W_{2N}^2 = W_N$

$$A_n = \frac{1}{2} \left\{ \frac{1}{N} \sum_{j'=0}^{N-1} X_{2j'} W_N^{-j'n} + \frac{1}{N} \sum_{j'=0}^{N-1} X_{2j'+1} W_N^{-j'n} W_{2N}^{-n} \right\}$$

$$A_n = \frac{1}{2} \left\{ A'_n + A''_n W_{2N}^{-n} \right\}$$

$$A_{n+N} = \frac{1}{2} \left\{ A'_n - A''_n W_{2N}^{-n} \right\}$$

Fig. 4. Successive doubling method.

calculated by first summing over n_1 to form an intermediate array $A_1(j_0, n_0)$, and then summing over n_0. This gives $A_1(j_0, n_0)$ as a phase factor $W_N^{j_0 n_0}$ times a set of r-term Fourier series with coefficients $A(n_1, n_0)$. The final result is then a set of s-term Fourier series with $A_1(j_0 n_0)$ as coefficients. It is thus seen how the amount of computation is reduced from N^2 for the original series to $r^2 s + s^2 r = N(r+s)$ for the two-stage process. By iterating on this procedure, the amount of calculation for a more composite N is reduced as shown in Fig. 3 to be N times the sum of the factors of N. If N is arbitrary, it is seen here by this counting that the lowest number of calculations is obtained if all factors are equal to 3. However, by avoiding multiplications when the powers of W_N are simple numbers like ± 1 or $\pm i$, one can reduce the calculations even further for $r = 2$, 4, 8, and 16.

The successive doubling algorithm is the special case when all factors of N are equal to 2, i.e., $N = 2^M$. A separate derivation is given for this algorithm in Fig. 4. The basic formula which can be copied into a program with some simple logic for generating the indices appears on the bottom two lines.

To demonstrate the simplicity of the basic procedure, a complete operable program in FORTRAN is given in Fig. 5. The first half of the program including the "DO 7" loop performs a reordering of the data. The second half including the "DO 20" loop actually does the calculation. This program requires $\log_2 N$ evaluations of the sine and cosine and N multiplications by W. For $N = 1024$, this represents about 15 percent more operations ("operation" being defined here as a complex addition or multiplication) than would be required by the simple expedient of storing tables of powers of W_N. Then, after that, a more complicated logic, as used in one of the available SHARE programs, would yield an additional saving of 10 percent in arithmetic operations. However, bookkeeping takes about as much time as the arithmetic, so the present simple program is estimated to be only about 12 percent less efficient than a more optimally programmed FORTRAN program.

Fig. 6 shows a slightly different fast Fourier transform

```
      SUBROUTINE FFT(A,M)
      COMPLEX A(1024),U,W,T
      N= 2**M
      NV2 = N/2
      NM1 = N-1
      J=1
      DO 7 I=1,NM1
      IF(I.GE.J) GO TO 5
      T = A(J)
      A(J) = A(I)
      A(I) = T
    5 K=NV2
    6 IF(K.GE.J) GO TO 7
      J = J-K
      K=K/2
      GO TO 6
    7 J= J+K
      PI = 3.14159265358979
      DO 20 L=1,M
      LE = 2**L
      LE1 = LE/2
      U = (1.0,0.)
      W=CMPLX(COS(PI/LE1),SIN(PI/LE1))
      DO 20 J=1,LE1
      DO 10 I=J,N,LE
      IP = I+LE1
      T=A(IP)*U
      A(IP)=A(I)-T
   10 A(I)=A(I)+T
   20 U=U*W
      RETURN
      END
```

FIG. 5. Program for computing DFT by FFT method.

Mappings $j \longrightarrow (j_1, j_0)$

$n \longrightarrow (n_1, n_0)$

defined by

$n = r\,n_0 + s\,n_1 \quad (\text{mod } N,\ 0 \leq n < N)$

and

$j_0 = j \;(\text{mod } r)$

$j_1 = j \;(\text{mod } s)$

Then, $j = s \cdot (s)_r^{-1} j_0 + r(r)_s^{-1}\, j_1$

where $(\ )_r^{-1}$ denotes reciprocal mod r.

Then $W_N^{jn} = W_N^{rn_0 j}\, W_N^{sn_1 j} = W_s^{n_0 j}\, W_r^{n_1 j} = W_s^{n_0 j_1}\, W_r^{n_1 j_0}$

Let $A_1(j_0, n_0) = \sum_{n_1=0}^{r-1} A(n_1, n_0)\, W_r^{n_1 j_0}$

$X(j_1, j_0) = \sum_{n_0=0}^{s-1} A_1(j_0 n_0)\, W_s^{j_1 n_0}$

Fig. 6. Prime factor algorithm.

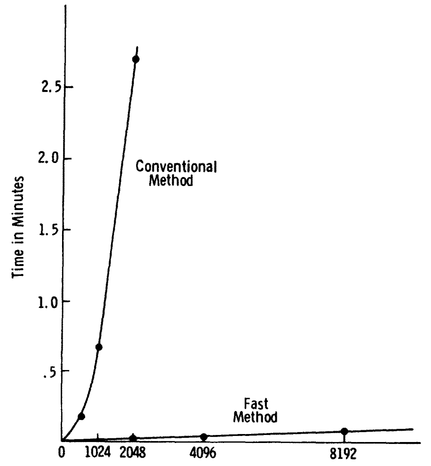

Fig. 7. Time required for calculation of Fourier transform of real data on IBM 7094 using FORTRAN with conventional and fast methods.

algorithm. This one requires that the factors of N be mutually prime and uses a different mapping of the one-dimensional indices j and n into the index pairs (j_1, j_0) and (n_1, n_0), respectively, from that given in Fig. 2. The result is a procedure quite similar to the one given above, except that the phase factor is missing in the calculation of $A_1(j_0, n_0)$. This can be used to advantage in combination with the previous algorithm. For example, to introduce an odd factor s, let $N = s \cdot r$ where r is a power of two. Then one can program the prime factor algorithm using the power of two subroutine for the r-point subseries.

The time required for computing Fourier series by a conventional program and by the FFT algorithm is illustrated in Fig. 7. It is seen here how quickly one can arrive at values of N which are so large as to make conventional methods unfeasible. The rate of growth with N of the fast methods shows that as technological advances permit the measurement and collection of data at increasing rates, the demands for computer speed rise only slightly more than linearly. Since advances in computer technology can be expected to increase at the same rate as the data collection, this indicates that the digital Fourier methods can be expected to continue to be feasible with technological advances. Such a prediction would not be possible if N^2 operation methods were necessary. Perhaps the future will even see radios with digital tuners.

The errors found in the calculation of a Fourier transform of a single complex exponential function are given in Table I. It can be shown that if one has data which are roughly evenly distributed, the FFT algorithm sequences the calculations in such a way that the numbers added on each successive intermediate step are approximately of the same expected value. This minimizes the error introduced by the shifting and chopping done in floating point calculations. A more careful error analysis of a sequentially ordered sum of products calculation would disclose an error whose expected value could be proportional to N, while the FFT method by the same estimation would give an error proportional to $\log_2 N$. To test this, the maximum error is divided by $\log_2 N$ in the third column of Table I and is found to follow this rule. Table II shows the results of a test on random data where the transform to frequency and back was effected and checked for accuracy. The same rate of growth of error with $\log_2 N$ instead of N is found.

To demonstrate a simple application of the FFT program (Fig. 8), data from a strain seismograph of the Rat Island, Alaska, earthquake were Fourier-analyzed by Dr. L. Alsop of IBM in 1966. The data were recorded at 2048 points over a $13\frac{1}{2}$-hour period after transients resulting from the initial shock had died out. A conventional program took 1567.8 seconds to compute the periodogram or estimated power spectrum (the modulus squared of the Fourier transform), which is shown in Fig. 9. The same task took 2.4 seconds with the FFT program and gave more accurate results. The result of

TABLE I

Calculated Maximum and rms Error in Computing the Fourier Transform of a Single Complex Exponential Function exp $(2\pi ijk/N)$, $j=0, 1, 2, \cdots, N-1$, $k=2$*

N	Errors in units of 10^{-8}			Time (minutes)
	Maximum Error	Max / $\log_2 N$	rms Error	
512	4.47	0.50	0.0133	0.008
1024	5.22	0.52	0.0072	0.017
2048	5.22	0.47	0.0036	0.037
4096	5.96	0.50	0.0019	0.085
8192	6.71	0.52	0.0010	0.175

* The program was compiled by IBSYS, FORTRAN IV and run on the IBM 7094. The time for calculating one Fourier transform is given in minutes.

TABLE II

Results of Transforming Random Data Distributed Evenly Between 0 and 1 to Frequency Data and Back*

N	Errors in Units of 10^{-8}			
	Maximum Error	Max / $\log_2 N$	rms Error	rms / $\log_2 N$
256	13.4	1.68	7.7	0.96
512	14.9	1.66	8.6	0.96
1024	17.8	1.78	9.6	0.96
2048	20.9	1.90	10.8	0.98
4096	24.5	2.04	13.0	1.08
8192	29.8	2.29	14.6	1.12

* The maximum and rms differences between the starting and resulting arrays are listed. These quantities, divided by $\log_2 N$, show the errors to be proportional to $\log_2 N$.

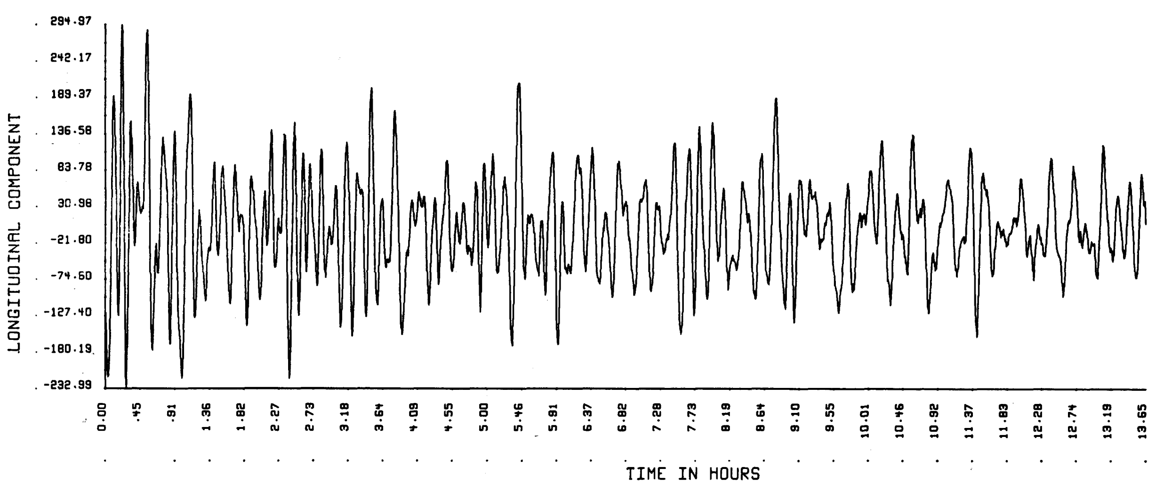

Fig. 8. Strain seismograph of Rat Island earthquake. $N=2048$; $T=13\frac{1}{2}$ hours.

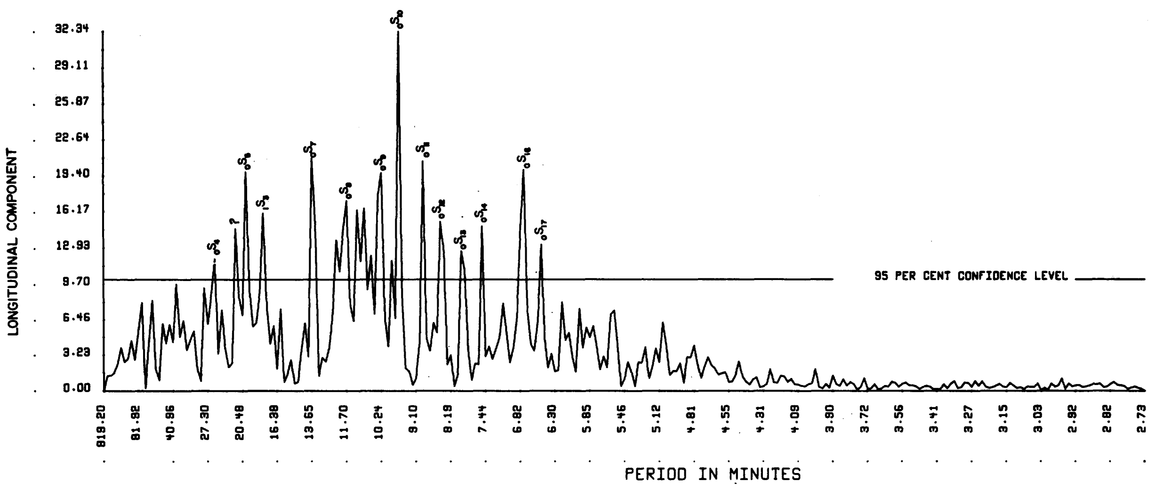

Fig. 9. Power spectrum of strain seismograph of Rat Island earthquake. $N=2048$; $T=13\frac{1}{2}$ hours.

the calculation is a set of possibly significant sharp peaks in the periodogram (power spectrum) corresponding to the natural modes of vibration of the earth. It should be noted that the 0.95 confidence band given is for individual values of $|A(n)|^2$.

Sampled Representation of Data

The data to be treated in Fourier analysis are often infinite in extent and/or continuous in the time or space domain and/or in the frequency domain. Of necessity one must represent and perhaps approximate such data in terms of the finite discrete sequences and their Fourier transforms, as was just discussed. Of fundamental importance, then, is the relationship between the "true" data and its sampled representation [4]. This relationship is expressed in the form of a theorem in Fig. 10. This theorem says that if $x(t)$ and $a(f)$ are an integral transform pair, one must first construct from them periodic aliased functions $x_p(t)$ and $a_p(f)$ with period T and F, respectively. The theorem further states that these periodized functions, each evaluated at N discrete points in its period, form a discrete finite Fourier transform pair.

As an illustration, a numerical example is worked out in Fig. 11 for the function $x(t)=0$ for $t<0$ and $x(t)=e^{-t}$ for $t>0$. In this example, $T=8$ and $N=16$ giving a sampling integral of $DT=\frac{1}{2}$. Since T is so large, $x(t)$ is indistinguishable from the aliased function $x_p(t)$, which is plotted in the upper curve. Sample values supplied to the discrete Fourier transform program are indicated by dots. Note that at the discontinuity at $t=0$, one must define $x_p(t)$ as the average of $x(-0)$ and $x(+0)$, since this is the value given by the inverse Fourier transform at a discontinuity. The correct transform $a(f)=(1+2\pi if)^{-1}$ is plotted in the solid line in the two lower graphs, the real part in one and the imaginary part in the other. The dots represent the computed values of $a_p(f)$ at $N=16$ points in the frequency period $F=1/DT=N/T=2$. Here, the effect of "aliasing" in the frequency domain is evident. The computed values follow a curve which is the sum of $a(f)$ and $a(f)$ displaced to the right by F. One can see that the error due to this aliasing is significant for f near $F/2$. If one seeks increased accuracy, it is obviously necessary to push the aliasing curve to the right by increasing F. To do this, DT is decreased by increasing N. The results for $N=32$ are given in Fig. 12. Here, the effect of the doubled F is shown to decrease the error in the range $0\leq f<F/2$.

The above theorem and computation also show how to compute Laplace transforms by using the Fourier transform. The example can be regarded as the case where one is computing the Laplace transform $a^*(s)$ of the step function $u(t)=0$ for $t<0$ and $u(t)=1$ for $t>0$. Then, letting the transform variable be $s=1+2\pi if$, the Laplace transform is the integral treated above and

$$a(f) = a^*(s) = \frac{1}{s}\cdot$$

Theorem: Suppose

$$x(t) = \int_{-\infty}^{\infty} a(f)e^{2\pi ift}\,df$$

We write: $x(t) \longleftrightarrow a(f)$.

Assume values of T, Δt, F, Δf, N such that

$$T = N\cdot\Delta t = 1/\Delta f$$
$$F = N\cdot\Delta f = 1/\Delta t$$

Construct periodic functions:

$x_p(t) = x(t) + x(t+T) + x(t-T) + x(t+2\ T) + x(t-2\ T) + \ldots$

for $0 \leq t < T$

$a_p(f) = a(f) + a(f+F) + a(f-F) + a(f+2\times F) + a(f-2\times F) + \ldots$

for $0 \leq f < F$.

Let:

$A_n = a_p(n\Delta f)$

$X_j = T\,x_p(j\Delta t)$

Then:

$X_j \longleftrightarrow A_n$

Fig. 10. Relation between discrete finite Fourier transform and integral Fourier transform.

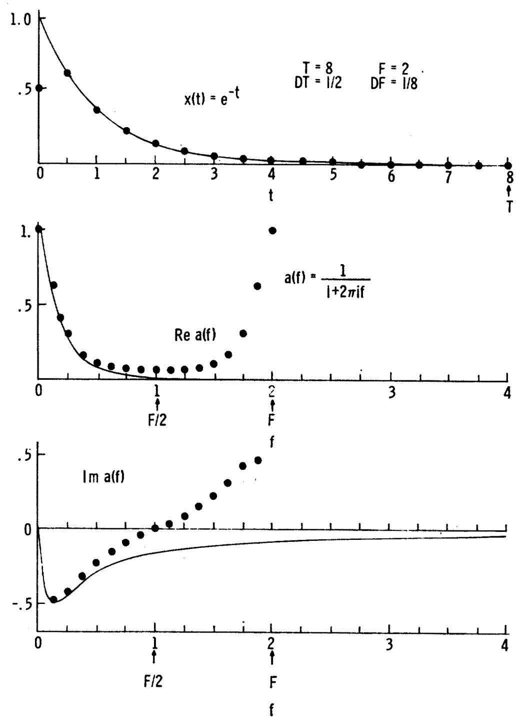

Fig. 11. Calculation of DFT of e^{-t} with $N=16$.

It should be noted that there is a serious problem in the above example. This is that the function $a_p(f)$ does not exist if the $a_p(f)$ is defined by the sum formula; the sum does not converge for $a^*(s)=1/s$. There is, however, a way of defining $a_p(s)$ so that it turns out to be the function computed here. This will not be discussed in detail now; we will just mention what one must do in the calculation to invert the Laplace transform, i.e., to compute $x(t)$ from $a(f)$. For this, instead of treating the transform pair $x(t)$, $a(f)$ as defined above, one can treat the pair $x(t)+x(-t)$ and $2\cdot\text{Re}\,a(f)$. The latter is a function for which the aliased function does exist. We are actually using the fact here that a *causal* function, i.e., one for which $f(t)=0$ for $t<0$, is completely determined

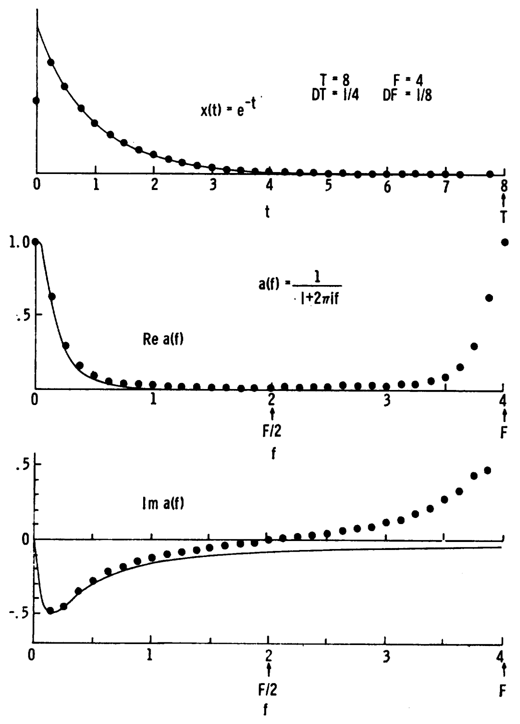

Fig. 12. Calculation of DFT of e^{-t} with $N=32$.

by either the real or imaginary parts of its Fourier transform. The real part of $a(f)$ happens to be summable when sampled.

Use of the Convolution Theorem

Most of the very important applications of the FFT involve the use of the convolution theorem. This theorem for discrete finite Fourier transforms is defined in Fig. 13. This simply states that the finite Fourier transform of the convolution is the product of the finite Fourier transforms of the two functions. A similar theorem holds for continuous functions, and in such a case, if the convolution integral were approximated by a sum formula, one would indeed obtain approximately the expression given in Fig. 13. However, as in the case of the inversion formula, these formulas are exact and are not simply numerical approximations of integrals.

An important aspect to be noted in connection with the convolution theorem is that while the convolution takes a number of operations proportional to N^2 by the direct method, the FFT method requires a number of operations proportional to $N \log N$. The latter method consists in Fourier-transforming the two sequences X_j and Y_j to the frequency domain, multiplying the frequency functions A_n and B_n, and transforming the product back. Note that, from here on, subscripts will be used on sequence elements.

For the convolution theorem to hold, it is necessary that sequences be periodic and of period N. Therefore, the convolution obtained is defined as if one is to define Y as the periodic repetition of its values on 0, $N-1$

Convolution:

$$Z_j = \sum_{k=0}^{N-1} X_k Y_{j-k}$$

i.e.,

$$\begin{pmatrix} Z_0 \\ Z_1 \\ Z_2 \\ \vdots \\ Z_{N-1} \end{pmatrix} = \begin{pmatrix} Y_0 & Y_{N-1} & Y_{N-2} & \cdots & Y_1 \\ Y_1 & Y_0 & Y_{N-1} & \cdots & Y_2 \\ Y_2 & Y_1 & Y_0 & \cdots & Y_3 \\ \vdots & \vdots & \vdots & & \vdots \\ Y_{N-1} & Y_{N-2} & Y_{N-3} & \cdots & Y_0 \end{pmatrix} \begin{pmatrix} X_0 \\ X_1 \\ X_2 \\ \vdots \\ X_{N-1} \end{pmatrix}$$

Theorem: If

$$X_j \leftrightarrow A_n$$
$$Y_j \leftrightarrow B_n$$
$$Z_j \leftrightarrow C_n$$

then

$$C_n = N \cdot A_n \cdot B_n$$

Fig. 13. Convolution theorem.

$$Z_j = \sum_{k=0}^{N-1} X_k Y_{j+k}$$

Then $C_n \leftrightarrow N A_{-n} B_n$

$$\begin{pmatrix} Z_0 \\ Z_1 \\ Z_2 \\ \cdot \\ \cdot \\ \cdot \\ Z_{N-1} \end{pmatrix} = \begin{pmatrix} Y_0 & Y_1 \cdots & Y_{N-3} & Y_{N-2} & Y_{N-1} \\ Y_1 & Y_2 \cdots & Y_{N-2} & Y_{n-1} & Y_0 \\ Y_2 & Y_3 \cdots & Y_{N-1} & Y_0 & Y_1 \\ & & & & \\ Y_{N-1} & Y_0 \cdots & Y_{N-4} & Y_{N-3} & Y_{N-2} \end{pmatrix} \begin{pmatrix} X_0 \\ X_1 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ X_{N-1} \end{pmatrix}$$

For $Z_0, Z_1, \ldots, Z_L$ to be correct, X must have L trailing 0's. If N' is length of original sequence,

$$\frac{\text{No. Ops. in sum of products method}}{\text{No. Ops. in F.T. method}} = \frac{(N'-L/2)(L+1)}{3(N'+L)\log_2(N'+L)}$$

Fig. 14. Lagged products.

when the subscript of Y_{j-k} goes out of the range 0 to $N-1$. In many actual situations, however, one really wants to assume the data to be zero or equal to their average value outside the region in which they are given. One then uses the simple expedient of lengthening the array by appending zeros or an average value and letting N of the present discussion be the length of the augmented array. The number of appended zeros, of course, depends upon how many lags or values of Z_j are wanted, and the number of nonzero values in the arrays.

Fig. 14 illustrates a slight variation on the convolution theorem obtained by replacing k with $-k$. As described in the figure, the last L values of the X_k vector will be multiplied by values in the periodic repetition of the Y_k vector in the calculation of $Z_0, Z_1, \cdots, Z_L$. Therefore, if the last L values of X_k are zero, the convolution will not be affected by the periodic nature of the sequences.

The formula for the speed ratio is given at the bottom of the figure; this is seen to be the number of operations in evaluating the convolution by the conventional

TABLE III

Number of Operations Required to Compute Lagged Products by Fourier Transform Method and Speed Ratio, i.e., the Number of Operations by the Conventional Divided by Fourier Transform Methods

$N+L$*	FT Method	Speed Ratio						
		$L=32$	$L=64$	$L=128$	$L=256$	$L=512$	$L=1024$	$L=2048$
128	2688	1.0	0.8					
256	6144	1.1	1.7	1.3				
512	13824	1.1	2.0	3.0	2.4			
1024	30720	1.1	2.0	3.5	5.4	4.3		
2048	67584	1.0	1.9	3.5	6.3	9.7	7.8	
4096	147456	0.9	1.8	3.4	6.5	11.6	17.8	14.2
8192	319488	0.8	1.7	3.2	6.3	11.9	21.4	32.8
16384	688128	0.8	1.5	3.0	6.0	11.6	22.1	39.6
32768	1474560	0.7	1.4	2.9	5.6	11.1	21.7	41.3
65536	3145728	0.7	1.4	2.7	5.3	10.6	20.9	40.7

* N = original record length. L = number of lags.

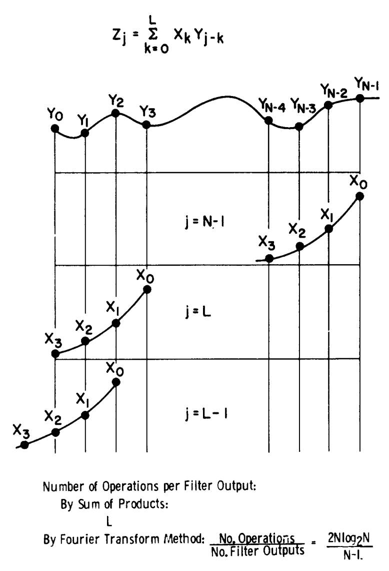
$Z_j = \sum_{k=0}^{L} X_k Y_{j-k}$
Y_0 Y_1 Y_2 Y_3 Y_{N-4} Y_{N-3} Y_{N-2} Y_{N-1}
j = N-1
X_3 X_2 X_1 X_0
j = L
j = L - 1
Number of Operations per Filter Output:
By Sum of Products:
L
By Fourier Transform Method: No. Operations / No. Filter Outputs = 2Nlog2N / N-L

Fig. 15. Digital filtering.

TABLE IV

Table of Values of $2N\log_2 N/(N-L)$*

L	N								
	16	32	64	128	256	512	1024	2048	8192
8	16	13	14	15	17	18			
16		20	16	16	17	19			
24			19	17	18	19			
32				18	18	19			
64				21	21	21			
96					26	22	22	23	
128					32	24	23	24	
1024									30

* $2N\log_2 N/(N-L)$ is the approximate number of complex multiply-adds required to compute each digital filter output for segments of length $N'=N-L$ where L is the number of filter weights. L is the number of operations per output by the direct method, and it is to be compared with table entries in evaluating the two methods.

method, divided by the number of operations taken by the FFT method when N' is the length of the original sequence and L is the number of lags required. Values of this ratio for various N' and L are given in Table III. It should be pointed out that for short sequences and few lags, the conventional method is more efficient. A turnover point is at $L=32$. For L less than or equal to 32, one will at best only improve the speed by a factor of 1.1, and for some N' values the FFT method will take longer. For $L=64$, unless N is as small as 64, one can obtain an improvement which as much as doubles the speed. Then, for high L, one gets high speed ratios, such as the 41 for $N'=32768$, $L=2048$.

An application to digital filtering is described in Fig. 15. This is again an example of a convolution calculation and corresponds to a situation where one may have a signal whose sampled values are the values of Y_j, and a filter with a point spread function described by a shorter sequence X_j of its sampled values. The output of such a filter is the sequence Z_j, the convolution of X and Y. Suppose the input signal is infinite in extent in both directions in the time domain. One then divides the input signal into time slices of N samples each, where N is chosen at one's convenience. Each such N-point sequence is Fourier-transformed and its transform is multiplied by the transform of X_j; the result is then transformed back to yield a filtered signal for that block of data. Now, due to the periodic nature of the convolution, the first L values of Z_j will have been convolved with the periodic repetition of the Y_j sequence and thereby contaminated. However, one can discard these values and take the blocks of N values of Y_j so that they overlap by an amount L.

To determine the block size N which minimizes the amount of arithmetic, one writes the formula for the number of operations per filter output. Two Fourier transforms are required, for which a conservative estimate of $2N \log_2 N$ operations is assumed. There will be $N-L$ good filter outputs, so that the number of operations per filter output is $2N \log_2 N/(N-L)$. This is to be compared with L, the number of operations by the conventional method. From the numerical values given in Table IV, it is seen that at $L=16$, the best N is

$N=128$; however, the amount of computation consists of 16 operations, the same as that required by the conventional method. As one goes to higher L, the optimal N increases and the speedup ratio goes up. For example, at $L=128$, a value of 1024 is obtained for N, so that the FFT method is almost six times as fast.

Summary

It has already been found that the use of the FFT methods has greatly increased the effectiveness of digital methods for a very wide range of problems such as spectral analysis, signal processing, Fourier spectroscopy, image processing, and the solution of differential equations.

Most efforts of the past several years have involved the reprogramming of previous procedures for the sake of economy in computer usage. Recent developments have been in the application of Fourier methods to problems which, due to computational effort, would not be tractable were it not for the use of the FFT method. Among these are the real-time digital Fourier methods, for which special-purpose computers are now being built. There are a number of areas in which a large amount of future development can be anticipated. Among these are the problems of the numerical solution of differential equations, image processing, and multiple time series analysis and filtering.

The preceding material has been a very rapid survey of the fast Fourier transform algorithm and of some of its applications. A much more detailed treatment is given in [5].

References

[1] L. E. Alsop and A. A. Nowroozi, "Faster Fourier analysis," *J. Geophys. Res.*, vol. 70, no. 22, p. 5482, 1966.
[2] W. T. Cochran *et al.*, "What is the fast Fourier transform?" *Proc. IEEE*, vol. 55, pp. 1664–1677, October 1967.
[3] J. W. Cooley, P. A. W. Lewis, and P. D. Welch, Historical notes on the fast Fourier transform," *Proc. IEEE*, vol. 55, pp. 1675–1677, October 1967.
[4] ——, "Application of the fast Fourier transform to computation of Fourier integrals, Fourier series, and convolution integrals," *IEEE Trans. Audio and Electroacoustics*, vol. AU-15, pp. 79–84, June 1967.
[5] ——, *The Fast Fourier Transform and Its Applications.* New York: Wiley, to be published.
[6] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Math. of Comp.*, vol. 19, no. 90, pp. 297–301, 1965.
[7] 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, pp. 365–380, 1942.
[8] W. M. Gentleman and G. Sande, "Fast Fourier transforms—for fun and profit," *Fall Joint Computer Conf.* 1966, *AFIPS Proc.*, vol. 29. Washington, D.C.: Spartan, pp. 563–578.
[9] I. J. Good, "The interaction algorithm and practical Fourier analysis," *J. Roy. Statist. Soc.*, series B., vol. 20, pp. 361–371, 1958; Addendum, vol. 22, pp. 372–375, 1960, MR 21 No. 1674; MR 23 No. A4231.
[10] C. Runge and H. König, "Die Grundlehren der Mathematischen Wissenschaften," Band XI, *Vorlesungen uber Numerisches Rechnen.* Berlin: Springer, 1924.
[11] P. Rudnick, "Note on the calculation of Fourier series," *Math. of Comp.*, vol. 20, no. 95, pp 429–430, 1966.
[12] K. Stumpff, *Grundlagen und Methoden der Periodenforschung.* Berlin: Springer, 1937.
[13] ——, *Tafelin und Aufgaben zur Harmonischen Analyse unde Periodogrammrechnung.* Berlin: Springer, 1939.
[14] L. H. Thomas, "Using a computer to solve problems in physics," *Applications of Digital Computers.* New York: Ginn, 1963.

Methods for Processing Complex Waveforms Through Filters for Use in Monte Carlo Noise Studies

MURLAN S. CORRINGTON, FELLOW, IEEE, T. C. HILINSKI, AND W. B. SCHAMING

Abstract—**Any N-pole linear lumped-constant time-invariant network can be represented by a linear homogeneous difference equation. If an arbitrary input waveform is sampled at equally spaced time intervals, so that eight or more samples are available at the highest significant frequency present, the output waveform can be computed in a simple manner from the difference equation. If K input samples are to be filtered, the saving in computer time, when compared to the usage of the convolution integral, is equal to K. The method is illustrated by the simulation of an FM receiver, operating at threshold, in the presence of Gaussian noise.**

Manuscript received August 14, 1968; revised September 4, 1968.
M. S. Corrington and W. B. Shaming are with the Radio Corporation of America, Camden, N. J.
T. C. Hilinski was with the Radio Corporation of America, Camden, N. J. He is now with MITRE Corporation, McLean, Va.

Introduction

Digital-computer simulation of electrical networks responding to signals contaminated by noise can be done by numerical integration using the convolution integral. This method is not very useful if long streams of sampled data are to be filtered, since the number of points required in the integrand increases linearly with time, and the computer time required is too great. It is not possible to update the previous calculation after each new input sample; all of the ordinates of the integrand must be recomputed and integrated each time.