NASA TECHNICAL NOTE

NASA TN D-4510

c.1

NASA TN D-4510

DIGITAL SPECTRAL ANALYSIS

by Anthony J. Villasenor
Goddard Space Flight Center
Greenbelt, Md.

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION • WASHINGTON, D. C. • JUNE 1968

DIGITAL SPECTRAL ANALYSIS

By Anthony J. Villasenor

Goddard Space Flight Center
Greenbelt, Md.

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

For sale by the Clearinghouse for Federal Scientific and Technical Information
Springfield, Virginia 22151 – CFSTI price $3.00

CONTENTS

DIGITAL SPECTRAL ANALYSIS

by
Anthony J. Villasenor
Goddard Space Flight Center

INTRODUCTION

Among the functions of the Test and Evaluation program at Goddard Space Flight Center are the following:

1. To evaluate the structural properties of spacecraft and spacecraft components by using spectral statistical analysis techniques to analyze vibration test data.

2. To simulate the spacecraft's solar environment as accurately as possible by using radiation sources with spectra that are matched with the true solar spectrum as determined by interferometric and spectroscopic techniques.

Test data from these two areas, structures and solar simulation, are sent to a digital computer facility for spectral analysis. It was natural, therefore, that the requirements from these areas led to this investigation of methods for efficient spectral analysis on a digital computer.

This paper presents some mathematical considerations of spectral analysis and some FORTRAN computer programs that use the Fast Fourier Transform algorithm of Cooley and Tukey. These programs compute Fourier amplitude and phase spectra, cross-power spectra, auto- and cross-correlation, and filtered spectra, as well as some other frequency domain functions.

SAMPLING CONSIDERATIONS

Let $f(t)$ be a continuous function of t. The first step in analyzing $f(t)$ by digital computer is to sample it and obtain a finite set of discrete points X_i ($i = 1, 2, 3 \ldots, N$), which will represent the function $f(t)$ in the computer. The process of sampling revolves around the Sampling Theorem (Reference 1), which states that the rate of sampling should be at least twice the maximum frequency contained in the data. Twice the maximum frequency is actually a bare minimum, and experience has shown that a sampling rate of five times the maximum frequency is usually adequate for most applications, where the value of the maximum frequency is not precisely known. Insufficiently sampled signals produce spectra containing "aliased" frequencies. For example,

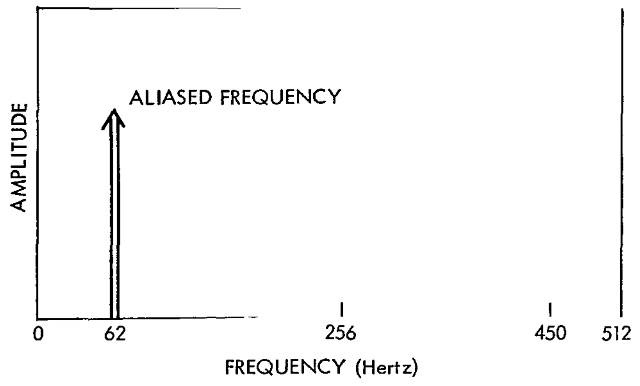


Figure 1—Sinusoid of 450 Hertz sampled 512 times per second.

Figure 1 depicts the Fourier amplitude spectrum of the curve

$$f(t) = \sin(2\pi 450\, t/512).$$

The spike occurs at f = 62 Hertz in the spectrum. The Aliasing occurs because the sampling rate of 512 Hertz was not enough to detect the true 450-Hertz signal and instead treated it as a 62-Hertz signal. We should have used a rate at least 2 × 450 or 900 samples per second. The rate of 512 samples per second gives rise to a maximum resolvable (cutoff) frequency f_c of 256 Hertz. The particular value of 62 comes from $f = 2f_c - 450 = 512 - 450 = 62$. This relation is due to the fact that the amplitude of any frequency f in a signal sampled at $2f_c$ samples per second equals the amplitude of the aliased frequency $2Nf_c \pm f$, where N is the number of points in the digitized sample. Thus for a time $t = f_c/2$

$$\cos(2\pi f t) = \cos\left(\frac{2\pi f_c N \pm f}{2 f_c}\right) = \cos\left(\frac{\pi f}{f_c}\right).$$

In other words, the Fourier coefficient of the frequency f is the same as the Fourier coefficient of the frequency $2Nf_c \pm f$ for the sampling rate of $2f_c$ samples per second.

DIGITAL SPECTRAL FUNCTIONS

Having sampled the function $f(t)$, we can perform various general mathematical operations on the data.

The *Fourier Amplitude Spectrum* describes the frequency content of the data in the form of complex amplitude versus frequency. The term "frequency" may be misleading because it refers not to frequencies of our original function $f(t)$ but rather to the number of times a particular sinusoid occurs within our finite sampled data. In fact, when working with a computer we are dealing with a Fourier Series expansion around the data even though we intend it to be a Fourier Integral expansion of $f(t)$.

The amplitude is given by

$$|R_k + i\, I_k|,$$

where

$$R_k + i\, I_k = \frac{1}{N}\sum_{j=1}^{N} X_j\ e^{-2\pi i k(j-1)}$$

The *Fourier Phase Spectrum* describes the phase content of our data in the form of phase angle (0 to 360 degrees or -180 to +180 degrees) versus frequency. The phase normally indicates changes of state of the $f(t)$ generator. In the case of interferometer data, an abrupt change in phase means that the light source has changed its temperature relative to the detector. In vibration data, an abrupt phase change indicates a resonance. The phase is computed as

$$\theta_k = \arctan (I_k/R_k).$$

The *Power Spectrum* is a description of the relative power of $f(t)$ as a function of frequency. The Power Spectrum $P(w)$ is the square of the Fourier Amplitude Spectrum,

$$P_k = R_k^2 + I_k^2 = (R_k + i\, I_k)(R_k - i\, I_k)$$

(Reference 2). The adjective "power" comes from an electrical analogy. The total energy E of a circuit is given by

$$E = \int_{\infty}^{\infty} P dt,$$

where P is the power of the circuit in watts. But $P = i^2R$, so that

$$E = \int_{-\infty}^{\infty} i^2 R d t.$$

If we set R to be a unit resistance, and the current i to be a time-varying function $x(t)$, then by Parseval's equality,

$$E = \int_{-\infty}^{\infty} |x(t)|^2 \, dt = \int_{-\infty}^{\infty} |F(w)|^2 \, dw,$$

where $F(w)$ is the Fourier amplitude spectrum of the current $x(t)$. The quantity $|F(w)|^2$ is thus called the power spectral density or the power spectrum.

When $x(t)$ is a finite, discrete function sampled at time intervals dt, we have a slightly different mathematical formulation. In this case, we can speak of the average power P_a over a finite time interval T of the current $x(t)$. The formulation is

$$P_a = E/T = (1/T) \int_t^{t+T} x(t)^2 \, dt .$$

If N samples are taken of the signal $x(t)$, then $T = Ndt$ and

$$P_a = (1/N) \sum_{j=1}^{N} x_j^2 .$$

We can now cite Parseval's equality, which holds that

$$\sum_{i=1}^{N} |x_i|^2 = N \sum_{k=1}^{N} |A_k|^2 .$$

Substituting this in the above,

$$P_a = \sum_{k=1}^{N} |A_k|^2 .$$

Hence the average power spectral density is $|A_k|^2$.

The power spectrum is used, for example, in acoustics to determine the system gain factor by finding the ratio of the output-power to the input-power spectrum. The energy or total power is also of frequent interest.

The *Auto-correlation* function describes the degree of interaction that a function has with itself at various intervals along the time or distance axis. Auto-correlation for finite, discrete data is presented in the form of correlation coefficient versus lag number. The correlation coefficient at any lag varies from 1, if the function is exactly duplicated at this lag, to zero, if the function is completely uncorrelated with itself at this lag, and to -1 if the function cancels itself out at this lag. For example, any two points of a straight line are always exactly identical, hence the auto-correlation is everywhere 1. A sinusoid is in phase with itself once every cycle, and 180 degrees out of phase once every cycle, so the auto-correlation is a cosine curve with amplitude 1. Figure 2 shows examples of auto-correlations and power spectra.

For a finite discrete data set x_i ($i = 1, 2, 3, \ldots, N$), sampled at equispaced intervals, the auto-correlation is computed as a function of "lag" number. For lag = 0, the data "comb" of N points is multiplied by itself; for lag = 1, the data comb is shifted over one point and N - 1 multiplications are performed; for lag = 2, the comb is shifted again for N - 2 multiplications. The process continues for as many lags as desired, usually up to 10 percent of N.

The auto-correlation function for finite discrete data is given by

$$R(L) = \frac{1}{R(0)} \frac{1}{N-L} \sum_{j=1}^{N-L} X_j X_{j+L},$$

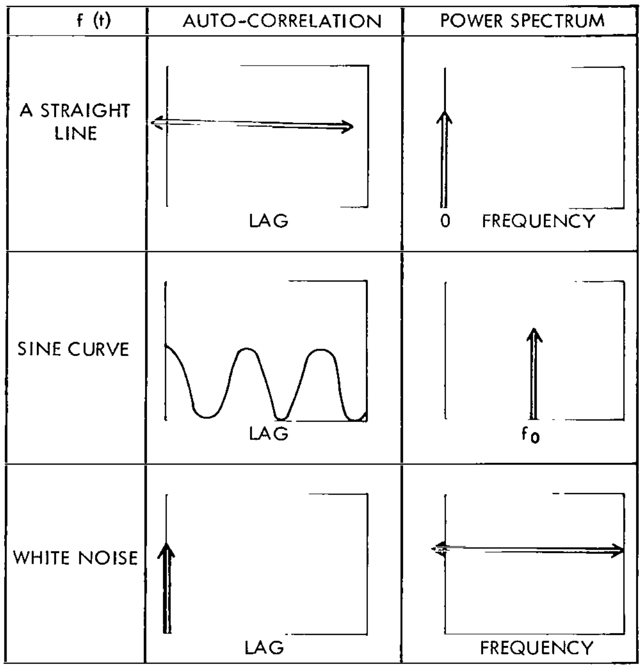


Figure 2—Examples of autocorrelation and power spectrum functions.

where L = lag number. Note the normalizing factor $1/R(0)$ without which this would be called an auto-covariance function. The background in statistics – variance, correlation coefficient – is plain. In fact, $R(0)$ is the sample estimate of the true mean-square value in the data, and is related to the sample variance s^2 by $R(0) = [(N - 1)/N]\, s^2$ (Reference 1).

An important relationship exists between the auto-correlation function $R(L)$ and the power spectral density $P(w)$: they are integral transform pairs. That is,

$$R(L) = \int_{-\infty}^{\infty} P(w)\, e^{2\pi i w}\, dw,$$

and

$$P(w) = \int_{-\infty}^{\infty} R(L)\, e^{-2\pi i L}\, dL.$$

Proof of this relationship is found in the literature under the heading "Wiener-Khintchine Theorem" (Reference 2). This relationship is important because the usual procedure for digitally computing power spectra is to compute first the autocovariance and then transform it to the frequency domain to obtain a first estimate of the power spectrum. The result is only an approximation because the auto-covariance was computed for a finite number of lags, whereas the true auto-covariance is an infinite function.

The *Cross-correlation* indicates the degree of relationship between two data samples in the form of correlation coefficient versus lag number. Again, complete correlation is 1, complete uncorrelation is 0, and anti-correlation is -1. Cross-correlation differs from auto-correlation, being defined as an imaginary quantity. The first step in this definition is to form $R_{xy}(L)$ and $R_{yx}(L)$, where

L = lag number,

$$R_{xy}(L) = \frac{1}{\sqrt{R_x(0)}\ \sqrt{R_y(0)}}\ \frac{1}{N-L} \sum_{n=1}^{N-L} x_n\, y_{n+L},$$

$$R_{yx}(L) = \frac{1}{\sqrt{R_x(0)}\ \sqrt{R_y(0)}}\ \frac{1}{N-L} \sum_{n=1}^{N-L} y_n\, x_{n+L}.$$

Then the even and odd parts of the cross-correlation function are given by

$$E_{xy}(L) = \frac{1}{2}\,[R_{xy}(L) + R_{yx}(L)],$$

$$O_{xy}(L) = \frac{1}{2}\,[R_{xy}(L) - R_{yx}(L)].$$

The *Cross Spectral Density* describes the frequency dependence between two signals. If two signals are mutually independent and uncorrelated, the cross-spectrum is zero, but if they are mutually dependent, the cross-spectrum is not zero and may indicate a frequency resonance for the device connecting the two signals. The cross-spectrum S_{xy} of two signals $x(t)$ and $y(t)$ is a complex number with a real part (the co-spectrum) and an imaginary part (the quadrature spectrum) The co-spectrum C_{xy} depicts the in-phase relationship of two signals, while the quadrature spectrum Q_{xy} depicts the out-of-phase relationship. The co-spectrum is traditionally computed as a cosine transform of the even part of a cross-correlation, and the quadrature spectrum as a sine transform of the odd part.

CO-SPECTRUM: $$C_{xy}(k) = \frac{2}{N}\left\{E_{xy}(0) + 2\sum_{\ell=1}^{L} E_{xy}(\ell)\cos\frac{\pi \ell k}{L^M} + (-1)^k E(L^M)\right\}$$

QUADRATURE SPECTRUM: $Q_{xy}(k) = \frac{4}{N}\left\{ \sum_{\ell=1}^{L^M-1} O_{xy}(\ell) \sin \frac{\pi \ell k}{L^M} \right\}$,

CROSS POWER SPECTRUM: $S_{xy} = C_{xy} - i\, Q_{xy}$,

where the cross-correlation is $E_{xy} + i\; Q_{xy}$. However, it is also possible to first compute the cross-spectral density of two signals by direct Fourier analysis, obtain $F_x(w)$ and $F_y(w)$, then combine these two series to get the cross-spectral density

$$S_{xy}(w) = F_x(w)\ \overline{F_y(w)},$$

where $\overline{F_y(w)}$ denotes the complex conjugate of $F_y(w)$ (Reference 3). Then, by applying the Wiener-Khintchine theorem, we immediately get the cross-correlation function R_{xy} as the inverse transform of S_{xy}.

FAST FOURIER TRANSFORM ALGORITHM

Since these spectral functions involve Fourier analysis computations, any algorithm that offers an efficient means of computing spectra on a digital computer is much to be desired. There are several sources of such algorithms in the literature. Two of these appear in a book (Reference 3). One algorithm by Goertzel makes a complex Fourier analysis of real data, using a simple matrix manipulation for obtaining sines and cosines. The other algorithm, by Southworth, first computes the auto-correlation for some percentage of the data, then performs a cosine Fourier transform of the result in order to obtain the power spectrum. This is the method proposed by Blackman and Tukey (Reference 4).

Both of these methods have drawbacks. Goertzel's program is exceptionally efficient, but still consumes too much computer time. Southworth's program loses phase information and can compute only a fraction of the total frequency content of a given data set.

More recently, James Cooley and John Tukey (Reference 5) introduced an algorithm that is hundreds of times faster than either of these methods. Called the Fast Fourier Transform, this algorithm analyzes or synthesizes a complex array of points whose size is (or can be built up with zeros to give) a power of 2. That is, their algorithm is set up to analyze 2^m points, where m is any positive integer. If the data set contains 1000 points, for example, it should be extended to $1024 = 2^{10}$ points by adding 24 zero-valued points. The Cooley-Tukey algorithm is based on the relationship existing between two analyses of N points each and one analysis of $2N$ points. That is, if we desire a Fourier analysis of the $2N$ points X_j ($j = 1, 2, 3, \ldots, 2N$), it is faster to make one analysis of the odd-index points (of which there are N) and another of the even-indexed points, and then merge the two results to obtain a single analysis covering all 2N points. Since this relation holds between one 2N series and two N-point series, then certainly each of the N-points analyses can in turn be obtained by pairs of N/2-point analyses, and so on. This successive halving accounts for the base 2. Cooley and Tukey have estimated that this method if $N/\log_2 N$ times faster than conventional methods. Note that the errors associated with digital computation are also reduced by this factor of $N/\log_2 N$.

Thus, the plan in the Fast Fourier Transform is to generate a 2N-Point analysis from two N-point analyses. This procedure is straightforward. Given 2N complex data points X_j (j = 1, 2, 3 . . . , 2N), we can split these points into an "odd" array indexed 1, 3, 5 . . . , 2N - 1, and an "even" array indexed 2, 4, 6 . . . , 2N. Let sequences of A_k's and AA_k's represent the spectra of these arrays, respectively. Then, if we consider separately the odd and even indexed points in the expression for the "total" representation, by substituting 2j - 1 and 2j in place of j, we will eventually end with a relationship between the A_k's, the AA_k's and all the C_k's which are the Fourier coefficients of the 2N points. The representations would be as follows:

Odd indexed points:

$$X_{2j-1} = \sum_{k=1}^{N} A_k \, e^{2\pi i(j-1)(k-1)/N},$$

Even indexed points:

$$X_{2j} = \sum_{k=1}^{N} AA_k \, e^{2\pi i(j-1)(k-1)/N},$$

where j = 1, 2, 3, . . . , N. In a 2N-point analysis, the series would be

$$X_j = \sum_{k=1}^{2N} C_k \, e^{2\pi i(j-1)(k-1)/2N}. \qquad (j = 1, 2, 3 - - - , 2N)$$

(The j and k indices range from 1 to N rather than the usual 0 to N - 1 to simplify programming notation.) If now we consider the even and odd indexed points in this last summation,

Odd:

$$X_{2j-1} = \sum_{k=1}^{2N} C_k \; e^{2\pi i(2j-2)(k-1)/2N}$$

$$= \sum_{k=1}^{2N} C_k \; e^{2\pi i(j-1)(k-1)/N}.$$

Even:

$$X_{2j} = \sum_{k=1}^{2N} C_k \; e^{2\pi i(2j-1)(k-1)/2N},$$

$$= \sum_{k=1}^{2N} C_k \; e^{2\pi i(j-1)(k-1)/N} \cdot e^{2\pi i(k-1)/2N}.$$

Replace the index k by $k' = k - N$ in the second half of each sum above (i.e. for $k = N+1$ to $k = 2N$).

Odd:

$$X_{2j-1} = \sum_{k=1}^{N} C_k \; e^{2\pi i(j-1)(k-1)/N} + \sum_{k'=1}^{N} C_{k'+N} \; e^{2\pi i(j-1)(k'+N-1)/N},$$

$$= \sum_{k=1}^{N} C_k \; e^{2\pi i(j-1)(k-1)/N} + \sum_{k'=1}^{N} C_{k'+N} \; e^{2\pi i(j-1)(k'-1)/N},$$

since $e^{2\pi i(j-1)N/N} = 1$.

Even:

$$X_{2j} = \sum_{k=1}^{N} C_k \; e^{2\pi i(j-1)(k-1)/N} \quad e^{2\pi i(k-1)/2N}$$

$$+ \sum_{k'=1}^{N} C_{k'+N} \; e^{2\pi i(j-1)(k-1)/N} \quad e^{2\pi i(N+k'-1)/2N},$$

$$= \sum_{k=1}^{N} C_k \, e^{2\pi(j-1)(k-1)/N} \, e^{2\pi i(k-1)/2N}$$

$$- \sum_{k'=1}^{N} C_{k'+N} \, e^{2\pi i(j-1)(k'-1)/N} \, e^{2\pi i(k'-1)/2N},$$

since

$$e^{2\pi i}(N/2N) = e^{\pi i} = -1.$$

In these sums, k and k' are independent indices, so we can collect terms to get

Odd: $$X_{2j-1} = \sum_{k=1}^{N} (C_k + C_{k+N}) \, e^{2\pi i(j-1)(k-1)/N}.$$

Even: $$X_{2j} = \sum_{k=1}^{N} (C_k - C_{k+N}) \, e^{2\pi i(j-1)(k-1)/N} \, e^{2\pi i(k-1)/2N}.$$

Comparing this result with the definition of A_k and AA_k, we have

$$A_k = C_k + C_{k+N},$$

$$AA_k = (C_k - C_{k+N}) \, e^{2\pi i(k-1)/2N}.$$

Solving this system of equations finally gives

$$C_k = \frac{1}{2}\left(A_k + AA_k \, e^{-2\pi i(k-1)/2N}\right),$$

$$C_{k+N} = \frac{1}{2}\left(A_k - AA_k \, e^{-2\pi i(k-1)/2N}\right),$$

where $k = 1, 2, 3 \ldots, N$.

These equations show that it takes N complex multiplications to go from the two N-point analyses to a 2N-point analysis. To find the total time needed to make an N-point analysis by this method, let M_N equal the number of multiplications needed for each of our N-point analyses. Let M_{2N} equal the total number of multiplications needed to end with the 2N-point analysis. Then obviously

$$M_{2N} = 2M_N + N.$$

Since $N = 2^m$, m successive doublings would give us the required N-point analysis – that is to say, for the above iterative equation the initial condition is $N_0 = 1$. Solving the equation

$$M_{2N_i} = 2M_{N_i} + N_i$$

(i = 0, 1, 2 . . . , m - 1) we get $M_N = (N/2)\log_2 N$. To compensate for various computer overhead operations, this can be overestimated as $N\log_2 N$ operations, as opposed to N^2 operations by Goertzel's method. Hence the saving in computer time using Cooley's method is $N/\log_2 N$ times the time by the old methods.

For a Control Data Corporation (CDC) 3100 computer with software floating point, Goertzel's algorithm took 45 minutes to Fourier-analyze 2048 points. It took Cooley's method 18.3 seconds – an enormous saving. It makes possible real-time digital spectral analysis. See Table 1 for comparisons of execution times.

Table 1
Table of Comparison Times in Seconds
(CDC 3100 with Software Floating Point)

	N, size of array					
	256	512	1024	2048	4096	8192
Goertzel	42.9	169.8	675.7	2699.5		
Harmon			14.0			
Fourier	1.7	3.8	8.4	18.3	31.9	
Aliasing	2.7	5.5	11.5	24.2		
Double			18.9		62.8	114.4

COMPUTER PROGRAMS

Subroutine FOURIER

The original subroutine HARMON written by Dr. Cooley to carry out these operations is somewhat limited because it requires a complex data array as input. Most numerical data are real, not complex. To solve the majority of problems, the subroutine FOURIER was written to take as input a real N-point array and to return $(N/2) + 1$ complex amplitudes. Subroutine FOURIER accomplishes this by the following procedure: First, it stores the odd-indexed points of the given N-point array in the real parts of $N/2$ complex numbers, and the even-indexed points in the imaginary parts. Next, it makes a call to subroutine HARMON with $N/2$ complex points as input. Finally, it extracts the odd and even pairs of amplitudes from the result and merges these to form the $(N/2) + 1$ complex amplitudes of the original N points. The number of complex amplitudes is $(N/2) + 1$ because the Fourier coefficients for real data satisfy:

$$A_k = A_{N-k+2} \quad (k = 1, 2, 3 \ldots N, N + 1) .$$

That is, the real and imaginary parts are respectively symmetric and antisymmetric about the element $k = (N/2) + 1$. This relation yields the following identities:

Real part of A_1 = average of input data points
Imaginary part of $A_1 = 0$
Imaginary part of $A_{(N/2)+1} = 0$
$A_1 = A_{N-1}$(periodicity).

Thus, for real analysis we need only be concerned with the $(N/2) + 1$ complex amplitudes A_1, $A_2, \ldots, A_{N/2}\, A_{(N/2)+1}$. At this point, the procedure for accomplishing the real analysis is straightforward. Given a complex array $X_k = X_k^o + i\, X_k^e$, where the superscript represent the odd and even indexed real data points, respectively, we call HARMON to get the complex amplitudes C_k $(k = 1, 2, 3 \ldots, N)$. Thus any original complex point X_k has the series representation

$$X_k = \sum_{j=1}^{N} C_j \, e^{2\pi i (j-1)(k-1)/N} \tag{1}$$

At this point we can also introduce the Fourier coefficients A_j and AA_j $(j = 1, 2, 3 \ldots, N)$ representing respectively, the odd- and even-indexed series coefficients. We can write

$$X_k^o = \sum_{j=1}^{N} A_j \, e^{2\pi i (j-1)(k-1)/N} , \tag{2}$$

and

$$X_k^e = \sum_{j=1}^{N} AA_j \, e^{2\pi i (j-1)(k-1)/N} . \tag{3}$$

Substituting Equations 2 and 3 in Equation 1 gives

$$X_k^o \pm iX_k^e = \sum_{j=1}^{N} (A_j \pm iAA_j)\, e^{2\pi i(j-1)(k-1)/N} . \tag{4}$$

Taking the complex conjugate in Equation 1 gives

$$\tilde{X}_k = \sum_{j=1}^{N} \tilde{C}_j\, e^{-2\pi i(j-1)(k-1)/N} = \sum_{\ell=1}^{N} \tilde{C}_{N-\ell+2}\, e^{2\pi i(\ell-1)(k-1)/N} , \tag{5}$$

where in the second summation the index j has been replaced by $\ell = N - j + 2$. That is, starting with

$$\sum_{j=1}^{N} \tilde{C}_j\, e^{-2\pi i(j-1)(k-1)/N}$$

and letting $j = N - \ell + 2$, immediately gives the following identities:

$$\tilde{C}_j = \tilde{C}_{N-\ell+2}$$

$$e^{-2\pi i(j-1)(k-1)/N} = e^{-2\pi i(N-\ell+1)(k-1)/N} = e^{2\pi i(\ell-1)(k-1)/N}$$

$$j = 1 \text{ becomes } \ell = N + 1$$

$$j = N \text{ becomes } \ell = 2$$

$$\sum_{j=1}^{N} \text{ becomes } \sum_{\ell=2}^{N+1} = \sum_{\ell=1}^{N} \text{ since } \tilde{C}_{N-(N+1)+2} \equiv \tilde{C}_{N+1} = \tilde{C}_1$$

so that (C) is demonstrated.

Combining Equation 1 and 4 gives

$$C_j = A_j + i AA_j$$

Combining Equations 4 and 5 gives

$$\tilde{C}_{N-j+2} = A_j - i AA_j$$

Thus,

$$A_j = \frac{1}{2} (C_j + \tilde{C}_{N-j+2}),$$

$$AA_j = -\frac{i}{2} (C_j - \tilde{C}_{N-j+2}).$$

Now we have the odd and even amplitudes, which can be recombined to give

$$C_j = \frac{1}{2} (A_j + AA_j \; e^{-2\pi i(j-1)/N}),$$

$$C_{N-j+2} = \tilde{C}_j = \frac{1}{2} (\tilde{A}_j - \tilde{AA}_j \; e^{2\pi i(j-1)/N}) = \frac{1}{2} (A_j - AA_j \; e^{-2\pi i(j-1)/N}),$$

which is the desired result.

Subroutine ALIASING

Another routine written around the original Cooley program is subroutine ALIASING. Its purpose is to determine, from N given real points, whether the sampling rate is sufficient. It also computes Parseval's formula to determine the magnitude of calculation errors attributable to digital roundoff. The routine returns with (N/2) + 1 complex amplitudes, just as if the routine FOURIER had been called instead.

The method used in ALIASING is similar to the one in FOURIER: An N-point array is treated as an (N/2)-point complex array, which is Fourier-analyzed with HARMON. The resulting spectrum is split up into two spectra A_k and AA_k representing, respectively, the frequency components of the odd and even indexed points of the given array. Each of these spectra represents a sampling density of N/2 points; that is, half the actual density. Then both the odd and the even spectra are merged to yield the desired final spectrum C_k. The A_k's and AA_k's represent the same sampling density at alternating points, so that their difference (between odd-indexed and even-indexed points) should be zero for each k. If the sampling rate is sufficient, the differences between A_k and C_k and between AA_k and C_k will also be close to zero. In addition, all the octaves C_{N-k} should be zero. If any or all of these quantities are larger than the difference between A_k and AA_k, then the sampling density is insufficient or, at best, questionable.

The quantities returned by subroutine ALIASING are computed by the following formulas:

Odd-even: $$\sum_{k=1}^{N/4} \left| |A_k| - |AA_k| \right|^2$$

Odd-both: $$\sum_{k=1}^{N/4} \left| |A_k| - |C_k| \right|^2$$

Even-both: $$\sum_{k=1}^{N/4} \left| |AA_k| - |C_k| \right|^2$$

Octave: $$\sum_{k=1}^{N/4} \left| C_{N/2-k+2} \right|^2 .$$

As an example, suppose that we generate a unit sinusoid of 450 Hertz, and sample it at 1024 points per second for 1 second. We know that all C_k's will be zero except C_{450}, which will equal 1. A call to ALIASING will yield zero amplitudes for all A's and AA's except for $k = 62$, when both will equal 1. The frequency of 62 Hertz is the aliased frequency corresponding to 450 Hertz for a sampling density of 512 points per-second. However, C_{62} will be zero, while C_{450} will be 1. This procedure would show the difference between the A's and AA's as zero, but obviously not zero for the other differences. In order to capture the 450-Hertz sinusoid, we would have to increase the sampling rate, say by a factor of 2.

Parseval's theorem simply says that the norm of the input data array should be equal to the norm of the output amplitude array. This is a property of linear transformations used in computer applications to determine the extent of error resulting from digital truncation. The statement of Parseval's equality is

$$\sum_{i=1}^{N} |X_i|^2 = N \sum_{k=1}^{N} |A_k|^2.$$

ALIASING deals with real data, however, so that

$$A_k = \tilde{A}_{N-k+2}.$$

Also, the amplitude of all components but A_1 and $A_{(N/2)+1}$ are doubled because the routine computes a symmetric two-sided spectrum, but only positive frequencies are of interest. Hence Parseval's equality is computed as

$$\sum_{i=1}^{N} |X_i|^2 = \frac{N}{2} \left\{ \sum_{k=2}^{N/2} |A_k|^2 + 2(A_1^2 + A_{(N/2)+1}^2) \right\}.$$

Subroutine DOUBLE

It sometimes happens that the sampling rate yields too many points for a particular computer. A Control Data Corporation (CDC) 3100 with 16,000 words of memory and standard FORTRAN can handle at most 4096 points, using the Fast Fourier Transform. The routine called DOUBLE was written to double any computer's capacity for Fourier analysis; it requires three scratch tapes. The user puts his oversized array on tape, makes a call to DOUBLE, then reads in that same tape to get his complex amplitudes. The user of course does not read and write his oversized N-point array in a single command; instead, he must divide his array into four records of N/4 points each. DOUBLE does the same processing as FOURIER except for input/output (I/O) calls and increased capacity. It should be mentioned that the complex amplitude $A_{(N/2)+1}$ is not returned from DOUBLE. Also, if only frequency amplitudes are required and phase information is discarded, then the user can save I/O usage and computer time by modifying DOUBLE to return just the N/2 amplitudes.

Subroutine POWER

Another subroutine power takes as input data two real arrays X and Y and returns in X with the cross-power spectrum

$$P_{xy}(w) = \tilde{\mathfrak{F}}(X)\, \mathfrak{F}(Y).$$

The power spectral density of a single array X may be computed by power as

$$P_{xx}(w) = \tilde{\mathfrak{F}}(X)\,\mathfrak{F}(X) = |\mathfrak{F}(X)|^2,$$

but this would not be as efficient as computing the squared modulus of the complex spectrum returned by FOURIER.

Subroutine COVAR

The routine COVAR uses CROSS to yield the cross-covariance of two given arrays. The covariance is computed as the inverse Fourier transform of the cross-power spectrum. COVAR first computes the cross-power spectrum by calling CROSS, then performs an inverse Fourier transform on this array to get the cross-covariance. For input arrays X and Y, the cross covariance is returned in Y and the cross power spectrum in X. To obtain the correlation function, normalize the covariance values by the "zero lag" value of the covariance – i.e., the value Y_1.

Subroutines HANNING and HAMMING

The subroutine HANNING takes as input a real array A of frequency amplitudes. The Hanning smoothing function is applied to the input, and the result is stored in array A, thereby replacing the original contents of A. The Hanning function is defined by

$$A'(1) = A(1) + A(2),$$

$$A'(N) = A(N-1) + A(N),$$

$$A'(i) = .5A(i-1) + A(i) + .5A(i+1)$$

$$(i = 2, 3, 4 \ldots N-1).$$

The coefficients are customarily given as (0.25, 0.5, 0.25), but we are dealing here with one-sided spectra so that a factor of 2 is involved.

The subroutine HAMMING applys the Hamming smoothing function to a given real array A. It is defined by

$$A'_1 = 1.08\,A_1 + 0.92\,A_2,$$

$$A'_N = 1.08\,A_N + 0.92\,A_{N-1},$$

$$A'_i = 0.46\,A_{i-1} + 1.08\,A_i + 0.46\,A_{i+1}$$

$$(i = 2, 3, \ldots N-1).$$

Note that the factor 2 appears again in the coefficients, which are usually given as (0.23, 0.54, 0.23).

The usefulness of these little subroutines for analyzing data with discrete frequencies should not be underrated; they compensate for the finite discrete nature of Fourier analysis when performed on a digital computer. These simple digital filters perform two functions: they reduce the resolution of spectral lines, and they increase the accuracy of relative height of spectral lines. For continuous, smooth spectra, this is not too important, but spectra with sharp spectral lines can be grossly misinterpreted if these two functions are not performed. In spite of the reduction in resolution, some filter should be used in digital Fourier analysis. No filter actually corresponds to using a $\sin x/x$ filter, which has undesirable high sidelobes that can lead to gross errors of computed results.

AVERAGING FOR BETTER RESOLUTION

Often, a sample size of N points does not adequately describe the frequency content of the data, probably twice as many points being needed. At this point it is often suggested that, rather than resample the data at a higher sampling rate, we should just perform linear interpolations between the given points to get the required point density – say, 2N points. The reply is that a linear interpolation – or rather, a linear transformation – adds no new information to the data, hence no *new information to the spectrum.* The proof is as follows:

Consider the N-point series representation

$$X_j = \sum_{k=1}^{N} A_k \, e^{2\pi i (j-1)(k-1)/N} .$$

Suppose that we define a new sequence of points XX_j located between the X_j, such that

$$XX_j = a_j X_j + b_j X_{j+1}$$

and such that the corresponding Fourier coefficients AA_j are represented by

$$XX_j = \sum_{k=1}^{N} AA_k \, e^{2\pi i (j-1)(k-1)/N} .$$

In order to get the AA_k's in terms of the A_k's, we set

$$X_{j+1} = \sum_{k=1}^{N} A_k \, e^{2\pi i j(k-1)/N}$$

$$= \sum_{k=1}^{N} A_k \, e^{2\pi i(j-1)(k-1)/N} \, e^{2\pi i(k-1)/N}.$$

Hence,

$$XX_j = a_j \sum_{k=1}^{N} A_k \, e^{2\pi i(j-1)(k-1)/N} + b_j \sum_{k=1}^{N} A_k \, e^{2\pi i(j-1)(k-1)/N} \, e^{2\pi i(k-1)/N}$$

$$= \sum_{k=1}^{N} (a_j A_k + b_j A_k \, e^{2\pi i(k-1)/N}) \, e^{2\pi i(j-1)(k-1)/N},$$

or

$$AA_k = a_j A_k + b_j A_k \, e^{2\pi i(k-1)/N}.$$

This equation can easily be made independent of j by assuming a uniform linear interpolation such that $a = a_j$ and $b = b_j$. Then we have

$$AA_k = aA_k + bA_k \, e^{2\pi i(k-1)/N}$$

$$= A_k (a + b \, e^{2\pi i(k-1)/N}).$$

Now if we merge these points X_j and XX_j, we form a 2N-point series X_j', such that our original points are the odd indexed X_j' and the linearly interpolated points are the even indexed X_j'. Using the equalities obtained earlier for merging series, we get, for $k = 1, 2, 3, \ldots, N+1$.

$$C_k = \frac{1}{2} (A_k + AA_k \, e^{-2\pi i(k-1)/2N})$$

$$= \frac{1}{2} [A_k + A_k (a + b \, e^{2\pi i(k-1)/N}) \, e^{-2\pi i(k-1)/2N}]$$

$$= \frac{1}{2} A_k [1 + (a + b \, e^{2\pi i(k-1)/N}) \, e^{-2\pi i(k-1)/2N}].$$

Also,

$$C_{N+k} = \frac{1}{2} A_k \left[1 - (a + b\, e^{2\pi i(k-1)/N})\, e^{-2\pi i(k-1)/2N} \right].$$

This shows that the new series, with alternating points being linearly interpolated values, is merely a modification of the old series by a complex exponential whose period is 2N points. The old frequency components will be modified and no new ones will appear. This effect is easily seen if we set $a = b = 1/2$, for then

$$C_k = \frac{1}{2} A_k \left[1 + \frac{1}{2} (1 + e^{2\pi i(k-1)/N})\, e^{-2\pi i(k-1)/2N} \right]$$

$$= \frac{1}{2} A_k \left[1 + \frac{e^{-2\pi i(k-1)/2N} + e^{2\pi i(k-1)/2N}}{2} \right]$$

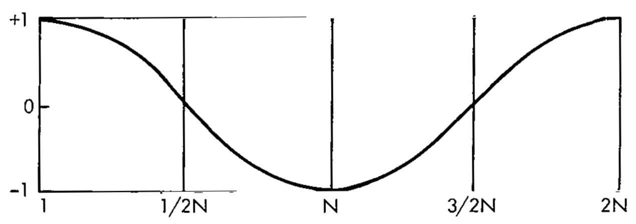
$$\left.\begin{aligned} &= \frac{1}{2} A_k \left(1 + \cos \frac{2\pi(k-1)}{2N} \right) \\ &\text{and} \\ C_{N+k} &= \frac{1}{2} A_k \left(1 - \cos \frac{2\pi(k-1)}{2N} \right) \end{aligned}\right\} \quad (k = 1, 2, 3 \ldots, N+1)$$

The curve $f(t)$ defined by

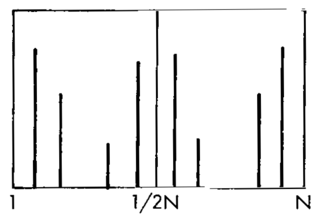
$$f(t) \begin{cases} \dfrac{1}{2}\left(1 + \cos \dfrac{2\pi(k-1)}{2N}\right) & \text{for first half of spectrum} \\[2ex] \dfrac{1}{2}\left(1 - \cos \dfrac{2\pi(k-1)}{2N}\right) & \text{for second half of spectrum} \end{cases}$$

$$(k = 1, 2, 3 \ldots, N+1)$$

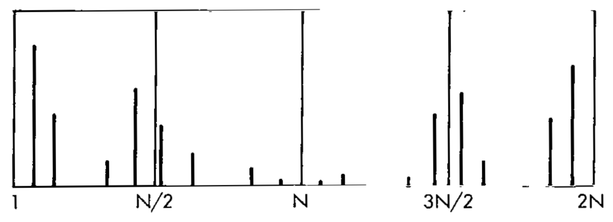
has the form shown in Figure 3(a). A typical spectrum of N points is shown in Figure 3(b). The result of linearly interpolating between points to get 2N points is shown in Figure 3(c).



(a) CURVE f (t)



(b) SPECTRUM (typical)



(c) SPECTRUM WITH CURVE f (t)

Figure 3—Effect of averaging neighboring points in a data sample.

DIGITAL FILTERS

When an infinite and continuous function $h(t)$ is sampled for digital processing, the actual function considered by the computer is $d(t) = g(t)h(t)$, where $g(t)$ is defined by

$$g(t) = \sum_{j=-M}^{M} \delta\ (t - j\,\Delta t).$$

It equals 1 when $t = j\Delta t$ and is otherwise zero. This finite sequence of $N, N = 2M + 1$, equispaced unit spikes, corresponding to N sampled points, is called a finite Dirac comb. Function $g(t)$ is a specimen of the so-called "lag windows" that modify $h(t)$ at different intervals or time lags.

When a Fourier transform is applied to $d(t)$, the result is the same as convolving the Fourier transforms of $g(t)$ and $h(t)$. That is, if $G(w)$ and $H(w)$ are the Fourier transforms of $g(t)$ and $h(t)$, respectively, then the Fourier transform of the product $g(t)h(t)$ is the convolution G^*H, defined as

$$G^*H(w) = \int_{-\infty}^{\infty} G(f)\,H(w - f)df = H^*G(w).$$

To see this, we compute

$$\int_{-\infty}^{\infty} g(t)h(t)e^{-2\pi i w t}\,dt = \int_{-\infty}^{\infty} h(t)\left[\int_{-\infty}^{\infty} G(f)e^{2\pi f i t}\,df\right] e^{-2\pi i wt}\,dt$$

$$\int_{-\infty}^{\infty} G(f)\left[\int_{-\infty}^{\infty} h(t)\ e^{-2\pi i(w-f)t}\,dt\right]df$$

$$\int_{-\infty}^{\infty} G(f)\ H(w - f)df.$$

Thus, instead of multiplying two functions before transformation, we could equivalently convolve their spectra represented by

$$G(w) = \int_{-\infty}^{\infty} g(t)\ e^{-iwt}\,dt$$

and

$$H(w) = \int_{-\infty}^{\infty} h(t)\, e^{-iwt}\, dt.$$

The convolution $G{*}H(w)$ represents an estimate of the true frequency content $H(f)$ as seen through a spectral window of variable transmission $G(w - f)$. This spectral window is referred to as a digital filter when applied to digital data; in that case, the discrete convolution takes the form

$$G{*}H(n) = \sum_{j=-M}^{M} G(j)\, H(n - j),$$

which has the effect of spreading the "true" spectral amplitude $H(n)$ by M weights on either side of n. The $H(n)$ are called "raw spectral estimates"; the $G{*}H(w)$ are "refined (or smoothed) spectral estimates". In most cases $G(w)$ has the form of a $(\sin L)/L$ function, which has a maximum peak at $L = 0$ (equal to 1) and an infinite number of relative maxima oscillating in sign. The peak at $L = 0$ is called the main lobe, and the relative maxima are called side lobes.

The choice of a particular digital filter is based on the following considerations:

1. $g(t) = 0$ for all $t > |T|$

   $\neq 0$ otherwise

2. The main lobe of $G(w)$ should be concentrated near $L = 0$ to reduce the resolution or bandwidth represented by a particular frequency at w. This means that $g(t)$ should be flat with sharp cutoffs at $\pm T$, such as a square wave.

3. The side lobes should be as small as possible to reduce the contribution of other frequencies at $w - f$ to the final amplitude at w. This means that $g(t)$ should be smooth and gradually trailing off the 0 at $t = \pm T$.

The only way to satisfy all these requirements is to effect some sort of compromise determined by experiment.

The HAMMING and HANNING subroutines described earlier are digital filters that smooth the spectrum and give discrete amplitudes a better relative representation at the cost of individual peak resolution. It would be worthwhile to see directly the relationship between lag windows and spectral windows for the popular Hanning function. The Hanning lag window is defined as

$$g(t) = \frac{1}{2}\left(1 + \cos\frac{\pi t}{T}\right) \qquad |t| \le T$$

$$= 0 \qquad\qquad |t| > T$$

Figure 4 is the graph of the function.

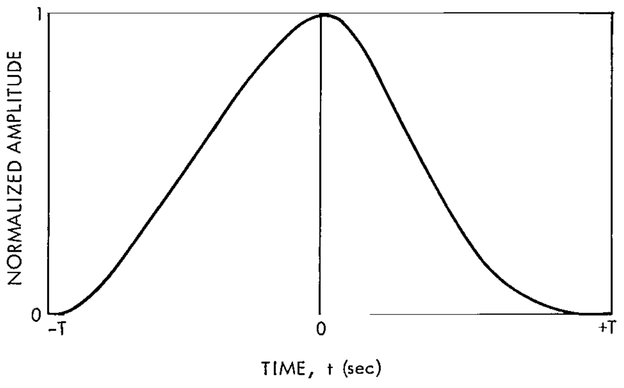


Figure 4—Hanning lag window function in time domain.

To obtain the Fourier spectrum of $g(t)$, we compute

$$G(f) = \int_{-\infty}^{\infty} g(t)\, e^{-2\pi i f t} dt = \frac{1}{T}\int_{-T}^{T} g(t) e^{-2\pi i f t} dt$$

$$= \frac{1}{2T}\int_{-T}^{T} e^{-2\pi i f t}\, dt + \frac{1}{2T}\int_{-T}^{T} \cos\frac{\pi t}{T}\, e^{-2\pi i f t}\, dt$$

$$= \frac{1}{2T}\int_{-T}^{T} (\cos 2\pi f t - i \sin 2\pi f t)\, dt + \frac{1}{2T}\int_{-T}^{T} \cos\frac{\pi t}{T}\, (\cos 2\pi f t - i \sin 2\pi f t)\, dt$$

$$= \frac{\sin 2\pi f T}{2\pi f} + \frac{1}{2}\,\frac{\sin 2\pi\left(f - \frac{1}{2T}\right)T}{2\pi\left(f - \frac{1}{2T}\right)} + \frac{1}{2}\,\frac{\sin 2\pi\left(f + \frac{1}{2T}\right)T}{2\pi\left(f + \frac{1}{2T}\right)} \,.$$

Thus the spectral form of the Hanning lag window is the sum of a central term at frequency f, and two other $(\sin x)/x$ terms displaced on either side of f by $1/2\,T$. The difference

$$\left(f + \frac{1}{2T}\right) - \left(f - \frac{1}{2T}\right) = \frac{1}{T}$$

is the digital bandwidth of the filter. For a sample of N points, the bandwidth is $2/(N\Delta t)$, since $N\Delta t = 2T$. Figure 5 is the graph of $G(f)$. Substituting $G(f)$ in the convolution formula, we have

$$G{*}H(f) = \int_{-\infty}^{\infty} G(\ell)\ H(f-\ell)\, d\ell = \frac{1}{T}\int_{-T}^{T} H(f-\ell)\left\{\frac{\sin 2\pi\ell T}{2\pi\ell} + \frac{1}{2}\,\frac{\sin 2\pi\left(\ell - \frac{1}{2T}\right)T}{2\pi\left(\ell + \frac{1}{2T}\right)} + \frac{1}{2}\,\frac{\sin 2\pi\left(\ell + \frac{1}{2T}\right)T}{2\pi\left(\ell + \frac{1}{2T}\right)}\right\} d\ell \,.$$

In the discrete case, a frequency $f = f_k$ represents the k^{th} harmonic in a finite sample of length $2T$, so the f_k can be expressed as $f_k = k/2T$. Hence if $f = f_k$ and $\ell = f_\ell$, we get

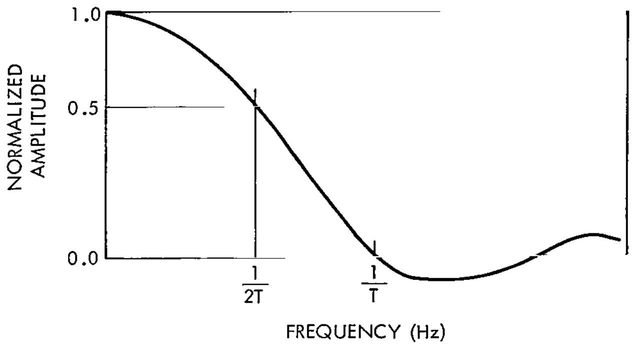


Figure 5—Hanning lag window function in frequency domain.

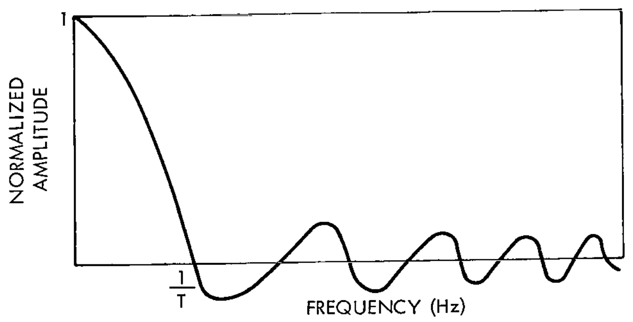


Figure 6—Dirac spectral window function.

$$G^*H(f_k) = G^*H\left(\frac{k}{2T}\right) = \int_{-T}^{T} H\left(\frac{k-\ell}{2T}\right)\left\{\frac{\sin \pi\ell}{\pi\ell} + \frac{1}{2}\,\frac{\sin \pi(\ell-1)}{\pi(\ell-1)} + \frac{1}{2}\,\frac{\sin\pi(\ell+1)}{\pi(\ell+1)}\right\} d\ell .$$

Integrating by parts gives

$$G^*H(f_k) = \frac{1}{2}\,H(f_k) + \frac{1}{4}\,H(f_{k+1}) + \frac{1}{4}\,H(f_{k-1}) .$$

The coefficients are the triplet (0.25, 0.5, 0.25) usually given for the Hanning spectral window. If we are dealing with one-sided spectra (i.e., $0 \leq f < \infty$) rather than two-sided spectra ($-\infty < f < \infty$), then these coefficients should be doubled.

The spectral window corresponding to the finite Dirac comb is

$$G(f) = N\Delta t \cos\left(\frac{\pi k}{N}\right) \frac{\dfrac{\sin 2\pi k}{2\pi k}}{\dfrac{\sin \pi k/N}{\dfrac{\pi k}{N}}}$$

Figure 6 is the graph of this function. The lobes are not as small as in the Hanning window, and for that reason the latter is usually preferred.

Further information on digital filters can be found in Reference 4.

Goddard Space Flight Center
National Aeronautics and Space Administration
Greenbelt, Maryland, December 12, 1967
125-23-02-13-51

REFERENCES

1. Bendat, J. S., and Piersol, A. G., "Measurement and Analysis of Random Data," New York: J. W. Wiley and Sons, 1966, pp. 200-201, 291.

2. Middleton, D., "Introduction to Statistical Communication Theory," New York: McGraw-Hill, 1960, pp. 143, 185.

3. Ralston, A., and Wilf, H. F., "Mathematical Methods for Digital Computers," New York: John Wiley and Sons, 1960.

4. Blackman, R. W., and Tukey, J., "The Measurement of Power Spectra," New York: Dover Publ. 1959.

5. Cooley, J. W., and Tukey J., "An Algorithm for the Machine Calculation of Complex Fourier Series," Mathematics of Computation, Vol. 19, No. 90, April 1965, pp. 297-301.

Appendix A

Subroutine HARMON

```
      SUBROUTINE HARMON(A,S,M,IFS,IFERR)
      DIMENSION A(1),S(1)
C         HARM, ONE-DIMENSIONAL BASIC FORTRAN VERSION.  J.W.COOLEY      HARM 001
C     MODIFIED TO RUN ON CDC 3100.
C                                                                       HARM 002
C                                                                       HARM 008
C     DOES EITHER FOURIER SYNTHESIS,I.E.,COMPUTES COMPLEX FOURIER SERIESHARM 009
C     GIVEN A VECTOR OF N COMPLEX FOURIER AMPLITUDES,OR, GIVEN A VECTOR HARM 010
C     OF COMPLEX DATA X DOES FOURIER ANALYSIS, COMPUTING AMPLITUDES.    HARM 011
C     A IS A COMPLEX VECTOR OF LENGTH N=2**M COMPLEX NOS. OR 2*N REAL   HARM 012
C     NUMBERS. A IS TO BE SET BY USER.                                  HARM 013
C     M  IS AN INTEGER 0.LT.M.LE.13, SET BY USER.                       HARM 014
C     S IS A VECTOR S(J)= SIN(2*PI*J/NP ), J=1,2,....,NP/4-1,           HARM 015
C     COMPUTED BY PROGRAM.                                              HARM 016
C     IFS IS A PARAMETER TO BE SET BY USER AS FOLLOWS-                  HARM 017
C     IFS=0 TO SET NP=2**M AND SET UP SINE TABLE S.                     HARM 018
C     IFS=1 TO SET N=NP=2**M, SET UP SIN TABLE, AND DO FOURIER          HARM 019
C     SYNTHESIS, REPLACING THE VECTOR A BY                              HARM 020
C                                                                       HARM 021
C     X(J)= SUM OVER K=0,N-1 OF A(K)*EXP(2*PI*I/N)**(J*K),              HARM 022
C     J=0,N-1, WHERE I=SQRT(-1)                                         HARM 023
C     THE  X-S  ARE STORED WITH   RE X(J) IN CELL  2*J+1                HARM 024
C     AND  IM X(J)  IN CELL  2*J+2  FOR  J=0,1,2,...,N-1.               HARM 025
C     THE  A-S  ARE STORED IN THE SAME MANNER.                          HARM 026
C                                                                       HARM 027
C     IFS=-1   TO SET  N=NP=2**M,SET UP SIN TABLE, AND DO FOURIER       HARM 028
C     ANALYSIS, TAKING THE INPUT VECTOR A AS X AND                      HARM 029
C     REPLACING IT BY THE A  SATISFYING THE ABOVE FOURIER SERIES.       HARM 030
C     IFS=+2 TO DO FOURIER SYNTHESIS ONLY, WITH A PRE-COMPUTED S.       HARM 031
C     IFS=-2 TO DO FOURIER ANALYSIS ONLY, WITH A PRE-COMPUTED S.        HARM 032
C     IFERR   IS SET BY PROGRAM TO-                                     HARM 033
C     =0 IF NO ERROR DETECTED.                                          HARM 034
C     =1 IF M IS OUT OF RANGE., OR, WHEN IFS=+2,-2, THE                 HARM 035
C     PRE-COMPUTED S TABLE IS NOT LARGE ENOUGH.                         HARM 036
C     =-1 WHEN IFS =+1,-1, MEANS ONE IS RECOMPUTING S TABLE             HARM 037
C     UNNECESSARILY.                                                    HARM 038
C                                                                       HARM 039
C     NOTE- AS STATED ABOVE, THE MAXIMUM VALUE OF M FOR THIS PROGRAM    HARM 040
C     ON THE IBM 7094 IS 13. ON 360 MACHINES HAVING GREATER STORAGE     HARM 041
C     CAPACITY, ONE SHOULD CHANGE THIS LIMIT BY REPLACING 13 IN         HARM 042
C     STATEMENT 3 BELOW BY LOG2 N, WHERE N IS THE MAX. NO. OF           HARM 043
C     COMPLEX NUMBERS ONE CAN STORE IN HIGH-SPEED CORE.                 HARM 044
C           IF THE CAPACITY OF  HARM  IS TO BE INCREASED, ONE MUST      HARM 045
C     ALSO ADD MORE  DO  STATEMENTS TO THE BINARY SORT ROUTINE          HARM 046
C     FOLLOWING STATEMENT  24  AND CHANGE THE EQUIVALENCE STATEMENTS    HARM 047
C     FOR THE  K-S.                                                     HARM 048
C                                                                       HARM 049
      DIMENSION K(12)
      EQUIVALENCE (K(11),K1),(K(10),K2),(K(9),K3),(K(8),K4),(K(7),K5)
      EQUIVALENCE (K(6),K6),(K(5),K7),(K(4),K8),(K(3),K9),(K(2),K10)
      EQUIVALENCE (K(1),K11),(K(1),N2)
      IF(M)2,2,3                                                        HARM 055
    3 IF(M-11) 5,5,2
    2 IFERR=1                                                           HARM 057
    1 RETURN                                                            HARM 058
    5 IFERR=0                                                           HARM 059
      N=2**M                                                            HARM 060
      IF( IABS(IFS) - 1 ) 200,200,10                                    HARM 061
C     WE ARE DOING TRANSFORM ONLY. SEE IF PRE-COMPUTED                  HARM 062
C     S TABLE IS SUFFICIENTLY LARGE                                     HARM 063
   10 IF( N-NP )20,20,12                                                HARM 064
   12 IFERR=1                                                         HARM 065
      GO TO 200                                                       HARM 066
C     SCRAMBLE A, BY SANDE-S METHOD                                   HARM 067
   20 K(1)=2*N                                                        HARM 068
      DO 22 L=2,M                                                     HARM 069
   22 K(L)=K(L-1)/2                                                   HARM 070
      DO 24 L=M,10
   24 K(L+1)=2                                                        HARM 072
C     NOTE EQUIVALENCE OF KL AND K(14-L)                              HARM 073
C     BINARY SORT-                                                    HARM 074
      IJ=2                                                            HARM 075
      J1=2
   25 DO 30 J2=J1,K2,K1
      DO 30 J3=J2,K3,K2                                               HARM 078
      DO 30 J4=J3,K4,K3                                               HARM 079
      DO 30 J5=J4,K5,K4                                               HARM 080
      DO 30 J6=J5,K6,K5                                               HARM 081
      DO 30 J7=J6,K7,K6                                               HARM 082
      DO 30 J8=J7,K8,K7                                               HARM 083
      DO 30 J9=J8,K9,K8                                               HARM 084
      DO 30 J10=J9,K10,K9                                             HARM 085
      DO 30 JI=J10,K11,K10
      IF(IJ-JI)28,30,30                                               HARM 089
   28 T=A(IJ-1 )                                                      HARM 090
      A(IJ-1)=A(JI-1)                                                 HARM 091
      A(JI-1)=T                                                       HARM 092
      T=A(IJ)                                                         HARM 093
      A(IJ)=A(JI)                                                     HARM 094
      A(JI)=T                                                         HARM 095
   30 IJ=IJ+2                                                         HARM 096
      J1=J1+2
      IF(K1-J1)31,25,25
   31 IF(IFS)32,2,36                                                  HARM 097
C     DOING FOURIER ANALYSIS,SO DIV. BY N AND CONJUGATE.              HARM 098
   32 FN = FLOAT(N)
      DO 34 I=1,N                                                     HARM 100
      A(2*I-1) = A(2*I-1)/FN                                          HARM 101
   34 A(2*I)=-A(2*I)/FN                                               HARM 102
C     SPECIAL CASE- L=1                                               HARM 103
   36 DO 40 I=1,N,2                                                   HARM 104
      T = A(2*I-1)                                                    HARM 105
      A(2*I-1) =T + A(2*I+1)                                          HARM 106
      A(2*I+1)=T-A(2*I+1)                                             HARM 107
      T=A(2*I)                                                        HARM 108
      A(2*I) = T + A(2*I+2)                                           HARM 109
   40 A(2*I+2)= T - A(2*I+2)                                          HARM 110
      IF(M-1) 2,1  ,50                                                HARM 111
C     SET FOR L=2                                                     HARM 112
   50 LEXP1=2                                                         HARM 113
C     LEXP1=2**(L-1)                                                  HARM 114
      LEXP=8                                                          HARM 115
C     LEXP=2**(L+1)                                                   HARM 116
      NPL= 2**MT                                                      HARM 117
C     NPL = NP* 2**-L                                                 HARM 118
      DO 130 L=2,M
C     SPECIAL CASE- J=0                                               HARM 120
      DO 80 I=2,N2,LEXP                                               HARM 121
      I1=I + LEXP1                                                    HARM 122
      I2=I1+ LEXP1                                                    HARM 123
      I3 =I2+LEXP1                                                    HARM 124
      T=A(I-1)                                                        HARM 125
      A(I-1) = T +A(I2-1)                                             HARM 126
      A(I2-1) = T-A(I2-1)                                             HARM 127
      T =A(I)                                                         HARM 128
      A(I) = T+A(I2)                                                  HARM 129
      A(I2) = T-A(I2)                                                 HARM 130
      T= -A(I3)                                                       HARM 131
      TI = A(I3-1)                                                    HARM 132
      A(I3-1) = A(I1-1) - T                                           HARM 133
      A(I3   ) = A(I1 )    - TI                                       HARM 134
      A(I1-1) = A(I1-1) +T                                            HARM 135
   80 A(I1)    = A(I1    )   +TI
      IF(L-2) 120,120,90
   90 KLAST=N2-LEXP
      JJ=NPL
      DO 110 J=4,LEXP1,2
      NPJJ=NT-JJ
      UR=S(NPJJ)
      UI=S(JJ)
      ILAST=J+KLAST
      DO 100 I= J,ILAST,LEXP
      I1=I+LEXP1
      I2=I1+LEXP1
      I3=I2+LEXP1
      T=A(I2-1)*UR-A(I2)*UI
      TI=A(I2-1)*UI+A(I2)*UR
      A(I2-1)=A(I-1)-T
      A(I2  )=A(I    ) - TI
      A(I-1) =A(I-1)+T
      A(I)    =A(I)+TI
      T=-A(I3-1)*UI-A(I3)*UR
      TI=A(I3-1)*UR-A(I3)*UI
      A(I3-1)=A(I1-1)-T
      A(I3)    =A(I1   )-TI
      A(I1-1)=A(I1-1)+T
  100 A(I1)    =A(I1)    +TI
C     END OF I LOOP
  110 JJ=JJ+NPL
C     END OF J LOOP
  120 LEXP1=2*LEXP1
      LEXP = 2*LEXP
  130 NPL=NPL/2
C     END OF L LOOP
      IF(IFS)145,2,1
CC    DOING FOURIER ANALYSIS. REPLACE A BY CONJUGATE.
  145 DO 150 I=1,N
  150 A(2*I) =-A(2*I)
      GO TO 1
C     RETURN
C     MAKE TABLE OF S(J)=SIN(2*PI*J/NP),J=1,2,....NT-1,NT=NP/4
  200 NP=N
      MP=M
      NT=N/4
      MT=M-2
      IF(MT) 260,260,205
  205 THETA=.7853981634
C     THETA=PI/2**(L+1)     FOR L=1
      JSTEP = NT
C     JSTEP = 2**( MT-L+1 ) FOR L=1
      JDIF = NT/2
C     JDIF = 2**(MT-L)  FOR L=1
      S(JDIF) = SIN(THETA)
      IF (MT-2)260,220,220
  220 DO 250 L=2,MT
      THETA = THETA/2.
      JSTEP2 = JSTEP
      JSTEP = JDIF
      JDIF = JDIF/2
      S(JDIF)=SIN(THETA)
      JC1=NT-JDIF
      S(JC1)=COS(THETA)
      JLAST=NT-JSTEP2
      IF(JLAST-JSTEP)250,230,230
  230 DO 240 J=JSTEP,JLAST,JSTEP
      JC=NT-J
      JD=J+JDIF
  240 S(JD)=S(J)*S(JC1)+S(JDIF)*S(JC)
  250 CONTINUE
  260 IF(IFS)20,1,20
      END
```

Appendix B

Subroutine FOURIER

```
      SUBROUTINE FOURIER (A,S,M,IFS)
C
C     THIS ROUTINE PERFORMS AN ANALYSIS OF 2**M POINTS BY FIRST DOING
C     AN ANALYSIS OF 2**M/2 COMPLEX POINTS AND THEN ARRANGING THE RESULTS
C
C     A R G U M E N T S
C     1. A - REAL DATA ARRAY - OF DIMENSION 2**M + 2
C     2. S - SIN/COS TABLE - DIMENSION 2**(M-3)
C     3. M - EXPONENT OF 2 - SIZE OF REAL ARRAY
C     4. IFS - -1 FOR FIRST TIME, -2 THERAFTER
C
      DIMENSION A(1),S(1)
      N = 2**(M-1)
      CALL HARMON(A,S,M-1,IFS,IFERR)
C     MERGE 2 N-POINT ANALYSIS INTO 1 2N-POINT ANALYSIS
      NHALF = N/2
      NTWO = N*2 + 4
      X = XO = COS(3.1415926536/FLOAT(N))
      Y = YO = SIN(3.1415926536/FLOAT(N))
      DO 1000 K2 = 4,N,2
      K1 = K2 - 1
      N2 = NTWO - K2
      N1 = N2 - 1
      BK1 = A(K1) + A(N1)
      BK2 = A(K2) - A(N2)
      BN1 = A(K2) + A(N2)
      BN2 = A(K1) - A(N1)
      XBN1 = X*BN1
      XBN2 = X*BN2
      YBN1 = Y*BN1
      YBN2 = Y*BN2
      A(K1) = .5 *(BK1 + XBN1 - YBN2)
      A(K2) = .5 *(-BK2 + XBN2 + YBN1)
      A(N1) = .5 *(BK1 - XBN1 + YBN2)
      A(N2) = .5 *(BK2 + XBN2 + YBN1)
      Q = X*XO - Y*YO
      Y = Y*XO + X*YO
 1000 X = Q
C     COMPLEX ELEMENT A(N)
      A(2*N+1) =(A(1) - A(2))*.5
      A(2*N+2) = 0.0
C     COMPLEX ELEMENT A(0)
      A(1) = .5*(A(1)+A(2))
      A(2) = 0.0
C     COMPLEX ELEMENT A(N/2)
C     A(N+1) = A(N+1)
C     A(N+2) = A(N+2)
      RETURN
      END
```

Appendix C

Subroutine ALIASING

```
      SUBROUTINE ALIASING (A,S,M, DATASUM,AMPSUM, ODDEVENS,ODDBOTHS,
     1 BOTHSUM, OCTAVES)
C
C     THIS PROGRAM COMPARES THE RESULTS OF THE GIVEN SAMPLING WITH
C     ARRAYS (ODD AND EVEN INDEXED) WHICH REPRESENT HALF THE SAMPLING DENSITY.
C     IF THE COMPARISONS ARE SIGNIFICANT, THEN ALIASING EXISTS FOR THE
C     HALF-DENSITY SAMPLE, AND PERHAPS ALSO FOR THE COMPLETE SAMPLE.
C
C     THE VALUES OF -ODDEVENS- AND -OCTAVES- SHOULD BE AS CLOSE TO
C     ZERO AS NUMERICAL TRUNCATION AND/OR NOISE ALLOWS.  IF THESE VALUES
C     ARE GREATER THAN, SAY 1.00E-2, THEN ALIASING EXISTS FOR THIS SAMPLE
C     RATE.
C
C     A = GIVEN REAL DATA ARRAY, OF DIMENSION 2**M + 2 LOCATIONS
C     S = SINE/COSINE ARRAY COMPUTED BY HARMON
C     M = EXPONENT OF 2
C     DATASUM - SUM OF INITIAL A(I)**2 FOR PARSEVALS EQUALITY
C     AMPSUM - SUM OF FINAL A(I)**2 FOR PARSEVALS EQUALITY
C     ODDEVENS - ERROR BETWEEN SPECTRA OF ODD- VS. EVEN-NUMBERED POINTS
C     ODDBOTHS - ERROR BETWEEN ODD-INDEXED VS ALL POINTS
C     BOTHSUM - ERROR BETWEEN SPECTRA OF EVEN-INDEXED VS ALL POINTS
C     OCTAVES - SUM OF OCTAVES
C
      DIMENSION A(1),S(1)
C
C     SET UP INDEX CONSTANTS
C
      N1024 = 2**M
      N512 = N1024/2
      N256 = N512/2
      N1025 = N1024 + 1
      N1026 = N1024 + 2
      N1028 = N1026 + 2
      N1030 = N1028 + 2
      N513 = N512 + 1
      N514 = N512 + 2
      N516 = N514 + 2
      N518 = N516 + 2
      N770 = N514 + N256
      N1544 = N1030 + N514
C
C     COMPUTE THE DATA SUM FOR PARSEVAALS EQUALITY
C
      DATASUM = 0.0
      DO 50 I = 1,N1024
   50 DATASUM = DATASUM + A(I)*A(I)
C
      CALL HARMON(A,S,M-1,-1,IFERR)
C
      A513 = A(N513)
      A514 = A(N514)
      A1025 = .5*(A(1)-A(2))
C     A1026 = 0.0




C
C     SORT COMPLEX SPECTRUM INTO ODD AND EVEN SPECTRA
```

```
      ODDBOTH = ABS(ODD-BOTH)
      ODDBOTHS = ODDBOTHS + ODDBOTH
      EVENBOTH = ABS(EVEN-BOTH)
      BOTHSUM = BOTHSUM + EVENBOTH
      AMPSUM = AMPSUM + BOTH + OCTAVE
      Q = C*DC - S*DS
      S = S*DC + C*DS
  300 C = Q
      AMPSUM = FLOAT(N512)*(AMPSUM+A513*A513+A514*A514+A1025*A1025*2.)
C
C     SORT THE SECOND HALF OF SPECTRUM
C
      N1542 = N1544-2
      DO 600 K2 = N516,N770,2
      K1 = K2-1
      KK2 = N1542 - K2
      KK1 = KK2 - 1
      AR = A(K1)
      AI = A(K2)
      A(K1) = A(KK1)
      A(K2) = A(KK2)
      A(KK1) = AR
  600 A(KK2) = AI
C     FILL IN CERTAIN LOCATIONS
      A(N513) = A513
      A(N514) = A514
      A(N1025) = A1025
      A(N1026) = 0.0
      RETURN
      END
```

Appendix D

Subroutine DOUBLE

```
      SUBROUTINE DOUBLE (A,S,M,N1,N2,N3)
C
C     ROUTINE TO DO FOURIER ANALYSIS OF REAL DATA OF SIZE 2**M WHEN
C     CORE CAN ONLY HANDLE ARRAYS OF SIZE 2**(M-1)
C
C     ARGUMENT LIST  --
C     1. A = DATA BUFFER WORKAREA - DIMENSION 2**(M-1)+ 2
C           THE EXTRA 2 LOCATIONS ARE FOR THE COMPLEX POINT AT THE
C           MIDPOINT OF THE SPECTRUM
C     2. S = SIN/COS TABLE - SIZE ASSUMED TO BE 2**(M-4)
C     3. M = EXPONENT OF REAL DATA ARRAY
C     4. N1 = SCRATCH TAPE WHERE INPUT DATA IS STORED IN FOUR RECORDS ...
C          RECORD 1 CONTAINS    A(1) ... A(N)
C          RECORD 2 CONTAINS    A(N+1) ... A(2N)
C          RECORD 3 CONTAINS    A(2N+1) ... A(3N)
C          RECORD 4 CONTAINS    A(3N+1) ... A(4N)
C          WHERE N = 2**(M-2)
C     5.,6. - N2 AND N3 ARE SCRATCH TAPES
C
      DIMENSION A(1),S(1),NTAPE(3)
      NTAPE(2) = N2
      NTAPE(3) = N3
      M4096 = 2**(M-1)
      M2048 = M4096/2
      M1024 = M2048/2
      M3072 = M2048 + M1024
      M3073 = M3072 + 1
      M2049 = M2048 + 1
      M2050 = M2048 + 2
      M4098 = M4096 + 2
      MINUS = -1
      REWIND N1
      REWIND N2
      REWIND N3
C
C     SORT DATA INTO ODD AND EVEN ARRAYS
C
      DO 100 I = 1,4
      READ (N1) (A(L),L=1,M2048)
      DO 110 J = 1,M1024
      J1 = J + M2048
      J2 = J + M3072
      A(J1) = A(2*J-1)
  110 A(J2) = A(2*J)
      WRITE (N2) (A(L),L=M2049,M3072)
  100 WRITE (N3) (A(L),L=M3073,M4096)
      REWIND N1
      REWIND N2
      REWIND N3
C
C     DO FOURIER ANALYSIS OF ODD AND EVEN ARRAYS SEPARATELY
C
      DO 200 II = 2,3
      NT = NTAPE(II)
      I1 = 1
      I2 = M1024
      DO 201 I = 1,4
      READ (NT) (A(L),L=I1,I2)
      I1 = I1 + M1024
  201 I2 = I2 + M1024
      REWIND NT
      CALL FOURIER (A,S,M-1,MINUS)
      MINUS = -2
      WRITE(NT) (A(L),L=1,M2050)
      WRITE(NT) (A(L),L=M2049,M4098)
  200 REWIND NT
C
C     MERGE THE EVEN AND ODD SPECTRA
C
      C = 1.0
      S = 0.
      DC = COS(3.1415926536/M4096)
      DS = SIN(3.1415926536/M4096)
      DO 300 II = 1,2
      READ (N2) (A(L),L=1,M2048),ARN,AIN
      READ (N3) (A(L),L=M2049,M4096),AARN,AAIN
      DO 301 K = 1,M1024
      K2 = 2*K
      K1 = K2 - 1
      AR = A(K1)
      AI = A(K2)
      KK1 = K1 + M2048
      KK2 = K2 + M2048
      AAR = A(KK1)
      AAI = A(KK2)
      DR = AAR*C + AAI*S
      DI = AAI*C - AAR*S
      A(K1) = .5*(AR+DR)
      A(K2) = .5*(AI+DI)
      A(KK1) = .5*(AR-DR)
      A(KK2) =.5*(-AI+DI)
      Q = C*DC - S*DS
      S = S*DC + C*DS
  301 C = Q
      GO TO (321,322),II
  321 A(M2049) = .5 * (ARN - C*(AARN+AAIN))
      A(M2050) = .5 * (-AIN + C*(AAIN-AARN))
      GO TO 323
  322 A(M2049) = ARN + AAIN
      A(M2050) = AIN - AARN
  323 CONTINUE
      WRITE (N1) (A(L),L=1,M2048)
      M1 = M2048+3
      MM = M4096 + M2048 + 2
      M2 = MM/2 - 2
      DO 302 K = M1,M2,2
      K1 = MM-K
      AR = A(K1)
      AI = A(K1+1)
      A(K1) = A(K)
      A(K1+1) = A(K+1)
      A(K) = AR
  302 A(K+1) = AI
  300 WRITE (N1) (A(L),L=M2049,M4096)
      REWIND N1
      REWIND N2
      REWIND N3
C
C     SORT SPECTRAL ELEMENTS INTO PROPER ORDER
C
      READ (N1) (A(L),L=1,M2048)
      READ (N1) (A(L),L=1,M2048)
      WRITE(N2) (A(L),L=1,M2048)
      READ (N1) (A(L),L=1,M2048)
      WRITE(N3) (A(L),L=1,M2048)
      READ (N1) (A(L),L=1,M2048)
      WRITE(N3) (A(L),L=1,M2048)
      REWIND N1
      REWIND N2
      REWIND N3
      READ (N1) (A(L),L=1,M2048)
      READ (N3) (A(L),L=1,M2048)
      WRITE (N1)(A(L),L=1,M2048)
      READ (N3) (A(L),L=1,M2048)
      WRITE(N1) (A(L),L=1,M2048)
      READ (N2) (A(L),L=1,M2048)
      WRITE(N1) (A(L),L=1,M2048)
      REWIND N1
      REWIND N2
      REWIND N3
      RETURN
C
C     COMPLEX AMPLITUDES ARE ON SCRATCH TAPE N1
C
      END
```

Appendix E

Subroutine HANNING

```
      SUBROUTINE HANNING(A,N)
      DIMENSION A(1)
      NN = N - 1
      X1 = A(1)
      A(1) = X1 + A(2)
      DO 100 I = 2,NN
      X2 = A(I)
      A(I) = .5* (X1 + A(I+1) ) + X2
  100 X1 = X2
      A(N) = A(N) + X1
      RETURN
      END
```

Appendix F

Subroutine HAMMING

```
      SUBROUTINE HAMMING(A,N)
      DIMENSION A(1)
      NN = N - 1
      X1 = A(1)
      A(1) = 1.08*X1 + .92*A(2)
      DO 100 I = 2,NN
      X2 = A(I)
      A(I) = .46 * ( X1 + A(I+1) ) + 1.08 * X2
  100 X1 = X2
      A(N) = .92 * X1 + 1.08*A(N)
      RETURN
      END
```

Appendix G

Subroutine COVAR

```
      SUBROUTINE COVAR (X, Y, M, S)
C
C     X =REAL ARRAY OF DIMENSION 2**M
C     Y =REAL ARRAY OF DIMENSION 2**M
C     M =EXPONENT, 2**M =NUMBER OF POINTS
C     S =SINE/COSINE ARRAY USED BY HARMON
C
C     CROSS-VOVARIANCE IS RETURNED IN Y
C     CROSS POWER SPECTRUM IS RETURNED IN X
C
      DIMENSION X(1), Y(1), S(1)
      N =2**M
      N1 =N + 1
      N2 =2N
      DO 50 I=N1, N2
      X(I) =0
 50   Y(I) =0
      M =M + 1
      CALL POWER (X, Y, M, S)
      DO 100 I =1, N2
100   Y(I) = X(I)
      CALL HARMON (Y, S, M-1, 2, IFERR)
      RETURN
      END
```

Appendix H

Subroutine POWER

```
      SUBROUTINE POWER(X,Y,M,S)
      DIMENSION X(1),Y(1),S(1)
      N = 2**M
      CALL FOURIER(X,S,M)
      CALL FOURIER(Y,S,M)
      DO 100 I = 1,N,2
      I1 = I+1
      TEMP = X(I)*Y(I) + X(I1)*Y(I1)
      X(I1) = X(I)*Y(I1) - X(I1)*Y(I)
  100 X(I) = TEMP
      RETURN
      END
```

"The aeronautical and space activities of the United States shall be conducted so as to contribute . . . to the expansion of human knowledge of phenomena in the atmosphere and space. The Administration shall provide for the widest practicable and appropriate dissemination of information concerning its activities and the results thereof."

—NATIONAL AERONAUTICS AND SPACE ACT OF 1958