AN ABSTRACT OF THE THESIS OF

Gil S. Ahn for the degree of Doctor of Philosophy

in Civil Engineering presented on _____

Title: DIGITAL SIMULATION OF MULTI-DIMENSIONAL

RANDOM PROCESSES AND ITS APPLICATIONS

Redacted for Privacy

Abstract approved: _____
Harold I. Laursen

This dissertation introduces a method which can simulate multi-dimensional Gaussian random processes by the superposition of sine functions. The fast Fourier transform technique is incorporated into the simulating method to facilitate saving of computer time.

In view of the fact that the fast Fourier transform play important roles in digital simulations, it is fully investigated to optimize computer memory and execution time requirements.

The essential features of the techniques are demonstrated by presenting three examples:

an artificial generation of an earthquake accelerogram, digital simulations of wind generated sea waves, and the generation of a two-dimensional wind field.

Digital Simulation of Multi-Dimensional Random Processes and Its Applications

by

Gil Soo Ahn

A THESIS

submitted to

Oregon State University

in partial fulfillmenet of
the requirements for the
degree of

Doctor of Philosophy

Completed on May 5, 1982

Commencement June 1982

APPROVED:

Redacted for Privacy

Professor of Civil Engineering
in charge of major

Redacted for Privacy

Head of Department of Civil Engineering

Redacted for Privacy

Dean of Graduate School

Date of thesis presented May 5, 1982

Typed by Gil Soo Ahn

ACKNOWLEDGEMENT

The author greatly appreciate his Major Professor for opening courses and seminars relative to Random Processes which have laid the foundation for this work.

The author further extends thanks to his family for their continuous encouragement and support during his study and the writing of this thesis.

TABLE OF CONTENTS

LIST OF FIGURES

DIGITAL SIMULATION OF MULTI-DIMENSIONAL RANDOM PROCESSES AND ITS APPLICATIONS

CHAPTER I

INTRODUCTION

The response of structures to random excitation is of wide engineering interest. The vibration environment may be generated by such diverse sources as atmospheric turbulence, ocean waves in a rough sea, ground motion due to earthquakes, acoustic pressures caused by jet engines and rocket motors. In each case, the excitation may be characterized by the fact that vibrational energy is generated in a random manner over a broad band of frequencies. A probabilistic outlook on the design of structures or equipments functioning in such an environment seems mandatory.

Those aspects of the theory of random vibration which deal with the first and second order statistics of random processes and with the input-output relations for linear devices has been developed and used for almost fifteen years in structural engineering. (1), (35) One aspect of the theory which presents itself as a

plausible alternative to a real random process for engineering use is the simulation method. The limitations and paucity of random process data together with the wide spread use of time history dynamic analysis for obtaining structural systems' response are the primary motivation for the development of a simulation method in this work.

The dissertation deals with an efficient method of simulating on a digital computer a set of correlated, stationary and Gaussian time series with zero mean from the given target cross spectral density matrix. The simulation method to be introduced is the superposition of sine functions with the aid of the Fourier transform technique.

Rice (3) originally introduced an one-dimensional simulation method by the sum of cosine functions, i.e.,

$$X(t) = \sum_{n=1}^{N} A_n \cos(\omega_n t + \phi_n) \qquad (1)$$

Shinozuka (1),(2) later expanded it to multi-dimensional processes and eventually succeeded in incorporating the fast Fourier transform technique into the simulating method of random processes by the sum of cosine functions.

Vanmarcke (23) noted in his paper without proof that one-dimensional simulation can be performed by the superposition of sine functions with the random number generator in a computer, i.e.,

$$X(t) = \sum_{n=1}^{N} A_n \sin(\omega_n + \phi_n) \qquad (2)$$

where A_n is the amplitude and ϕ_n is the random phase angle of the n-th contribution of sinusoid and they are distributed uniformly between 0 and 2π. In Chapter III of the thesis the validity of the simulating method by the superposition of sine functions is proved and expanded to multi-dimensional processes. Chapter IV presents the method of incorporating the Fourier transform technique into the simulation method by the superposition of sine functions.

The essential features of the simulating technique are demonstrated by presenting three examples in Chapter V: an artificial generation of an earthquake accelerogram, digital simulation of wind generated sea waves and the generation of a two-dimensional wind field. Most of the work in Chapter III through V is believed to be original.

It is to be noted that all the computation in this work have been carried out by an Apple II computer with a disk drive; the core memory is 48K and the disk can store up to 186K. The special commands pertinent to the Apple II computer should be replaced with specific ones in the listed programs when using other microcomputers.

CHAPTER II

LITERATURE SURVEY

Rice (3) derived a form similar to Equation (2) after discussion of the Fourier series representation of the shot effect current in his paper. Numerous studies dealing with simulation of random process have been published in various forms since then. Although many authors dealt with the simulation of single random processes utilizing trignometric series (3), filtered white noise (4), filtered shot noise (5), and correlated random pulse train (6), only Borgman (7), Shinozuka (8), and Whittig and Sinha (9) investigated the simulation of multicorrelated processes.

I. In the simulation of ocean surface elevation, Borgman used wave superposition by choosing the frequency in such a way that the amplitude of each wave function was an equal portion of the cumulative spectrum. Borgman also presented a method for simulating several simultaneous time series by passing a white noise vector through filters. He proposed the following model (7):

$$y_m(t) = \sum_{j=1}^{m} \int_{-\infty}^{\infty} k_{mj}(\tau) x_m(t-\tau) d\tau \qquad m = 1,2,\ldots M \quad (3)$$

where

$y_m(t)$ = m th time series

$x_j(t)$ = independent random inputs

$k_{mj}(\tau)$ = kernels

If $S_{mr}(f)$ represents the cross-spectral density between $y_m(t)$ and $y_r(t)$, then kernels $k_{mj}(\tau)$ and its Fourier transform $K_{mj}(f)$ are determined from the relation

$$S_{mr}(f) = \sum_{j=1}^{r} \overline{K_{mj}(f)} K_{rj}(f) S_{xj}(f)$$
$$r = 1,2,\ldots, m; \quad m = 1,2,\ldots, M \qquad (4)$$

where the bar denotes the complex conjugate.

The system of equations can be solved sequentially:

$$S_{11}(f) = \left|K_{11}(f)\right|^2$$
$$S_{21}(f) = K_{21}(f)\, K_{11}(f)$$
$$S_{22}(f) = \left|K_{21}(f)\right|^2 + \left|K_{22}(f)\right|^2$$
$$S_{31}(f) = K_{31}(f)\, K_{11}(f)$$
$$S_{32}(f) = K_{31}(f)\, K_{21}(f) + K_{32}(f)\, K_{22}(f)$$
$$S_{33}(f) = \left|K_{31}(f)\right|^2 + \left|K_{32}(f)\right|^2 + \left|K_{33}(f)\right|^2$$

Etc.

Upon solution:

$$K_{11}(f) = \left[S_{11}(f)\right]^{\frac{1}{2}}$$

$$K_{21}(f) = \left[\frac{S_{21}(f)}{S_{11}(f)}\right]^{\frac{1}{2}}$$

$$K_{22}(f) = \left[S_{22}(f) - K_{21}(f)\right]^{\frac{1}{2}}$$

Etc.

This, in turn, determines the digital filter coefficients, a_{nmj}, needed to approximate the kernel associated with the system function $k_{mj}(f)$. Then the final simulation equation reduces to

$$y_m(k\Delta t) = \sum_{j=1}^{N} \sum_{n=-N}^{N} a_{nmj} x_{j\ k-n} \qquad m = 1,2,\ldots M \qquad (6)$$

in which $x_{j,n}$ for n = 1, 2, 3, ... is the j-th generated sequence of independent, zero mean, unit variance, normal random variables.

II. Wittig and Sinha indicated in their paper (9) that complex random numbers can be used to generate sample functions. Given the spectral matrix, $S_{pg}(k/Nh)$ which represents the one-sided cross-spectral density function between process $x_p(nh)$ and process $x_g(nh)$, it can generally be factored into a lower triangular matrix $H_{pg}(k/Nh)$ and its complex transpose, i.e.,

$$S_{pg}(k/Nh) = \left[H_{pg}(k/Nh)\right]\left[H^{*}_{pg}(k/Nh)\right]^{T} \qquad (7)$$

where * denotes the complex conjugate and T the transpose. Equation (7) can be written in summation notation as

$$S_{pg}(k/Nh) = \sum_{i=1}^{p} H_{pi}(k/Nh) H_{gi}(k/Nh) \qquad p,g = 1,2,\ldots,M \qquad (8)$$

Then a set of sample time series can be simulated using the following model:

$$x_p(nh) = \frac{1}{N}\sum_{k=0}^{N-1} X_p(k/Nh)\exp\left(j\frac{2\pi kn}{N}\right) \qquad (9)$$

It is to be noted that $x_p(nh)$ is the Fourier transform of $X_p(k/Nh)$. To define and obtain the terms $X_p(k/Nh)$ first a set of completely independent Gaussian random numbers $\zeta_{ik} = \xi_{ik} + i\eta_{ik}$ is generated using a standard random number generator, such that

$$E\left[\xi_{ik}\right] = E\left[\eta_{ik}\right] = 0$$

and $$E\left[\xi_{ik}^2\right] = E\left[\eta_{ik}^2\right] = 0.5$$

where E is the expectation operator. Then

$$X_p(k/Nh) = (N/2h)^{\frac{1}{2}}\sum_{i=1}^{p} H_{pi}(k/Nh)\,\zeta_{ik} \qquad (10)$$

or, in matrix form,

$$\begin{Bmatrix} X_1 \\ X_2 \\ \cdot \\ \cdot \\ \cdot \\ X_M \end{Bmatrix} = \left(\frac{N}{2h}\right)^{\frac{1}{2}} \begin{bmatrix} H & 0 & \cdots & 0 \\ H & H & \cdots & 0 \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ H & H & \cdots & H \end{bmatrix} \begin{Bmatrix} \zeta_{1k} \\ \zeta_{2k} \\ \\ \\ \\ \zeta_{Mk} \end{Bmatrix} \qquad (11)$$

The cross correlation between time series $x_p(nh)$ and $x_g(mh)$ is

$$R_{pg}(m,n) = \left\{ E\,\frac{1}{N^2}\sum_{k=0}^{N-1}\sum_{l=0}^{N-1} X_p(k/Nh)X_g^*(l/Nh) \times \exp\left[j\frac{2\pi(kn - lm)}{N}\right]\right\} \qquad (12)$$

where, since $x_g(nh)$ is a real function, it has been replaced by its complex conjugate. Because only the X_p and X_g terms are random in nature, the expectation operator can be brought inside the summation. If this is done, Equation (12) reduces to

$$R_{pg}(m,n) = \frac{1}{N^2}\sum_{k=0}^{N-1}\sum_{l=0}^{N-1} E\left[X_p(k/Nh)X_g^*(l/Nh)\right] \times \exp\left[j\frac{2\pi(kn - lm)}{N}\right] \tag{13}$$

Considering Eq. (10) and the properties of ζ_{ik}, it follows that

$$E\left[X_p(k/Nh)X_g^*(l/Nh)\right] = \frac{N}{2h}\sum_{i=1}^{p}\sum_{j=1}^{p} H_{pi}(k/Nh)H_{gj}^*(l/Nh) \times E\left[\zeta_{ik}\zeta_{jl}\right] = \frac{N}{2h}\sum_{i=1}^{p} H_{pi}(k/Nh)H_{gi}(l/Nh)\delta_{kl} \tag{14}$$

where δ_{kl} is the Kronecker delta function. From Eqs. (12) and (13), it can be shown that

$$R_{pg}(m,n) = \frac{1}{2Nh}\sum_{k=1}^{N-1}\sum_{i=1}^{p} H_{pi}(k/Nh)H_{gi}(k/NH) \times \exp\left[j\frac{2\pi k(n - m)}{N}\right] \tag{15}$$

The right-hand side of Eq. (15) is dependent upon only the difference between m and n, but not on their separate values. From this it can be concluded that the time series generated by this technique are stationary.

By applying the Fourier transform and multiplying by a factor of two to get an one-sided spectrum, the cross-spectral density between processes p and g can be obtained, which confirms that the time series given by Equation (9) does have the required power and cross-spectral densities.

III. Shinozuka expanded the concept of Equation (2) to the multi-dimensional processes and systematized the use of series of cosine functions in simulations (8).

For an n-dimensional homogeneous process with mean zero and spectral density function $S_o(k)$ which is of insignificant magnitude outside the region defined by

$$-\infty < \underline{k}_l \leq \underline{k} \leq \underline{k}_u < \infty$$

and denote the interval by

$$(\Delta k_1, \Delta k_2, \ldots, \Delta k_n) = \left(\frac{k_{1u} - k_{1l}}{N}, \frac{k_{2u} - k_{2l}}{N}, \cdots \frac{k_{nu} - k_{nl}}{N} \right)$$

where usually $\underline{k}_l = -\underline{k}_u$, then the process can be simulated as

$$f(x) = \sqrt{2} \sum_{j=1}^{N_1} \sum^{N_2} \cdots, \sum_{j=1}^{N_n} \Big[S_o(k_{1j_1}, k_{2j_2}, \ldots, k_{nj_n}) \Delta k_1 \Delta k_2 \Delta k_n \Big]^{\frac{1}{2}} \cos(k_{1j_1} x_1 + k_{2j_2} x_2 + \quad + k_{nj_n} x_n + \phi_{j_1, \ldots, j_n}) \tag{16}$$

where

$\phi_{j_1 j_2 \ldots, j_n}$ = independent random phase angles

uniformly distributed between 0 and 2 .

$$k_{ij_m} = k_{il} + j_m \Delta k_m$$
$$(j_m = 1,2,\ldots,N, \quad m = 1,2,\ldots,N)$$

Equation (16) can be written in the following form to which the FFT technique is readily applicable.

$$f(\underline{x}) = \mathrm{Re}\sqrt{2}\sum_{j_1=1}^{N_1}\sum_{j_2=1}^{N_2}\cdots,\quad\sum_{j_n=1}^{N_n} B_{j_1 j_2 \cdots j_n} e^{i\sum_{m=1}^{n} k_m j_m x_m} \tag{17}$$

where Re indicates the real part and

$$B_{j_1 j_2 \cdots j_n} = A_{j_1 j_2 \cdots j_n} e^{i\phi_{j_1 j_2 \cdots j_n}} \tag{18}$$

with

$$A_{j_1 j_2 \cdots j_n} = \left[S_0(k_{1j_1}, k_{2j_2}, \ldots, k_{nj_n})\Delta k_1 \Delta k_2 \cdots, \Delta k_n\right]^{\frac{1}{2}} \tag{19}$$

For the case of an one dimensional process with symmetric $S_0(k)$, Equation (16) reduces to

$$f(\underline{x}) = \sqrt{2}\sum_{j=1}^{N}\left[S_0'(j\Delta k)\Delta k\right]^{\frac{1}{2}} \cos(j\Delta k + \phi_j)$$

where

$$\Delta k = \frac{k_u}{N} \quad \text{and} \quad S_0'(j\Delta k) = 2S_0(j\Delta k) \tag{20}$$

When $S_o^1(k)$ can be discomposed into

$$S_0'(k) = H(k)H(k)^*$$

where

$$k = k_{1j}, k_{2j}, \ldots, k_{nj},$$

then $S_o^1(k)$ in Equation (20) can be substituted by $H(k)$.

IV. Comparison of various methods.

It appears that the method of simulation proposed by Shinozuka (8) is the most efficient and systematic so far. With the incorporation of the FFT technique into the simulation method, the one proposed by Shinozuka takes less computer time and is more practical than that proposed by Borgman, which requires (a) the inverse Fourier Transform and (b) $N(N+1)/2$ number of integrations in the time domain. The method proposed by Wittig and Sinha which utilizes the complex random numbers can be reduced to the one of using sinusoids. Then there appears to be not much difference between the methods proposed by Shinozuka and Wittig and Sinha.

CHAPTER III

DIGITAL SIMULATION OF RANDOM PROCESSES

I. A basic form.

A basic form of a homogeneous one-dimensional random process f(x) with mean zero and spectral density $S_0(\omega)$ can be represented by:

$$f(x) = \sqrt{2}\sum_{k=1}^{N} A_k \sin(\omega_k x + \phi_k) \tag{21}$$

where ϕ_k are random phase angles distributed uniformly between 0 and 2π and

$$A_k = \left[S_0'(\omega)\Delta\omega\right]^{\frac{1}{2}}, \quad \omega_k = (k - \tfrac{1}{2})\Delta\omega \tag{22}$$

with

$$S_0'(\omega) = 2S_0(\omega) \tag{23}$$

being the one-sided spectral density function. Fig. 1 shows the relationship between the frequency and the one-sided spectral density function.

II. Simulation of a multidimensional process.

The autocorrelation function of an n-dimensional homogeneous real process $f_0(\underline{x})$, defined by

$$R_0(\underline{\xi}) = E\left[f_0(\underline{x}_1) f_0(\underline{x}_2)\right] \tag{24}$$

is even in

$$R_0(\underline{\xi}) = R_0(-\underline{\xi}) \tag{25}$$

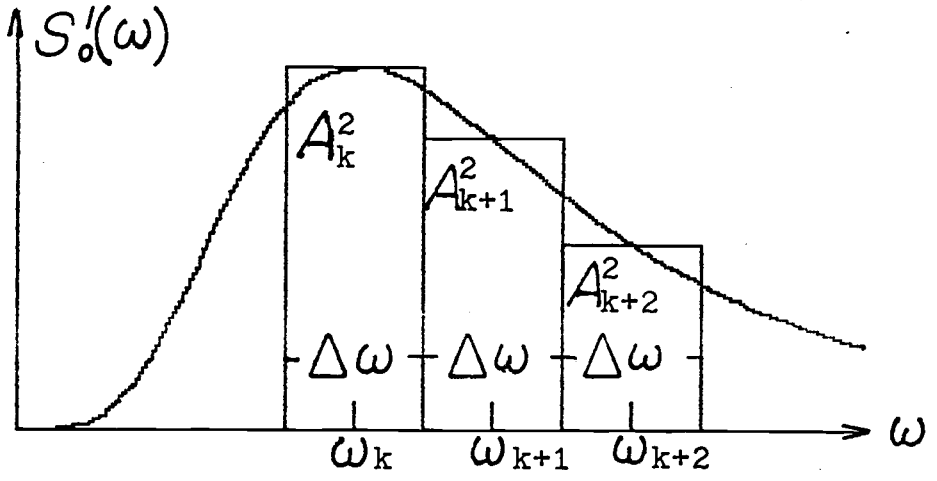

Figure 1

where E indicates expectation, $\underline{x}_1$ and $\underline{x}_2$ are space vectors and $\underline{\xi} = \underline{x}_2 - \underline{x}_1$ is the separation vector. The density function of $f_0(\underline{x})$ is then defined as

$$S_0(\underline{k}) = \frac{1}{(2\pi)^n} \int_{-\infty}^{\infty} R_0(\underline{\xi}) e^{-i\underline{k}\cdot\underline{\xi}} \, d\underline{\xi} \qquad (26)$$

where $\underline{k}$ is the frequency (or wave number) vector and $\underline{k}\cdot\underline{\xi}$ is the inner product of $\underline{k}$ and $\underline{\xi}$, and for simplicity

$$\int_{-\infty}^{\infty} (\quad) d\underline{\xi} = \int_{-\infty}^{\infty} \overset{n\text{-fold}}{\cdots\cdots} \int_{-\infty}^{\infty} (\quad) d\xi_1 d\xi_2 \cdots d\xi_n$$

with n being the dimension of the vector $\underline{\xi}$. It follows from equation (25) that

$$\int_{-\infty}^{\infty} R_0(\underline{\xi}) \sin(\underline{k}\cdot\underline{\xi}) d\underline{\xi} = 0 \qquad (27)$$

and therefore from Equation (26)

$$S_0(\underline{k}) = S_0(-\underline{k})$$

Then

$$S_0(\underline{k}) = \frac{1}{(2\pi)^n}\int_{-\infty}^{\infty} R_0(\underline{\xi})\cos(\underline{k}\cdot\underline{\xi})d\underline{\xi} \qquad (28)$$

and is real.

It can be shown that if the autocorrelation function $R_0(\underline{\xi})$ is nonnegative definite, then according to Bochner (10), it has a nonnegative n-fold Fourier transform, i.e., $S_0(\underline{k}) \geq 0$.

Based on these properties of $S_0(\underline{k})$, an efficient method of simulating $f_0(\underline{x})$ can be proposed as follows. For an n-dimensional homogeneous process with zero mean and spectral density function $S_0(\underline{k})$ which has values of negligible magnitude outside the region defined by

$$-\infty < \underline{k}_l \leq \underline{k} < \underline{k}_u < \infty$$

and denote the interval vector by

$$(\Delta k_1, \Delta k_2, \ldots, \Delta k_n) = \left(\frac{k_{1u} - k_{1\ell}}{N_1}, \frac{k_{2u} - k_{2\ell}}{N_2}, \ldots, \frac{k_{nu} - k_{n\ell}}{N_n}\right)$$

where usually $\underline{k}_l = -\underline{k}_u$. Then the process can be simulated as

$$f(\underline{x}) = \sqrt{2}\sum_{j_1=1}^{N_1}\sum_{j_2=1}^{N_2}\cdots\sum_{j_n=1}^{N_n}\left[S_0^1(k_{1j_1}, k_{2j_2}, \ldots, k_{nj_n})\Delta k\Delta k\ldots\Delta k\right]^{\frac{1}{2}}$$

$$\sin\left(k_{1j_1}x_1 + k_{2j_2}x_2 + \cdots + k_{nj_n}x_n + \phi_{j_1 j_2 \ldots j_n}\right) \qquad (29)$$

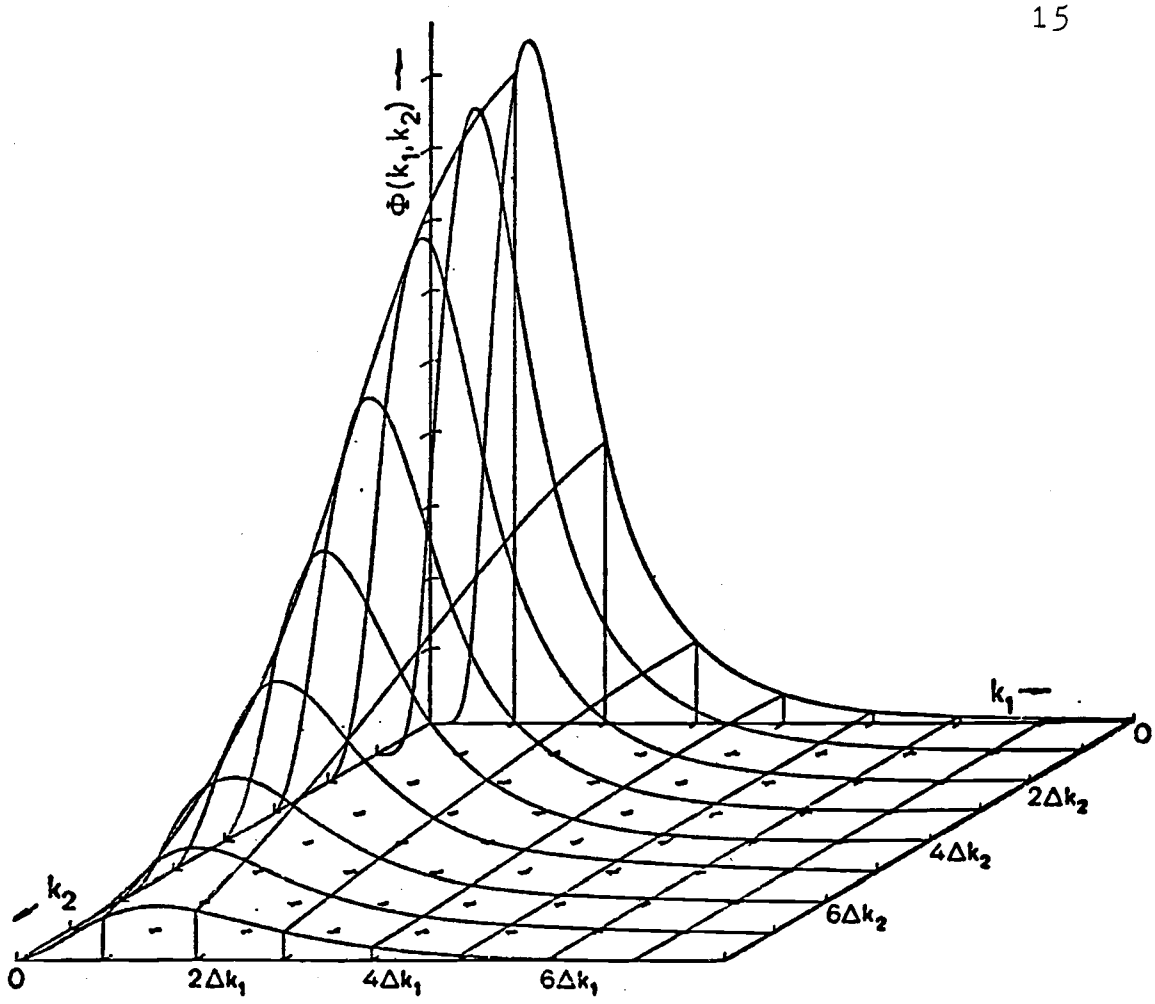

Figure 2

where

$\phi_{j_1 j_2 \ldots j_n}$ = independent random phase angles uniformly distributed between 0 and 2π.

$$k_{i j_m} = k_{il} + j_m \Delta k_m$$

$$(j_m = 1,2,\ldots,N, \quad m = 1,2,\ldots,n)$$

To avoid the lengthy expression in the subsequent discussion, $f(\underline{x})$ will be written in the following form:

$$f(\underline{x}) = \sqrt{2} \sum_{j=1}^{N} A(\underline{k}) \sin(\underline{k}_j \cdot \underline{x} + \phi_j) \qquad (30)$$

where

$$N = N_1 N_2 \ldots N_n$$

$$A(\underline{k}_j) = \left[S_0(\underline{k}_j)\Delta k_1 \Delta k_2 \ , \ldots , \Delta k_n \right]^{\frac{1}{2}} = \left[S_0(\underline{k}_j)\, \Delta \underline{k} \right]^{\frac{1}{2}}$$

It can be shown (8) that when the ensemble average $f(\underline{x})$ is zero, then the autocorrelation function $R(\underline{\xi})$ of $f(\underline{x})$ becomes:

$$R(\underline{\xi}) = \sum_{j=1}^{N} A^2(\underline{k})\cos(\underline{k} \cdot \underline{\xi}) \qquad (31)$$

Upon substituting $A^2(\underline{k}) = S_0(\underline{k})\Delta \underline{k}$ and taking the limit as $N \rightarrow \infty$ one obtains

$$R(\underline{\xi}) = \int_{-\infty}^{\infty} S_0(\underline{k})\cos(\underline{k} \cdot \underline{\xi})\, d\underline{k} = R_0(\underline{\xi}) \qquad (32)$$

where it is assumed $S_0(\underline{k}) = 0$ outside the region. This indicates that, when the ensemble average is considered, the simulated process $f(\underline{x})$ possesses the target autocorrelation $R_0(\underline{\xi})$ and therefore the target spectral density $S_0(\underline{k})$. It can be shown (8) that the spatial (temporal) mean $< f(\underline{x}) >$ is also zero and the spatial autocorrelation approaches $R_0(\underline{\xi})$ as $N \rightarrow \infty$. This makes the method directly applicable to the space (time) domain anaysis in which the ensemble average can be evaluated in the form of the spatial (temporal) average. It is to be noted that the simulated process is Gaussian by virtue of the central limit theorem (11).

In order to make it feasible for the FFT technique, Eq.(29) has to be written in the following form:

$$f(\underline{x}) = \mathrm{Im}\sqrt{2}\sum_{j_1=1}^{N_1}\sum_{j_2=1}^{N_2}\cdots\sum_{j_n=1}^{N_n} B_{j_1 j_2\cdots j_n} e^{i\sum_{m=1}^{n} k_{m j_m} x_m} \tag{33}$$

where Im indicates the imaginary part and

$$B_{j_1 j_2\cdots j_n} = A_{j_1 j_2\cdots j_n} e^{i\phi_{j_1 j_2\cdots j_n}} \tag{34}$$

with

$$A_{j_1 j_2\cdots j_n} = \left[S_o^1(k_{1j_1}, k_{2j_2}, \ldots, k_{nj_n})\Delta k_1 \Delta k_2 \cdots \Delta k_n\right]^{\frac{1}{2}}$$

For one-dimensional processes with symmetric $S_o(k)$, Equation (30) reduces to

$$f(x) = \sqrt{2}\sum_{j=1}^{N}\left[S_o^1(j\,\Delta k)\,\Delta k\right]^{\frac{1}{2}} \sin(j\,\Delta k + \phi_j) \tag{35}$$

where

$$\Delta k = \frac{k_u}{N} \quad \text{and} \quad S_o^1(j\Delta k) = 2S_o(j\Delta k)$$

III. Decomposition of the cross-spectral density matrix

Discussed in this section will be a set of homogeneous Gaussian n-dimensional processes $f_j(\underline{x})$ (j= 1,2,...m) with zero mean and with the cross-spectral matrix $S_o(\underline{k})$ defined by:

$$\begin{bmatrix} S^o_{11}(\underline{k}) & S^o_{12}(\underline{k}) & \dots & S^o_{1m}(\underline{k}) \\ S^o_{21}(\underline{k}) & S^o_{22}(\underline{k}) & \dots & S^o_{2m}(\underline{k}) \\ \dots\dots\dots\dots\dots\dots\dots\dots & & & \\ S^o_{m1}(\underline{k}) & S^o_{m2}(\underline{k}) & & S^o_{mm}(\underline{k}) \end{bmatrix} \tag{36}$$

where $S^o_{lj}(\underline{k})$ is the n-fold Fourier transform of the cross correlation $R^o_{lj}(\underline{\xi})$. Because of the fact that $R^o_{lj}(\underline{\xi}) = R^o_{lj}(-\underline{\xi})$, it follows

$$S^o_{lj}(\underline{k}) = S^o_{lj}(\underline{k})^*$$

where the (*) indicates the complex conjugate. The matrix $S_o(\underline{k})$ is therefore Hermitian and nonnegative definite. If a matrix $H(\underline{k})$ which possesses an n-dimensional Fourier transform can be found and satisfies the following equation

$$S^o(\underline{k}) = H(\underline{k})H^*(\underline{k})^T \tag{37}$$

where $S_o(\underline{k})$ is the specified target cross-spectral matrix and (T) indicates the transpose. Then $f_l(\underline{x})$ (l=1,2,..,m) can be simulated by the following filtering technique (7).

$$f_l(\underline{x}) = \sum_{j=1}^{m} \int_{-\infty}^{x} h_{lj}(\underline{x} - \underline{\xi})\, \eta_j(\underline{\xi})\, d\underline{\xi} \tag{38}$$

where $h_{lj}(\underline{x})$ is the n-dimensional Fourier transform of

$H_{1j}(\underline{k})$:

$$h_{1j}(\underline{x}) = \int_{-\infty}^{\infty} H_{1j}(\underline{k})\, e^{-i\underline{k}\cdot\underline{x}}\, d\underline{k}$$

and $\eta_j(\underline{k})$ is an independent n-dimensional normalized white noise component such that

$$E\left[\eta_1(\underline{x})\eta_j(\underline{x})\right] = \delta(\underline{x}_1 - \underline{x}_2)\delta_{1j}$$

with

$$\delta(\underline{x}_1 - \underline{x}_2) = \delta(x_{11} - x_{21})\delta(x_{12} - x_{22})\ldots\delta(x_{1n} - x_{2n})$$

To find the matrix $H(\underline{k})$ in an efficient way, it can be assumed that $H(\underline{k})$ is a lower triangular matrix (12):

$$H(\underline{k}) = \begin{bmatrix} H_{11}(\underline{k}) & 0 & & 0 \\ H_{21}(\underline{k}) & H_{22}(\underline{k}) & & 0 \\ \cdots & \cdots & \cdots & \cdots \\ H_{m1}(\underline{k}) & H_{m2}(\underline{k}) & \cdots & H_{mm}(\underline{k}) \end{bmatrix}$$

Inserting the above $H(\underline{k})$ into equation (37), solutions are obtained as

$$H_{jj}(\underline{k}) = \left[\frac{D_k(\underline{k})}{D_{j-1}(\underline{k})}\right]^{\frac{1}{2}} \qquad j = 1,2,\ldots m \qquad (39)$$

where $D_j(\underline{k})$ is the j-th principal minor of $S^o(\underline{k})$ with D_o being defined as unity, and

$$H_{lj}(\underline{k}) = H_{jj}(\underline{k}) \frac{S^o\begin{pmatrix}1,2,\ldots,j-1,l\\1,2,\ldots,j-1,j\end{pmatrix}}{D_j(\underline{k})}$$

$$j = 1,2,\ldots,m$$

$$l = j+1 \ \ldots,m \qquad (40)$$

where

$$S^o\begin{pmatrix}1,2,\ldots j-1,l\\1,2,\ldots j-1,j\end{pmatrix} = \begin{vmatrix} S^o_{11} & S^o_{12} & & S^o_{1,j-1} & S^o_{1j} \\ S^o_{21} & S^o_{22} & & S^o_{2,j-1} & S^o_{2j} \\ \cdot & \cdot & \cdots & \cdot & \cdot \\ S^o_{j-1,1} & S^o_{j-1,2} & & S^o_{j-1,j-1} & S^o_{j-1,j} \\ S^o_{11} & S^o_{12} & \ldots & S^o_{1,j-1} & S^o_{1j} \end{vmatrix}$$

is the determinant of a sub-matrix obtained by deleting all elements except the $(1,2,\ldots j-1,1)$th row and $(1,2,\ldots j-1,j)$th column of $S^o(\underline{k})$. It is noted that the above decomposition is valid only when the matrix $S^o(\underline{k})$ is Hermitian and positive definite as can be seen from Equation (39). Because the cross-spectral density matrix $S^o(\underline{k})$ is known to be only nonnegative definite, special consideration is needed in those cases where $S^o(\underline{k})$ has a zero principal minor. This is discussed in Ref.(8). Since the real and the imaginary parts of cross-spectral density functions are respectively even and odd in $\underline{k}$, it can be shown from successive substitutions using equations (39) and (40) that

$$\text{Re } H_{lj}(\underline{k}) = \text{Re } H_{lj}(-\underline{k})$$

$$\text{Im } H_{lj}(\underline{k}) = -\text{Im}H_{lj}(-\underline{k})$$

for $l > k$, and

$$H_{ll}(\underline{k}) = H_{ll}(-\underline{k}) \geq 0$$

from which it follows that $h_{lj}(\underline{x})$ is real.

Once $H(\underline{k})$ is computed using equations (39) and (40), then instead of passing a white noise vector through filters, the process $f_l(\underline{x})$ can be simulated in terms of the following series:

$$f_l(\underline{x}) = \sum_{p=1}^{l} \sum_{q=1}^{N} |H_{pq}(\underline{k}_q)| \sqrt{2\,\Delta\underline{K}} \sin\left[\underline{k}_q \cdot \underline{x} + \Theta_{lp}(\underline{k}_q) + \phi_{pq}\right] \tag{41}$$

where $\underline{k}_q$, $\Delta\underline{k}$, N and ϕ_{pq} are essentially the same as defined previously for n-dimensional processes and

$$\Theta_{lp}(\underline{k}) = \tan^{-1}\left(\frac{\text{Im } H_{lp}(\underline{k})}{\text{Re } H_{lp}(\underline{k})}\right).$$

CHAPTER IV

FAST FOURIER TRANSFORM

I. The Fast Fourier transform.

The present study is concerned with an efficient numerical single and double Fourier transform and its inversion which plays a crucial role in a stochastic approach to dynamic structural analysis. A new concept of the application of the FFT technique in simulating sample fuctions will be introduced and used in the subsequent chapter.

The mathematical tools employed in the present study are the complex Fourier series technique and the Cooley -Tukey algorithm (13), as used in the numerical evaluation of the inverse FFT (14),(15). Based on the Cooley-Tukey algorithm, the number of operations required for evaluating the complex Fourier coefficients for the first N terms of the expansion of a given spectrum is $2N \log_2 N$ against N^2 operations in the usual computation, in which an operation is defined as that consisting of one complex multiplication followed by one complex addition. Therefore, for large values of N, especially in the double Fourier inversion, the number of operations in the numerical inversion is signifi-

cantly reduced by applying the FFT in computation.

1. Discrete Fourier transform.

Let a Fourier series subroutine whose input is the complex sequence A(n), n=0,1,...,N-1 and whose output is the Fourier series (IDFT),

$$X(j) = \sum_{n=0}^{N-1} A(n)W_N^{jn}, \quad j=0,1,\ldots,N-1 \tag{42}$$

Then, it can easily be seen that by letting X*(n)/N replace A(n) as input and A*(j) replace X(j) as output, the above subroutine will be capable of computing the discrete Fourier transform (DFT)

$$A(n) = \frac{1}{N}\sum_{j=0}^{N-1} X(j)W_N^{-nj} \tag{43}$$

2. Two sets of real data in one pass through a DFT subroutine.

Using the linearity property, it can be seen that if $X_1(j)$ and $X_2(j)$ are real sequences such that

$$X_1(j) \longleftrightarrow A_1(n)$$
$$X_2(j) \longleftrightarrow A_2(n)$$

and

$$X(j) = X_1(j) + iX_2(j) \tag{44}$$

then X(j) has the transform

$$A(n) = A_1(n) + iA_2(n) \tag{45}$$

Replacing n by N-n, taking complex conjugates of both sides, and applying conjugate relations gives

$$A^*(N-n) = A_1(n) - iA_2(n) \tag{46}$$

Solving (45) and (46) for $A_1(n)$ and $A_2(n)$, we obtain

$$\begin{aligned} A_1(n) &= \tfrac{1}{2}\left[A^*(-n) + A(n)\right] \\ A_2(n) &= i\tfrac{1}{2}\left[A^*(-n) - A(n)\right] . \end{aligned} \tag{47}$$

The procedure can be outlined as follows:

(a) form X(j) as defined by (44);

(b) compute A(n) by means of the DFT subroutine;

(c) compute $A_1(n)$ and $A_2(n)$ according to (47) for n=0,1,...,N/2

3. Computing 2N from two N-point sequence.

Given the 2N data points Y(j) with $Y(j) \longleftrightarrow C(n)$, where n and j = 0,1,...,2N-1. If the two N-point sequences

$$\begin{aligned} X_1(j) &= Y(2j), \\ X_2(j) &= Y(2j+1), \quad j = 0,1,\ldots,N-1 \end{aligned} \tag{48}$$

have the N-point transforms $A_1(n)$ and $A_2(n)$,

$$\begin{aligned} X_1(j) &\longleftrightarrow A_1(n) \\ X_2(j) &\longleftrightarrow A_2(n). \end{aligned} \tag{49}$$

Separating even and odd indexed points in the series for C(n), it becomes

$$C(n) = \frac{1}{2N} \sum_{j=0}^{2N-1} Y(j) W_{2N}^{-nj}$$

$$= \frac{1}{2N} \left[\sum_{j=0}^{N-1} Y(2j) W_{2N}^{-2nj} + \sum_{j=0}^{N-1} Y(2j+1) W_{2N}^{-2nj-n} \right] \quad (50)$$

Because

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

$$C(n) = \frac{1}{2N} \left[\sum_{j=0}^{N-1} Y(2j) W_N^{-nj} + \sum_{j=0}^{N-1} Y(2j+1) W_N^{-nj} W_{2N}^{-n} \right] \quad (51)$$

The sums appearing in (51) define $A_1(n)$ and $A_2(n)$, so (51) can be written

$$C(n) = \frac{1}{2} \left[A_1(n) + A_2(n) W_{2N}^{-n} \right]. \quad (52)$$

Substituting N+n for n, and using the fact that $W_{2N}^N = -1$,

$$C(N+n) = \frac{1}{2} \left[A_1(n) - A_2(n) W_{2N}^{-n} \right] \quad (53)$$

It can be summarized as follows:

(a) compute the two N-point DFT's of the sequences $X_1(j) = X(2j)$ and $X_2(j) = X(2j+1)$

(b) apply (52) and (53) to the resulting transforms. This procedure, when iterated upon to produce successive doublings up to N, the total number of points is the FFT algorithm with radix 2.

4. Calculation of the DFT of real data.

The transform of a single sequence of 2N real data points can be done by means of one DFT of N complex points (16) by using procedure 2 and procedure 3. First, assuming Y(j), j=0,1,...,2N-1 to be the real sequence, let

$$Y(j) \longleftrightarrow C(n).$$

Then form the N-point sequences,

$$\begin{aligned} X_1(j) &= Y(2j), \\ X_2(j) &= Y(2j+1), \\ X(j) &= X_1(j) + iX_2(j) \end{aligned} \tag{54}$$

Compute the N-point Fourier transform of X(j) and use procedure 2, Equation (47), obtain the transforms of the two real sequences $X_1(j)$, $X_2(j)$ in terms of the DFT of X(j). For convenience, Equation (47) is repeated

$$\begin{aligned} A_1(n) &= \frac{1}{2}\left[A^*(-n) + A(n)\right], \\ A_2(n) &= i\frac{1}{2}\left[A^*(-n) - A(n)\right]. \end{aligned} \tag{55}$$

By using procedure 3, the doubling algorithm (52), the transform of the full array Y(j) is obtained:

$$C(n) = \frac{1}{2}\left[A_1(n) + A_2(n)W_{2N}^{-n}\right]. \tag{56}$$

It is to be noted that the upper half of the C(n) array

is redundant;i.e., Y(j) real implies that C(n) = C*(2N-n). Since $X_1(j)$ and $X_2(j)$ are real, $A_1(n)$ and $A_2(n)$ are conjugate even. Replacing N by N+n in (56);

$$C(N+n) = \frac{1}{2}\left[A_1(n) - A_2(n)W_{2N}^{-n} \right] . \qquad (57)$$

It is efficient to use (56) and (57) for n=0,1,...,N/2. It can be easily shown that the C(n)'s in

$$Y(j) = \sum_{n=0}^{2N-1} C(n)W_{2N}^{nj} , \quad j = 0,1,...,2N-1 \qquad (58)$$

can be identified with the sine-cosine coefficients of the series

$$Y(j) = \frac{1}{2}a(0) + \sum_{n=1}^{N-1}\left[a(n)\cos\frac{\pi jn}{N} + b(n)\sin\frac{\pi jn}{N}\right] + \frac{1}{2}(-1)^j a(N), \qquad (59)$$

as follows:

$$a(n) = 2 \text{ Re } C(n), \quad \text{for } 0 \leq n \leq N;$$

$$b(n) = -2 \text{ Im } C(n), \quad \text{for } 0 < n < N.$$

The procedure for this section is outlined below;

(a) let the 2N-point array Y(j) be put into a complex N-point array X(j) as defined by (54); in many computer programs, this means one does nothing since complex arrays are stored so that real and imaginary parts are already in alternating locations;

(b) compute the N-point DFT of X(j);

(c) apply Equation (55) for n=0,1,...,N/2 to get $A_1(n)$ and $A_2(n)$;

(d) apply equations (56) and (57) for n=0,1,...N/2 to get C(n) for n=0,1,...,N.

It is to be noted that the 2N real data points Y(j) are transformed to N+1 complex frequency values C(n), n=0, 1,2,...,N. Since C(0) and C(N) are real and since the remaining C(n)'s are complex the sequence C(n) contains 2N independent real numbers. It is to be noted again that C(n) is defined for $0 \leq n < 2N$, but due to the property C(n) = C*(-n), values for $N < n < 2N$ are redundant and need not be computed or stored.

5. The calculation of Fourier series for real data.

This is performed by reversing the previous procedure. To derive this formula , solve (56) and (57) for $A_1(n)$ and $A_2(n)$ in terms of C(n) and C(N + n):

$$A_1(n) = C(n) + C(N + n) \tag{60}$$

$$A_2(n) = \left[C(n) - C(N + n)\right] W_{2N}^2 \quad . \tag{61}$$

Then solving (55), it becomes

$$A(n) = A_1(n) + iA_2(n) \tag{62}$$

$$A^*(N-n) = A_1(n) - iA_2(n). \tag{63}$$

Therefore, procedure 5 is as follows:

(a) generate $A_1(n)$ and $A_2(n)$ for n=0,1,...,N/2 according to (60) and (61);

(b) generate A(n) by using (62) and (63) with n=0,1,...,N/2;

(c) compute the inverse DFT of the N-element complex array A(n); the result will be the sequence X(j) whose real and imaginary parts are the real elements of Y(j) as defined by (54).

Procedures 4 and 5 also permit one to transform a conjugate even data sequence X(j) to a real frequency A(n) and vice versa. This is easily seen to be accomplished by letting X*(n)/N replace A(n) and A*(j) replace X(j) in procedures 4 and 5, thereby switching the roles of "frequency" and "data".

6. Calculation of cosine series for real data.

It can be easily demonstrated that Y(j) being real and even implies that its IDFT will be a cosine series. Thus, if Y(j), j=0,1,...,2N-1, is real and Y(j)=Y(2N-j),

$$Y(j) = \sum_{n=0}^{2N-1} C(n) W_{2N}^{nj}$$

or

$$Y(j) = \frac{1}{2}a(0) + \sum_{n=1}^{N-1} a(n)\cos\frac{\pi jn}{N} + \frac{1}{2}(-1)^j a(n) \quad (64)$$

where the a(n)'s are real and

$$a(n) = 2C(n).$$

It follows that the DFT of Y(j) can be expressed as a cosine series:

$$a(n) = \frac{2}{N}\left[\frac{1}{2}Y(0) + \sum_{j=1}^{N-1} Y(j)\cos\frac{\pi jn}{N} + \frac{1}{2}(-1)^j Y(N)\right] \tag{65}$$

The procedure will be derived for computing a cosine transform of real even data Y(j), j=0,1,...,N-1, where

$$Y(2N-j) = Y(j).$$

It is to be noted that actually only Y(0),...,Y(N) need be given. First define the complex sequence

$$X(j) = Y(2j) + \left[Y(2j+1) - Y(2j-1)\right] i \tag{66}$$

for j=0,1,...,N-1. It is to be noted that only terms with j=0,1,...,N/2 have to be formed. This is a complex conjugate even sequence and, therefore, its transform, A(n), must be real. Procedure 5 gives an efficient way to transform a conjugate even sequence to a real sequence. It is to be noted that the former was a frequency function and the latter was data. Here the roles are reversed. Letting the conjugate even sequence X*(j)/N be the input to procedure 5, the output will be A(n), the real DFT of X(j).

Having A(n), procedure 2 can be used to obtain the transforms of the real and imaginary parts of X(j). Let it be defined that

$$Y(2j) \longleftrightarrow A_1(n) \tag{67}$$

$$Y(2j+1) \longleftrightarrow A_2(n)$$

$$Y'(2j) = Y(2j+1) - Y(2j-1) \longleftrightarrow A_2'(n) \tag{68}$$

From procedure 2, equation (47)

$$A_1(n) = \frac{1}{2}\left[A(n) + A(-n)\right], \tag{69}$$

$$A_2'(n) = \frac{1}{2i}\left[A(n) - A(-n)\right], \tag{70}$$

where A(n) is real. From reference (14), it can be derived that

$$Y(2j-1) \longleftrightarrow W_N^{-n} A_2(n),$$

and therefore,

$$A_2'(n) = A_2 - W_N^{-n} A_2(n), \tag{71}$$

giving

$$A_2(n) = A_2'(n)/(1 - W_N^{-n}). \tag{72}$$

A special calculation must be made for n=0 and n=N. For this, it must be computed

$$A_2(0) = \frac{1}{N}\sum_{j=0}^{N-1} Y(2j+1). \tag{73}$$

Finally, procedure 3, the doubling algorithm, leads from $A_1(n)$ and $A_2(n)$ to C(n), the DFT of Y(j). Substituting (69) and (71) in Equation (56) of procedure 4, it becomes

$$C(n) = \frac{1}{2}\left[A_1(n) + A_2(n) W_{2N}^{-n}\right],$$

$$2C(n) = \frac{1}{2}\left[A(n)+A(-n)\right] + \frac{1}{2i}\left[A(n)-A(-n)\right]\frac{1}{1-W_N^{-n}}W_{2N}^{-n}$$

Therefore

$$a(n)=2C(n)=\frac{1}{2}\left\{\left[A(n)+A(-n)\right]-\frac{\left[A(n)-A(-n)\right]}{2\sin\frac{\pi n}{N}}\right\} \qquad (74)$$

for n=1,2,...,N-1. For n=0 and n=N the following equation must be used

$$C(0) = \frac{1}{2}\left[A_1(0) + A_2(0)\right] ,$$

$$C(N) = \frac{1}{2}\left[A_1(0) - A_2(0)\right] . \qquad (75)$$

Summarizing, procedure 6 for the cosine transform is:

(a) given the real even sequence Y(j), j=0,1,.., 2N-1, define X(j) according to (66);

(b) let X*(j)/N be the input to procedure 5; the output will be A*(n) = A(n);

(c) compute C(n) using (74) for n=1,2,...,N-1; for n=0 and n=N, let $A_1(0) = A(0)$ and compute $A_2(0)$ with (73); then, use (75) to obtain C(0) and C(N); finally, let a(n) = 2C(n) for n=0,1,...,N.

7. Calculation of sine series for real data.

It can be easily demonstrated that if Y(j), j=0,1,...,2N-1 is real and odd, it is expressible as a sine series,

$$Y(j) = \sum_{n=0}^{2N-1} C(n) W_{2N}^{nj} = \sum_{n=1}^{N-1} b(n) \sin \frac{\pi nj}{N}, \qquad (76)$$

where the b(n)'s are real and

$$b(n) = 2iC(n).$$

It should be noted that Y(0) = Y(N) = 0. It also follows that

$$b(n) = \frac{2}{N} \sum_{j=1}^{N-1} Y(j) \sin \frac{\pi nj}{N} \qquad (77)$$

For the calculation, the following fact should be noted; if Y(j) is real and odd, then iY(j) is conjugate even and its transform is real. Therefore,

$$Y(j) \longleftrightarrow C(n) = \frac{b(n)}{2i}$$

and

$$+iY(j) \longleftrightarrow +C(n) = \frac{1}{2} b(n).$$

Therefore, letting

$$X(j) = -\left[Y(2j+1) - Y(2j-1)\right] + Y(2j)i, \qquad (78)$$

and, following a derivation similar to that for equation (74) in procedure 6, Equation (79) can be arrived at for the coefficients of the cosine series.

Summarizing, procedure 7 is as follows:

(a) given values Y(j), j=1,2,...,N-1 of a real odd sequence Y(j), j=0,1,2,...,2N-1, form X(j) according to (78);

(b) let X*(j)/N be input to procedure 5; the

output will be $A^*(n) = A(n)$, where $X(j) \longleftrightarrow A(n)$;

(c) compute

$$b(n) = 2iC(n) = \frac{1}{2}\left\{\frac{\left[A(n) - A(-n)\right] - \left[A(n) + A(-n)\right]}{2\sin\frac{\pi n}{N}}\right\} \tag{79}$$

for n=1,2,...,N-1.

Use of procedure 6 and 7 yields a fourfold decrease in computation and storage requirements since only half the real input data arrays need be supplied. Actually, N/2 + 1 data for the cosine transform and N/2 - 1 for the sine transform. The complex data X(j) to be transformed contains only N/2 terms rather than the 2N required if the full array Y(j) is supplied to a complex DFT subroutine.

II. Application of the FFT to the simulation processes

Unless the given data has a special property such as real, imaginary, even and odd, the N data require 2N inputs because of the characteristic of the simulation function.

Two expressions have been presented in Chapter I, they are:

$$X(t) = \sum_{n=1}^{N} A_n \sin(\omega_n t + \phi_n) \tag{1}$$

and

$$X(t) = \sum_{n=1}^{N} A_n \cos(\omega_n t + \phi_n). \tag{2}$$

In order to take advantage of the Fast Fourier transform technique, alternate forms of equations (1) and (2) will be introduced,

$$Z(t) = \sum_{n=1}^{N} A_n \exp(i\phi_n)\exp(i\omega_n t) \quad . \tag{80}$$

Then,

$$Z(t) = X_1(t) + X_2(t)$$

$$= \sum_{n=1}^{N} A_n \left[\cos(\omega_n t + \phi_n) + i\sin(\omega_n t + \phi_n)\right]. \tag{81}$$

Therefore Equation (1) is:

$$X(t) = X_2(t) = \text{Im} \sum_{n=1}^{N} A_n \exp(i\phi_n)\exp(i\omega_n t) \tag{82}$$

and Equation (2):

$$X(t) = X_1(t) = \text{Re} \sum_{n=1}^{N} A_n \exp(i\phi_n)\exp(i\omega_n t) \tag{83}$$

The discrete form of Equation (6) is a prerequisite to FFT application. The following conversion is also necessary for the FFT application:

$$\omega_n = 2\pi f_n,$$

$$\Delta t = \frac{T}{N}$$

$$f_n = nf = \frac{n}{T} = \frac{n}{N\Delta t},$$

$$t = \frac{kT}{N}, \qquad k, n = 0,1,\ldots, N-1 \tag{84}$$

Thus,

$$X(k\frac{T}{N}) = \mathrm{Im}\sum_{n=0}^{N-1} A_n \exp(i\phi_n)\exp(i2\pi\frac{nk}{N}) \qquad (85)$$

The limits in Equation (85) requires special scrutiny. Rice (3) originally expressed the digital simulation in the following form:

$$X(t) = \sum_{n=1}^{N} C_n \cos(\omega_n t - \phi_n) \qquad (86)$$

The limits of summation are from 1 to N. The FFT form is:

$$X(t) = \mathrm{Im}\sum_{n=0}^{N-1} A_n \exp(i\phi_n)\exp(i2\pi\frac{nk}{N}) \qquad (87)$$

Equation (87) is zero at limits n=0 and n=N since

$$\sin(2\pi\frac{nk}{N}) = 0$$

at n=0 and n=N. The choice of Equation (1) over Equation (2) in the subsequent section is of course motivated by the reason just mentioned above. Equation (87) is the imaginary part of Z(t), which is the inverse Fast Fourier transform. The inverse FFT of a subroutine, however, has generally a built-in scale factor, i.e., 1/N. This scale factor has to be removed prior to the inverse FFT computation.

To show the effectiveness of the FFT technique when incorporated into the simulating function Equation (87), the following data has been used in a computer program as input :

Sample data points = 128

Amplitude $A_n = 1.0$

The program and output are listed in Figures (3) through Figure (6). The effectiveness of the FFT computation is well demonstrated by the ratio of computations, i.e.,

$$\frac{2N\log_2 N}{NxN} = \frac{2\log_2 N}{N} = \frac{14}{128}.$$

The computer program in Fig. 2 has been written in BASIC programming language to implement the previously discussed calculations. The parameters are defined bellow:

Program symbol	Description
P	2π
PI	π
A1	Amplitude A_n
W	Frequency increment
R	Random phase angle ϕ_n
A(k)	Simulated time series by Eq.(1)
FI(k)	Simulated time series by FFT.

The loop 100 - 300 computes the simulated time series by the standard method, i.e., by Equation (1) , except statements 210 and 220 extract values of $A_n \exp(i\phi_n)$. The loop 1000 - 3000 is the implementation of the FFT.

The result of the simulation using Equation (1) is shown in rows designated as "STD" meaning the standard method and the "FFT" is the result by implementing Equation (87) in Figure 6.

In APPLESOFT BASIC, which is used in the program, numbers are stored internally to over nine digits of accuracy and when printed, only nine digits are shown. Figure 6 indicates that the results are the same up to seven digits.

```
LIST 1,300

10  DIM FR(512),FI(512),SR(512),S
    I(512),A(512)
20  HOME
30  HTAB 16: VTAB 7: INVERSE : PRINT
    "COMPUTING"
40  HTAB 12: VTAB 11: PRINT "PLEA
    SE BE PATIENT"
50  NORMAL
60 P = 8 *  ATN (1)
70 PI = P / 2
80 N = 128
90 F = P / N
100  FOR I = 1 TO N
110 A(I) = 0
120  NEXT I
130  VTAB 5: PRINT QW
140  FOR L = 1 TO N
150 QW = QW + 1
160 A1 = 1
170 W = F
180 R =  RND (L) * P
190 S2 =  SIN (R)
200 C2 =  COS (R)
210 SR(L) = C2 * A1
220 SI(L) = S2 * A1
230 U =  - 1
240 S1 =  SIN (W)
250 C1 =  COS (W)
260 A(1) = A(1) + A1 * S2
270  FOR K = 2 TO N
280 QW = QW + 1
290 S3 =  SIN (W * (K - 1) * (L -
    1) + R)
300 A(K) = A(K) + A1 * S3
]
```

Figure 3.

```
]
```

```
 LIST 310,540

310  VTAB 5: PRINT QW
320  NEXT K,L
330 N4 = N / 8
340  GOSUB 1000
350  PR# 1
360  PRINT  CHR$ (15) +  CHR$ (1)
     ;
370  PRINT  CHR$ (1) + "132N"
380  PRINT "NO. OF COMPUTATION=";
     QW
390  PRINT "NO. OF FFT COMPPUTATI
     ON=";QX
400  PRINT : PRINT
410  FOR J = 1 TO N4
420  PRINT "STD  ";
430  FOR K = 1 TO 8
440  PRINT A(8 * (J - 1) + K); SPC(
     2)
450  NEXT K
460  PRINT
470  PRINT "FFT  ";
480  FOR K = 1 TO 8
490  PRINT FI(8 * (J - 1) + K); SPC(
     2)
500  NEXT K
510  PRINT
520  NEXT J
530  PR# 0
540  END

]
```

Figure 4.

```
LIST 1000,3000

1000  REM
1010  FOR IT = 1 TO N
1020 I = IT - 1
1030 J = 0
1040 M2 = 1
1050 M1 = M2
1060 M2 = M2 + M2
1070 MI =  INT (I / M2)
1080 MS = I - MI * M2
1090  IF MS - M1 < 0 THEN 1110
1100 J = J + N / M2
1110  IF M2 - N < 0 THEN 1050
1120 JF = J + 1
1130 FR(IT) = SR(JF):FI(IT) = SI(
     JF)
1150  NEXT IT
1160  PRINT : PRINT
1210 L = 1
1220  IF L - N =  > 0 THEN 3000
1230 ST = 2 * L
1240 EL = L
1250  FOR M = 1 TO L
1260 A = PI * (1 - M) / EL
1270 WR =  COS (A)
1280 WI =  SIN (A)
1281  IF U < 0 THEN 1285
1282  GOTO 1290
1285 WI =  - WI
1290  FOR I = M TO N STEP ST
1295 QX = QX + 1
1300 J = I + L
1310 TR = WR * FR(J) - WI * FI(J)

1320 TI = WR * FI(J) + WI * FR(J)

1330 FR(J) = FR(I) - TR
1340 FI(J) = FI(I) - TI
1350 FR(I) = FR(I) + TR
1360 FI(I) = FI(I) + TI
1365  NEXT I,M
1370 L = ST
1380  GOTO 1220
3000  RETURN

]
```

Figure 5.

```
        132N
NO. OF COMPUTATION=16384
NO. OF FFT COMPPUTATION=448

STD  -13.5071041  .925342795  -.91965651  -2.02184861  -8.06727744  -12.4235314  -6.95233814  -6.00746897
FFT  -13.507104  .925342815  -.919656508  -2.02184857  -8.06727742  -12.4235313  -6.95233824  -6.00746894
STD  .0699925864  .797093071  12.3110947  -8.11595969  -3.00893509  -13.9600725  -9.03665644  -9.90095831
FFT  .0699925125  .797093092  12.3110946  -8.11595962  -3.00893533  -13.9600725  -9.03665649  -9.90095817
STD  -.990155118  11.5152396  3.42345453  6.19365518  -12.1814656  .276834135  3.04800914  .0558410252
FFT  -.990155013  11.5152397  3.42345465  6.19365518  -12.1814655  .276834079  3.04800909  .0558409877
STD  -11.6684336  -1.4879015  -.913069759  -2.27256963  -4.21068191  2.28047262  19.4170723  -6.62977362
FFT  -11.6684337  -1.48790114  -.913069813  -2.27256981  -4.21068182  2.28047261  19.4170723  -6.62977317
STD  -3.88431406  -17.7666349  10.5244671  -.30629541  2.44750068  13.1253808  2.19596293  -2.8107277
FFT  -3.88431386  -17.7666347  10.5244671  -.306295384  2.44750099  13.1253807  2.19596309  -2.81072781
STD  -2.7906381  .864539589  -.586356806  -.600376181  -.52258663  -15.4033209  -9.38582525  .218064269
FFT  -2.79063818  .864539675  -.586357018  -.600376463  -.5225864  -15.4033211  -9.38582523  .218063863
STD  -4.19621226  -12.0647145  .539652218  -11.6194848  5.16418244  8.01066313  7.70594693  -16.5376199
FFT  -4.19621259  -12.0647146  .539652852  -11.6194845  5.16418309  8.0106633  7.70594694  -16.5376197
STD  1.57837461  6.13485166  -12.9850639  -3.36867451  .665564401  -4.41090463  -4.93392232  -6.31755217
FFT  1.57837458  6.13485144  -12.9850642  -3.36867446  .665564272  -4.41090409  -4.93392203  -6.31755224
STD  7.64438394  .265750928  14.094952  -11.1177575  -3.40274293  .951005932  -11.5951081  10.8721279
FFT  7.64438448  .265750984  14.0949519  -11.1177575  -3.40274274  .951006334  -11.595108  10.872128
STD  -8.39872447  -7.57264635  -16.634389  6.69521189  9.14398593  5.50475832  -4.62668768  1.00121415
FFT  -8.3987239  -7.57264614  -16.6343898  6.69521206  9.14398587  5.50475852  -4.626688  1.00121396
STD  12.3405139  7.34962399  .273304494  1.89974124  -6.01223223  7.73500329  4.46050584  -17.301121
FFT  12.340514  7.34962377  .273304409  1.89974192  -6.01223248  7.73500272  4.46050595  -17.3011202
STD  14.9588226  7.07121836  -1.45206619  -10.9942145  -5.97717412  -4.59636378  2.33655537  1.79058254
FFT  14.9588229  7.07121835  -1.45206518  -10.9942142  -5.97717461  -4.59636204  2.33655583  1.79058276
STD  11.454465  -10.197553  9.98383047  9.55265385  -3.43028685  .375046034  2.63434403  7.81709493
FFT  11.454465  -10.1975531  9.98383031  9.55265363  -3.4302873  .375046207  2.6343452  7.81709503
STD  -7.8992595  -2.86176983  12.5788115  .230840301  -2.44481278  5.38738906  16.5427908  1.4930809
FFT  -7.89926019  -2.8617698  12.5788111  .230840205  -2.44481317  5.38738946  16.5427901  1.49308084
STD  -10.5530043  -5.38288846  -10.4345255  .0815321351  -4.77078338  -3.20996125  1.85674685  11.2397068
FFT  -10.5530046  -5.38288887  -10.4345249  .0815320388  -4.77078364  -3.20996144  1.85674593  11.2397058
STD  -5.65925615  15.6301138  -3.31248065  -11.3399604  8.57572503  -2.44785739  -14.5659191  .693130892
FFT  -5.65925662  15.6301137  -3.31248019  -11.3399601  8.57572418  -2.44785696  -14.5659182  .693131643
```

Figure 6.

III. A two dimensional simulation process.

A two dimensional homogeneous random process $f_o(x,y)$ with spectral density $S_o(k_1,k_2)$ can be derived by logical extension of the one-dimensional case (17). According to Equation (30), the simulated process $f(x,y)$ corresponding to $f_o(x,y)$ can be written as

$$f(x,y) = \sqrt{2}\sum_{m=1}^{N_1}\sum_{n=1}^{N_2} A_{mn}\sin(k_{1m}x + k_{2n}y + \phi_{mn}) \quad (89)$$

with

$$A_{mn} = \sqrt{4S_o(k_{1m},k_{2n})\Delta k_1 \Delta k_2} \quad (90)$$

where symmetry of $S_o(k_1,k_2)$ with respect to k_1 and k_2 has been assumed and

$$\Delta k_1 = k_{1u}/N_1 \qquad \Delta k_2 = k_{2u}/N_2$$
$$k_1 m = m\Delta k_1 \qquad k_2 n = n\Delta k_2 \quad (91)$$
$$\phi_{mn} = \text{random phase angles distributed uniformly between 0 and } 2\pi.$$

In Equation (91), k_{1u} and k_{2u} are the upper cut-off frequencies in the first quadrant of the k_1 - k_2 plane such that $S_o(k_1,k_2)$ is significant only in the domain D_o specified by

$$D_o : 0 \leq k_1 \leq k_{1u}, \quad 0 \leq k_2 \leq k_{2u}$$

To take advantage of computing efficiency, here again Equation (89) is rewritten in the following form

$$f(x,y) = \mathrm{Im}\sqrt{2}\sum_{m=1}^{N_1}\sum_{n=1}^{N_2} A_{mn} e^{i\phi_{mn}} e^{i(k_{1m}x + k_{2n}y)} \qquad (92)$$

$f(x,y)$ is evaluated at

$$\begin{aligned} x &= p\Delta x \qquad (p=0,1,\ldots,N_1-1) \\ y &= q\Delta y \qquad (q=0,1,\ldots,N_2-1) \end{aligned} \qquad (93)$$

where Im indicates the imaginary part and

$$\begin{aligned} \Delta x &= \frac{2\pi}{N_1 \Delta k_1} = \frac{2\pi}{k_{1u}} \\ \Delta y &= \frac{2\pi}{N_2 \Delta k_2} = \frac{2\pi}{k_{2u}} \end{aligned} \qquad (94)$$

Using Equations (91) and (93), Equation (92) can be rewritten as

$$f(p\Delta x, q\Delta y) = \mathrm{Im}\sqrt{2}\sum_{m=0}^{N_1-1}\sum_{n=0}^{N_2-1} B_{mn}\, e^{2\pi i(\frac{mp}{N_1} + \frac{nq}{N_2})} \qquad (95)$$

where

$$B_{mn} = A_{mn}\, e^{i\phi_{mn}} \qquad (96)$$

Now the FFT technique can be directly applied to Equation (95) and the use of FFT technique is almost mandatory because of the large size of the sample points resulting from $N_1 \times N_2$.

The identity between Equations (89) and (95) is established by setting in Equation (96)

$$A_{m0} = A_{0n} = 0$$

and

$$A_{mn} = 0.$$

A computer program written in BASIC is again listed with its output. The program computes the simulated random fuctions by both the standard and the FFT method with the view of comparing resulting values. The parameters which differ from the previous listing are listed below.

Program symbol	Description
A(k,j)	Simulated time series in 2-D
TI(k,j)	Simulated time series by FFT

The loop 110 - 340 computes the simulated time series by Equation (89). The statements 230 and 240 computes Equation (96). $A_{m0} = A_{0n} = 0$ are introduced at statements 320 - 330. The loop 740 - 800 is the matrix transposition, the special method of which will be discussed in the following section. The statements 840 - 1220 is the loop for the FFT computation (21),(22). $A_{mn} = 0$ is automatically taken care of at the statement 280 by proper indexing and in the loop of FFT computations.

```
LIST 1,300

10  DIM FR(16),FI(16),A(16,16),TR
    (16,16),TI(16,16),SR(16),SI(
    16)
20 N = 8
30  HTAB 16: VTAB 7: INVERSE : PRINT
    "COMPUTING"
40  HTAB 12: VTAB 11: PRINT "PLEA
    SE BE PATIENT"
50  NORMAL
60 P = 8 *  ATN (1)
70 PI = P / 2
80 F = P / N
90 W = F
100 QW = 0
110  FOR I = 1 TO N
120  FOR J = 1 TO N
130 A(I,J) = 0
140  VTAB 5: PRINT QW
150  NEXT J,I
160  FOR J = 1 TO N
170  FOR L = 1 TO N
180 QW = QW + 1
190 A1 = N * (J - 1) + L
200 R =  RND (A1) * P
210 S2 =  SIN (R)
220 C2 =  COS (R)
230 TR(L,J) = C2 * A1
240 TI(L,J) = S2 * A1
250  FOR I = 1 TO N
260  FOR K = 1 TO N
270 QW = QW + 1
280 S3 =  SIN (W * ((K - 1) * (L -
    1) + (I - 1) * (J - 1)) + R)

290 A(I,K) = A(I,K) + A1 * S3
300  VTAB 5: PRINT QW


]
```

Figure 7.

```
LIST 310,600

310  NEXT K,I,L
320 TR(J,0) = 0:TR(0,J) = 0
330 TI(0,J) = 0:TI(J,0) = 0
340  NEXT J
350  GOSUB 560
360  PR# 1
370  PRINT  CHR$ (15) +  CHR$ (1)
     ;
380  PRINT  CHR$ (1) + "132N"
390  PRINT "NO. OF COMPUTATION=";
     QW
400  PRINT "NO. OF FFT COMPPUTATI
     ON=";QX
410  PRINT : PRINT
420  FOR J = 1 TO 8
430  PRINT "STD  ";
440  FOR K = 1 TO 8
450  PRINT A(K,J); SPC( 2)
460  NEXT K
470  PRINT
480  PRINT "FFT  ";
490  FOR K = 1 TO 8
500  PRINT TI(K,J); SPC( 2)
510  NEXT K
520  PRINT
530  NEXT J
540  PR# 0
550  END
560  REM
570 U =  - 1
580  FOR K = 1 TO N
590  FOR J = 1 TO N
600 SR(J) = TR(K,J):SI(J) = TI(K,
     J)

]
```

Figure 8.

```
 LIST 610,900

610  NEXT J
620  GOSUB 830
630  FOR M = 1 TO N
640 TR(K,M) = FR(M):TI(K,M) = FI(
     M)
650  NEXT M
660  NEXT K
670  IF SW < 0 THEN 710
680  GOSUB 720
690 SW =  - 1
700  GOTO 580
710  RETURN
720  REM
730  FOR I = 2 TO N
740 LL = I - 1
750  FOR J = 1 TO LL
760 X = TR(I,J):Y = TI(I,J)
770 TR(I,J) = TR(J,I)
780 TI(I,J) = TI(J,I)
790 TR(J,I) = X:TI(J,I) = Y
800  NEXT J,I
810  RETURN
820 QX = 0
830  REM
840  FOR IT = 1 TO N
850 I = IT - 1
860 J = 0
870 M2 = 1
880 M1 = M2
890 M2 = M2 + M2
900 MI =  INT (I / M2)

]
```

Figure 9.

```
 LIST 910,1400

910 MS = I - MI * M2
920  IF MS - M1 < 0 THEN 940
930 J = J + N / M2
940  IF M2 - N < 0 THEN 880
950 JF = J + 1
960 FR(IT) = SR(JF):FI(IT) = SI(J
    F)
970  NEXT IT
980  PRINT : PRINT
990 L = 1
1000  IF L - N =  > 0 THEN 1220
1010 ST = 2 * L
1020 EL = L
1030  FOR M = 1 TO L
1040 A = PI * (1 - M) / EL
1050 WR =  COS (A)
1060 WI =  SIN (A)
1070  IF U < 0 THEN 1090
1080  GOTO 1100
1090 WI =  - WI
1100  FOR I = M TO N STEP ST
1110 QX = QX + 1
1120 J = I + L
1130 TR = WR * FR(J) - WI * FI(J)

1140 TI = WR * FI(J) + WI * FR(J)

1150 FR(J) = FR(I) - TR
1160 FI(J) = FI(I) - TI
1170 FR(I) = FR(I) + TR
1180 FI(I) = FI(I) + TI
1190  NEXT I,M
1200 L = ST
1210  GOTO 1000
1220  RETURN

]

]
```

Figure 10.

```
132N
NO. OF COMPUTATION=4160
NO. OF FFT COMPPUTATION=192

STD  168.463181  153.122407  -68.1804928  -196.555777  147.772747  -48.9815142  -21.0300458  -76.7114471
FFT  168.46318  153.122407  -68.1804933  -196.555777  147.772748  -48.9815128  -21.0300465  -76.7114478
STD  -19.7631101  117.068057  -310.424627  366.829792  113.116516  -180.046591  -131.860369  -4.13069721
FFT  -19.7631097  117.068057  -310.424627  366.829789  113.116516  -180.046589  -131.860369  -4.13069672
STD  -191.329798  158.668122  -270.751995  -51.9876561  -507.430468  123.771956  341.920224  237.341941
FFT  -191.329798  158.668123  -270.751994  -51.9876561  -507.430467  123.771956  341.920223  237.34194
STD  46.0528117  -8.44475029  40.2930351  -175.344035  -261.22257  134.738506  8.66409435  249.436503
FFT  46.0528118  -8.44475049  40.2930352  -175.344036  -261.222571  134.738506  8.66409238  249.436503
STD  158.597389  -93.0257152  227.783027  245.482221  -352.133456  -194.993083  238.54861  -157.973399
FFT  158.597389  -93.0257156  227.783025  245.482221  -352.133455  -194.993083  238.54861  -157.9734
STD  -320.842537  85.2720386  -144.08918  153.547831  428.974041  -27.0075637  -50.9678987  -79.6149231
FFT  -320.842537  85.2720387  -144.08918  153.54783  428.974042  -27.0075618  -50.9678995  -79.6149244
STD  281.906888  307.332581  205.065764  -75.8691008  -189.714622  -63.641971  -97.0695095  -342.414005
FFT  281.906886  307.332581  205.065762  -75.8690987  -189.714621  -63.6419729  -97.0695088  -342.414003
STD  -107.734476  335.913585  44.7738794  -45.9754901  -344.866302  153.007326  -110.531961  -10.980645
FFT  -107.734475  335.913587  44.7738799  -45.9754898  -344.866301  153.007328  -110.531961  -10.9806491
```

Figure 11.

IV. The FFT with external storage.

A characteristic feature of the FFT, as presented by Cooley and Tukey (13), is the assumption that all data points in the array to be transformed are kept in the main storage simultaneously. This can be a serious restriction on the range of data points when a mini- or micro-computer set-up is the only alternative for the FFT computation. In the following sections, some of the special techniques will be presented which are intended to remove those restrictions.

1. One dimensional FFT.

Singleton (18) presented a method using four external storage files, such as magnetic tapes or serial disk files. In his method the $n = 2^m$ complex data points are first written on two of the files, with the first n/2 complex values in sequence on the first file and the remaining n/2 on the second. The Fast Fourier transform is computed while copying the data back and forth between the two pairs of files. An ALGOL program is also listed in his paper for those who may want to experiment with it.

When the random-access files are available, however, two files are sufficient for the Fast Fourier transform computing. The n data points

are initially written on the first file and subsequent computings are written on the second file. The implementation of the second method will be demonstrated by listing a BASIC computer program which can be compared with the core memory program in Fig. (12).

In Figure 12, the file "DATA1" contains the initial n-data and the file "data" is the computing one.

2. Two dimensional FFT.

When the FFT computing is implemented on Equation (95) of the two-dimensional case; first, one-dimensional Fourier transform is performed row by row starting from the first to the last row, then column by column another one-dimensional Fourier transform is carried out in a similar fashion to complete the required two-dimensional Fourier transform.

Because of the inherently large data points in the two-dimensional Fourier transform, the data are generally stored in the external storage. This may impose a serious restriction on the method described above. To state the problem more explicitly, let

$$B_{mn} = B(m,n), \quad m,n = 0,1,\ldots,N-1 \qquad (97)$$

be the matrix to be transformed. The transform is defined by

```
LIST 1,1100

250  PRINT D$;"OPEN DATA1"
260  PRINT D$;"DELETE DATA1"
270  PRINT D$;"OPEN DATA1,L16"
480  PRINT D$;"WRITE DATA1,R";J1
490  PRINT SR
500  PRINT D$;"WRITE DATA1,R";J2
510  PRINT SI
520  NEXT L
530 U =  - 1
530  PRINT D$;"CLOSE DATA1"
800  REM
810  FOR IT = 1 TO N
820 I = IT - 1
830 J = 0
840 M2 = 1
850 M1 = M2
860 M2 = M2 + M2
870 MI =  INT (I / M2)
880 MS = I - MI * M2
890  IF MS - M1 < 0 THEN 910
900 J = J + N / M2
910  IF M2 - N < 0 THEN 850
920 JF = J + 1
930 J2 = JF * 2
940 J1 = J2 - 1
950  PRINT D$;"OPEN DATA1,L16"
960  PRINT D$;"READ DATA1,R";J1
970  INPUT SR
980  PRINT D$;"READ DATA1,R";J2
990  INPUT SI
1000  PRINT D$;"CLOSE DATA1"
1010 I2 = 2 * IT
1020 I1 = I2 - 1
1030  PRINT D$;"OPEN DATA ,L16"
1040  PRINT D$;"WRITE DATA ,R";I1

1050  PRINT SR
1060  PRINT D$;"WRITE DATA ,R";I2

1070  PRINT SI
1080  PRINT D$;"CLOSE DATA"
1090  NEXT IT


]
```

Figure 12

```
]
```

```
LIST 1110,2000

1110  PRINT D$;"OPEN DATA,L16"
1120 L = 1
1130  IF L - N =  > 0 THEN 1540
1140 ST = 2 * L
1150 EL = L
1160  FOR M = 1 TO L
1170 A = PI * (1 - M) / EL
1180 WR =  COS (A)
1190 WI =  SIN (A)
1200  IF U < 0 THEN 1220
1210  GOTO 1230
1220 WI =  - WI
1230  FOR I = M TO N STEP ST
1240 J = I + L
1250 J2 = 2 * J
1260 J1 = J2 - 1
1270  PRINT D$;"READ DATA,R";J1
1280  INPUT FR
1290  PRINT D$;"READ DATA,R";J2
1300  INPUT FI
1310 TR = WR * FR - WI * FI
1320 TI = WR * FI + WI * FR
1330 I2 = 2 * I
1340 I1 = 2 * I - 1
1350  PRINT D$;"READ DATA,R";I1
1360  INPUT SR
1370  PRINT D$;"READ DATA,R";I2
1380  INPUT SI
1390 R1 = SR - TR
1400 R2 = SR + TR
1410 F1 = SI - TI
1420 F2 = SI + TI
1430  PRINT D$;"WRITE DATA,R";J1
1440  PRINT R1
1450  PRINT D$;"WRITE DATA,R";J2
1460  PRINT F1
1470  PRINT D$;"WRITE DATA,R";I1
1480  PRINT R2
1490  PRINT D$;"WRITE DATA,R";I2
1500  PRINT F2
1510  NEXT I,M
1520 L = ST
1530  GOTO 1130
1540  PRINT D$;"CLOSE DATA"
1550  RETURN
```

]

Figure 13

$$f(p,q) = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} B(m,n)\exp\, i2\pi\left(\frac{mp}{M} + \frac{nq}{N}\right) \qquad (98)$$
$$p = 0,1,\ldots,M-1,\ q = 0,1,\ldots,N-1$$

B(m,n) is stored on an external unit, e.g., a disk or tape, row by row, that is in the order B(0,0),B(0,1), B(0,2),...,B(0,N-1),B(1,0),B(1,1),...,B(M-1,N-1). B(m,n) is then read row by row and the following sum is calculated

$$g(m,q) = \sum_{n=0}^{N-1} B(m,n)\exp\left(i2\pi\frac{mq}{N}\right),\ q=0,1,\ldots,N-1 \qquad (99)$$

To minimize required storage, g(m,q) is stored in the location that earlier contained B(m,n). If the resulting arrays are written on an external data set, the matrix elements are still stored row by row. Now the next operation is to calculate

$$f(p,q) = \sum_{m=0}^{M-1} g(m,q)\exp\left(i2\pi\frac{mp}{M}\right),\ p=0,1,\ldots,M-1 \qquad (100)$$

that is,to form the sum of the elements in the matrix g multiplied by their corresponding exponential factor column-wise. But since g is stored row by row, it cannot be obtained by the elements in one column without reading through the whole matrix, picking from each row the proper element. Doing this straightforwardly, it is required to go through the matrix M times, which will result in excessive input output time, even in the

case of moderate matrix sizes. One way to avoid this is evidently to transpose the matrix g, which is by no means trivial. The problem can be formulated as one of constructing a specific permutation of a given sequence when only part of the sequence can be kept in the main storage and operated on at the same time, minimizing the number of the times the matrix is worked through, and each element is taken into the core. Theoretically, the algorithm requires that the matrix - that is supposedly stored on an external data set unit - is brought into the main storage and returned again N times. This number, however, can be reduced to a quarter of that by efficient programming. The algorithm will be presented by an example of 8x8 matrix A.

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} & a_{16} & a_{17} & a_{18} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} & a_{26} & a_{27} & a_{28} \\ & & & & & & & \\ & & & & & & & \\ & & & & & & & \\ a_{81} & a_{82} & a_{83} & a_{84} & a_{85} & a_{86} & a_{87} & a_{88} \end{bmatrix}$$

The matrix will be transposed in the following steps.

Step 1: First read row 1 and 2 and let a_{12}, a_{14}, a_{16}, and a_{18} change places with a_{21}, a_{23}, a_{25} and a_{27} respectively. This means that the elements of even columns in row 1 swap the elements of odd columns in row 2. Then write row 1 and 2 in their former places and repeat the procedure with row 3 and 4, row 5 and 6, and row 7 and 8. The result is:

$$A_1 = \begin{bmatrix} a_{11} & a_{21} & a_{13} & a_{23} & a_{15} & a_{25} & a_{17} & a_{27} \\ a_{12} & a_{22} & a_{14} & a_{24} & a_{16} & a_{26} & a_{18} & a_{28} \\ a_{31} & a_{41} & a_{33} & a_{43} & a_{35} & a_{45} & a_{37} & a_{47} \\ a_{32} & a_{42} & a_{34} & a_{44} & a_{36} & a_{46} & a_{38} & a_{48} \\ a_{51} & a_{61} & a_{53} & a_{63} & a_{55} & a_{65} & a_{57} & a_{67} \\ a_{52} & a_{62} & a_{54} & a_{64} & a_{56} & a_{66} & a_{58} & a_{68} \\ a_{71} & a_{81} & a_{73} & a_{83} & a_{75} & a_{85} & a_{77} & a_{87} \\ a_{72} & a_{82} & a_{74} & a_{84} & a_{76} & a_{86} & a_{78} & a_{88} \end{bmatrix}$$

Step 2: Read row 1 and row 3 of A_1, interchange the elements 3,4,5,6,7, and 8 in row 1, with elements 1,2,3,4,5, and 6 in row 3, and finally write row 1 and row 3 in their proper locations and repeat the procedure with row 2 and row 4, row 3 and row 5, row 4 and row 6, row 5 and row 7, and row 6 and row 8. The intermediate result is:

$$A_2 = \begin{bmatrix} a_{11} & a_{21} & a_{31} & a_{41} & a_{33} & a_{43} & a_{35} & a_{45} \\ a_{12} & a_{22} & a_{32} & a_{42} & a_{34} & a_{44} & a_{36} & a_{46} \\ a_{13} & a_{23} & a_{51} & a_{61} & a_{53} & a_{63} & a_{55} & a_{65} \\ a_{14} & a_{24} & a_{52} & a_{62} & a_{54} & a_{64} & a_{56} & a_{66} \\ a_{15} & a_{25} & a_{71} & a_{81} & a_{73} & a_{83} & a_{75} & a_{85} \\ a_{16} & a_{26} & a_{72} & a_{82} & a_{74} & a_{84} & a_{76} & a_{86} \\ a_{17} & a_{27} & a_{37} & a_{47} & a_{57} & a_{67} & a_{77} & a_{87} \\ a_{18} & a_{28} & a_{38} & a_{48} & a_{58} & a_{68} & a_{78} & a_{88} \end{bmatrix}$$

Step 3: Columns 1 and 2, and rows 7 and 8 of A_2 have been transposed. The upper-right corner elements are yet to be transposed. The procedure of step 2 is repeated for the 6x6 matrix. The result is:

$$A_3 = \begin{bmatrix} a_{11} & a_{21} & a_{31} & a_{41} & a_{51} & a_{61} & a_{53} & a_{63} \\ a_{12} & a_{22} & a_{32} & a_{42} & a_{52} & a_{62} & a_{54} & a_{64} \\ a_{13} & a_{23} & a_{33} & a_{43} & a_{71} & a_{81} & a_{73} & a_{83} \\ a_{14} & a_{24} & a_{34} & a_{44} & a_{72} & a_{82} & a_{74} & a_{84} \\ a_{15} & a_{25} & a_{35} & a_{45} & a_{55} & a_{65} & a_{75} & a_{85} \\ a_{16} & a_{26} & a_{36} & a_{46} & a_{56} & a_{66} & a_{76} & a_{86} \\ a_{17} & a_{27} & a_{37} & a_{47} & a_{57} & a_{67} & a_{77} & a_{87} \\ a_{18} & a_{28} & a_{38} & a_{48} & a_{58} & a_{68} & a_{78} & a_{88} \end{bmatrix}$$

Step 4: Now the 4x4 submatrix in the upper corner is left to be transposed. The same procedure again of step 2 is repeated here to get finally

the transposed matrix A*.

$$A^* = \begin{bmatrix} a_{11} & a_{21} & a_{31} & a_{41} & a_{51} & a_{61} & a_{71} & a_{81} \\ a_{12} & a_{22} & a_{32} & a_{42} & a_{52} & a_{62} & a_{72} & a_{82} \\ a_{13} & a_{23} & a_{33} & a_{43} & a_{53} & a_{63} & a_{73} & a_{83} \\ a_{14} & a_{24} & a_{34} & a_{44} & a_{54} & a_{64} & a_{74} & a_{84} \\ a_{15} & a_{25} & a_{35} & a_{45} & a_{55} & a_{65} & a_{75} & a_{85} \\ a_{16} & a_{26} & a_{36} & a_{46} & a_{56} & a_{66} & a_{76} & a_{86} \\ a_{17} & a_{27} & a_{37} & a_{47} & a_{57} & a_{67} & a_{77} & a_{87} \\ a_{18} & a_{28} & a_{38} & a_{48} & a_{58} & a_{68} & a_{78} & a_{88} \end{bmatrix}$$

When the random-access file is available, all the complications inherent to the sequential file can be eliminated. The random access file will be used in the subsequent computer programs.

CHAPTER V

APPLICATIONS OF THE SIMULATION METHOD

1. Introduction.

Vibratory motions in mechanical and structural systems can in many cases be represented by random vibrations. Examples include such motions as (a) the response of aircraft to aerodynamic noise and turbulent boundary layers, (b) the response of ground vehicles to rough roadways, (c) analysis of ship oscillations caused by ocean waves, (d) the statistical strength analysis of engineering materials, (e) the response of off-shore structures to wave and wind forces, and (f) the response of tall structures to wind and seismic forces.

In subsequent sections, 3 examples will be presented to demonstrate the usefulness of the digital simulations to various vibratory motions in structural engineering. The first two of the examples are both one-dimensional cases and Eq. (87) will be applied to them. The third example shows the application of Equation (95) to a tall structure subject to buffeting of a two-dimensional wind field.

2. An Artificial Earthquake Accelerogram.

Because of the statistical nature of earthquake ground motions, the information that can be obtained from structural response studies based on only a single earthquake record is limited, and it would be desirable to have a large sample of recorded strong-motion earthquake accelerograms for each class of intensity and duration at various epicentral distances. For this reason, it is desirable to develop a practical method of constructing random functions that can be used in place of earthquake accelerograms in making studies of the response of structures. A classical method of modelling the strong earthquake motion will first be demonstrated. Then the simulation method introduced in Chapter III will be contrasted with the classical method.

In a constant parameter linear system with a weighting function $h(\tau)$, the response $y(t)$ can be defined (19) by:

$$y(t) = \int_0^\infty h(\tau)x(t-\tau)d\tau \tag{101}$$

For a pair of times t and $t+\tau$, the product $y(t)y(t+\tau)$ is given (20) by

$$y(t)y(t+\tau) = \int_0^\infty\int h(\xi)h(\eta)x(t-\xi)x(t+\tau-\eta)d\xi\,d\eta \tag{102}$$

The expected value of Equation (102) yields

$$R_y(\tau) = \iint_0^\infty h(\xi)h(\eta)R_x(\tau+\xi-\eta)\,d\xi\,d\eta \qquad (103)$$

The transformation of Equation (103) to a complex-valued frequency domain by taking the Fourier transform yields the power spectrum relation. i.e.,

$$S_y(\omega) = |H(\omega)|^2 S_x(\omega) \qquad (104)$$

where $H(\omega)$ is the Fourier transform of $h(\tau)$ and $S_x(\omega)$ is the power spectral density of the excitation. If the input power spectral density is given by a Gaussian white noise with a mean zero and magnitude 1, then

$$S_y(\omega) = |H(\omega)|^2 \qquad (105)$$

The response power spectral density to be used here is that developed by Tajimi (23) from the work of Housner and Jennings:

$$S(\omega) = \frac{\left[1 + 4\zeta_g^2 (\omega/\omega_g)^2\right] G_o}{\left[1-(\omega/\omega_g)^2\right]^2 + 4\zeta_g^2(\omega/\omega_g)^2} \qquad (106)$$

With $S_x(\omega) = 1$ and $S_y(\omega)$ given by Equation (105), it is found that

$$H(\omega) = \frac{\sqrt{G_o}(1 + i2\zeta\omega/\omega_g)}{(1 - \omega^2/\omega_g) + i2\zeta\omega/\omega_g} \qquad (107)$$

and

$$h(\tau) = \sqrt{G_o}\left[\sqrt{1 - 2\zeta^2}\,\frac{\omega_g^2}{\omega_d}\,e^{-\zeta\omega_g\tau}\sin\omega_d\tau + 2\zeta\omega_g e^{-\zeta\omega_g\tau}\cos\omega_d\tau\right] \tag{108}$$

If $\underline{N}(t)$ represents a white noise ensemble with power spectral density 1, then a sample of the pseudo-earthquake ensemble will be given by

$$\ddot{\underline{y}}(t) = \int_{-\infty}^{t} \underline{N}(t)\,\underline{h}(t - \tau)\,d\tau \tag{109}$$

in which $\ddot{\underline{y}}(t)$ is the pseudo-earthquake accelerations. However, Eq. (109) is not in a form convenient for generating $\ddot{\underline{y}}(t)$ on the digital computer. It would be more desirable to represent Equation (109) by a differential equation that could be used more readily for producing samples of the process. It will now be shown that the desired differential equation is:

$$\ddot{z}(t) + 2\zeta\omega_g\dot{z}(t) + \omega_g^2 z(t) = -F(t) \tag{110}$$

This equation can describe the relative displacement of a linear, viscously damped oscillator whose base is excited by the acceleration F(t). By differentiating the integral form of the relative displacement, the absolute acceleration is found to be

$$\ddot{z}(t) + F(t) = \sqrt{1 - 2\zeta^2}\frac{\omega_g^2}{\omega_d}\int_o^t F(\tau)\exp\left[-\zeta\omega_g(t-\tau)\right]$$
$$\sin\omega_d(t-\tau)d\tau + 2\zeta\omega_d\int_o^t F(\tau)\exp$$
$$\left[-\zeta\omega_g(t-\tau)\right]\cos\omega_d(t-\tau)d\tau \qquad (111)$$

where $\omega_d = \omega_g\sqrt{1-\zeta^2}$

From Equation (111), it is seen that the weighting function for the absolute acceleration is identical with Eq. (108), except for a factor of $\sqrt{G_o}$. The differential equation to be used instead of Equation (111) is therefore obtained by replacing F(t) by $\sqrt{G_o}N(t)$:

$$\ddot{z}(t) + 2\zeta\omega_g\dot{z}(t) + \omega_g z(t) = -\sqrt{G_o}N(t) \qquad (112)$$

The pseudo-earthquake accelerogram is

$$\ddot{y}(t) = \ddot{z}(t) + \sqrt{G_o}N(t) = -\left[2\zeta\omega_g\dot{z}(t) + \omega_g z(t)\right] \qquad (113)$$

Housner and Jennings (4) have shown a convenient method of choosing initial conditions to integrate Equation (113) and their results can be found in the reference (4).

A basic form of a homogeneous one-dimensional random process y(t) with zero mean and spectral density $S_y(\omega)$ can be expressed by

$$\ddot{y}(t) = \sum_{n=1}^{N}\sqrt{2S_y(\omega_n)\Delta\omega}\,\sin(\omega_n t + \phi_n) \qquad (114)$$

with parameters according to Housner and Jennings (4):

$G_o = 0.01238$, $\zeta^2 = .410$, and $\omega_g = 242.$

Equation (106)of the response spectral density becomes

$$S_y(\omega) = 0.01238(1 + \omega^2/147.8)/(1 - \omega^2/242)^2 + \omega^2/147.8$$

Figure (14) shows the relation between $S_y(\omega)$ and ω. The generation of the random function, i.e., Eq.(114) is fairly straightforward, once parameters are chosen. In order to compare the results from Equation (114) with those of Housner and Jennings (4), the same data are used in Equation (114), i.e.,

ω_u = 30 rad/sec; t = 26.808 sec; N = 128

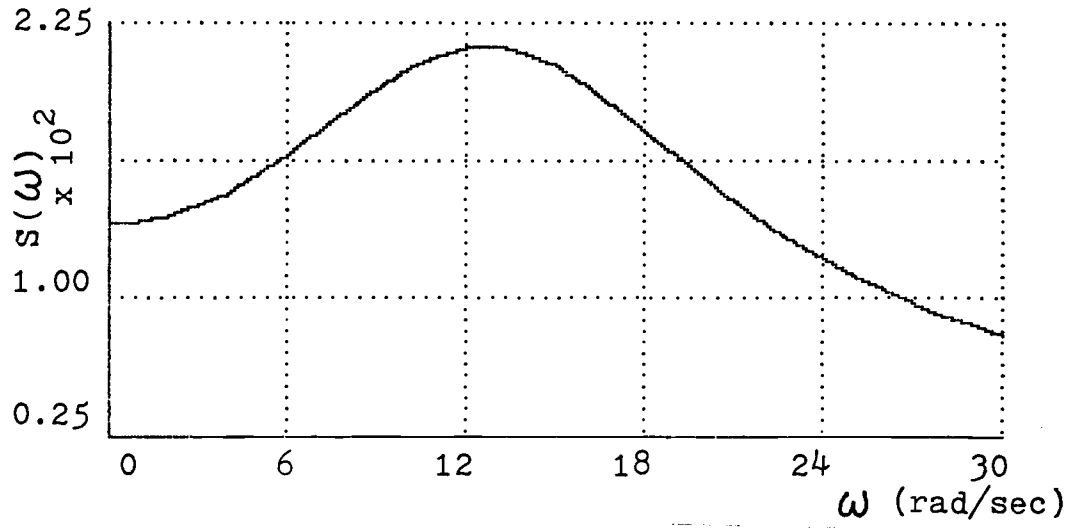

Figure 14

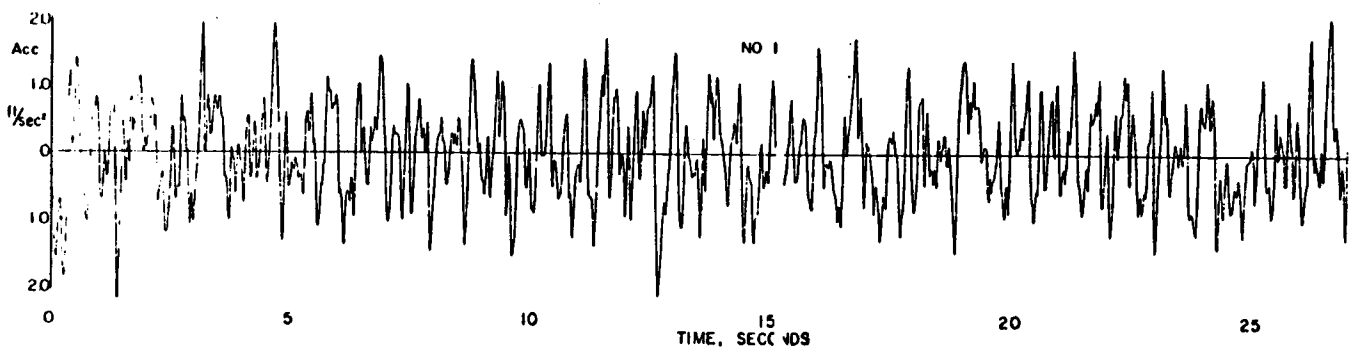

Figure 15 Simulated Accelerogram
by Ref.(4)

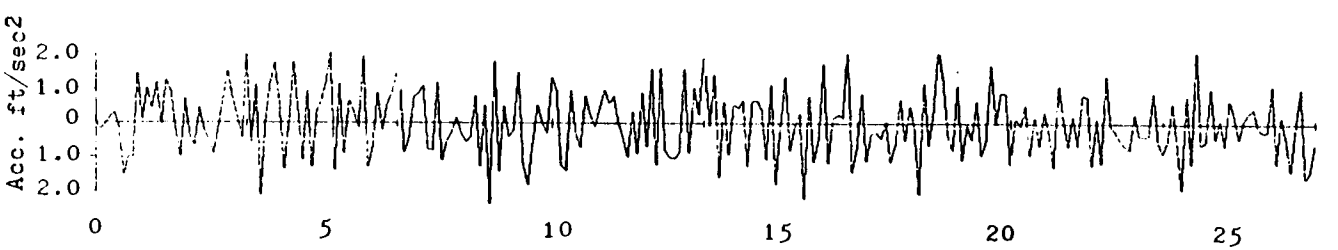

Figure 16 Simulated Accelerogram
by the Proposed Method

Figure (15) shows the result of the simulation method by Housner and Jennings(4) and Figure (16) is the result of the simulated accelerogram by implementing Equation (114). The results obtained by the proposed method compare very favorably with the ones obtained by Ref. (4) in terms of statistical charateristics, i.e. the number of zero crossings and the peak value. Fig. (15) by Ref.(4) has 158 zero crossings and 2.00 ft/sec^2 for its maximum peak value, while the proposed method has 156 zero crossings and 2.00 ft/sec^2 for its maximum peak value.

The use of Equation (114) facilitates generation of random functions and when it is incorporated into a computer program with the aid of FFT, the time expended for computation is far less than the classical method (24). The BASIC computer program is listed in the Appendix for the problem.

3. Simulation of Wind Generated Waves

Random irregular waves induced by wind are difficult to incorporate into engineering design. Yet, interest in the dynamic response of offshore structures due to ocean wave forces has increased considerably in the past decade. It is frequently not possible to analytically model the statistical properties of waves. In fields of this type, the use of simulation techniques has been the only successful method for obtaining solutions. The wave velocity and acceleration, which are prerequisite for the dynamic analysis of offshore structures, will be simulated in the following.

For waves induced by wind under fully developed sea conditions, the following one-sided wave spectrum (25),(26) will be assumed:

$$S_{hh}(\omega) = \frac{\alpha g^2}{\omega^5} \exp\left[-\sigma\left(\frac{\omega_o}{\omega}\right)^4\right] \quad (0 \leq \omega < \infty) \quad (115)$$

where $\alpha = 8.1 \times 10$, $\sigma = 0.74$, and $\omega_o = g/V$, with V being the mean wind velocity. A plot of this spectrum for three different values of wind velocity is given in Figure 17.

For linear wave theory, the horizontal component of wave particle velocity, v, for deep water waves can be represented by (27)

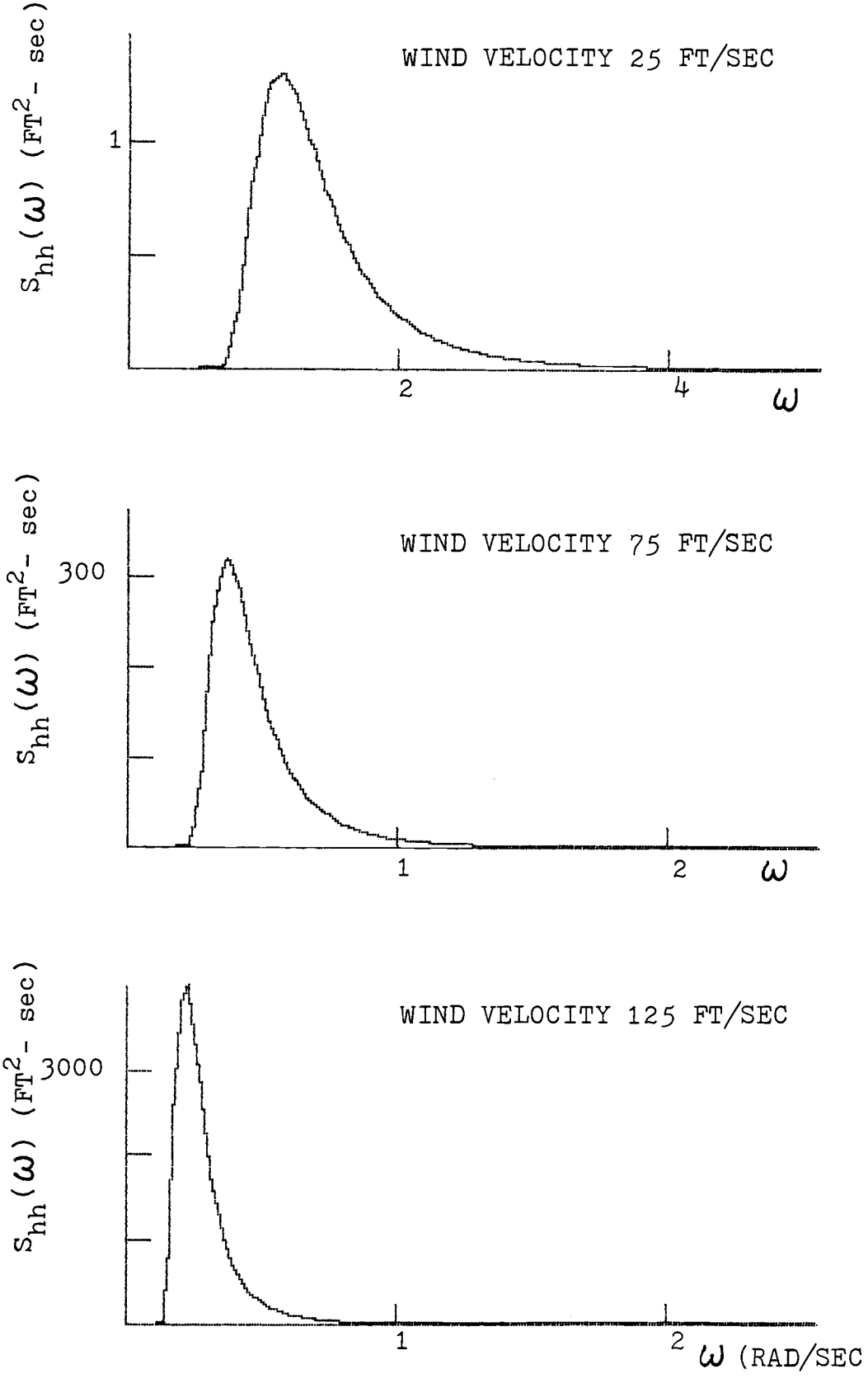

Figure 17. One-sided spectral densities

$$\dot{v}(y,t)=\sum_{n=1}^{N}\sqrt{2S_{nn}(\omega_i)\Delta\omega}\;\omega_i \exp(-\omega_i^2\,\frac{y}{g})\cos(\omega_i t+\phi_i) \tag{116}$$

where ϕ_i = random phase angles uniformly distributed between 0 and 2π

ω_u = the upper cut-off frequency and equal to $N\Delta\omega$

$\omega_i = i\Delta\omega$

The acceleration, $\ddot{v}$, is obtained by differentiating Eq. (116) with respect to time:

$$\ddot{v}(y,t)=-\sum_{n=1}^{N}\sqrt{2S_{nn}(\omega_i)\Delta\omega}\;\omega_i^2 \exp(-\omega_i^2\,\frac{y}{g})\sin(\omega_i t+\phi_i) \tag{117}$$

Sample simulations of wind velocity $\dot{v}$ = 25, 75 and 125 have been performed with the following data:

N = 512

ω_u = 24.0 rad/sec

$\Delta t = 2\pi/24$ sec

g = 32.2 ft/sec2

y = 10 ft

Therefore, $T = N\Delta t$ = 134.04 sec. Figure 18 shows the time history of the velocity at 75 ft/sec. A computer listing in BASIC of Eq. (116) is also appended. Once the velocity and acceleration have been obtained, a standard procedure of dynamic analysis can be carried out to design an offshore structure. According to Malhotra and Penzien (28), the equation of motion for a discrete mass

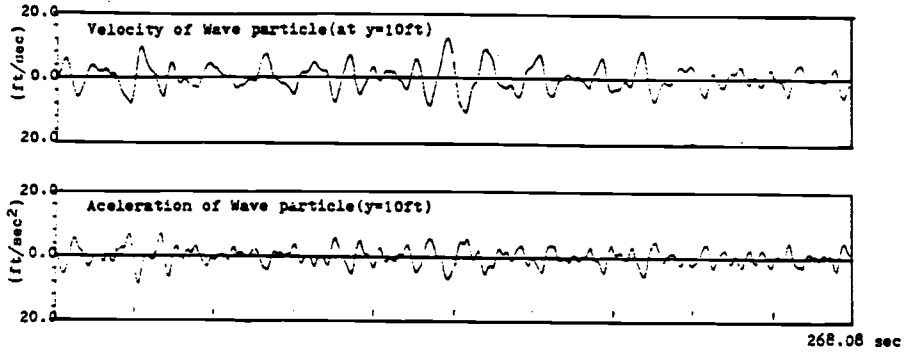

Fig. 18 Simulation by Eq. (116)
Wave Velocity = 75 ft/sec

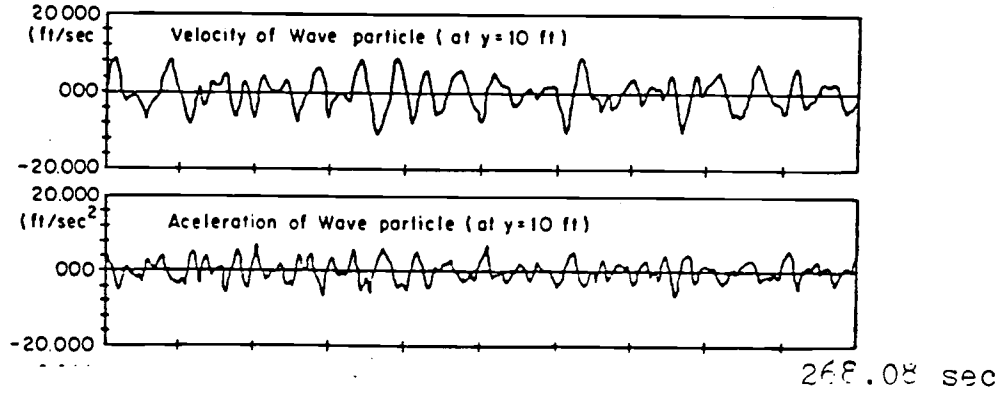

Fig. 19 Simulation by Ref. 27
Wave Velocity = 75 ft/sec

system can be written as follows:

$$\left.\begin{aligned} &[M]\{\ddot{u}\} + [C]\{\dot{u}\} + [K]\{u\} = \{F\} \\ &\{F\} = [C_D]\{(\dot{v}-\dot{u})|\dot{v}-\dot{u}|\} + \rho[\nabla]\{\ddot{v}\} + [C_M - \rho\nabla]\{\ddot{v}-\ddot{u}\} \end{aligned}\right\} \quad (118)$$

where $[M]$ is the mass matrix; $[C]$ is the structural damping matrix; $[K]$ is the structural stiffness matrix; $\{F\}$ is the resultant wave force vector consisting of the contribution from the drag force $[C_D]\{(\dot{v} - \dot{u})|\dot{v} - \dot{u}|\}$, the inertia force due to the mass of the fluid displaced by the structure $\rho[\nabla]\{\ddot{v}\}$, and the inertia force due to the added mass based on the relative acceleration $[C_M - \rho\nabla]\{\ddot{v} - \ddot{u}\}$; $[C_M] = \rho K_M[\nabla]$; $[C_D] = \rho K_D[A]$; ρ is the mass density of water; K_M is the empirical coefficient of inertia; K_D is the empirical coefficient of drag; $[A]$ is the diagonal matrix indicating area projected in the direction of the flow. Assuming the normal mode solution, the generalized coordinate $\{Y\}$ is introduced, i.e.,

$$\{u\} = [\phi]\{Y\} \quad (119)$$

then Equation (118) can be expressed:

$$[m]\{\ddot{Y}\} + [c]\{\dot{Y}\} + [k]\{Y\} = [F_M]\{\ddot{v}\} + [F_D]\{(\dot{v} - [\phi]\{\dot{Y}\})|\dot{v} - [\phi]\{\dot{Y}\}|\} \quad (120)$$

in which $[m] = [\phi]^T[M + (C_M - \rho\nabla)][\phi]$ is the generalized mass matrix; $[c] = [\phi]^T[C][\phi]$ is the generalized damping

matrix; $[k] = [\phi]^T[K][\phi]$ is the generalized stiffness matrix; $[F_M] = [\phi]^T[C_M]$ and $[F_D] = [\phi]^T[C_D]$ are the coefficient matrices of the generalized fluid forces; and the superscript T indicates matrix transposed. The modal matrix $[\phi]$ can be obtained by solving the linearized, homogeneous, and undamped equation

$$[M + (C_M - \rho\nabla)]\{-\omega^2 u\} + [K]\{u\} = \{0\} \qquad (121)$$

After the time histories of the wave particle velocity $\dot{v}$ and the acceleration $\ddot{v}$ are generated, Eq. (120) can be solved for $\{Y(t)\}$ directly in the time domain utilizing a numerical integration procedure. Once $\{Y(t)\}$ is evaluated, $\{u\}$ can be obtained by means of Eq. (119).

Being a one-dimensional example, the results shown in Figure 18 match well in terms of statistical properties the results presented in Ref. (27), which is shown in Figure (19): the number of zero crossings is 48 and the maximum peak value is 12.01 ft/sec. The proposed method has 47 zero crossing and 12.08 ft/sec for the maximum peak value.

4. Two Dimensional Wind Field Generation.

With the view of limiting the memory space in a computer, a two dimensional frozen wind field is defined as follows:

$$u_o(x,y) = \sum_{m=1}^{N_1}\sum_{n=1}^{N_2} A_{mn}\sin(k_{1m}x + k_{2n}y + \phi_{mn}) \qquad (122)$$

$$\text{where } A_{mn} = \left[8S_1(k_{1m},k_{2n})\Delta k_1 \Delta k_2\right]^{\frac{1}{2}} \qquad (123)$$

in which $k_1 = k_{1u}/N_1$; $k_2 = k_{2u}/N_2$;

$$k_{1m} = m\Delta k_1; \quad k_{2n} = n\Delta k_2 \qquad (124)$$

and ϕ_{mn} = random phase angle

In Eq. (124), k_{1u} and k_{2u} are the upper cutoff wave numbers in the first quadrant of the k_1 - k_2 plane, such that $S_1(k_1,k_2)$ is significant only in the domain D_o specified by

$$D_o: \quad 0 \leq k_1 \leq k_{1u}, \quad 0 \leq k_2 \leq k_{2u}$$

Prior to implementing the FFT technique to Equation (122), the spectral density function will be defined. Let $u(x,y,t)$ denote the fluctuating components of the wind velocity in the along wind direction. Assuming that $u(x,y,t)$ is a homogeneous random process in x,y and t, the autocorrelation function, $R(\xi,\eta)$ of $u(x,y,t)$ at a specified altitude y is defined as

$$R(\xi,\eta) = E\left[u(x+\xi,y,t+\tau)u(x,y,t)\right] \qquad (125)$$

According to Davenport (29), the following relation can be established:

$$\Phi_o(\xi,\omega) = \frac{1}{2\pi}\int_{-\infty}^{\infty} R(\xi,\tau)\exp(-i\omega\tau)d\tau$$

$$= \exp(-\bar{c}|\omega||\xi|)\Phi_o(\omega) \qquad (126)$$

where $\exp(-\bar{c}|\omega||\xi|)$ is the space correlation and $\bar{c}$ is a positive constant related to the scale of turbulence; and Φ_o is the two-sided mean square spectral density of u(x,y,t), i.e.,

$$\Phi_o(\omega) = \frac{1}{2\pi}\int_{-\infty}^{\infty} R_u(\tau)\exp(-i\omega\tau)d\tau \qquad (127)$$

where $R_u(\tau) = E\left[u(x,y,t+\tau)u(x,y,t)\right]$ (128)

Generally in designing engineering structures, the dynamic structure-wind interaction is not paramount, thus the space correlation may be approximated as $\exp(-|\xi|/L)$, which is independent of ω. Therefore

$$\Phi(\xi,\omega) = \exp\left(\frac{-|\xi|}{L}\right)\Phi_o(\omega) \qquad (129)$$

where L = correlation distance in space. Writing $S_o(\omega)$ for the one-sided version of $\Phi_o(\omega)$, it is assumed for analytical convenience that $S_o(\omega)$ is of the following form (30),(31);

$$S_0(\omega) = \frac{K_1 \lambda \overline{u^2}}{2\pi U(y)(1 + X^2)^{5/6}} \tag{130}$$

in which $\overline{u}^2$ = mean-square value of $u(x,y,t) = 6.0 K_r U_1^2$ (131)

$$X = \frac{\lambda \omega}{2\pi U(y)} \tag{132}$$

$$K_1 = \frac{2\Gamma\left(\frac{5}{6}\right)}{\Gamma\left(\frac{1}{2}\right)\Gamma\left(\frac{1}{3}\right)} = 0.4751 \tag{133}$$

$U(y)$ = mean wind velocity at altitude $y = U_1 \left(\frac{y}{y_1}\right)^{\alpha}$ (134)

$$\lambda = \frac{(\overline{u^2})^{3/2} y_1}{K_2(\alpha U_1)} \tag{135}$$

with K_r = surface drag coeffieient; α = exponent of the power law governing the profile of mean wind velocity; y_1 = reference height = 33 ft; U_1 = mean wind velocity at reference height y_1; $\Gamma(\ )$ = gamma function defined as follows (32):

$$\Gamma(x) = \lim_{n \to \infty} \frac{1\ 2\ 3 \cdots (n-1)}{x(x+1)(x+2)\cdots(x+n-1)}\, n^x$$

$$K_2 = (2\pi)^{-1} K^2 a^{3/2} \left(\frac{1}{K_1}\right) \tag{136}$$

where κ = the Karman constant = 0.4; and a = the universal Kolmogorov constant = 0.5.

The intensity of $S_o(\omega)$ varies with altitude y. For the first approximation and for analytical simpli-

fication, the geometrical center, $y = y^*$ will be elected to be used in computation (31).

Then Eq. (130) can be written as

$$S_o(\omega) = \frac{K_1 \lambda^* \bar{u}^2}{2\pi U(y^*)(1 + X^{*2})^{5/6}} \tag{137}$$

where $\lambda^* = \frac{(\bar{u}^2)^{3/2} y_1}{K_2 (\alpha U_1)^3} \left(\frac{y^*}{y_1}\right)^{1-3\alpha}$, $X^* = \frac{\lambda^* \omega}{2\pi U(y^*)}$ (138)

For ease of algebraic manipulation, the following forms are defined:

$$c = \frac{K_1 \lambda^* u^2}{2\pi U(y^*)}, \quad b = \frac{\lambda^*}{2\pi U(y^*)} \tag{139}$$

Then

$$S_o(\omega) = \frac{c}{(1 + b^2\omega^2)^{5/6}} \tag{140}$$

The inverse Wiener-Khintchine transform of Equation (129)with respect to ω gives the autocorrelation function, $R(\xi, \tau)$:

$$R(\xi, \tau) = \exp\left(\frac{-|\xi|}{L}\right) \int_0^\infty S_o(\omega) \cos\omega\tau \, d\omega \tag{141}$$

Substituting Equation (140)into Equation (141) it follows

$$R(\xi, \tau) = \exp\left(\frac{-|\xi|}{L}\right) \frac{c\pi^{\frac{1}{2}}}{b\Gamma\left(\frac{5}{6}\right)} \left(\frac{\tau}{2b}\right)^{1/3} K_{1/3}\left(\frac{\tau}{b}\right) \tag{142}$$

where $K_\nu(x)$ = the modified Bessel function of the second kind. It can be shown that

$$\lim_{\tau \to 0} \left(\frac{\tau}{2b}\right)^{\frac{1}{3}} K_{1/3}\left(\frac{\tau}{b}\right) = \tfrac{1}{2}\Gamma\left(\tfrac{1}{3}\right) = 1.340 \tag{143}$$

thus

$$R(\xi, 0) = R_o(\xi) = \frac{\pi^{\frac{1}{2}}\Gamma\left(\frac{1}{3}\right) c}{2b\Gamma\left(\frac{5}{6}\right)} \exp\left(\frac{-|\xi|}{L}\right) \tag{144}$$

Now the frozen field, which has been mentioned earlier, will be formally defined. With the assumption that the fluctuating component of wind velocity, $u(x,y,t) = u_o(x,y)$, at a fixed value of t, which is the frozen wind field, depends only upon the distance between two points in the x-y plane. It follows that the autocorrelation function of the frozen field, i.e.,

$$R_1(\xi, \eta) = E\left[u_o(x+\xi, y+\eta)u_o(x,y)\right] \tag{145}$$

can be written as

$$R_1(\xi, \eta) = R_o(\rho) \tag{146}$$

where $\rho = (\xi^2 + \eta^2)^{\frac{1}{2}}$ (147)

Then

$$R_1(\xi, \eta) = R_o(\rho) = \frac{\pi^{\frac{1}{2}}\Gamma\left(\frac{1}{3}\right) c}{2b\Gamma\left(\frac{5}{6}\right)} \exp\left(\frac{-|\rho|}{L}\right) \tag{148}$$

The two-dimensional Wiener-Khintchine transform of $R_1(\xi, \eta)$ leads to the corresponding two-dimensional

spectrum, $S_1(k_1,k_2)$, of the frozen field:

$$S_1(k_1,k_2)=\frac{1}{(2\pi)^2}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}R_1(\xi,\eta)\exp -i(k_1\xi+k_2\eta)\,d\xi\,d\eta \qquad (149)$$

in which k_1 and k_2 = the wave numbers in the x and y directions, respectively. It can be shown (33) then that $S_1(k_1,k_2) = S(k)$ is obtained from

$$S(k) = \frac{1}{2\pi}\int_0^{\infty}\rho R_o(\rho)J_o(\rho k)d\rho \qquad (150)$$

where $k = (k_1^2 + k_2^2)^{\frac{1}{2}}$ and $J_o(\ )$ = the Bessel function of order zero.

According to Tables (34)

$$\int_0^{\infty}\exp(-\alpha x)J_\nu(\beta x)x^{\nu+1}dx = \frac{2\alpha(2\beta)^\nu\,\Gamma\left(\nu+\frac{3}{2}\right)}{\left[\pi(\alpha^2+\beta^2)^{\nu+3/2}\right]^{\frac{1}{2}}} \qquad (151)$$

Therefore

$$S(k) = \frac{\Gamma(\frac{1}{2})c}{4\sqrt{\pi}\,\Gamma\left(\frac{5}{6}\right)b}\,\frac{L^2}{(1+L^2k^2)^{3/2}} \qquad (152)$$

Equation (152) can be rewritten in the following form:

$$S_1(k_1,k_2) = \frac{Q}{\left[1+L^2(k_1^2+k_2^2)\right]^{3/2}} \qquad (153)$$

where

$$Q = \frac{\Gamma\left(\frac{1}{3}\right)L^2 c}{4\sqrt{\pi}\,\Gamma\left(\frac{5}{6}\right)b} \tag{154}$$

Now Equation (95) can be applied to any numerical example. As an example, a structure having the following parameters will be considered:

Height = 400 ft; Width = 400 ft

K_r = 0.03; L = 280 ft; y* = 200 ft

U_1 = 30 fps; α = 1/7

The conceptual relationship between the structure and wind field is shown in Figure 20.

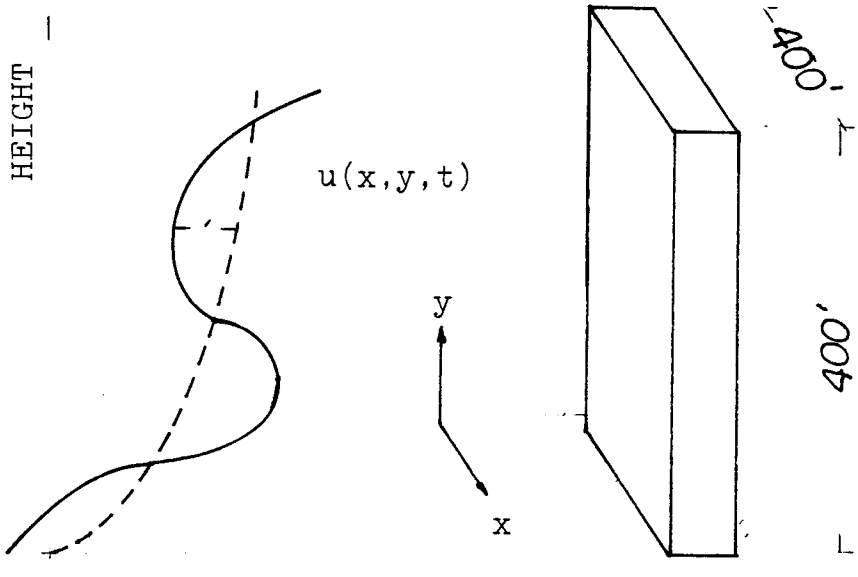

Figure 20.

A two-dimensional wind field is generated by telescoping Equations (153), (90), and (95). The associated parameters with the FFT technique are:

$$N_1 = N_2 = 64; \quad \Delta x = \Delta y = 400/64$$

$$k_{1u} = k_{2u} = 2\pi/\Delta x.$$

The BASIC computer program of the above wind field is listed in the Appendix. To get the same result, the simulation method in Ref (31) requires a two-dimensional array of 256 x 256 complex numbers while the proposed method requires an array of only 64 x 64. For ready comparison, two results are shown in Figures (21) and (22): Figure (21) shows a two-dimensional wind field simulation by the superposition of sine functions using an array of N_1 x N_2 = 16 x 16 and Figure (22) shows the same simulation by the sum of cosine functions with an array of N_1 x N_2 =64 x 64. Note the requirement of a larger array for the simulation method by the sum of cosine functions. The statistical properties of two results are very similar, which shows the advantage of the simulation method proposed here over the one in Ref. (31). Sample simulations of mean wind velocity U_1 = 15, 22.5, and 30 have been performed by using both methods and the results are shown in Table 1.

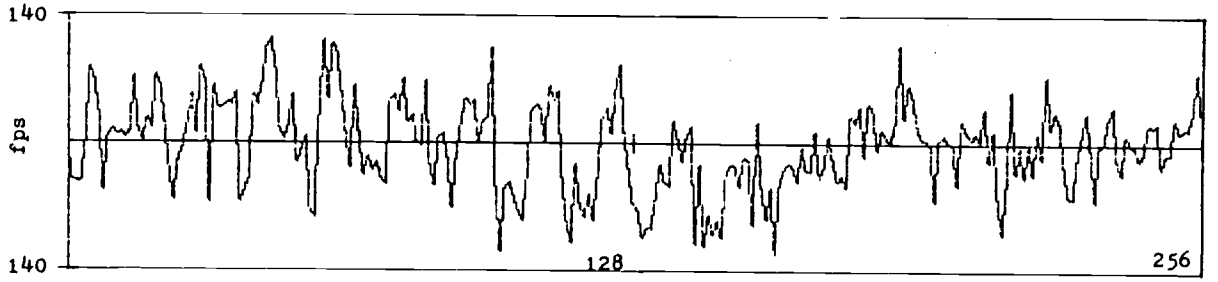

Figure 21 Sample simulation by the proposed method

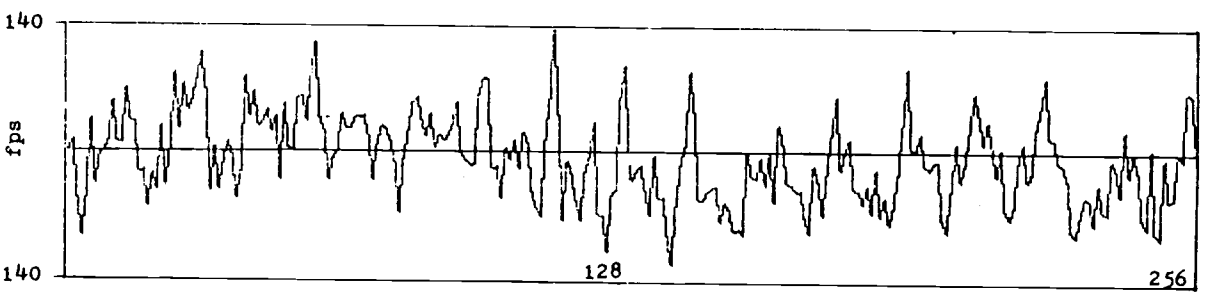

Figure 22 Sample simulation by Ref. (31)

TABLE I Numerical Results

U_1 = Mean Velocity (ft/sec)	Simulation Method by Ref. (31)		Simulation by the proposed method	
	Maximum Peak Value (ft/sec)	Number of Zero Crossings	Maximum Peak Value (ft/sec)	Number of Zero Crossings
15.0	30.06	64	29.10	62
22.5	81.44	63	80.21	65
30.0	140.01	68	138.04	64

CHAPTER VI
CONCLUSION

A method of simulating a multivariate and multi-correlated random process has been presented by the sum of sine function. Then, methods of incorporating the fast Fourier transform into the simulating functions have been reviewed in order to optimize computer memory and execution time requirements.

In Chapter 2, it has been pointed out that the simulating method by the sum of cosine functions which is proposed in Ref. (8) seems to be most efficient in the past. The method which has been presented in this document is a refinement to that presented in Ref (8): a. the limits of summation in Equation (1), i.e.,

$$X(t) = \sum A_n \sin(\omega_n + \phi_n)$$

are from 1 to N. In the FFT, however, the limits are from 0 to N-1. The sum of sine functions $\sin(2\pi nk/N)$ are zero at n=0 and n=N. Therefore the function (1) is more definite than the function (2). b. In generating and modelling a one dimensional field, there is no clear advantage using either Equation (1) or (2). The

disadvantage of using Equation (2) comes into play when it is applied in simulating two or three dimensional fields. For example, in Ref. (31), the two dimensional wind field is represented by the expanded domain of an array of 512 x 512 complex numbers. The method introduced in Ch. 4 requires an array of 128 x 128 complex numbers.

Occasionally investigators still use Equations (1) or (2) for sample simulations (24). The only drawback is the amount of computer storage, which can be prohibitive if there are too many correlated processes to be simulated or if a very large number of discretized points are necessary for analysis. It is well known that the FFT method works much faster in terms of computer time compared to the sinusoid models. The incorporation of the FFT techniques to the digital simulation is all but indispensible except for special cases. Some of the pertinent characteristics of FFT which are very effective in reducing computer time requirements are presented in Chapter 4. For example, the transform of a single sequence of 2N real data points can be done by means of one DFT of N complex points which has been also presented by Hudspeth and Borgman in Ref. (16).

In many problems in structural engineering, it is desirable to perform three-dimensional simulation. The methodology does not present any difficulty; the three-dimensional package program like HARM has been around for a few years. It is rather a matter of available computer times and its cost. For that matter, the mini- or micro-computer presents a very attractive alternative. A three dimensional array of 128 x 128 x 128, which is a minimum set of data points for any practical application, requires about 40 mega bytes. Even micro-computers plus disk drives having that much memory have been around from many companies for several years. The FFT program in Appendix can be readily expanded to three dimensional cases.

REFERENCES

1. Shinozuka, M., "Monte Carlo Solution of Structural Dynamics," Int. Journal of Computers and Structures, Vol. 2, 1972, pp. 855 - 874.

2. Shinozuka, M., "Digital Simulation of Random Processes in Engineering Mechanics with the Aid of FFT Technique," Stochastic Problems in Mechanics, University of Waterloo Press, Waterloo, Ontario, Canada, 1974, pp 277 - 286

3. Rice, S.O., "Mathematical Analysis of Random Noise" in Selected Papers of Noise and Stochastic Processes, Edited by N. Wax, Dover Publications, Inc, New York, 1954, pp. 180 - 181.

4. Housner, G.W., and Jennings, P.C., "Generation of Artificial Earthquake," Journal of Engineering Mechanics Division, ASCE, Vol. 90, 1964, pp. 113-150.

5. Amin, M., and Ang, A.H.S., "Nonstationary Stochastic Model of Earthquake Motions," Journal of Engineering Mechanics Division, ASCE, Vol. 94,1968, pp. 559-583.

6. Liu, S.C., "Dynamics of Correlated Pulse Trains," Journal of Engineering Mechanics Div., Vol. 96, Aug., 1970.

7. Borgman, L.E., "Ocean Wave Simulation for Engineering Design," Journal of Waterways and Harbor Div., ASCE, Vol. No. WW4, November, 1969, pp. 557 - 583.

8. Shinozuka, M., and Jan, C.-M., "digital Simulation of Random Processes and Its Applications," Journal of Sound and Vibration, Vol. 25, No.1, 1972, pp. 111 - 128.

9. Wittig, L.E., and Sinha, A.K., "Simulation of Multi-Correlated Random Processes Using the FFT Algorithm," Journal of Acoust. Society of America, Vol. 58, No. 3, September, 1975, pp. 630 - 634.

10. Bochner, S.,Lectures on Fourier Integrals, Annals of Mathematical Studies No. 42, Princeton University Press, Princeton, N.J., 1959, pp. 325 - 328.

11. Lanning, J. H. and Battin, N. H., Random Processes in Automatic Control, McGraw-Hill, New York, 1956, Sects. 2.12 and 2.13.

12. GantMacher, F. R., The Theory of Matrices, Vol. 1, Chelsea, New York, p. 37.

13. Cooley, J. W. and Tukey, J. W., "An Algotithm for the Machine Calculation of Complex Fourier Series," Mathematics of Computation, Vol. 19, April, 1965.

14. Cooley, J. W., Lewis, P. A. and Welch, P. D., "The Fast Fourier Transform Algorithm," Journal of Sound and Vibration, Vol. 12, July 1970, pp. 315 - 337.

15. Brigham, E. O., The Fast Fourier Transform, Prentice-Hall, New Jersey, 1974, Ch. 4.

16. Hudspeth, R. T. and Borgman, L.E., "Efficient FFT Simulation of Digital Time Sequences," Journal of the Engineering Mechanics Division, ASCE, April, 1979, pp. 223-235.

17. Gonzalez, R. C.and Wintz, P., Digital Image Processing, Addison-Wesley Publishing Co., Massachusetts, 1977.

18. Singleton, R.C., "A Method Computing the Fast Fourier Transform with Auxiliary Memory and Limited High-speed Storage," IEEE Trans. Audio and Electroacoustics, Vol. AU-15, June 1967 pp. 91 - 98.

19. Meirovitch, L., Analytical Methods in Vibrations, The Macmillan Co., London, 1967.

20. Bendat, J.S. and Piersol, A.G., Engineering Applications of Correlation and Spectral Analysis, Wiley-Interscience, New York, 1980.

21. Steiglitz, K., An Introduction to Discrete System, Wiley, New York, 1974.

22. Stearns, S.D., Digital Signal Analysis, Hayden Book Co., New Jersy, 1975.

23. Tajimi, H., "A Statistical Method of Determining the Maximum Response of Building During an Earthquake," Proceedings, 2nd World Conference on Earthquake Engineering, held at Tokyo, Japan, 1960.

24. Vanmarcke, E. H.,SIMQKE: A Program for Artificial Motion Generation, MIT Dept. of Civil Engineering, November 1976.

25. Pierson, W.J. and Moskowitz, L.A., "A Proposed Spectral Form for Fully Developed Wind Sea Based on the Similarity Theory of S.A. Kitaigorodskii," Journal of Geophysical Research, Vol. 69, No. 24, 1964, pp. 5181 - 5190.

26. Wen, Y.K. and Shinozuka, M., "Analysis of Floating Plate under Ocean Waves," Journal of Waterways, Harbors and Coastal Engineering Div., Proceedings of ASCE, Vol. 98, No. WW2, May 1972, pp. 177-190.

27. Shinozuka, M., Yun, C., and Vaicaitis, R.,"Dynamic Analysis of Fixed Offshore Structures Subjected to Wind Generated Waves," Journal of Structural Mechnics, Vol. 5, No. 2, 1977, pp. 135-146.

28. Malhotra, A.K. and Penzien, J.,"Nondeterministic Analysis of Offshore Structures," Engineering Mechnics Div., ASCE, December 1970, pp. 985-1003.

29. Davenport, A.G.,"The Response of Slender Line-like Structures to a Gusty Wind," Proceedings of the Institute of Civil Engineering, Vol. 23, 1962, pp. 389-408.

30. Davenport, A.G., "The Spectrum of Horizontal Gustiness near the Ground in High Wind," Quarterly Journal of Royal Met. Soc., 87, 1961.

31. Shinozuka, M. and Levy, R.,"Digital Generation of Alongwind Velocity Field," Journal of the Engineering Mechanics Div., ASCE, August 1977.

32. Tsuboi, S., Tagimi, H., and Okano, S., Mathematics in Structural Engineering, Corona Book Co., Tokyo 1967.

33. Barlett, M.S., An Introduction to Stochastic Processes, Cambridge University Press, New York, N.Y. 1960, p. 160.

34. Gradshteyn, I.S. and Ryzhik, I.M., Tables of Integrals, Series and Products, Academic Press, New York, 1965, p. 711.

35. Lin, Y. K., Probabilistic Theory of Structural Dynamics, McGraw-Hill Book Company, New York, 1967.

APPENDIX

APPENDIX

COMPUTER PROGRAM

This Appendix deals with the listing of some of the important input parameters in the computer programs included. On the right -hand column is the computer language (BASIC) symbol and on the left -hand side, just opposite, is the equivalent variable used in the derivation of the equations in Chapter V.

Listing for Program I

(An Artificial Earthquake Accelerogram)

π	PI
2π	P2
W_u	WU, upper cutoff frequency
ΔW	DW, Increment of frequency
ω_n	W
ω_n^2	WC
$S(\omega)$	GW( ), the response power spectral density
$\sqrt{2S(\omega)\Delta\omega}$	A1

The statements 20 - 130 and 380 - 420 are for graphic outputs. The loop 220 - 330 computes simulation of Eq.(114). The loop 450 - 830 is the FFT computing.

```
LIST 1,400

10  HOME
20  HGR
30  HCOLOR= 3
40  HPLOT 18,9 TO 23,9
50  HPLOT 18,149 TO 23,149
60  HPLOT 23,9 TO 23,79
70  HPLOT 23,150 TO 23,79
80  HPLOT 18,44 TO 23,44
90  HPLOT 18,114 TO 23,114
100  HPLOT 151,74 TO 151,84
110  HPLOT 279,74 TO 279,84
120  HPLOT 279,79 TO 23,79
130 XØ = 23:YØ = 79
140  DIM FR(128),FI(128),SR(128),
     SI(128),GW(128)
150 N = 128
160 PI = 4 *  ATN (1)
170 P2 = PI * 2
180 WU = 30
190 DT = P2 / WU
200 T = N * DT
210 DW = 30 / N
220  FOR L = 1 TO N
230 W = (L - 1) * DW + 1E - 5
240 WC = W ^ 2
250 GW(W) = .01238 * (1 + WC / 14
     7.8) / ((1 - WC / 242) ^ 2 +
     WC / 147.8)
260 GW(W) = GW(W) * DW * 2
270 A1 =  SQR (GW(W))
280 R =  RND (L) * P2
290 S2 =  SIN (R)
300 C2 =  COS (R)
310 SR(L) = C2 * A1
320 SI(L) = S2 * A1
330  NEXT L
340 N2 = N / 2
350 N4 = N / 4
360 U =  - 1
370  GOSUB 450
380  FOR I = 1 TO N2
390 X = 4 * I + 23
400 Y = 79 - FI(I) * 35

]
```

```
LIST410,1000

410  HPLOT X0,Y0 TO X,Y
420 X0 = X:Y0 = Y
430  NEXT I
440  END
450  REM
460  FOR IT = 1 TO N
470 I = IT - 1
480 J = 0
490 M2 = 1
500 M1 = M2
510 M2 = M2 + M2
520 MI =  INT (I / M2)
530 MS = I - MI * M2
540  IF MS - M1 < 0 THEN 560
550 J = J + N / M2
560  IF M2 - N < 0 THEN 500
570 JF = J + 1
580 FR(IT) = SR(JF):FI(IT) = SI(J
    F)
590  NEXT IT
600  PRINT : PRINT
610 L = 1
620  IF L - N =  > 0 THEN 830
630 ST = 2 * L
640 EL = L
650  FOR M = 1 TO L
660 A = PI * (1 - M) / EL
670 WR =  COS (A)
680 WI =  SIN (A)
690  IF U < 0 THEN 710
700  GOTO 720
710 WI =  - WI
720  FOR I = M TO N STEP ST
730 J = I + L
740 TR = WR * FR(J) - WI * FI(J)
750 TI = WR * FI(J) + WI * FR(J)
760 FR(J) = FR(I) - TR
770 FI(J) = FI(I) - TI
780 FR(I) = FR(I) + TR
790 FI(I) = FI(I) + TI
800  NEXT I,M
810 L = ST
820  GOTO 620
830  RETURN

]
```

Listing for Program II

(Simulation of Wind Generated Waves)

π	PI
2π	P2
W_u	WU, Upper cutoff frequency
Δt	DT, Increment of time
T	T = N x DT
y	H = 10 ft
α	AL = 8.1E-3
σ	SG = 0.74
g	G = 32.2
Δw	DW, Increment of frequency
$S_{hh}(\omega)$	GW(ω), The one-sided wave spectrum

The statements 70 - 520 are implementation of Equation (116)and writing them on the file "DATA1". The loop 800 - 1400 is the FFT computing.

```
]LIST 1,300

1  REM  THIS PROGRAM SIMULATES WI
     ND GENERATED WAVES WHICH CAN
      BE USED FOR DESIGNING OFF-S
     HORE STRUCTURES.
10  REM    STATEMENTS FROM 10 TO
     380 ARE VARIOUS DATA INPUT A
     ND COMPUTING OF THE SQUARE O
     F THE SPECTRAL DENSITY.

20  DIM GW(1024)
30  HOME
40  HTAB 16: VTAB 7: INVERSE : PRINT
     "COMPUTING"
50  HTAB 12: VTAB 11: PRINT "PLEA
     SE BE PATIENT"
60  NORMAL
70 N = 1024
80 PI = 4 *  ATN (1)
90 P2 = PI * 2
100 WU = 24
110 DT = P2 / WU
120 T = N * DT
130 H = 10
140 N4 = N / 4: GOTO 180
150  HGR
160  HPLOT 0,0 TO 0,79
170  HPLOT 279,79 TO 0,79
180 AL = 8.1E - 3
190 SG = 0.74
200 G = 32.2
210 V = 75
220 WO = G / V
230 DW = 24 / N
240 D$ =  CHR$ (4)
250  PRINT D$;"OPEN DATA1"
260  PRINT D$;"DELETE DATA1"
270  PRINT D$;"OPEN DATA1,L16"
280  FOR L = 1 TO N
290 W = (L - 1) * DW + 1E - 5
300 GW(W) = (AL * G ^ 2) / (W ^ 5
     ) *  EXP ( - SG * (WO / W) ^
     4)


]
```

```
LIST 310,660

310  GOTO 350
320 Y = 79 - 50 * (GW(W))
330 X = 20 * W
340  HPLOT  TO X,Y
350 A2 =  - (W ^ 2 / G) * H
360 A3 = W *  EXP (A2)
362 STSTEMENT365ACCOUNTS FOR ACCE
    LERASTI ON .
365 A3 =  - W * A3
370 GW(W) = GW(W) * DW * 2
380 A1 =  SQR (GW(W))
390  REM  STATEMENTS FROM 380 TO
    590 COMPUTE COMPLEX NUMBERS
    AND STORE THEM IN RANDOM ACC
    ESS FILE DATA1
400 A1 = A1 * A3
410 R =  RND (L) * P2
420 S2 =  SIN (R)
430 C2 =  COS (R)
440 SR = C2 * A1
450 SI = S2 * A1
460 J2 = 2 * L
470 J1 = J2 - 1
480  PRINT D$;"WRITE DATA1,R";J1
490  PRINT SR
500  PRINT D$;"WRITE DATA1,R";J2
510  PRINT SI
520  NEXT L
530 U =  - 1
530  PRINT D$;"CLOSE DATA1"
550  REM  PRINTOUT
560  GOSUB 800
570  PRINT D$;"OPEN DATA,L16"
580  FOR I = 1 TO N
590 I2 = 2 * I
600 I1 = I2 - 1
610  PRINT D$;"READ DATA,R";I1
620  INPUT FR
630  PRINT D$;"READ DATA,R";I2
640  INPUT GW(I)
650  NEXT I
660  PR# 1


]

]
```

```
LIST 670,1000

670  PRINT  CHR$ (15) +  CHR$ (1)
     ;
680  PRINT  CHR$ (1) + "132N"
690  FOR J = 1 TO N4
700  FOR K = 1 TO 4
710 NN = N4 * (K - 1) + J
720  PRINT NN; SPC( 2);GW(NN); SPC(
     5);
730  NEXT K
740  PRINT
750  NEXT J
760  PR# 0
770  PRINT D$;"CLOSE DATA"
780  END
790  REM  STATEMENTS FROM 860 TO
     THE LAST COMPUTE  FFT
800  REM
810  FOR IT = 1 TO N
820 I = IT - 1
830 J = 0
840 M2 = 1
850 M1 = M2
860 M2 = M2 + M2
870 MI =  INT (I / M2)
880 MS = I - MI * M2
890  IF MS - M1 < 0 THEN 910
900 J = J + N / M2
910  IF M2 - N < 0 THEN 850
920 JF = J + 1
930 J2 = JF * 2
940 J1 = J2 - 1
950  PRINT D$;"OPEN DATA1,L16"
960  PRINT D$;"READ DATA1,R";J1
970  INPUT SR
980  PRINT D$;"READ DATA1,R";J2
990  INPUT SI
1000  PRINT D$;"CLOSE DATA1"

]
```

```
 LIST 1010,1400

1010 I2 = 2 * IT
1020 I1 = I2 - 1
1030  PRINT D$;"OPEN DATA ,L16"
1040  PRINT D$;"WRITE DATA ,R";I1

1050  PRINT SR
1060  PRINT D$;"WRITE DATA ,R";I2

1070  PRINT SI
1080  PRINT D$;"CLOSE DATA"
1090  NEXT IT
1110  PRINT D$;"OPEN DATA,L16"
1120 L = 1
1130  IF L - N =  > 0 THEN 1540
1140 ST = 2 * L
1150 EL = L
1160  FOR M = 1 TO L
1170 A = PI * (1 - M) / EL
1180 WR =  COS (A)
1190 WI =  SIN (A)
1200  IF U < 0 THEN 1220
1210  GOTO 1230
1220 WI =  - WI
1230  FOR I = M TO N STEP ST
1240 J = I + L
1250 J2 = 2 * J
1260 J1 = J2 - 1
1270  PRINT D$;"READ DATA,R";J1
1280  INPUT FR
1290  PRINT D$;"READ DATA,R";J2
1300  INPUT FI
1310 TR = WR * FR - WI * FI
1320 TI = WR * FI + WI * FR
1330 I2 = 2 * I
1340 I1 = 2 * I - 1
1350  PRINT D$;"READ DATA,R";I1
1360  INPUT SR
1370  PRINT D$;"READ DATA,R";I2
1380  INPUT SI
1390 R1 = SR - TR
1400 R2 = SR + TR


]

]
```

```
 LIST 1410,1550

1410 F1 = SI - TI
1420 F2 = SI + TI
1430  PRINT D$;"WRITE DATA,R";J1
1440  PRINT R1
1450  PRINT D$;"WRITE DATA,R";J2
1460  PRINT F1
1470  PRINT D$;"WRITE DATA,R";I1
1480  PRINT R2
1490  PRINT D$;"WRITE DATA,R";I2
1500  PRINT F2
1510  NEXT I,M
1520 L = ST
1530  GOTO 1130
1540  PRINT D$;"CLOSE DATA"
1550  RETURN

]
```

Listing for Program III

(Two Dimensional Wind Field Generation)

N	N, Number of sample points
Width	LO = 400 ft
L	LL, Correlation distance in space
$\Gamma(1/2)$	G1 = 1.77245
$\Gamma(1/3)$	G2 = 2.67893
$\Gamma(5/6)$	G3 = 1.12878
K_1	K1 = 0.4751
K_r	KR, Surface drag coefficient
$U(y^*)$	UC, Mean wind velocity at altitude y*
Q	Q, Quantity defined by Eq.(54)'
Δx	X1
W_u	$2\pi/\Delta x$

The statements 490 - 950 is the implementation of Equation (122). The FFT computation is done by the loop 40 - 400. Putting a subroutine in the beginning of program saves some of the searching times. The program in page 110 is a way of extracting a section of a large two dimensional array.

```
]LIST 1,400

30  GOTO 440
40 U =  - 1
50  REM
60  FOR IT = 1 TO N
70 I = IT - 1
80 J = 0
90 M2 = 1
100 M1 = M2
110 M2 = M2 + M2
120 MI =  INT (I / M2)
130 MS = I - MI * M2
140  IF MS - M1 < 0 THEN 160
150 J = J + N / M2
160  IF M2 - N < 0 THEN 100
170 JF = J + 1
180 FR(IT) = SR(JF):FI(IT) = SI(J
     F)
190  NEXT IT
200 QW = QW + 1
210 L = 1
220  IF L - N =  > 0 THEN 430
230 ST = 2 * L
240 EL = L
250  FOR M = 1 TO L
260 A = PI * (1 - M) / EL
270 WR =  COS (A)
280 WI =  SIN (A)
290  IF U < 0 THEN 310
300  GOTO 320
310 WI =  - WI
320  FOR I = M TO N STEP ST
330 J = I + L
340 TR = WR * FR(J) - WI * FI(J)
350 TI = WR * FI(J) + WI * FR(J)
360 FR(J) = FR(I) - TR
370 FI(J) = FI(I) - TI
380 FR(I) = FR(I) + TR
390 FI(I) = FI(I) + TI
400  NEXT I,M
```

```
]LIST 410,800

410 L = ST
420  GOTO 220
430  RETURN
440  HOME
450  HTAB 16: VTAB 7: INVERSE : PRINT
     "COMPUTING"
460  HTAB 12: VTAB 11: PRINT "PLE
     ASE BE PATIENT"
470  NORMAL
480  DIM FR(64),FI(64),SR(64),SI(
     64),TI(8,8)
490 N = 8
500 PI = 4 *  ATN (1)
510 P2 = 2 * PI
520 L0 = 400
530 LL = 282
540 G1 = 1.77245
550 G2 = 2.67893
560 G3 = 1.12878
570 G4 = G1 * G2 * .5 * G3
580 K1 = 0.4751
590 KR = .03
600 AL = 1 / 7
610 K2 = (1 / P2) * .4 ^ 2 * .5 ^
     (3 / 2) * G4
620 ZC = 200
630 Z1 = 30
640 U1 = 30
650 UC = U1 * (ZC / Z1) ^ AL
660 U2 = (6 * KR * U1 ^ 2) ^ 2
670 U3 = (AL * U1) ^ 3
680 LD = U2 ^ 1.5 * Z1 / (K2 * U3
     ) * (ZC / Z1) ^ (1 - 3 * AL)

690 BC = P2 * UC
700 C = K1 * LD * U2 / BC
710 B = LD / BC
720 Q = G2 * LL ^ 2 * C / (4 *  SQR
     (PI) * G3 * B)
730 X1 = L0 / N
740 Y1 = X1
750 WU = P2 / X1
760 DW = WU / N
770 QW = 0
780 D$ =  CHR$ (4)
790  PRINT D$;"OPEN DATA"
800  PRINT D$;"DELETE DATA"


]
```

```
]LIST 810,1200

810  PRINT D$;"OPEN DATA,L16"
820  FOR IA = 1 TO N
830  FOR L = 1 TO N
850 KM = IA * DW:KN = L * DW
860 KK = (KM * KM + KN * KN)
870 KK = (1 + KK * LL * LL) ^ 1.5

880 A1 = Q / KK
890 A1 =  SQR (8 * A1 * DW * DW)
900 R =  RND (IA + L) * P2
910 S2 =  SIN (R)
920 C2 =  COS (R)
930 SR(L) = C2 * A1
940 SI(L) = S2 * A1
950  NEXT L
960  GOSUB 50
970  FOR K = 1 TO N
980 Y1 = N * (IA - 1) + K
990 J2 = 2 * Y1
1000 J1 = J2 - 1
1010  PRINT D$;"WRITE DATA,R";J1
1020  PRINT FR(K)
1030  PRINT D$;"WRITE DATA,R";J2
1040  PRINT FI(K)
1050  NEXT K
1060  NEXT IA
1070  PRINT D$;"CLOSE DATA"
1080 N4 = N / 4
1090  PRINT D$;"OPEN DATA,L16"
1100  FOR IB = 1 TO N
1110  FOR K = 1 TO N
1120 Y1 = N * (K - 1) + IB
1130 J2 = Y1 * 2
1140 J1 = J2 - 1
1150  PRINT D$;"READ DATA,R";J1
1160  INPUT SR(K)
1170  PRINT D$;"READ DATA,R";J2
1180  INPUT SI(K)
1190  NEXT K
1200  GOSUB 50


]
```

```
]LIST 1210,1520

1210  FOR KB = 1 TO N
1220 Y2 = N * (KB - 1) + IB
1230 J2 = 2 * Y2
1240 J1 = J2 - 1
1250  PRINT D$;"WRITE DATA,R";J1
1260  PRINT FR(KB)
1270  PRINT D$;"WRITE DATA,R";J2
1280  PRINT FI(KB)
1285  NEXT KB
1290  NEXT IB
1300  PRINT D$;"CLOSE DATA"
1310  PRINT D$;"OPEN DATA,L16"
1320  FOR I = 1 TO N
1330  FOR K = 1 TO N
1340 Y1 = N * (I - 1) + K
1350 J2 = Y1 * 2
1360 J1 = J2 - 1
1370  PRINT D$;"READ DATA,R";J1
1380  INPUT TR
1390  PRINT D$;"READ DATA,R";J2
1400  INPUT TI(K,I)
1410  NEXT K
1420  NEXT I
1430  PRINT D$;"CLOSE DATA"
1440  GOTO 1524
1450  FOR I = 1 TO N
1460 ZX = X1 * (I - 1)
1470  IF ZX - Z1 <  = Ø THEN 1510

1480 UX = U1 * (ZX / Z1) ^ AL
1490 FR(I) = UX
1500  GOTO 1520
1510 FR(I) = U1
1520  NEXT I


]

]LOAD TTT
]
```

```
]LIST 1,500

10 D$ =  CHR$ (4)
20  PRINT D$;"MON C,I,O"
30  DIM FR(16,32)
40 N = 64
50 R = 0:S = 0
80  PRINT D$;"OPEN DATA,L16"
90  FOR I = 49 TO 64
94 R = 0
95 S = S + 1
96  FOR K = 1 TO 32
97 R = R + 1
100 Y1 = N * (I - 1) + K
110 J2 = 2 * Y1
120 J1 = J2 - 1
250  PRINT D$;"READ DATA,R";J1
260  INPUT TR
270  PRINT D$;"READ DATA,R";J2
280  INPUT FR(S,R)
290  NEXT K
300  NEXT I
310  PRINT D$;"CLOSE DATA"
320  PR# 1
330  PRINT  CHR$ (15) +  CHR$ (1)
     ;
340  PRINT  CHR$ (1) + "132N"
350  FOR J = 1 TO 16
360  FOR L = 1 TO 4
370  FOR K = 1 TO 8
380 M = 8 * (L - 1) + K
430  PRINT FR(J,M); SPC( 2)
435  NEXT K
437  PRINT
440  NEXT L
450  PRINT
460  NEXT J
470  PR# 0
480  END
```