2

NAVAL POSTGRADUATE SCHOOL
Monterey, California

AD-A257 496

DTIC
S ELECTE DEC 01 1992 D
A

THESIS

WAVELET-BASED MULTIRESOLUTION ANALYSES OF SIGNALS

by

Mark Russell Kalmbach

June 1992

Thesis Advisor: Alex W. Lam

Approved for public release; distribution is unlimited

92-30490

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 0704-0188
1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED		1b. RESTRICTIVE MARKINGS		
2a. SECURITY CLASSIFICATION AUTHORITY		3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited		
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE				
4. PERFORMING ORGANIZATION REPORT NUMBER(S)		5. MONITORING ORGANIZATION REPORT NUMBER(S)		
6a. NAME OF PERFORMING ORGANIZATION Naval Postgraduate School	6b. OFFICE SYMBOL *(If applicable)* EC	7a. NAME OF MONITORING ORGANIZATION Naval Postgraduate School		
6c. ADDRESS *(City, State, and ZIP Code)* Monterey, CA 93943-5000		7b. ADDRESS *(City, State, and ZIP Code)* Monterey, CA 93943-5000		
8a. NAME OF FUNDING/SPONSORING ORGANIZATION	8b. OFFICE SYMBOL *(If applicable)*	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER		
8c. ADDRESS *(City, State, and ZIP Code)*		10. SOURCE OF FUNDING NUMBERS		
		PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE *(Include Security Classification)*
WAVELET-BASED MULTIRESOLUTION ANALYSES OF SIGNALS

12. PERSONAL AUTHOR(S)
KALMBACH, Mark Russell

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT *(Year, Month, Day)*	15. PAGE COUNT
Master's Thesis	FROM ____ TO ____	June 1992	102

16 SUPPLEMENTARY NOTATION The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the US Government.

17. COSATI CODES			18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
FIELD	GROUP	SUB-GROUP	wavelet; Daubechies; Haas; multiresolution analysis of signals; Discrete Wavelet Transform

19 ABSTRACT *(Continue on reverse if necessary and identify by block number)*
Signal analysts have traditionally relied on the Discrete Fourier Transform and various data windowing schemes for signal detection and classification. Some signals, notably those of a transient nature, are inherently difficult to analyze with these traditional tools. The Discrete Wavelet Transform has recently generated considerable interest in several areas of digital signal processing and a determination of its suitability as a signal analysis tool is necessary. Associated with wavelet theory is the concept of multiresolutional analyses which allow examination of a signal at different scales.

This thesis investigates dyadic discrete wavelet decompositions of signals. A new multiphase wavelet transform is proposed and investigated. The multiphase transform technique is shown to be useful in transient signal analysis. Several MATLAB™ programs that perform multiresolutional analyses with various supporting features are provided.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION	
[✓ UNCLASSIFIED/UNLIMITED] ☐ SAME AS RPT ☐ DTIC USERS	UNCLASSIFIED	
22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE *(Include Area Code)*	22c. OFFICE SYMBOL
LAM, Alex W.	408-646-3044	EC/La

DD Form 1473, JUN 86 *Previous editions are obsolete.* SECURITY CLASSIFICATION OF THIS PAGE
S/N 0102-LF-014-6603 UNCLASSIFIED

Approved for public release; distribution is unlimited

Wavelet-Based Multiresolution Analyses of Signals

by

Mark Russell Kalmbach
Captain, United States Marine Corps
BS, United States Naval Academy, 1983

Submitted in partial fulfillment of the
required of degree for

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
June 1992

Author: Mark Russell Kalmbach

Mark Russell Kalmbach

Approved by: [illegible]

Alex W. Lam, Thesis Advisor

[illegible]

Roberto Cristi, Second Reader

M.A. Morgan

Michael A. Morgan, Chairman
Department of Electrical and Computer Engineering

ABSTRACT

Signal analysts have traditionally relied on the Discrete Fourier Transform and various data windowing schemes for signal detection and classification. Some signals, notably those of a transient nature, are inherently difficult to analyze with these traditional tools. The Discrete Wavelet Transform has recently generated considerable interest in several areas of digital signal processing and a determination of its suitability as a signal analysis tool is necessary. Associated with wavelet theory is the concept of multiresolutional analyses which allow examination of a signal at different scales.

This thesis investigates dyadic discrete wavelet decompositions of signals. A new multiphase wavelet transform is proposed and investigated. The multiphase transform technique is shown to be useful in transient signal analysis. Several MATLAB^{TM} programs that perform multiresolutional analyses with various supporting features are provided.

DTIC QUALITY INSPECTED 2

Accesion For	
NTIS CRA&I	✓
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution /	
Availability Codes	
Dist	Avail and / or Special
A-1	

TABLE OF CONTENTS

I. INTRODUCTION

The Discrete Fourier Transform has traditionally been the dominant tool in digital signal processing for extracting waveform features required in the analyses of signals. Characteristic of the Fourier transform is the global manner in which it operates on the input signal, thereby rendering this technique incapable of providing time localization of frequency components. The analyses of signals that are transitory in nature will particularly suffer the deleterious effects of this limitation and the utility of the discrete Fourier transform for transient signal analyses may therefore be diminished. Various data windows have been used in the past in an attempt to improve performance, but all involve tradeoffs in resolution and sidelobe levels that may unacceptably affect the analysis. [Ref. 1:pp. 63-82]

Recent appearance in the literature of several articles describing wavelet transforms and the related multiresolution analyses have created interest in determining their suitability for signal analysis applications; specifically, to overcome the time localization limitation of the Fourier transform. This thesis examines the potential of the wavelet transform for signal analysis purposes.

II. REVIEW OF WAVELET THEORY

A. DISCRETE WAVELET TRANSFORM

Wavelets are families of functions,

$$h_{a,b}(x) = |a|^{-1/2} h\left(\frac{x-b}{a}\right) \quad \begin{matrix} a,b \in \Re \\ a \neq 0 \end{matrix} \qquad (2.1)$$

generated by the dilations and translations of a single function. For digital signal processing applications it is necessary to discretize the parameter values and fix the initial dilation step a_0 and the translation step b_0 such that

$$a_0 > 1,\ b_0 \neq 0$$

Equation 2.1 becomes,

$$h_{m,n}(x) = a_0^{m/2} h(a_0^m x - nb_0) \quad m,n \in Z \qquad (2.2)$$

with $a = a_0^{-m}$ and $b = nb_0 a_0^{-m}$

The translation parameter is therefore dependent on the dilation parameter. A large, positive m will contract the function $h_{m,0}$ and cause the translation step to shorten while a large negative m dilates the function $h_{m,0}$ and lengthens the translation step size. The inverse relationship ensures coverage of the entire range. [Ref. 2: pp. 909-910]

Associated with the discrete wavelet is the discrete wavelet transform, T, which maps the function f to a sequence indexed by Z^2,

$$(Tf)_{mn} = \langle h_{mn}, f \rangle = a_0^{m/2} \int \overline{h(a_0^m x - nb_0)}\, f(x)\, dx \qquad (2.3)$$

If

$$\int |\xi|^{-1} |\mathscr{F}[h(\xi)]|^2 d\xi < \infty, \qquad (2.4)$$

and h has sufficient decay, and if T has a bounded inverse on its range, the set $\langle h_{mn} \rangle$ forms a frame for all square-integrable, one-dimensional functions f,

$$f(x) \in L^2(\Re) \qquad (2.5)$$

The significance of a frame is that numerically stable algorithms exist which allow f to be reconstructed from the wavelet coefficients $\langle h_{mn}, f \rangle$. [Ref. 2: p. 911]

Selection of appropriate h, a_0, and b_0 is influenced by various factors. The desire to take advantage of previously published work and to minimize redundancy in the wavelet representation resulted in choices of $a_0 = 2$, $b_0 = 1$, and several h (discussed below) such that the h_{mn} constitute an orthonormal basis of compact support.

B. MULTIRESOLUTION ANALYSIS

Multiresolution analysis, as the name implies, describes the L^2 function f as a series of approximations of the function at different levels of resolution and consists of a family of embedded closed subspaces

$$V_m \subset L^2(\Re): \quad \ldots \subset V_{-2} \subset V_{-1} \subset V_0 \subset V_1 \subset V_2 \ldots \qquad (2.6)$$

such that

$$\bigcap V_m = \{0\}, \quad \overline{\bigcup V_m} = L^2(\Re) \qquad (2.7)$$

and

$$f(\cdot) \in V_m \leftrightarrow f(2\cdot) \in V_{m+1} \qquad (2.8)$$

Also, there is a scaling function $\phi(x) \in V_0$ such that the

ϕ_{mn} (where $\phi_{mn}(x) = 2^{m/2}\phi(2^m x - n)$) constitute a basis for V_m

$$V_m = \overline{span(\phi_{mn})} \qquad (2.9)$$

Let $P_m f$ be the orthogonal projection of f onto V_m. From the preceding, it can be seen that $\lim_{m\to\infty} P_m f = f$, for all $f \in L^2(R)$.

[Ref. 3:pp. 967-968]

Considering our parameter choices and the decision to use orthonormal bases, if we define

$$c_{mn}(f) = \langle \phi_{mn}, f \rangle \tag{2.10}$$

then

$$P_m f = \sum c_{mn}(f)\phi_{mn} \in V_m \tag{2.11}$$

Now define $Q_m f = P_{m+1} f - P_m f$ and W_m as the orthogonal complement of V_m ($W_m \perp V_m$) so that $V_{m+1} = V_m \oplus W_m$. Then Q_m is the orthogonal projection of any $f \in L^2(\mathbb{R})$ onto W_m, the orthogonal complement of V_m in V_{m+1} . The W_m are scaled versions of W_0 ,

$$f(\cdot) \in W_m \leftrightarrow f(2^m \cdot) \in W_0 \tag{2.12}$$

and the W_m are orthogonal spaces which sum to L^2(R)

$$L^2(\mathbb{R}) = \oplus W_m \tag{2.13}$$

Similar to the V_m, there exists in W_m a wavelet function ψ such that

$$W_m = \overline{span[\psi_{mn}]} \tag{2.14}$$

where $\psi_{mn}(x) = 2^{m/2}\psi(2^m x-n)$. If we define

$$d_{mn}(f) = \langle\psi_{mn}, f\rangle \tag{2.15}$$

then

$$Q_m f = \sum d_{mn}(f)\,\psi_{mn} \in W_m \tag{2.16}$$

Naturally, the function may be reconstructed from its projections onto the bases vector spaces. [Ref. 2:pp. 916-918]

1. Properties of the Wavelet-Scaling Function Pair

Following is a brief summary of the relationship between the ϕ and ψ families. Since $\phi_0(x)\in V_0\subset V_1$, it can be written as

$$\phi_0(x) = \sum_{n=-\infty}^{\infty} \langle\phi_0(x), \phi_1(x-\frac{n}{2})\rangle\, \phi_1(x-\frac{n}{2}) \tag{2.17}$$

Let

$$h(n) = \int_{-\infty}^{\infty} \phi_0(x)\,\phi_0(2x-n)\,dx$$
$$= 2^{-\frac{1}{2}}\langle \phi_0(x), \phi_1(x-\tfrac{n}{2})\rangle \qquad (2.18)$$

so that

$$\phi_0(x) = \sqrt{2}\sum_{n=-\infty}^{\infty} h(n)\,\phi_1(x-\tfrac{n}{2})$$
$$= 2\sum_{n=-\infty}^{\infty} h(n)\,\phi_0(2x-n) \qquad (2.19)$$

The Fourier transform of this equation is

$$\Phi(2\omega) = H(\omega)\,\Phi(\omega) \qquad (2.20)$$

where we have defined

$$H(\omega) = \sum_{n=-\infty}^{\infty} h(n)\,e^{-jn\omega} \qquad (2.21)$$

To satisfy the above relationships, the following properties must hold:

$$|H(0)| = 1$$
$$|H(\omega)|^2 + |H(\omega+\pi)|^2 = 1 \qquad (2.22)$$

From Equation 2.20 it can be seen that the ϕ can thus be derived if $H(\omega)$ is known since

$$\Phi(\omega) = \prod_{p=1}^{\infty} H(2^{-p}\omega) \tag{2.23}$$

Knowledge of ϕ can subsequently be used to find ψ . Since $\psi_0(x) \in W_0 \subset V_1$, it can be written as

$$\psi_0(x) = \sum_{n=-\infty}^{\infty} \langle \psi_0(x), \phi_1(x-\frac{n}{2}) \rangle \, \phi_1(x-\frac{n}{2}) \tag{2.24}$$

Let

$$\begin{aligned} g(n) &= \int_{-\infty}^{\infty} \psi_0(x)\phi_0(2x-n) \\ &= 2^{-\frac{1}{2}} \langle \psi_0(x), \phi_1(x-\frac{n}{2}) \rangle \end{aligned} \tag{2.25}$$

and by a similar approach to that used previously we can find

$$\begin{aligned} \psi_0(x) &= \sqrt{2} \sum_{n=-\infty}^{\infty} g(n)\phi_1(x-\frac{n}{2}) \\ &= 2 \sum_{n=-\infty}^{\infty} g(n)\phi_0(2x-n) \end{aligned} \tag{2.26}$$

The Fourier transform of Equation 2.26 is

$$\Psi(2\omega) = G(\omega)\Phi(\omega) \tag{2.27}$$

The following properties result:

$$\begin{gathered} |G(0)| = 0 \\ |G(\omega)|^2 + |G(\omega+\pi)|^2 = 1 \\ H(\omega)\overline{G(\omega)} + H(\omega+\pi)\overline{G(\omega+\pi)} = 0 \end{gathered} \tag{2.28}$$

Equations 2.22 and 2.28 are recognized as characteristic of "conjugate" filters and are necessary to ensure orthogonality among the ϕ and ψ families of functions. Equation 2.28 ensures that each function of the ϕ (commonly referred to as the scaling function) is orthogonal to each function of the ψ family. This last condition results from the fact that $V_m \perp W_m$. [Ref. 4: pp. 16-23]

An example of a function that fulfills the stated conditions is

$$G(\omega) = e^{-j\omega}\overline{H(\omega+\pi)} \tag{2.29}$$

from which we can find

$$g(n) = (-1)^{1-n}h(1-n) \tag{2.30}$$

Equation 2.30 is the defining relationship for describing wavelet-scaling function dependency in the application algorithms. [Ref. 3: p. 679]

2. Decomposition and Reconstruction Algorithms

Recalling that $c_m(n) = \langle\phi_{mn}, f\rangle$, we can derive the decomposition recursion algorithm:

$$\begin{aligned} c_{m-1}(n) &= \langle\phi_{(m-1)n}, f\rangle \\ &= \langle\phi_{m-1}(x-2^{-(m-1)}n), f\rangle \\ &= \int\phi_{m-1}(x-2^{-(m-1)}n)\, f(x)\, dx \\ &= \int[\sqrt{2}\sum_k h(k)\,\phi_m(x-2^{-(m-1)}n-2^{-m}k)]\, f(x)\, dx \\ &= \sqrt{2}\sum_k h(k)\int\phi_m(x-2^{-m}(2n+k))\, f(x)\, dx \\ &= \sqrt{2}\sum_k h(k)\, c_m(2n+k) \end{aligned} \tag{2.31}$$

Likewise we find

$$d_{m-1}(n) = \sqrt{2}\sum_k g(k)\, c_m(2n+k) \tag{2.32}$$

To determine the reconstruction recursion algorithm recall

$$P_m f = P_{m-1}f + Q_{m-1}f \tag{2.33}$$

$$P_{m-1}f + Q_{m-1}f = \sum_{k=-\infty}^{\infty} c_{m-1}(k)\,\phi_{(m-1)k} + \sum_{k=-\infty}^{\infty} d_{m-1}(k)\,\psi_{(m-1)k} \tag{2.34}$$

and substitute terms to get

$$\begin{aligned} c_m &= \langle \phi_{mn}, P_m f \rangle \\ &= \sum_k c_{m-1}(k)\langle \phi_{mn}, \phi_{(m-1)k}\rangle + \sum_k d_{m-1}(k)\langle \phi_{mn}, \psi_{(m-1)k}\rangle \end{aligned} \tag{2.35}$$

Using a change of variables the following can be found

$$\langle \phi_{mn}, \phi_{(m-1)k}\rangle = \sqrt{2}h(n-2k) \tag{2.36}$$

$$\langle \phi_{mn}, \psi_{(m-1)k}\rangle = \sqrt{2}g(n-2k) \tag{2.37}$$

so that

$$c_m(n) = \sqrt{2}\,[\sum_k c_{m-1}(k)h(n-2k) + \sum_k d_{m-1}(k)g(n-2k)] \tag{2.38}$$

The equations of this section demonstrate the pyramidal structure of multiresolution analysis and readily lend themselves to digital signal processing applications. [Ref 4: pp.25-28]

The important conclusion is that, assuming knowledge of the *g* and *h* vectors, the *c* and *d* coefficients at any level can be completely determined from the *c* coefficients at the next higher level; also, the *c* coefficients at any level can be determined from the c and d coefficients at the adjacent lower level. Notice that calculations within this pyramidal algorithm do not require explicit use of the ϕ and ψ functions since interlevel relationships are defined entirely by the g and h vectors. Obviously, the same results would be reached if dilations and translations of the orthogonal basis functions were used directly at each level in the decomposition and reconstruction, but would be found at great computational cost because of the requirement to calculate inner products.

III. COMPUTER IMPLEMENTATION OF MULTIRESOLUTION ANALYSIS

A. INTRODUCTION

Our goal is to investigate the use of multiresolution analyses to extract information from an input signal. In this case, a one-dimensional data sequence will represent the input. To avoid having to numerically evaluate any inner products we will equate the $c_0(n)$ coefficients with the data sequence, thereby constructing an auxiliary function f, with $f=\Sigma c_0(n)\phi_0(n)$, which clearly resides in V_0. Since $c_0(n)$ entirely describes the input, it represents the highest resolution level possible and the apex of the pyramidal algorithm. The multiresolution framework previously described can now be used to decompose f (thus the data sequence $c_0(n)$) into lower resolution, i.e. $m < 0$, approximation coefficients $c_m(n)$, and detail coefficients $d_m(n)$. The reconstruction recursive equations can be used to recover the original f (and therefore the data sequence). A different multiresolution analysis exists for each scaling function/wavelet set. This thesis is limited to the Haar wavelet and to Daubechies' group of compactly supported wavelets. The h vectors for these wavelets were published in Reference 1 and are listed in Appendix H for convenience. Describing the h vectors is the common method of defining scaling function/wavelet sets because all other desired information can subsequently be derived from them.

Several computer programs were written to implement various multiresolution analyses. The programs were written in MATLAB^{TM} to take advantage of the transportability of m-files between various operating systems. The capabilities of these programs, listed in Appendices B-H, will be described in general. Specific information can be obtained by referring to program comments in the appropriate appendix.

The normal entry point into the collection of programs is the calling program wavelet.m (Appendix B). A brief description of the various possible options is presented upon entry and the user can make a decision based on the choices provided. Specifically, the user may decide to use the Haar wavelet, his own wavelet, or one from the Daubechies group of nine compactly supported wavelets and analyze with either a single phase or multiphase approach.

The single phase approach implies the decomposition start point is the data start point, i.e. a "snapshot" of the entire data sequence is processed. The term multiphase refers to starting a decomposition at every possible data sequence/wavelet phase relationship, i.e. a "sliding window" approach, to maximize the signal analysis capability. As desired, wavelets are phase sensitive but feature extraction may be hindered if the "best" input phase relationship is not used. The multiphase analysis therefore contains multiple sets of coefficients, with each set containing all the information necessary for reconstruction.

B. SINGLE PHASE MULTIRESOLUTION ANALYSIS

The second program, haarwave.m (Appendix C), does the "snapshot" analysis of an input data sequence using the Haar basis. Because of the simple nature of the Haar, it is easy to generate the actual approximation and detail function at each resolution level. The input data is viewed as a piecewise constant function (equating to the sample and hold operation) to allow easy inner product calculation. Representations of the properly weighted scaling function and wavelet at each level are provided to clearly indicate the dyadic relationship between levels. Additionally, the reconstruction of the original sequence can be viewed as well as a plot of the reconstruction error. Finally, distribution of the energy of each approximation and detail coefficient and the total energy of each resolution level are displayed. These energy distributions are the foundation for signal analysis.

The third program, daubwave.m (Appendix D), allows the user to pick one of the wavelets from Daubechies group or to enter a valid user-defined *h* vector to define the basis set. Basically, the same output plots that were generated for wvhaar.m can be seen. Because of the complexity and irregularity of the basis functions, however, only the value of the coefficients which are generated for specific points will be plotted instead of the inner product representation.

C. MULTIPHASE WAVELET ANALYSIS

The last analytical program, multiphs.m (Appendix E), allows the use of any of the previously defined wavelets with the multiphase algorithms. The different coefficient sets are displayed simultaneously, which is possible because their coefficient indices interleave without interference. Clearly, redundant information is provided in the sense that there are more coefficients than would be necessary for reconstruction of the data sequence over its interval of support. However, because of the different zero padding necessary for each set of coefficients a different $P_m f$ in V_m is defined for each set. The user will therefore be able to see the $P_m f$ with the most distinctive set of coefficients to ease analysis.

D. SUPPORTING PROGRAMS

The program basisplt.m (Appendix F) supports the daubwave.m and multiphs.m programs by generating iterative plots of the scaling function and wavelet bases. It can also be called up directly and will use the vector "hcoeff" as the h coefficients to determine the ϕ and ψ based on a graphical recursion method.

The functions wvinput.m (Appendix G) and daubdata.m (Appendix H) are called as necessary by any of the multiresolution analysis programs to provide selected input data signals or the h vectors from Daubechies' group, respectively. The input signal options consist of a sinusoid, a pseudo-noise (PN) sequence, a sinusoid modulated with a PN sequence, and any of the above corrupted by noise. Various parameters can be modified to satisfy the user's needs.

IV. SIGNAL ANALYSIS

All plots associated with this section can be found in Appendix A.

A. SINGLE PHASE ANALYSIS OF A SINUSOIDAL WAVEFORM

The first signal to be analyzed will be the sinusoid shown in Figure 1, along with its level 0 approximation. Figures 2-5 show the Haar multiresolution analysis, stopping at level -4 since there is no energy at any lower level for this decomposition. The reconstruction plots (not shown) are indistinguishable from the decomposition plots because the maximum reconstruction error is 15 orders of magnitude below signal levels. The energy contained at each resolution level for both approximation and detail coefficients can be seen in Figure 6. Clearly, the maximum energy change occurs in the detail coefficients at level -4 and the energy in the approximation coefficients at every level is the sum of the energy of the detail and approximation coefficients from the level below, as expected. Figures 7 and 8 provide a clear summary of exactly where the signal and filter best match with respect to sample number and resolution level. In this example, the Haar decomposition highlights the phase changes associated with the switching between positive and negative half-cycles because of the signal/wavelet phase coherence.

The decomposition is repeated with a different multiresolution analysis based on the 6-coefficient (third order) Daubechies wavelet. Only the coefficient energy plots are shown in Figures 9-11 for comparison with the Haar plots. It can be shown that the time index of coefficients goes beyond the original support length of the data (potentially out to sample 257 in this example, although the plots were truncated at sample 90 for display purposes and because very little energy is associated with samples beyond 90). Although the coefficients outside the signal support range are generally very small, they are necessary for completeness. Computing these coefficients requires affixing zeros to the sequence, most significantly at the trailing edge. These "edge effects" are minimized at the leading edge (no coefficients are generated for negative time indices) by shifting the *h* vector so that nonzero values may initially exist only over [-(m-2),...,0,1] and then assigning the coefficient value to the index of the second term from the right (initially 0). This is intuitively satisfying from a causality viewpoint and also aids interpretation by only having coefficients outside the data sequence on one side. Leading edge effects consist of the influence of leading zeros necessary for filter coefficients with negative time indices. Note that edge effects increase as the resolution level decreases because lower resolution coefficients are influenced by a larger number of data points. The extent of edge effects is also dictated by the number of *h* coefficients. Coefficients beyond the data sequence support will not have time indices greater than [$(2^{-m}-1)$(number of *h* coefficients - 1)] for any of the decomposition programs and often significantly fewer (e.g., if the data sequence length is a power of 2 the Haar decomposition will not generate any terms beyond the data length, as can be seen in the previous plots).

The next set of plots (Figs. 12-18) show the phase sensitivity of the decomposition when the input sinusoid phase is shifted from 0 to 45 degrees. The energy distribution is no longer as concentrated as in the previous set of plots and interpretation is consequently more difficult. Changing the sampling frequency relative to the sinusoid frequency will have a similar effect.

B. SINGLE PHASE ANALYSIS OF A BINARY PHASE SHIFT KEYED SIGNAL

The Binary Phase Shift Keyed signal shown in Figure 19 will serve to highlight a deficiency in the single phase approach. Notice with the Haar wavelet in Figures 20-22 that the decomposition is exactly the same as the pure sinusoid, i.e., because of the phase alignment we cannot discriminate between the two signals' coefficient energy plots. The Daubechies 6-coefficient wavelet decomposition (Figs. 23-25) does appear significantly different than the sinusoid decomposition but there is no clear indication of every phase reversal. Thus, the results are ambiguous.

C. MULTIPHASE ANALYSIS OF A BINARY PHASE SHIFT KEYED SIGNAL

Since these responses would be unsuitable for signal analysis purposes the multiphase approach (described previously) was developed. It can be viewed as starting decompositions at each of the 2^{-m} phases of each level so that the most distinctive representation can be used for analysis regardless of initial signal phase. The multiphase plots are shown in Figures 26 and 27 for the Haar and in Figures 28 and 29 for the Daubechies 6-coefficient filter. There is clear indication in both sets of plots of the signal's phase inversions.

D. MULTIPHASE ANALYSIS OF A TRANSIENT SIGNAL

Finally, the transient signal of Figure 30 was investigated. Use of the "zoom" feature in the programs was necessary to clearly see the response. Figures 31-34 show the responses associated with the Haar and Daubechies 6-coefficient wavelets that have been the standard throughout this thesis. Higher order Daubechies (more regular) wavelets were also experimented with in an attempt to best fit the signal and it can be seen that the 18-coefficient (ninth order) wavelet used for Figures 35 and 36 does a good job of clearly indicating the presence of the signal and of maximizing the concentration of energy into only a few points in a single resolution level. This good "match" of the signal with the wavelet filter could be used for signal classification purposes as well as detection if an a priori selection of the wavelet filter based on a similar known signal had occurred.

V. CONCLUSIONS

Wavelet-based multiresolution analysis is another tool for the signal analyst and great potential exists for its use in various applications. Signals that can only be represented in the frequency domain with large numbers of significant terms provide special motivation for wavelet-based decomposition, as can be seen in the transient signal example above. Signal detection and pattern recognition through the use of "matched" wavelets may also prove useful, especially in areas where analysis at different resolution levels (i.e., possible signal dilation/contraction) is required. The extent of the ultimate usefulness and popularity of wavelets will become known as others gain knowledge of basic wavelet characteristics and incorporate multiresolution analyses into their research.

APPENDIX A

SIGNAL ANALYSIS PLOTS

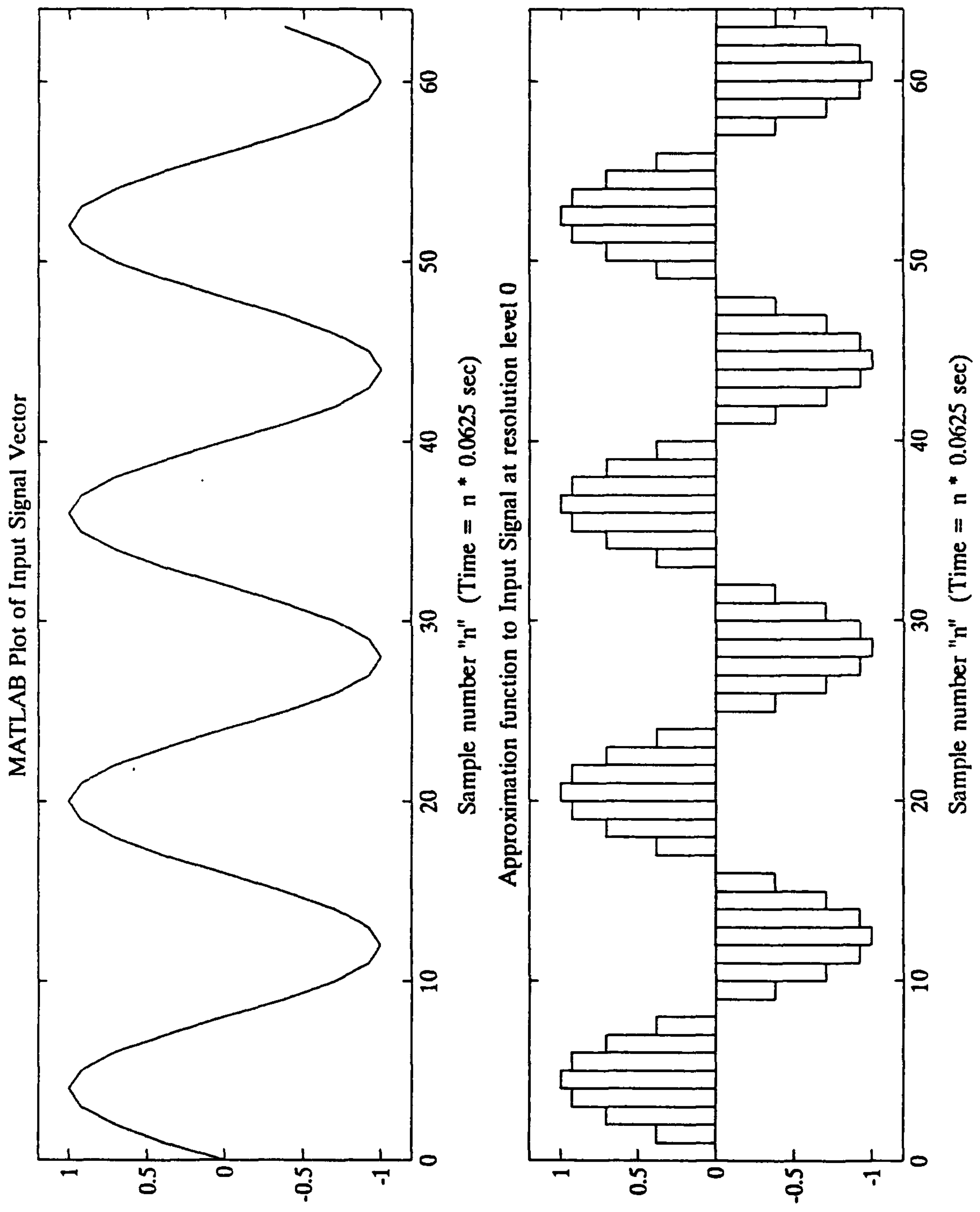

Figure 1. Input Sinusoid and Approximation

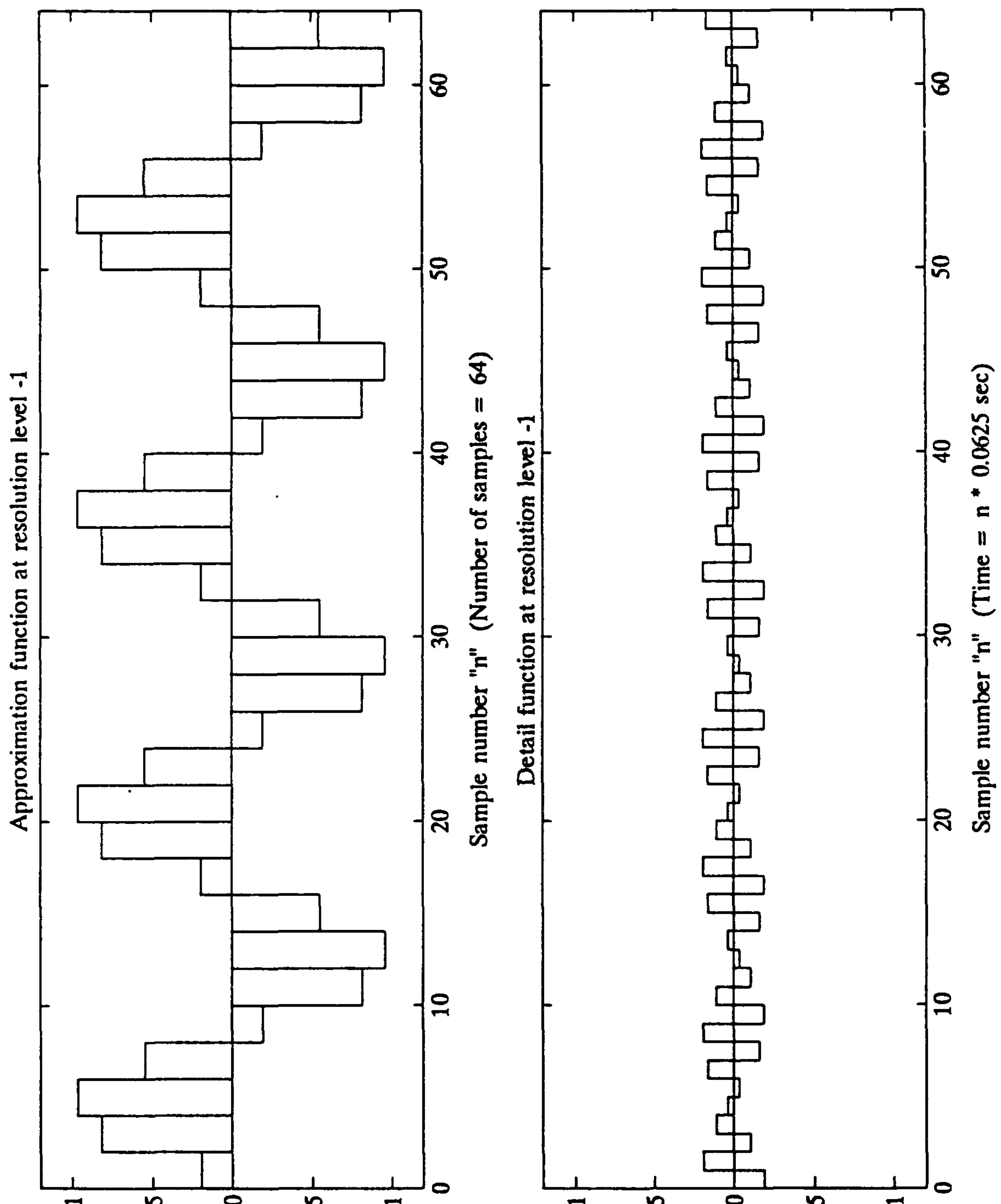

Figure 2. Haar Response at Resolution Level -1

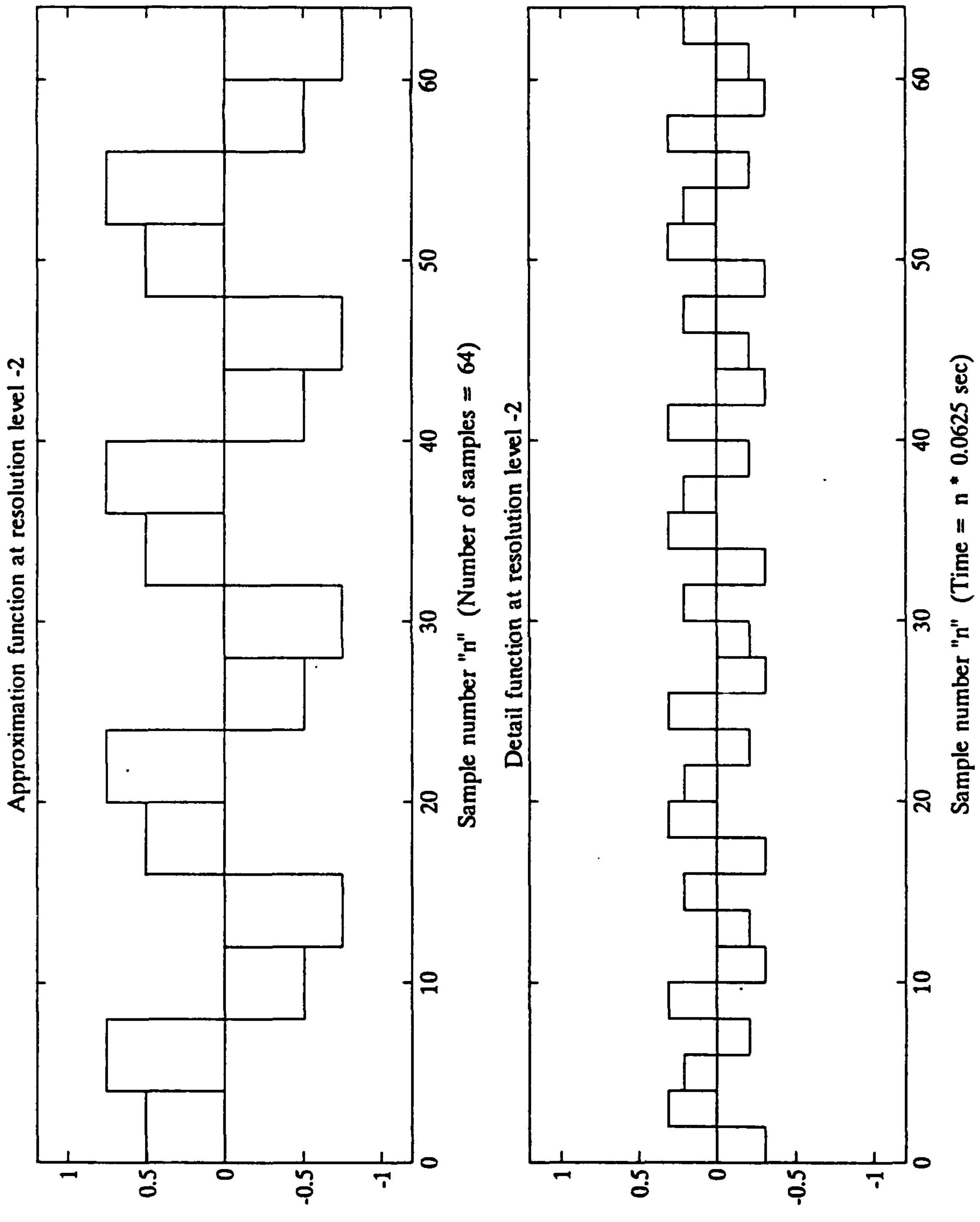

Figure 3. Haar Response at Resolution Level -2

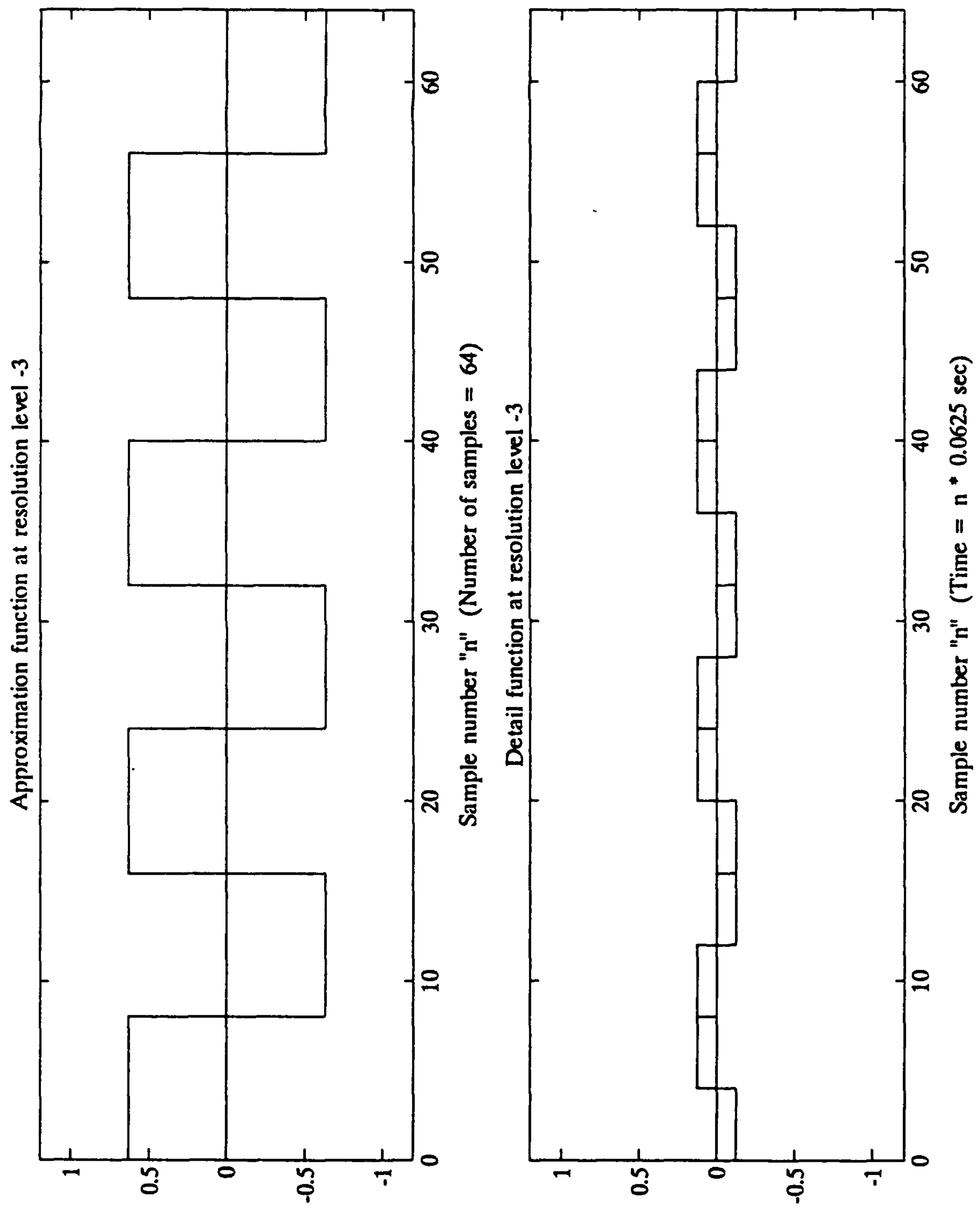

Figure 4. Haar Response at Resolution Level -3

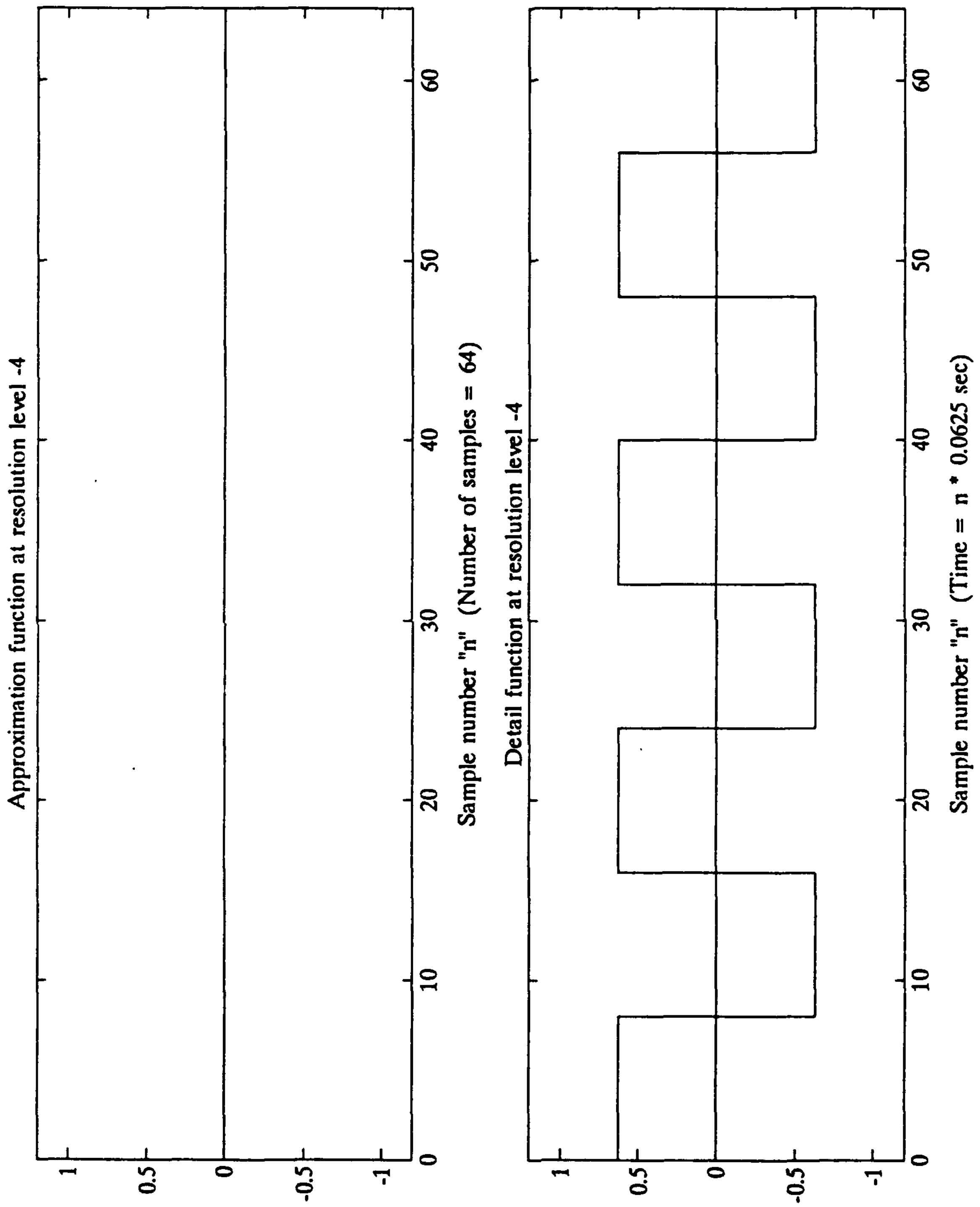

Figure 5. Haar Response at Resolution Level -4

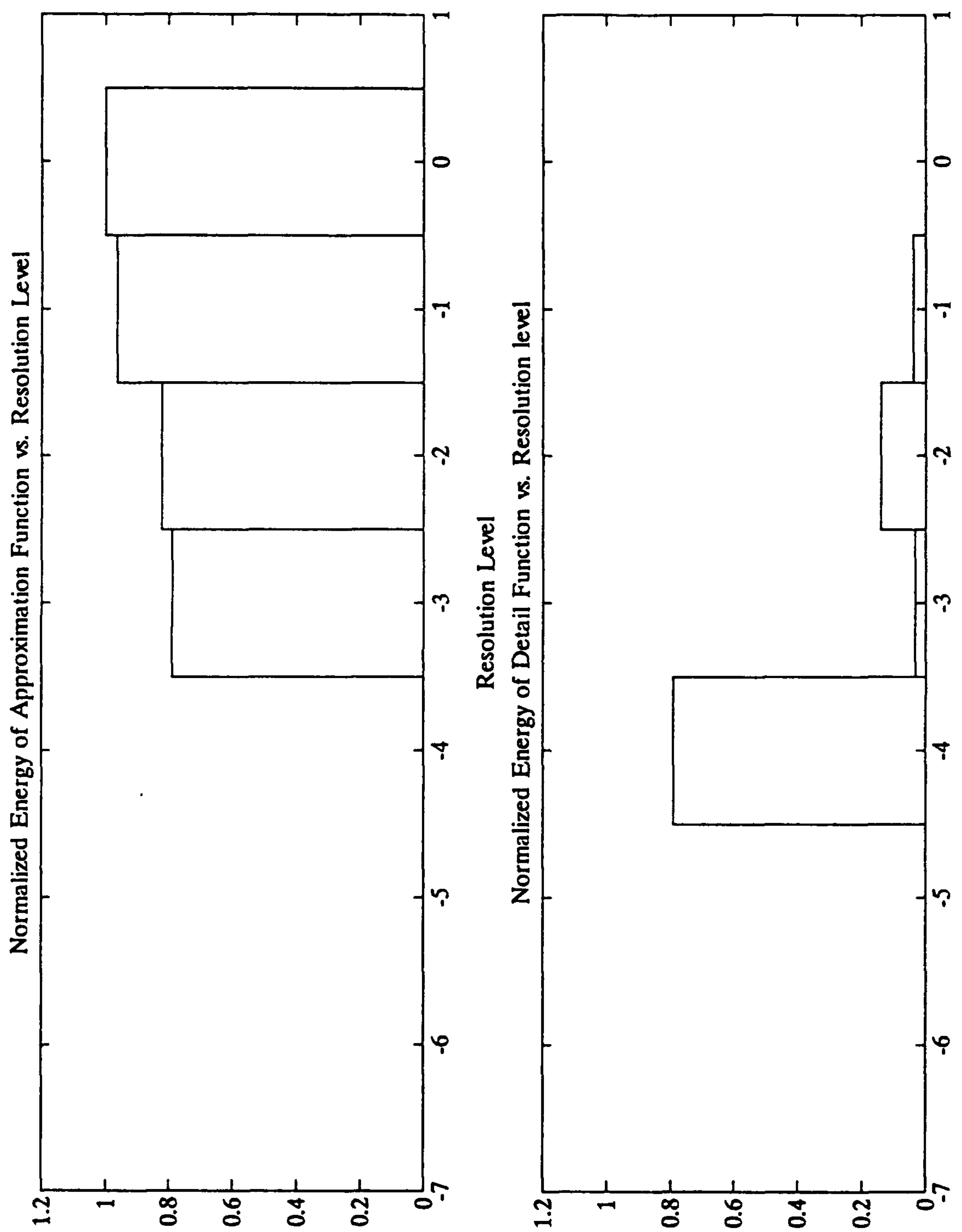

Figure 6. Haar Resolution Level Energy Distribution

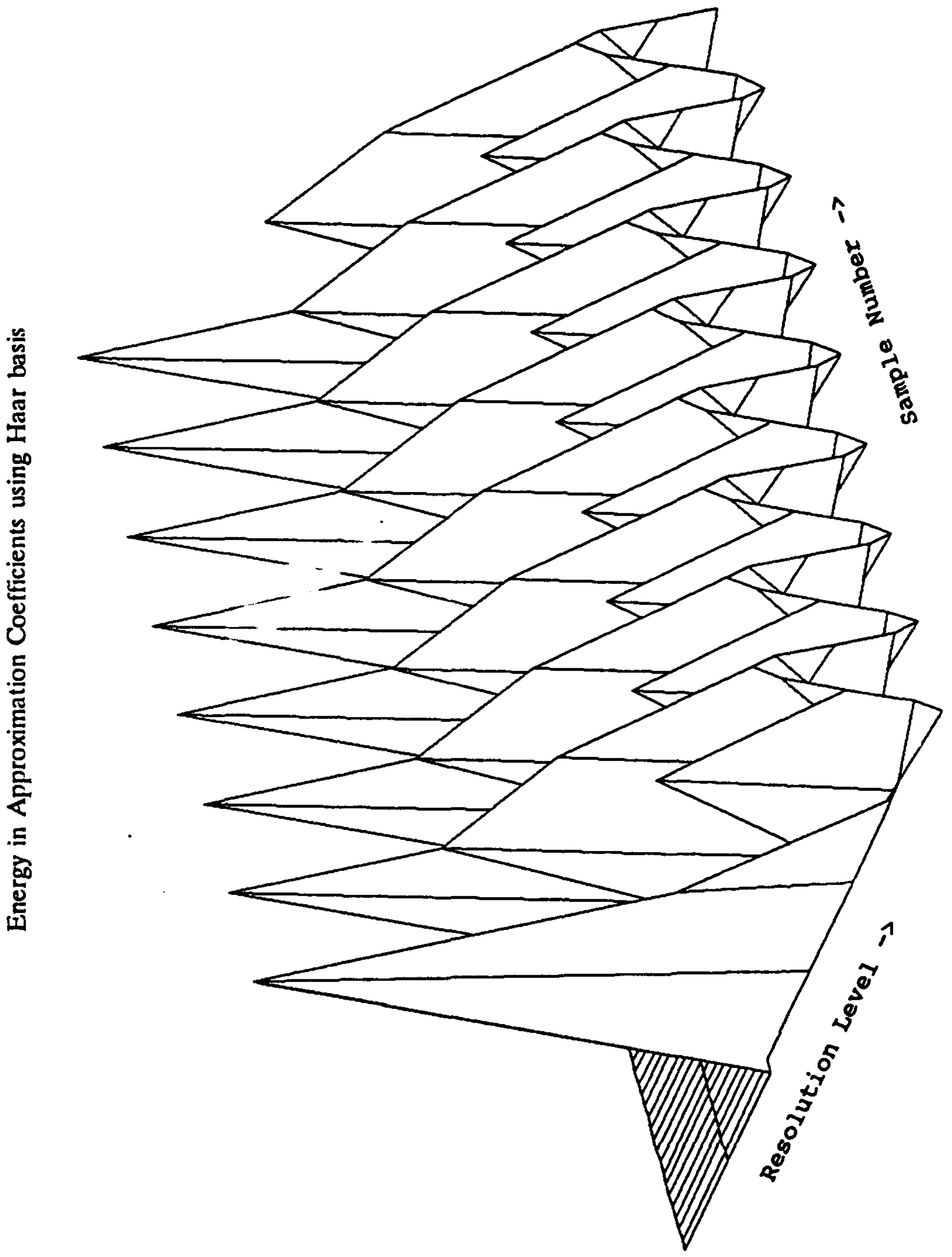

Figure 7. Haar Approximation Coefficient Distribution

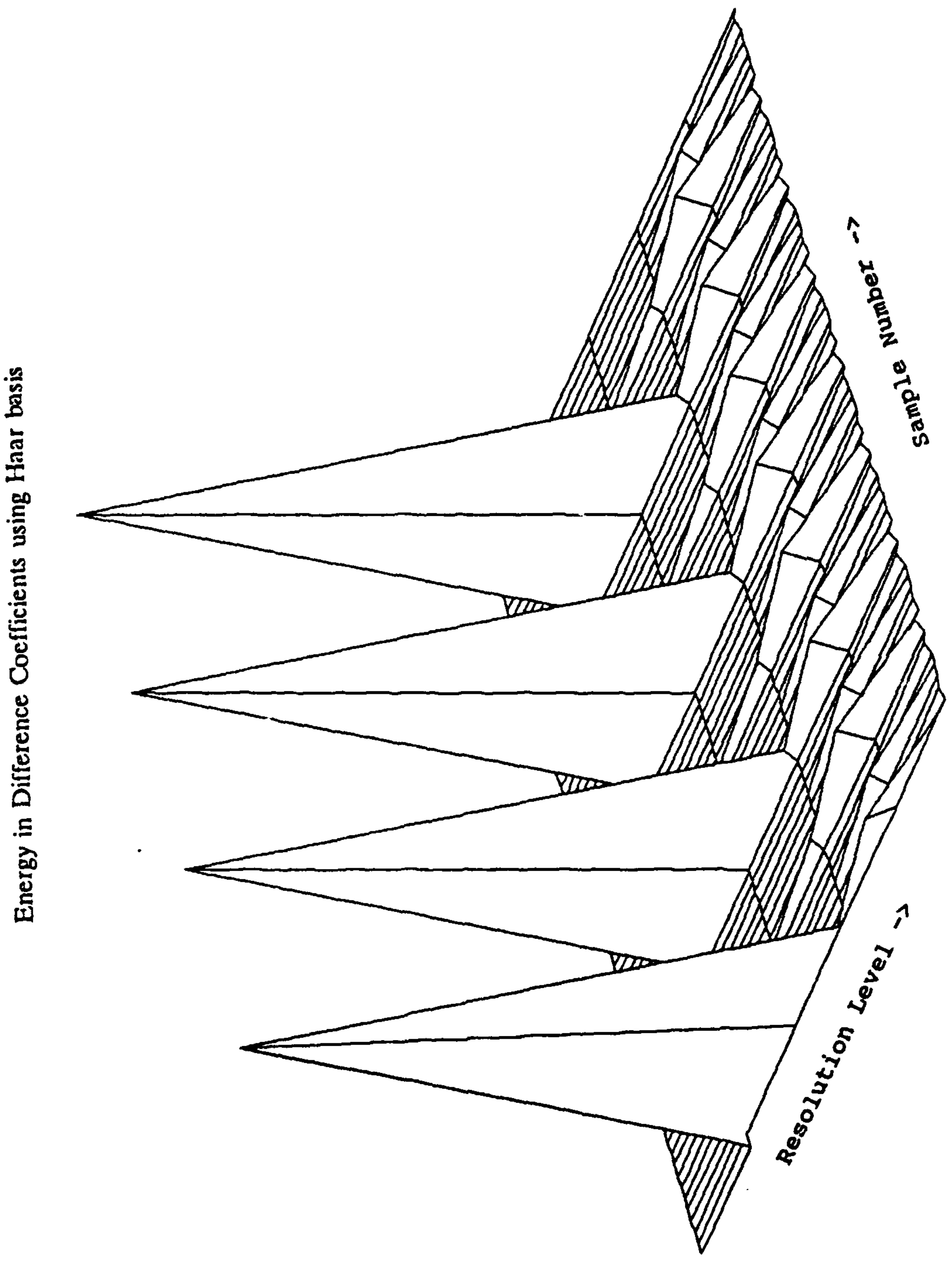

Figure 8. Haar Detail Coefficient Distribution

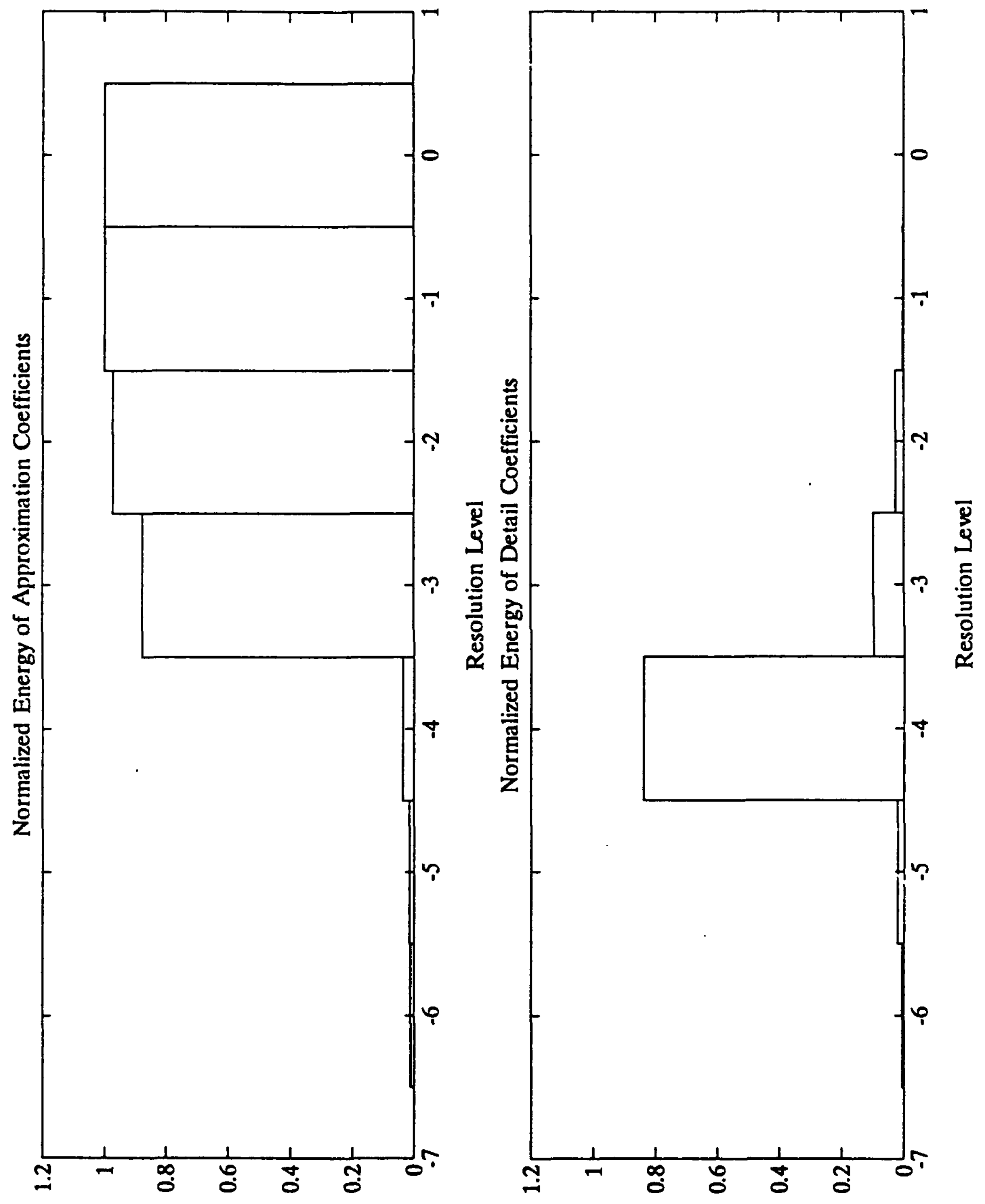

Figure 9. Daubechies Resolution Level Energy Distribution

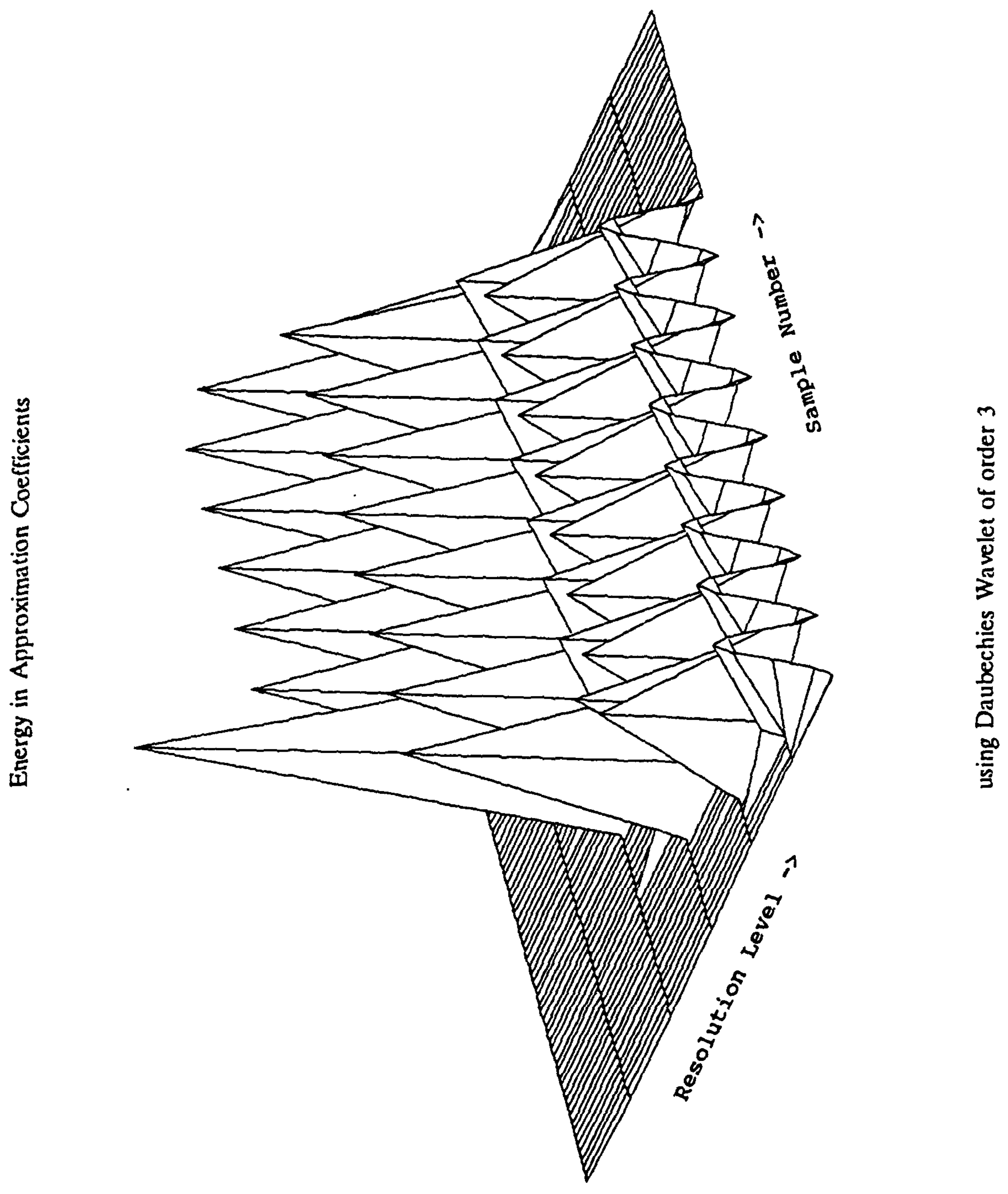

Figure 10. Daubechies Approximation Coefficient Distribution

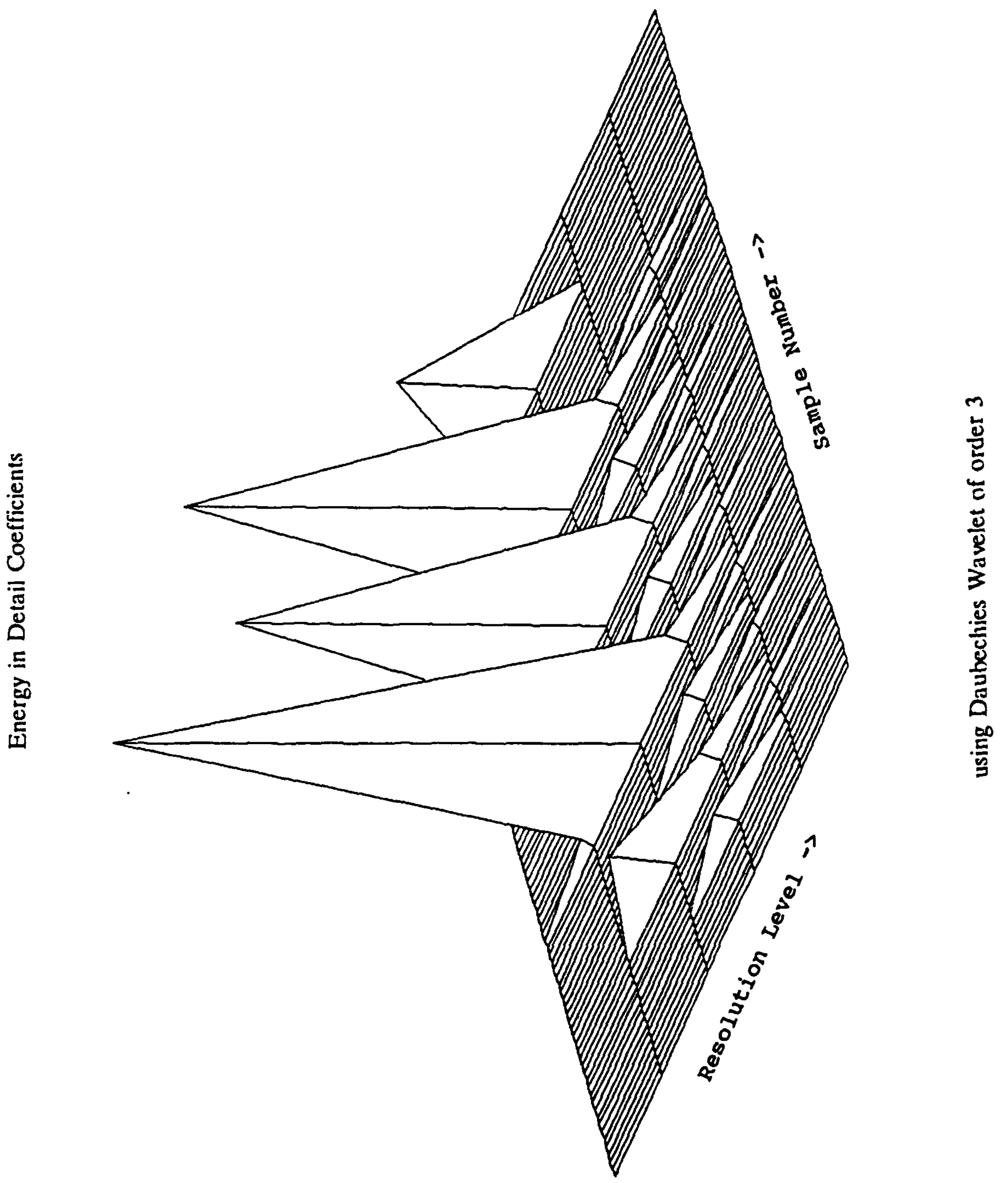

Figure 11. Daubechies Detail Coefficient Distribution

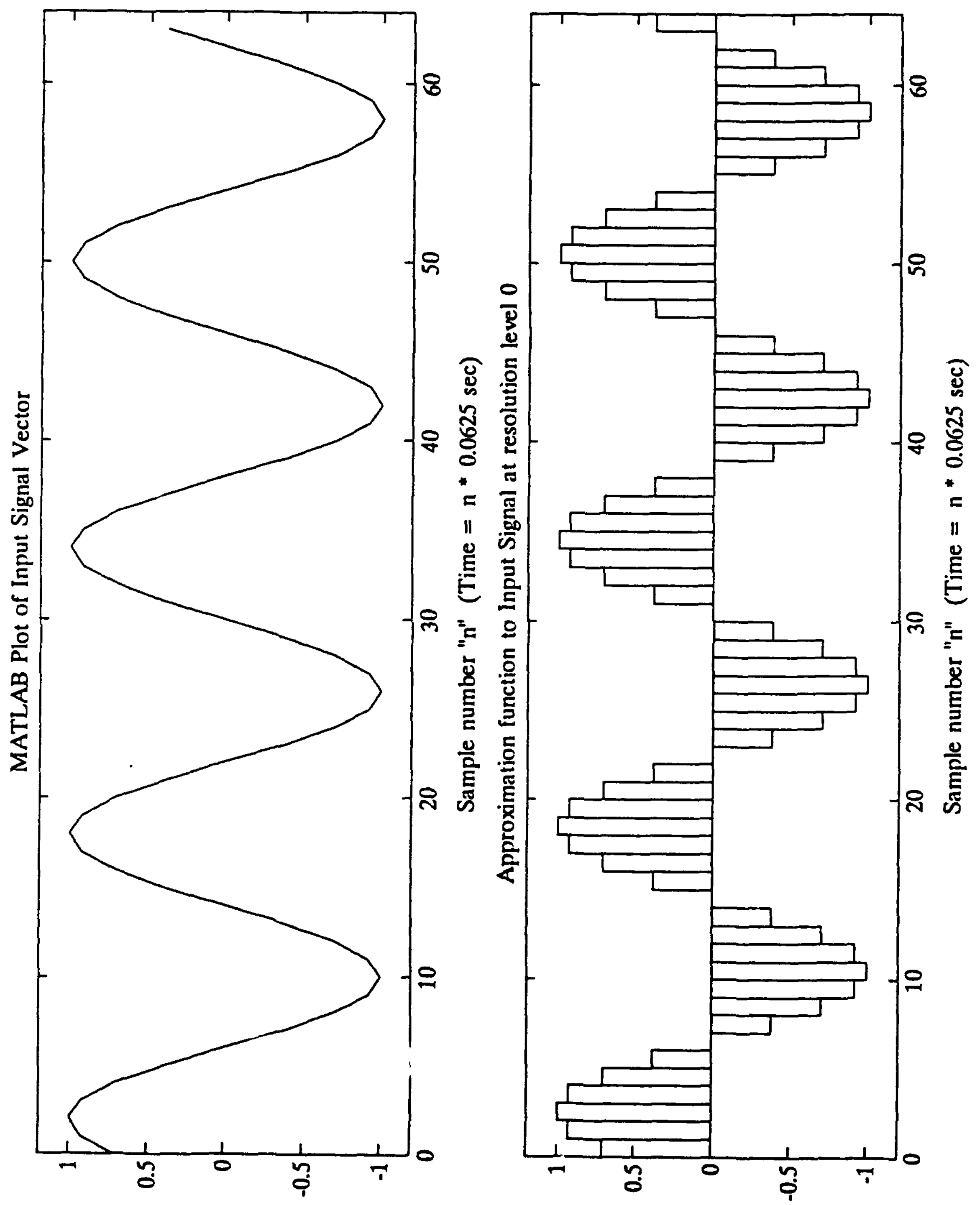

Figure 12. Input Sinusoid Shifted by 45^{o} and Approximation

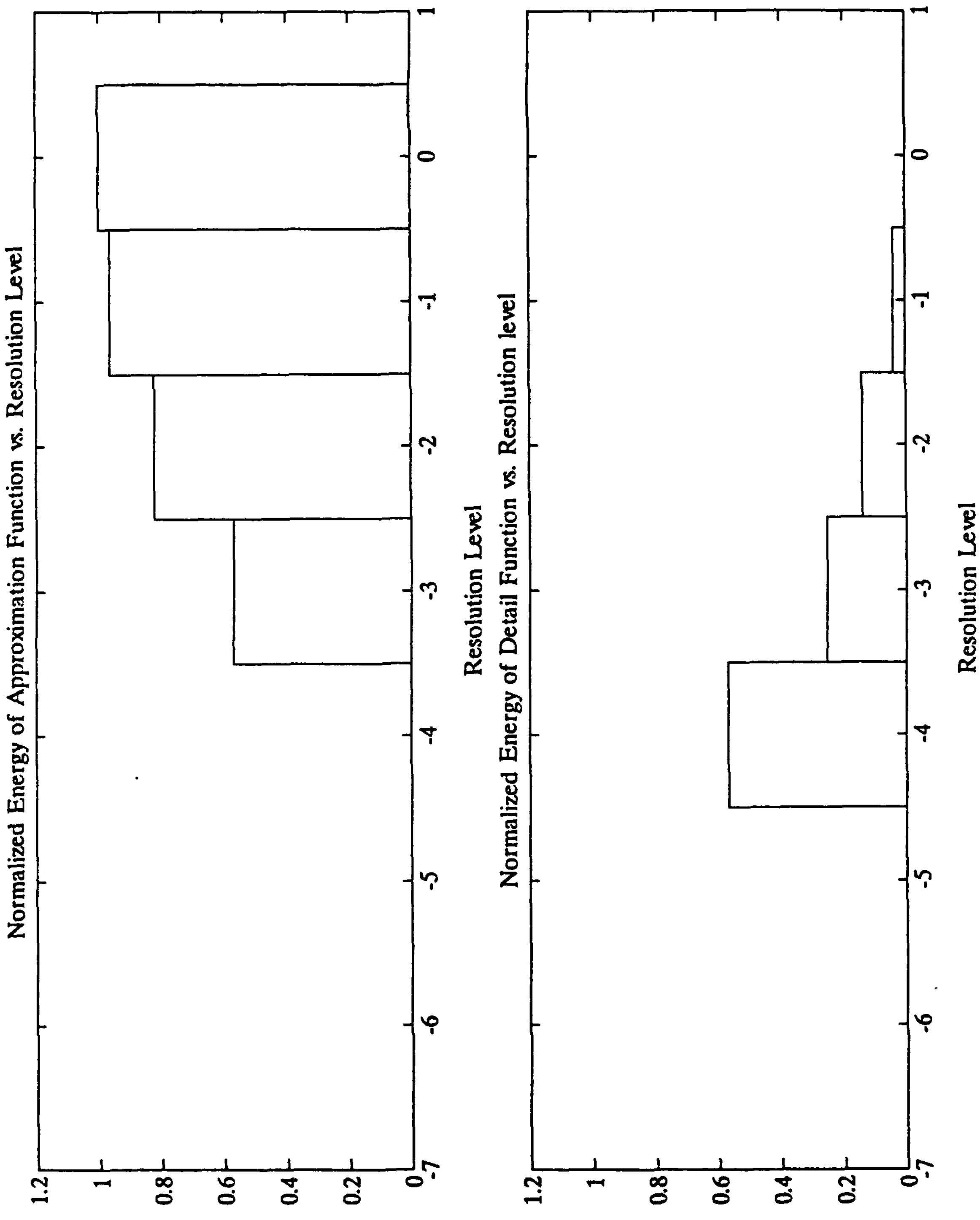

Figure 13. Haar Resolution Level Energy Distribution

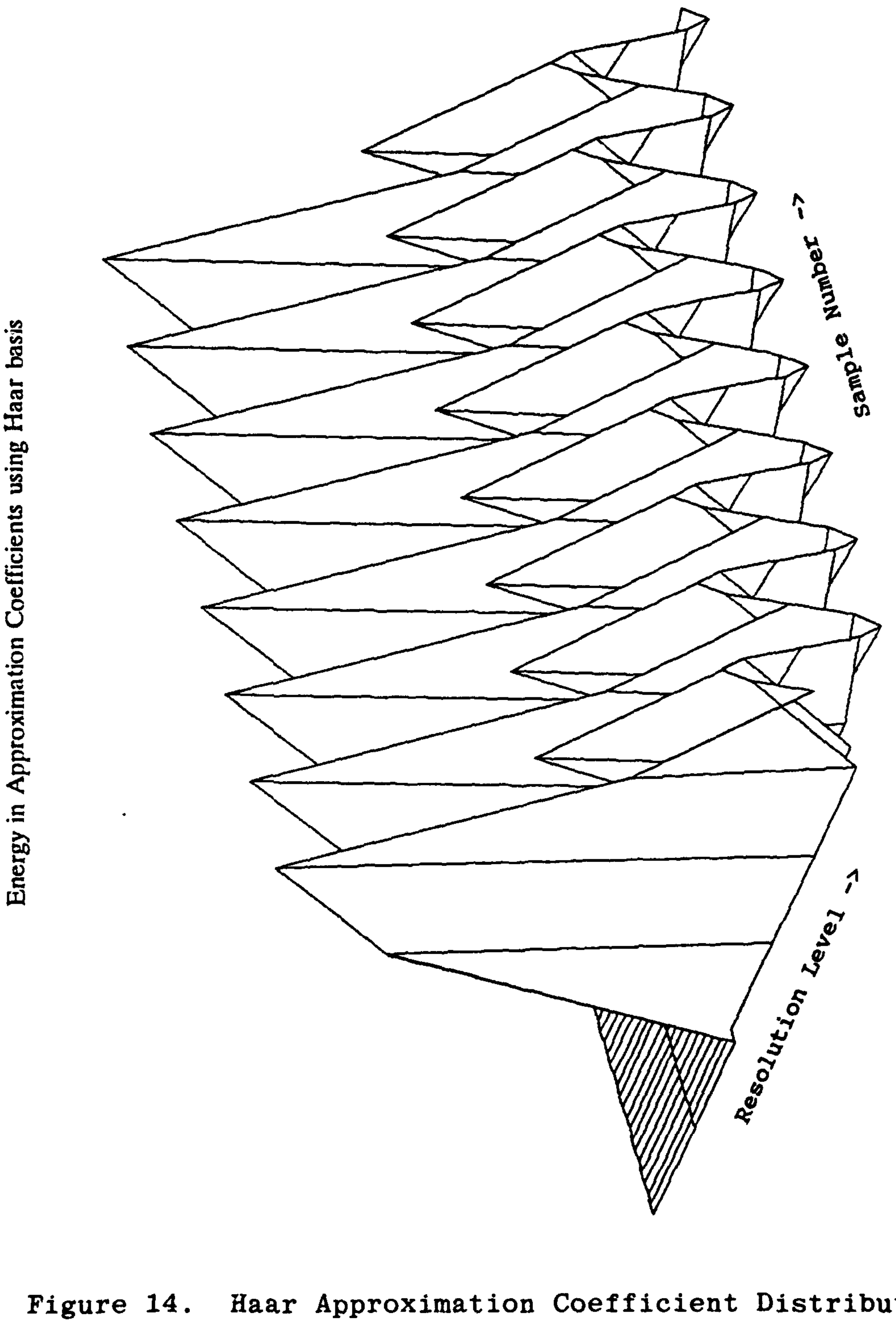

Figure 14. Haar Approximation Coefficient Distribution

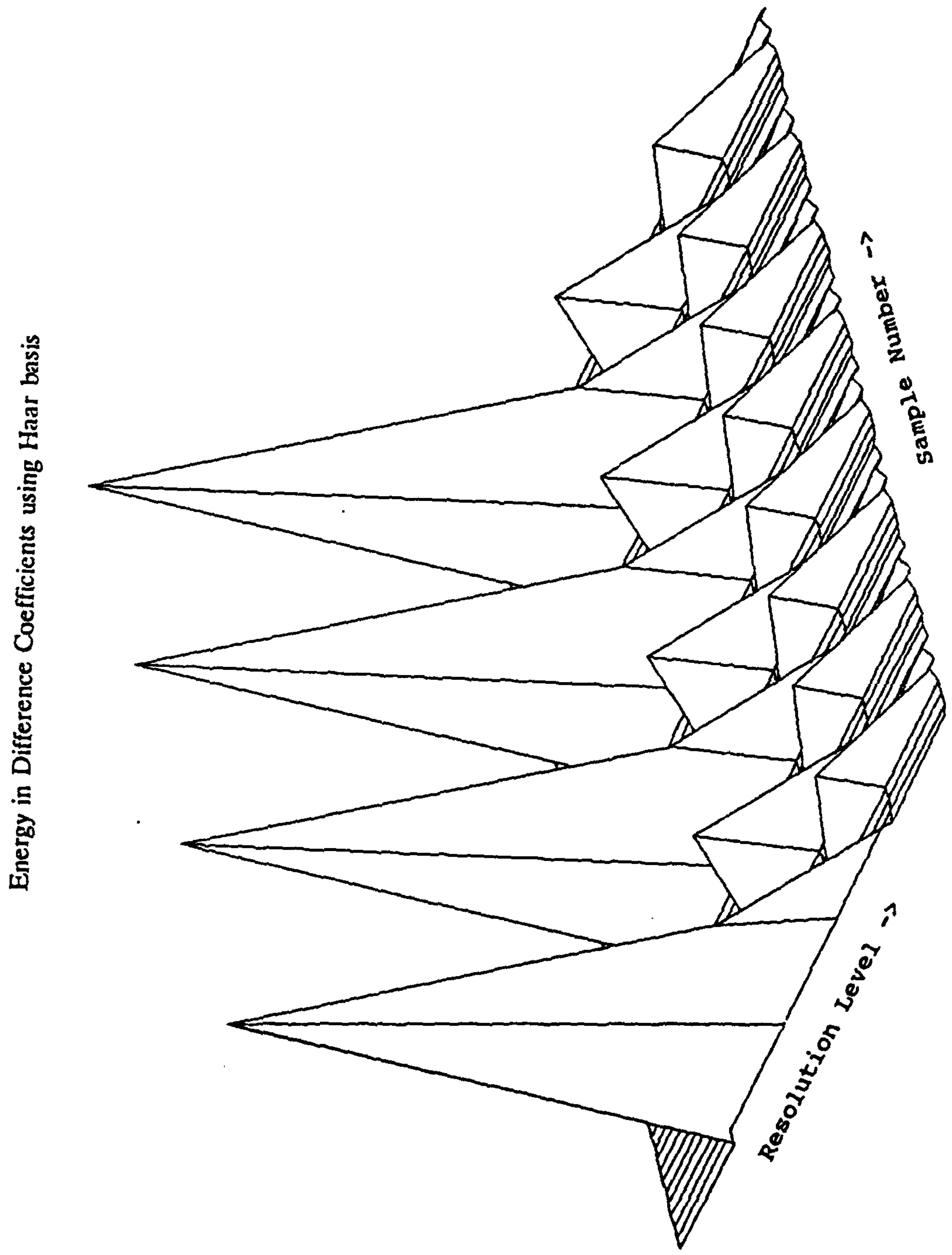

Figure 15. Haar Detail Coefficient Distribution

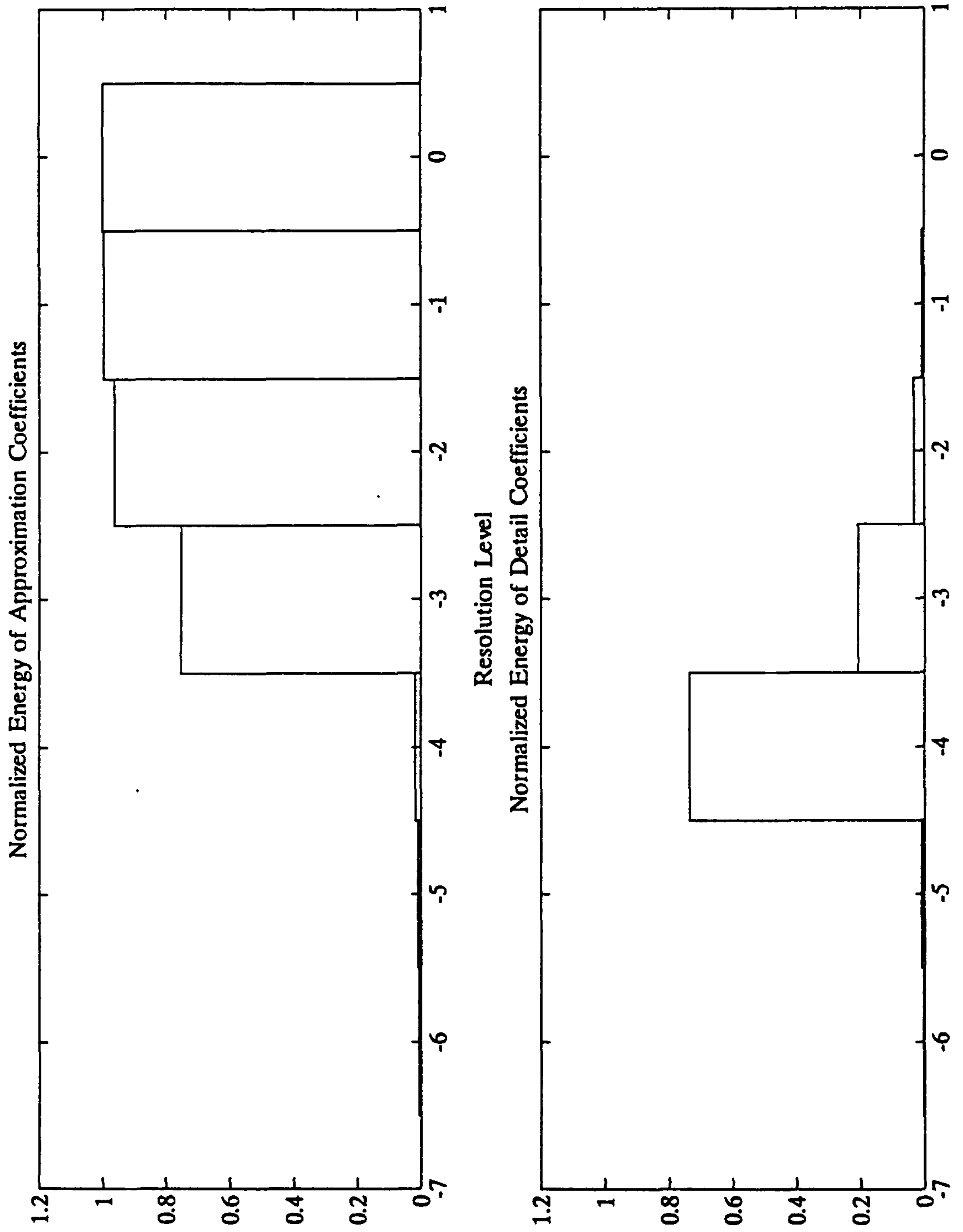

Figure 16. Daubechies Resolution Level Energy Distribution

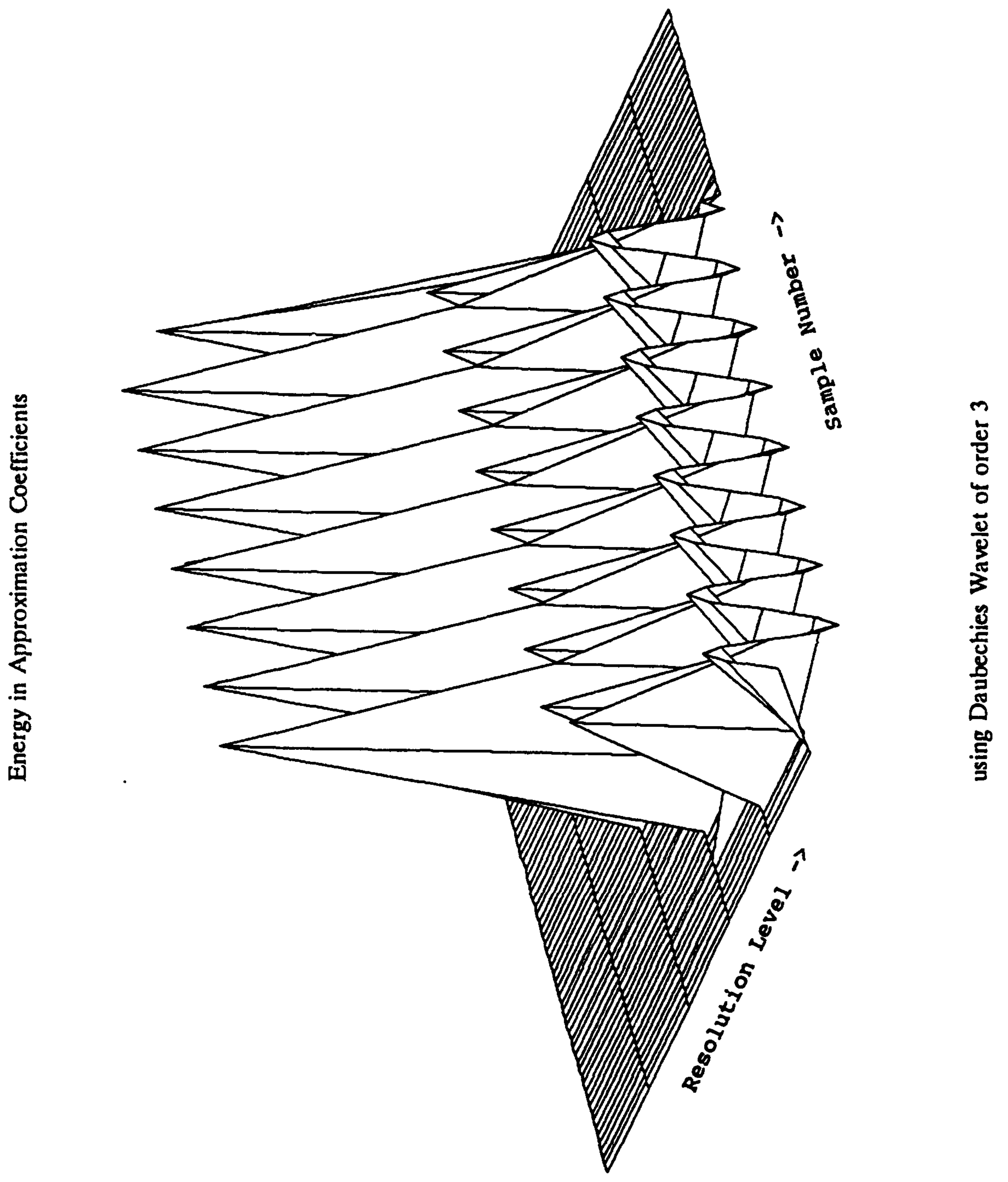

Figure 17. Daubechies Approximation Coefficient Distribution

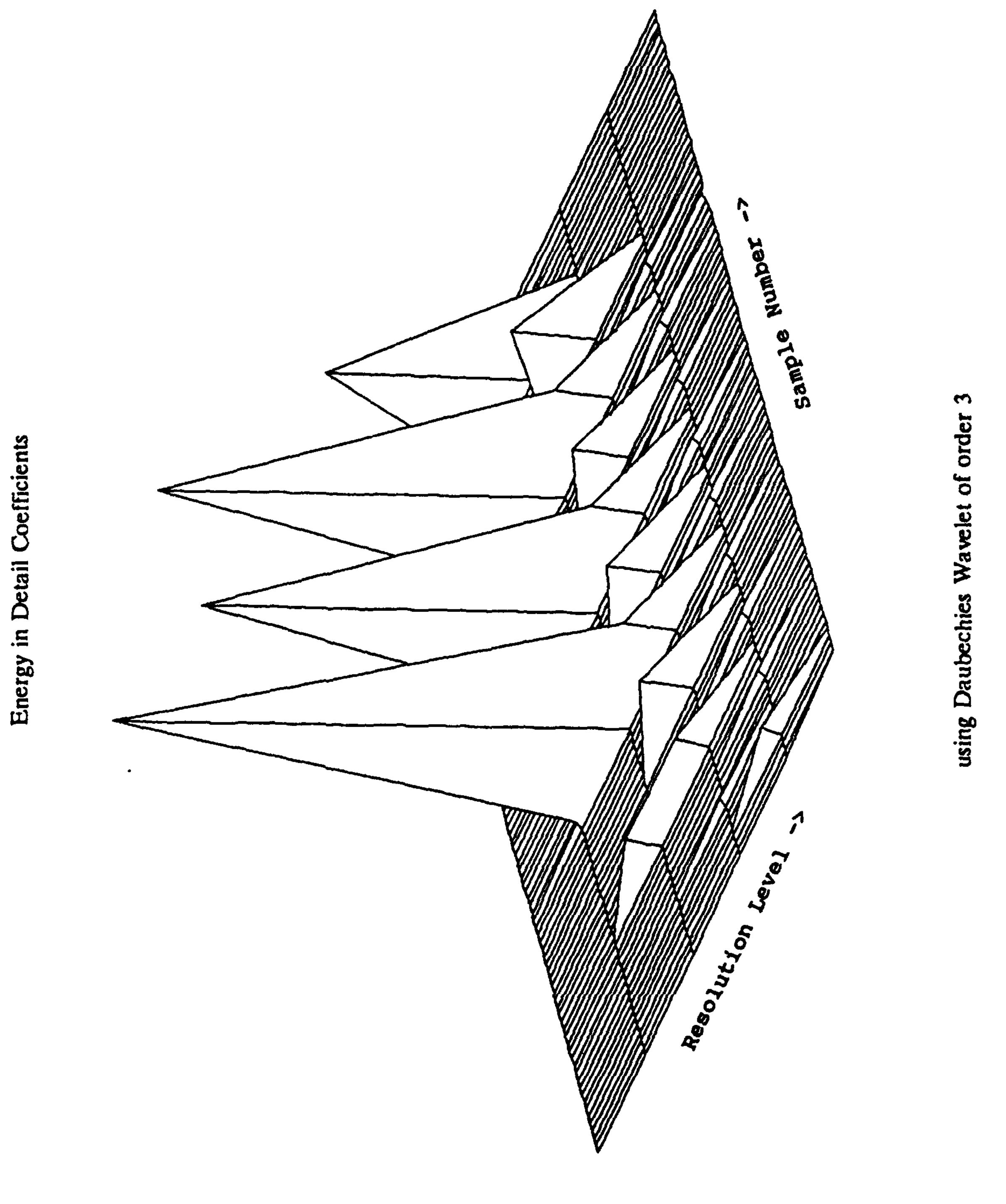

Figure 18. Daubechies Detail Coefficient Distribution

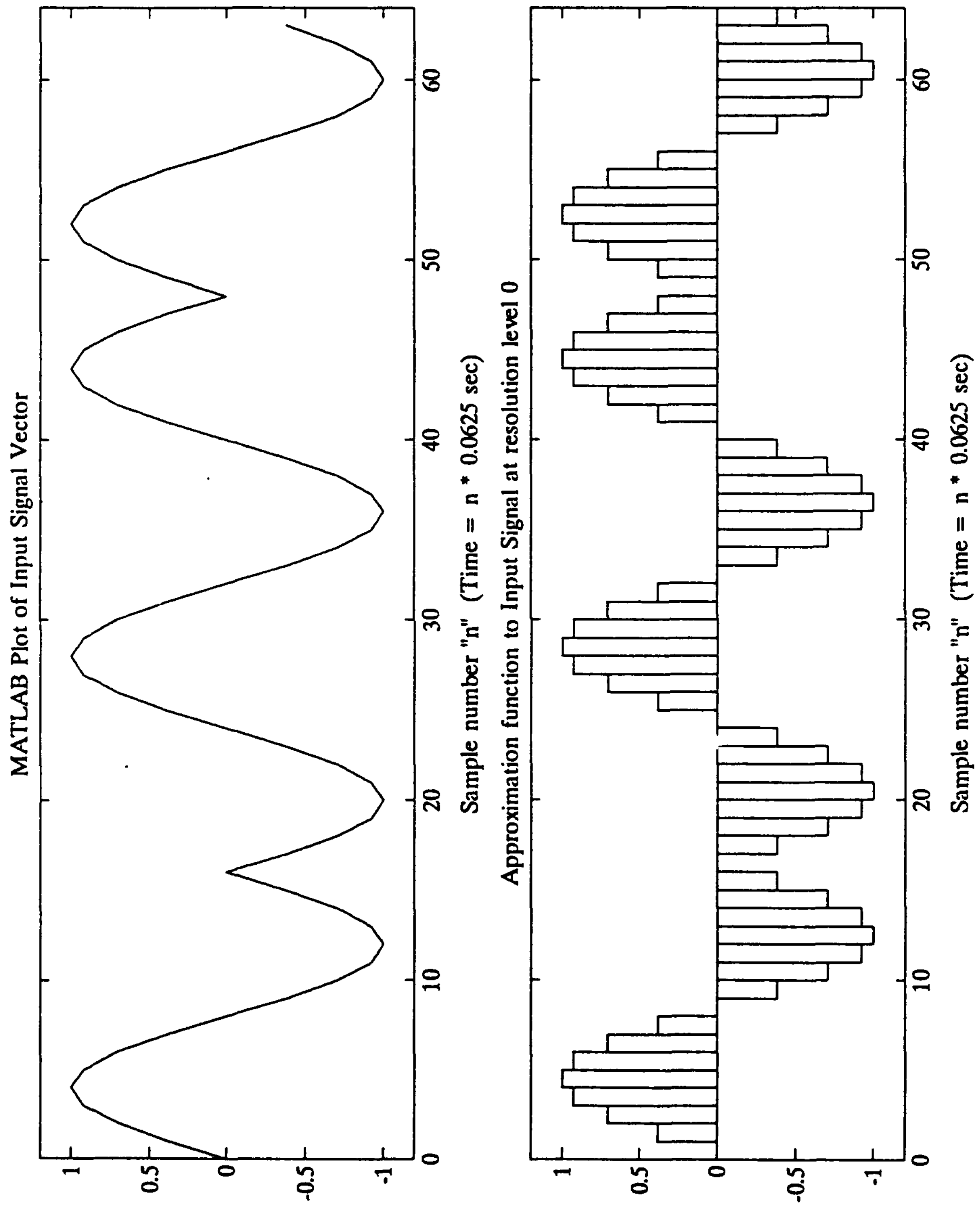

Figure 19. Binary Phase Shift Keyed Signal and Approximation

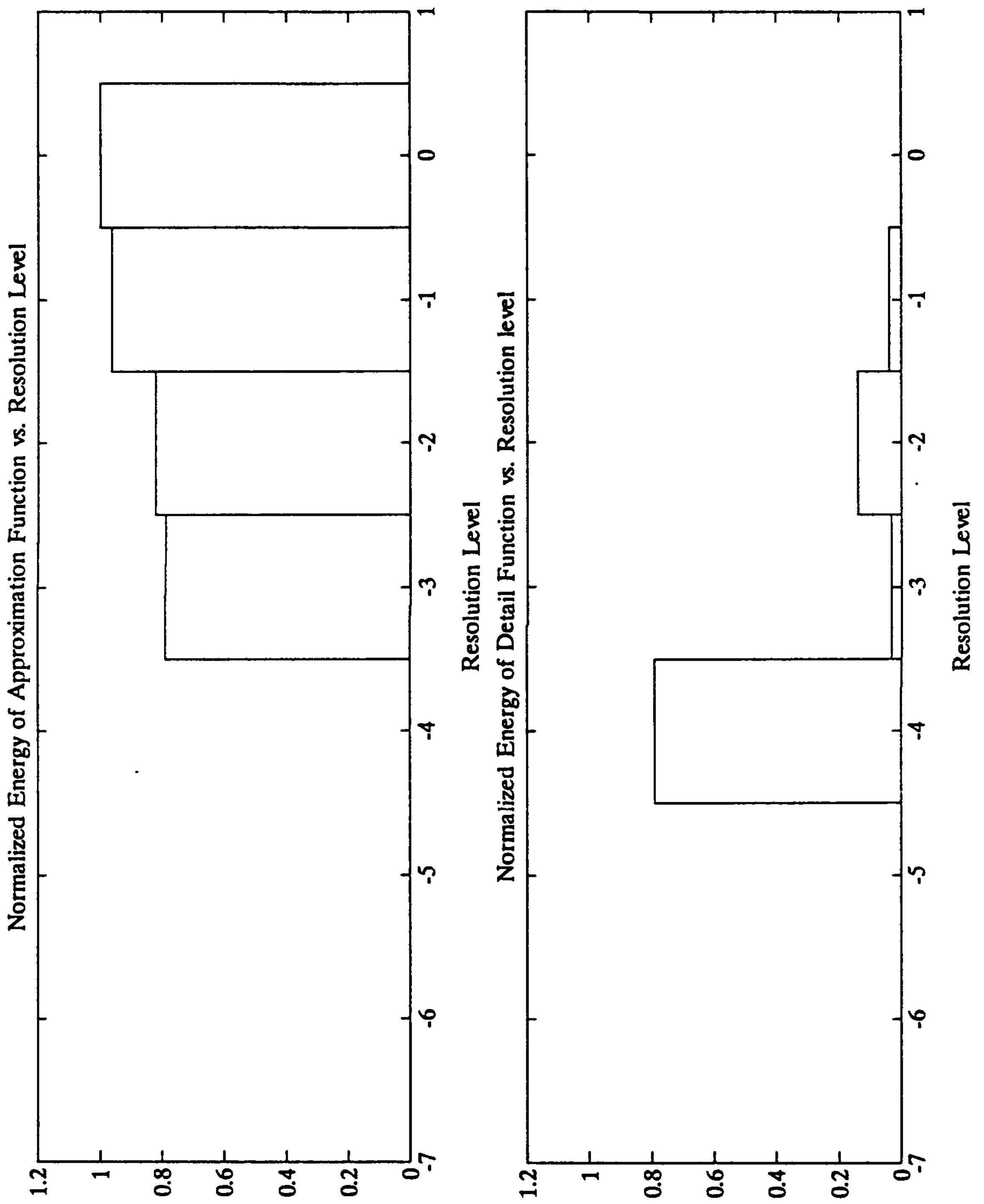

Figure 20. Haar Resolution Level Energy Distribution

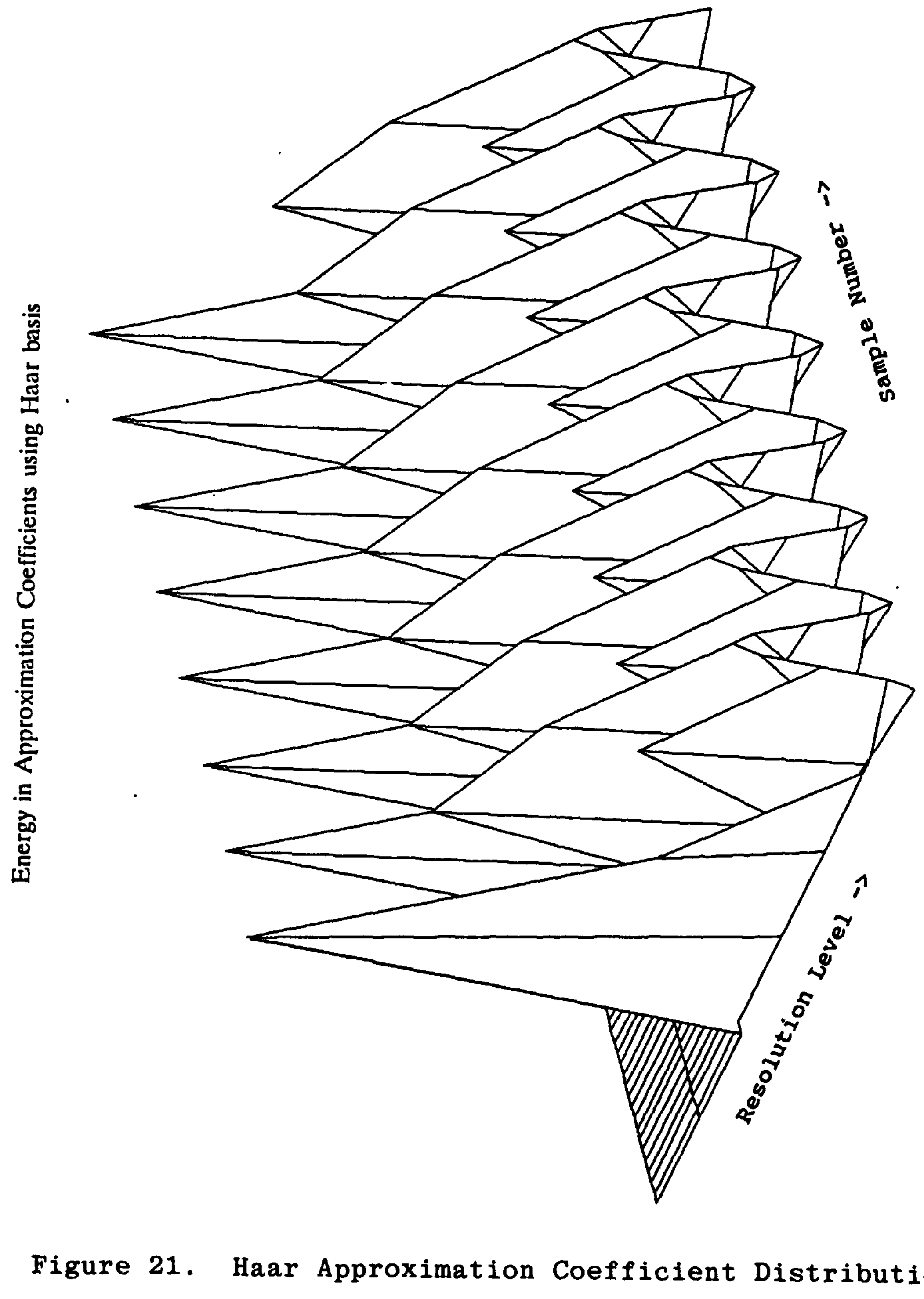

Figure 21. Haar Approximation Coefficient Distribution

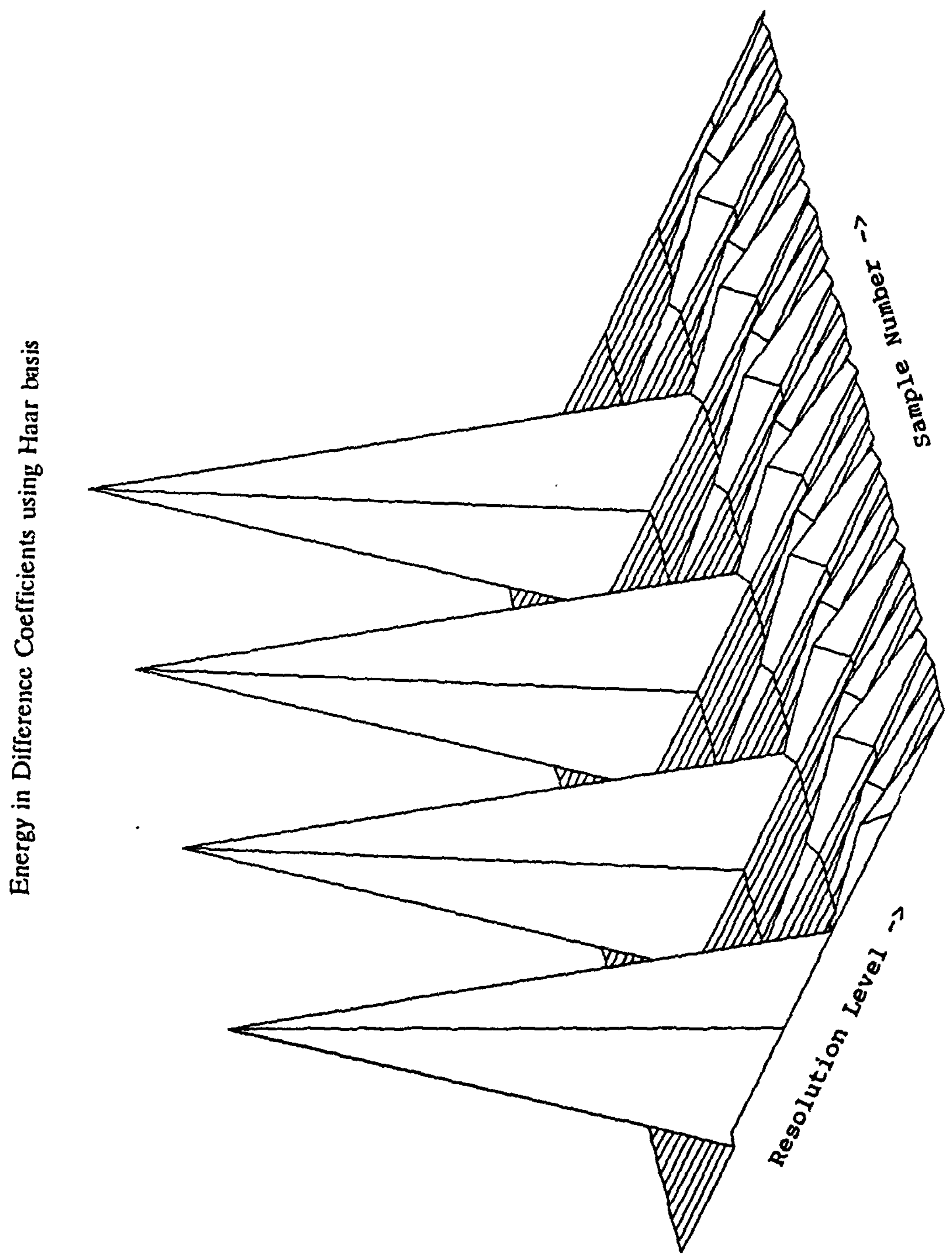

Figure 22. Haar Detail Coefficient Distribution

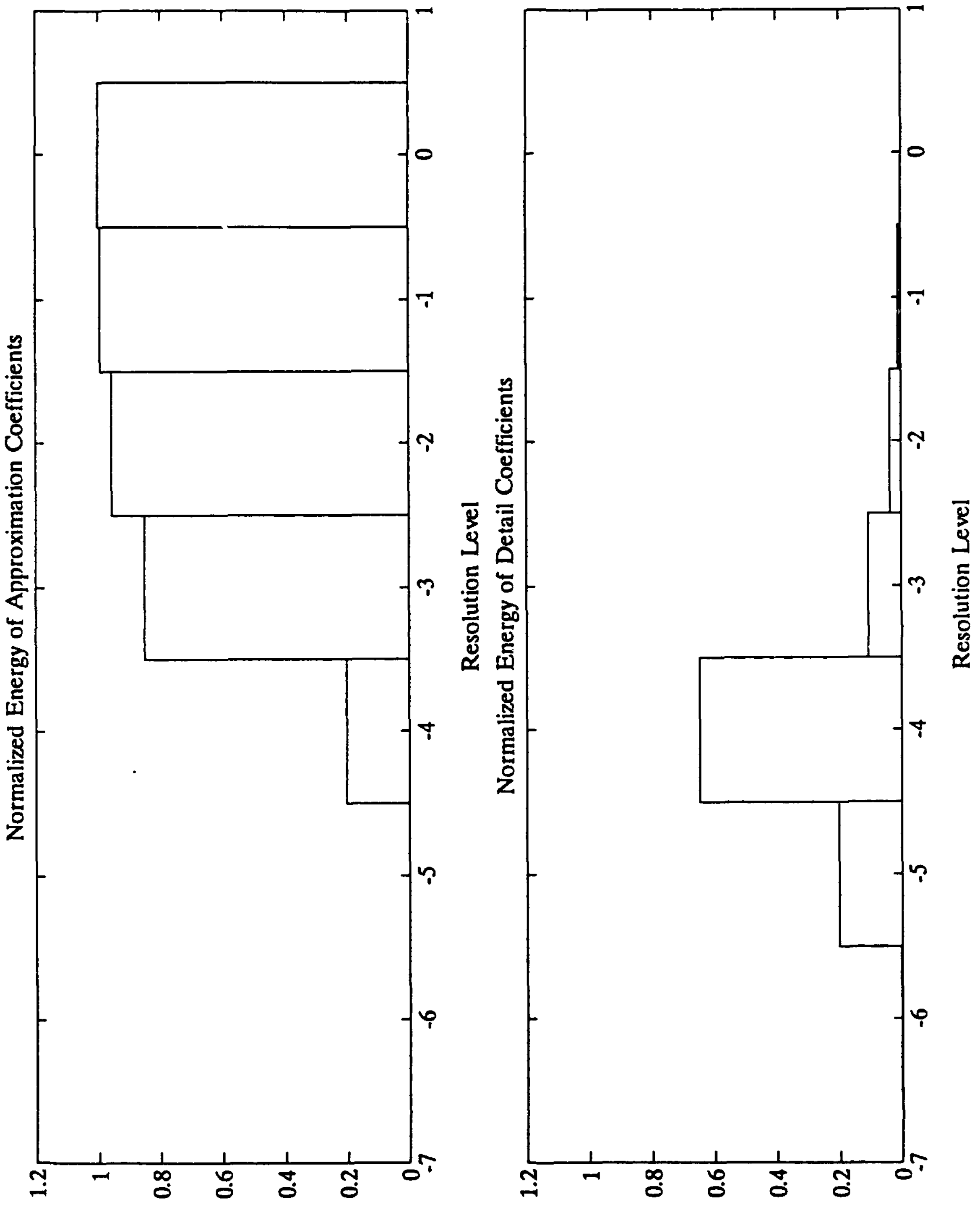

Figure 23. Daubechies Resolution Level Energy Distribution

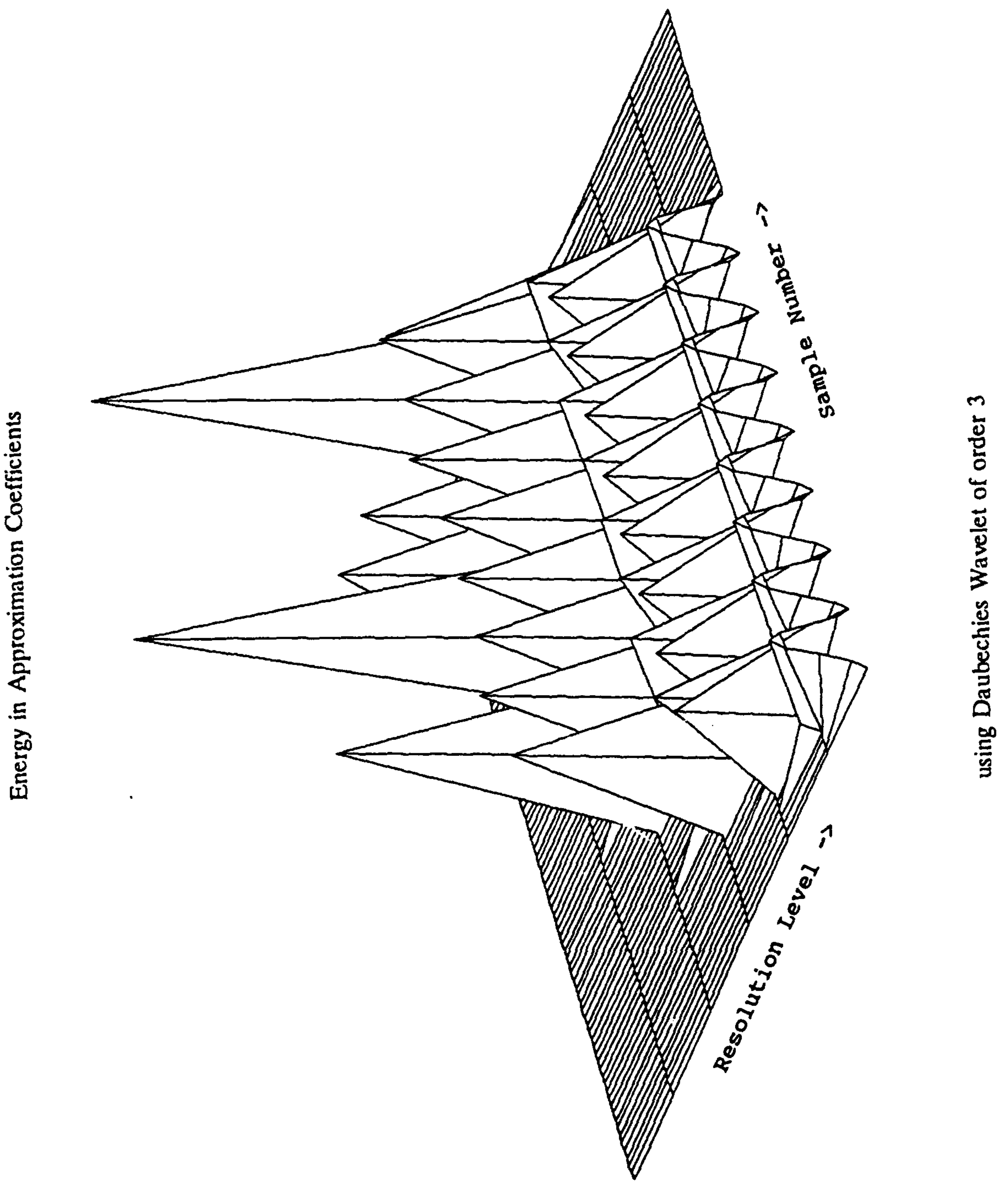

Figure 24. Daubechies Approximation Coefficient Distribution

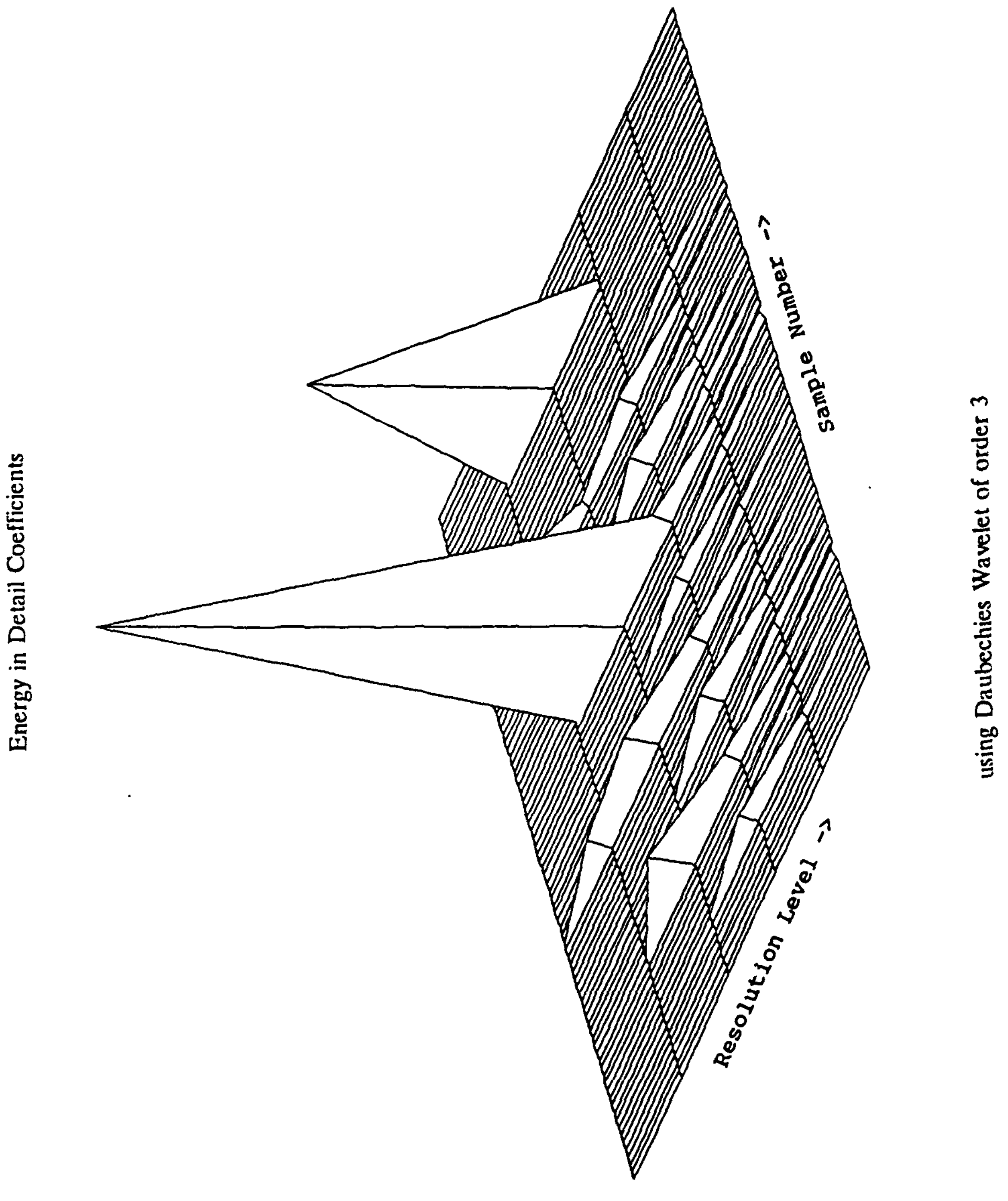

Figure 25. Daubechies Detail Coefficient Distribution

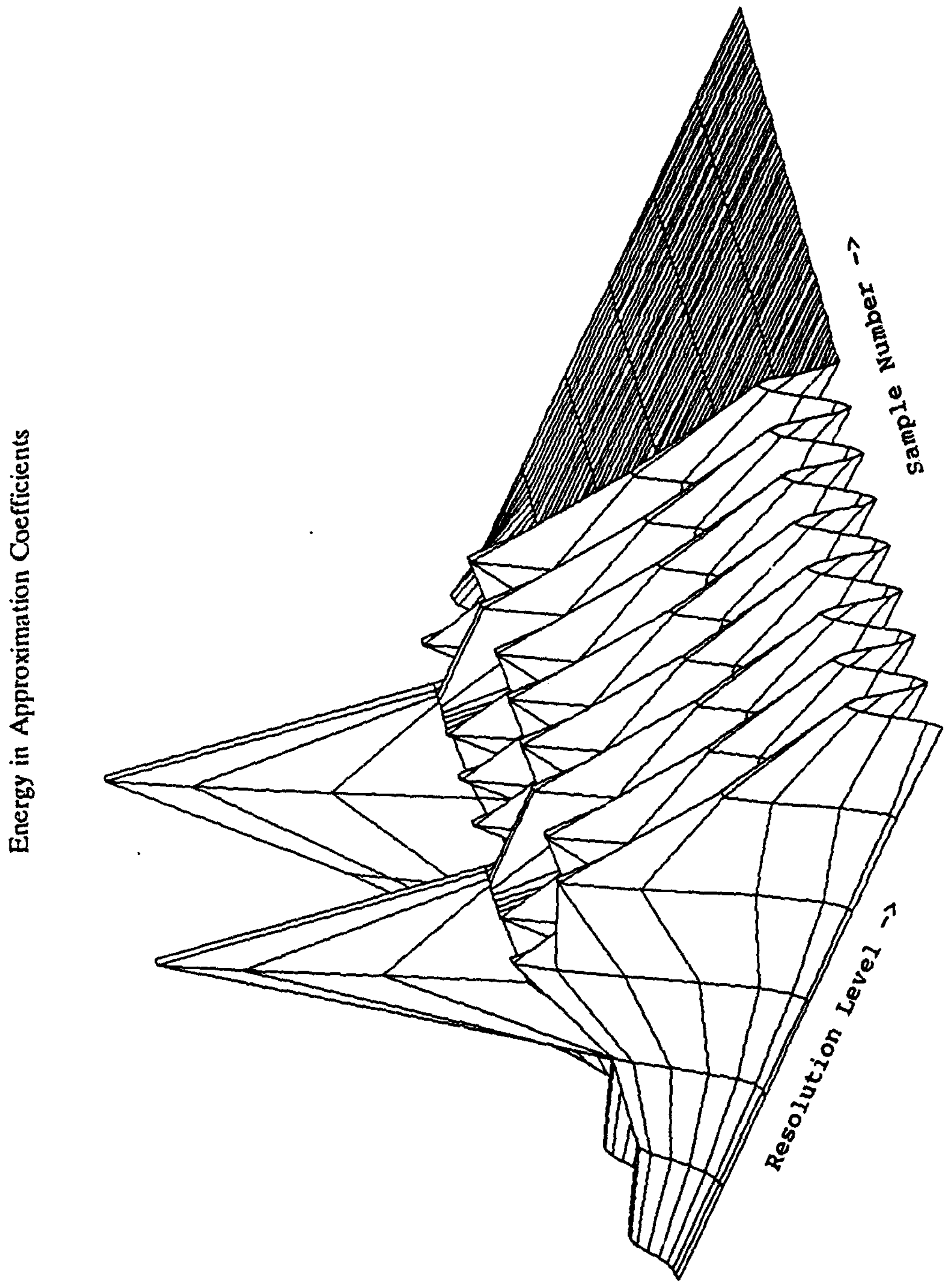

Figure 26. Multiphase Haar Approximation Coefficients

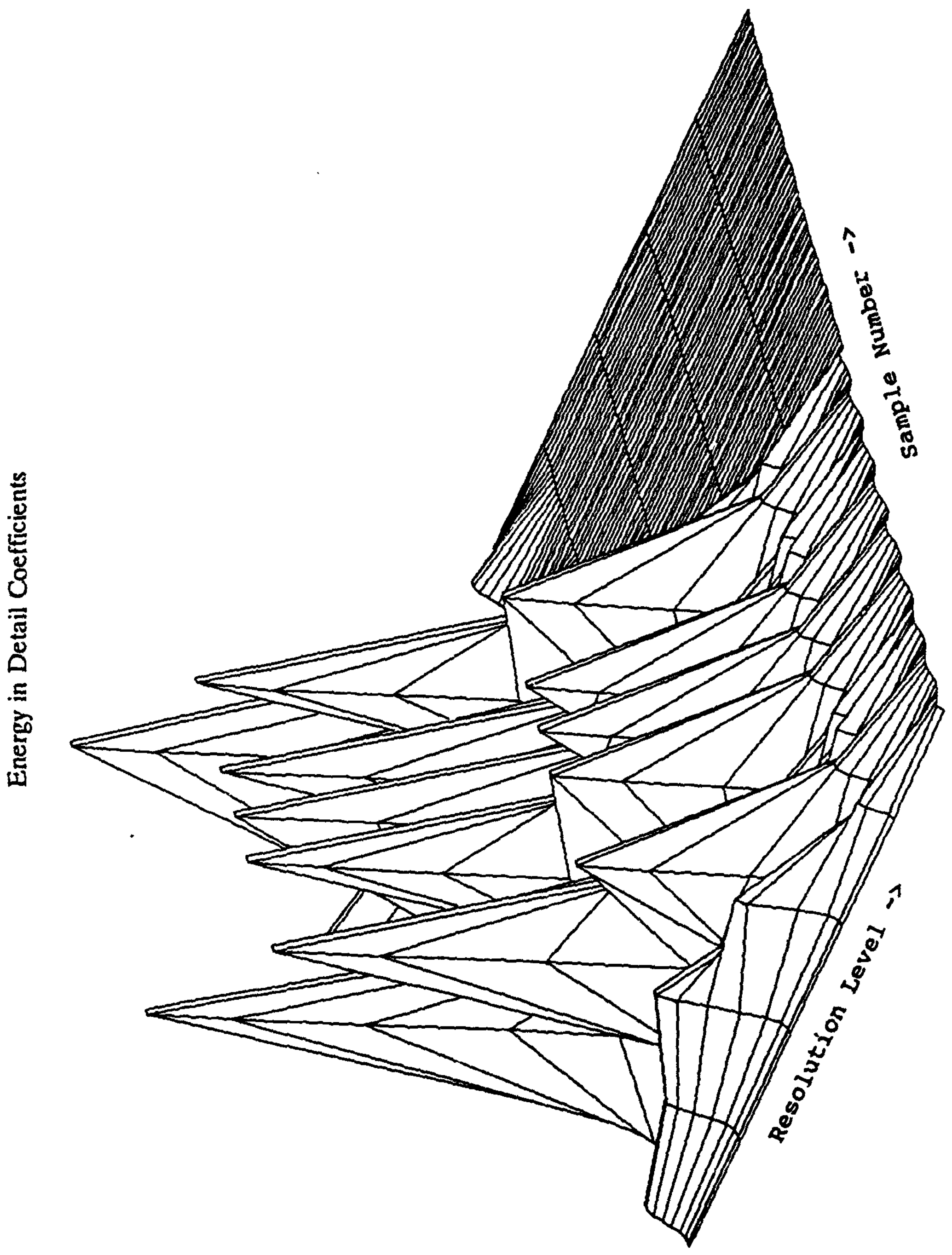

Figure 27. Multiphase Haar Detail Coefficients

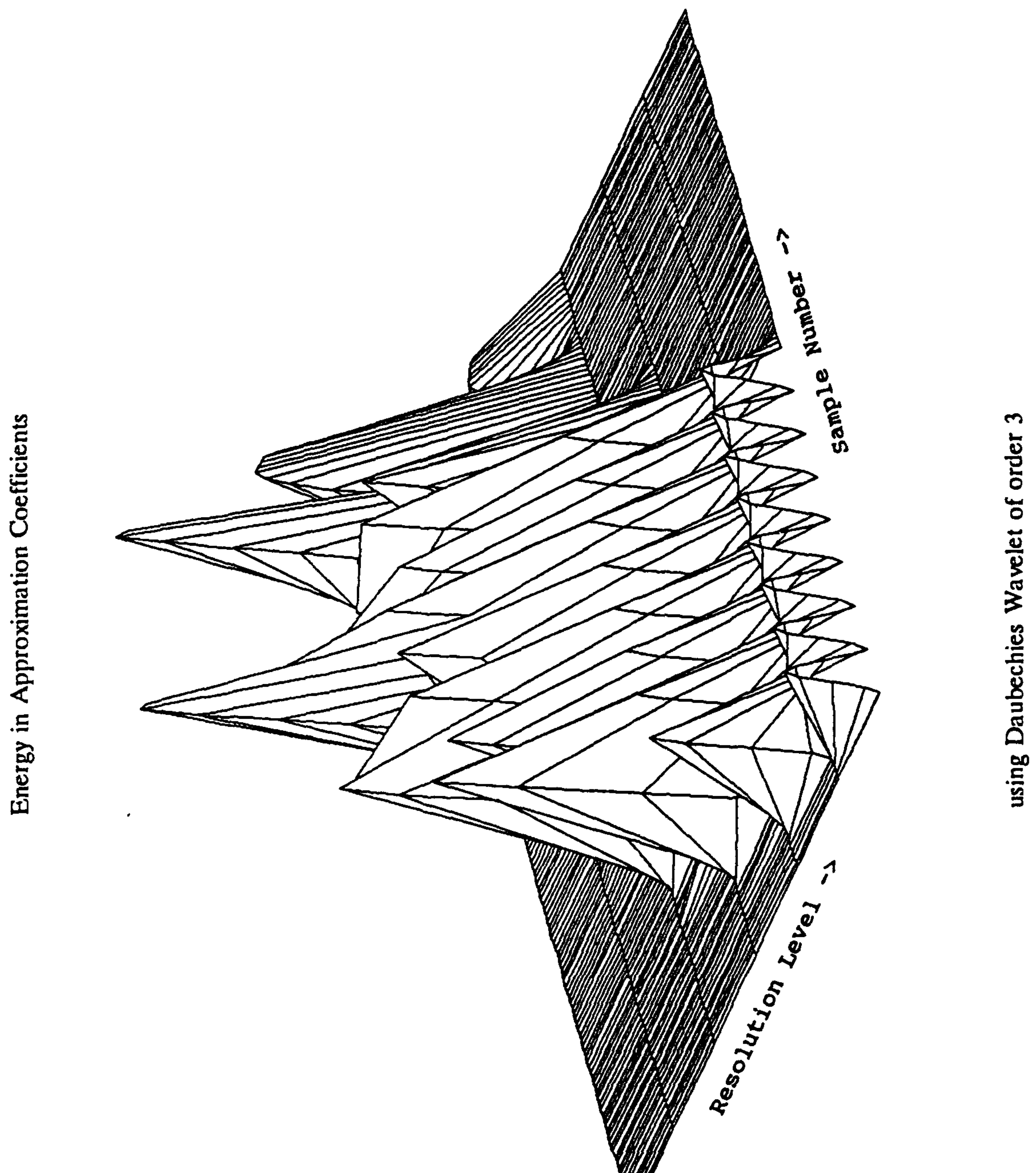

Figure 28. Multiphase Daubechies Approximation Coefficients

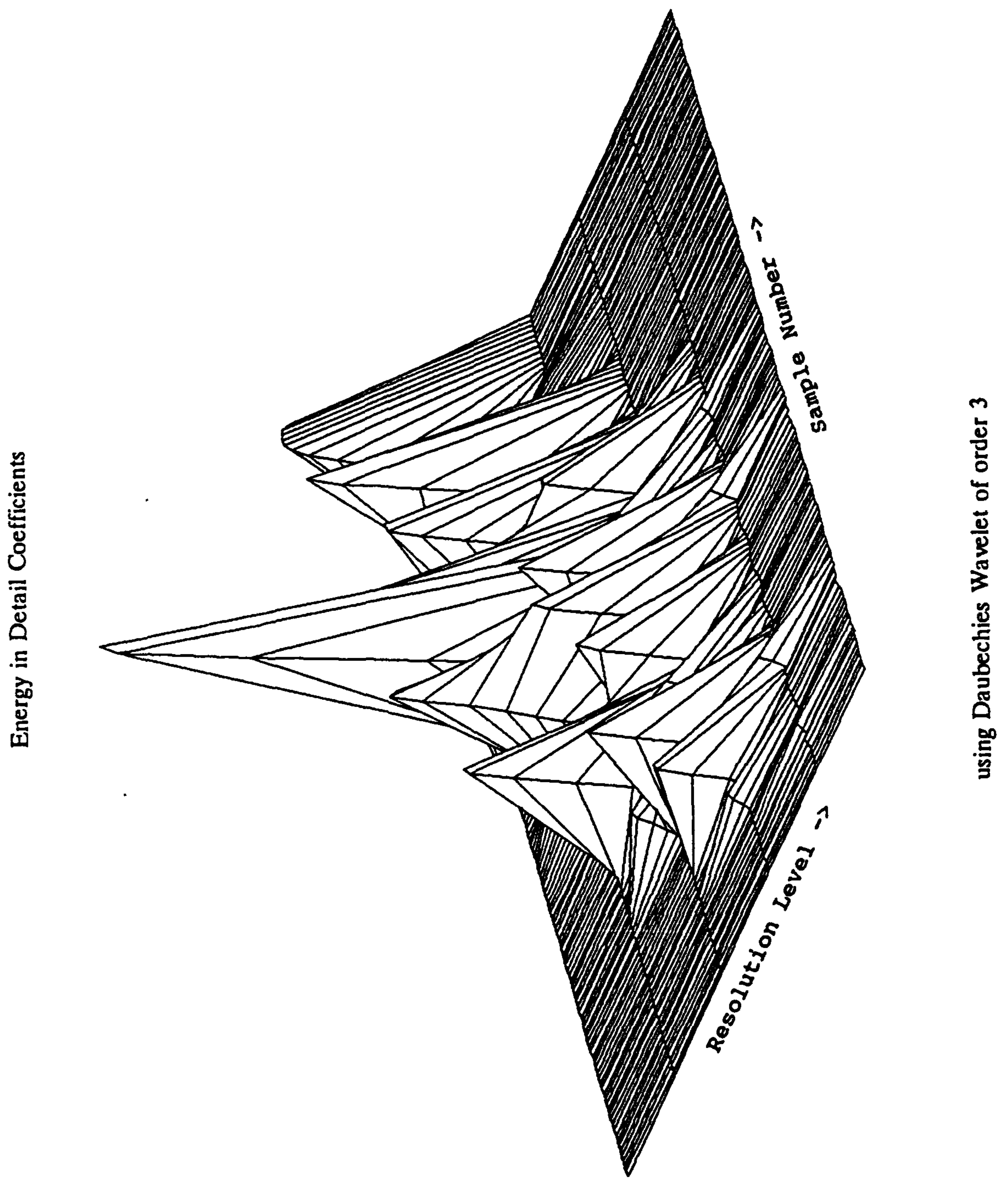

Figure 29. Multiphase Daubechies Detail Coefficients

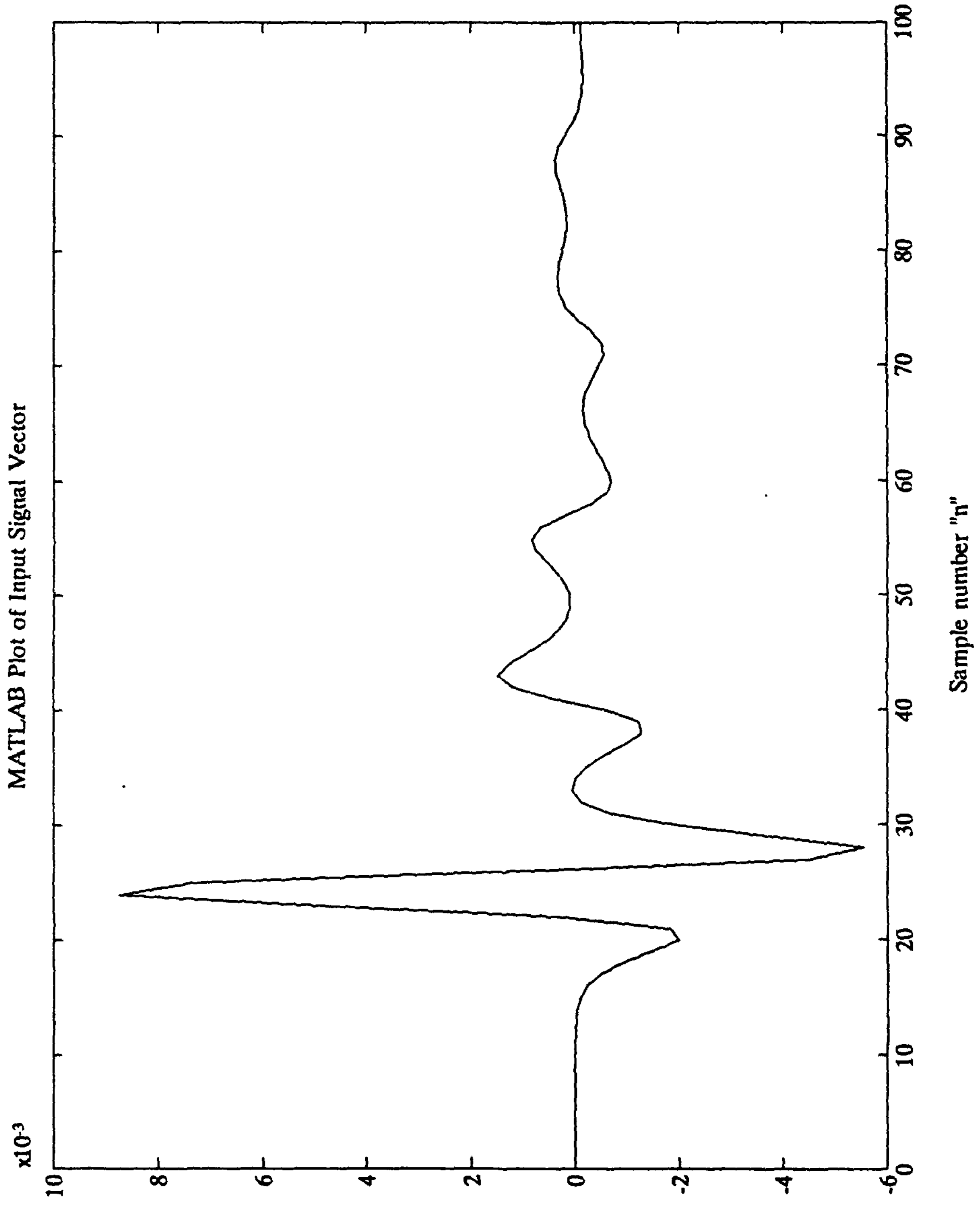

Figure 30. Transient Signal

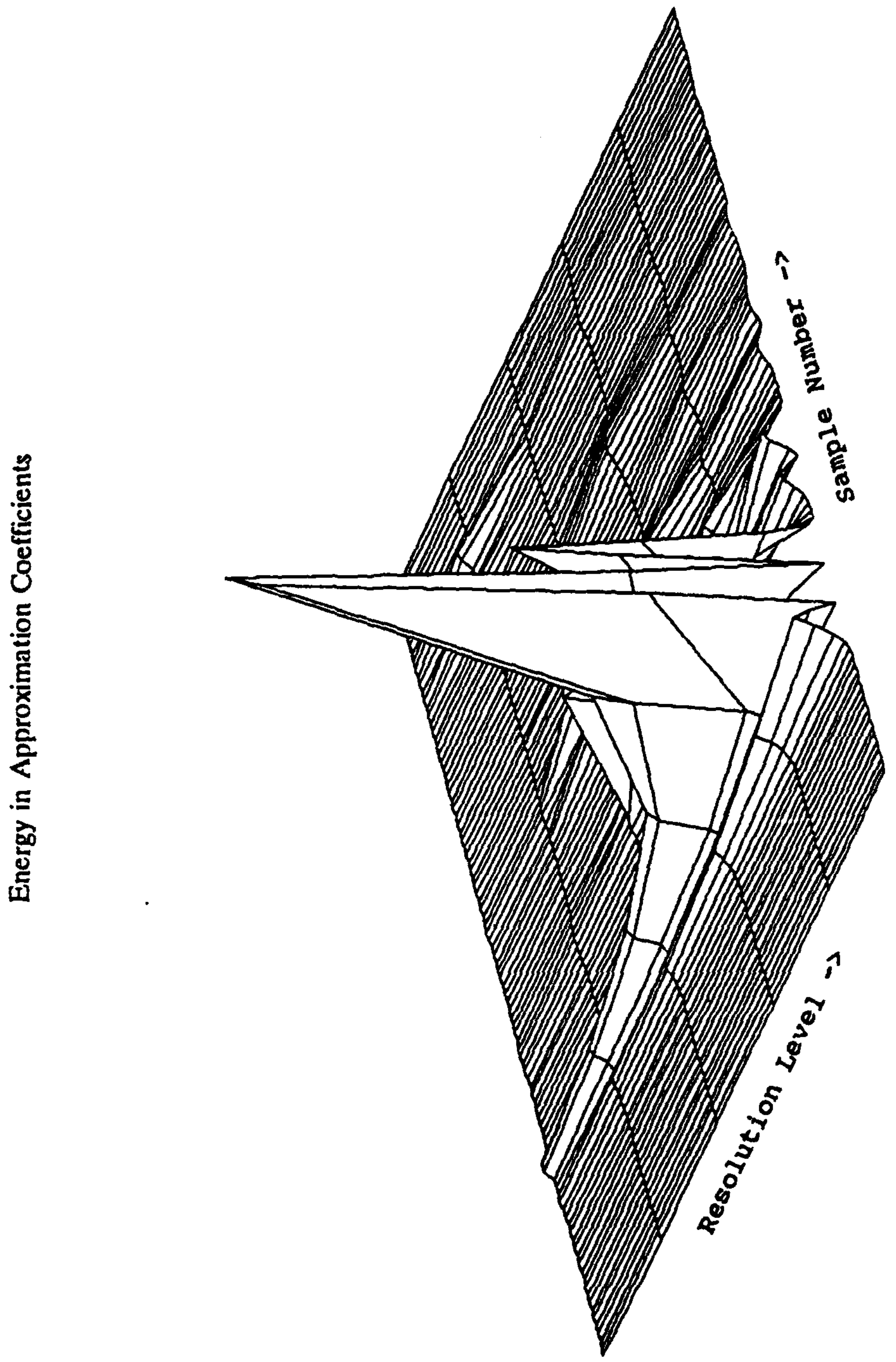

Figure 31. Multiphase Haar Approximation Coefficients

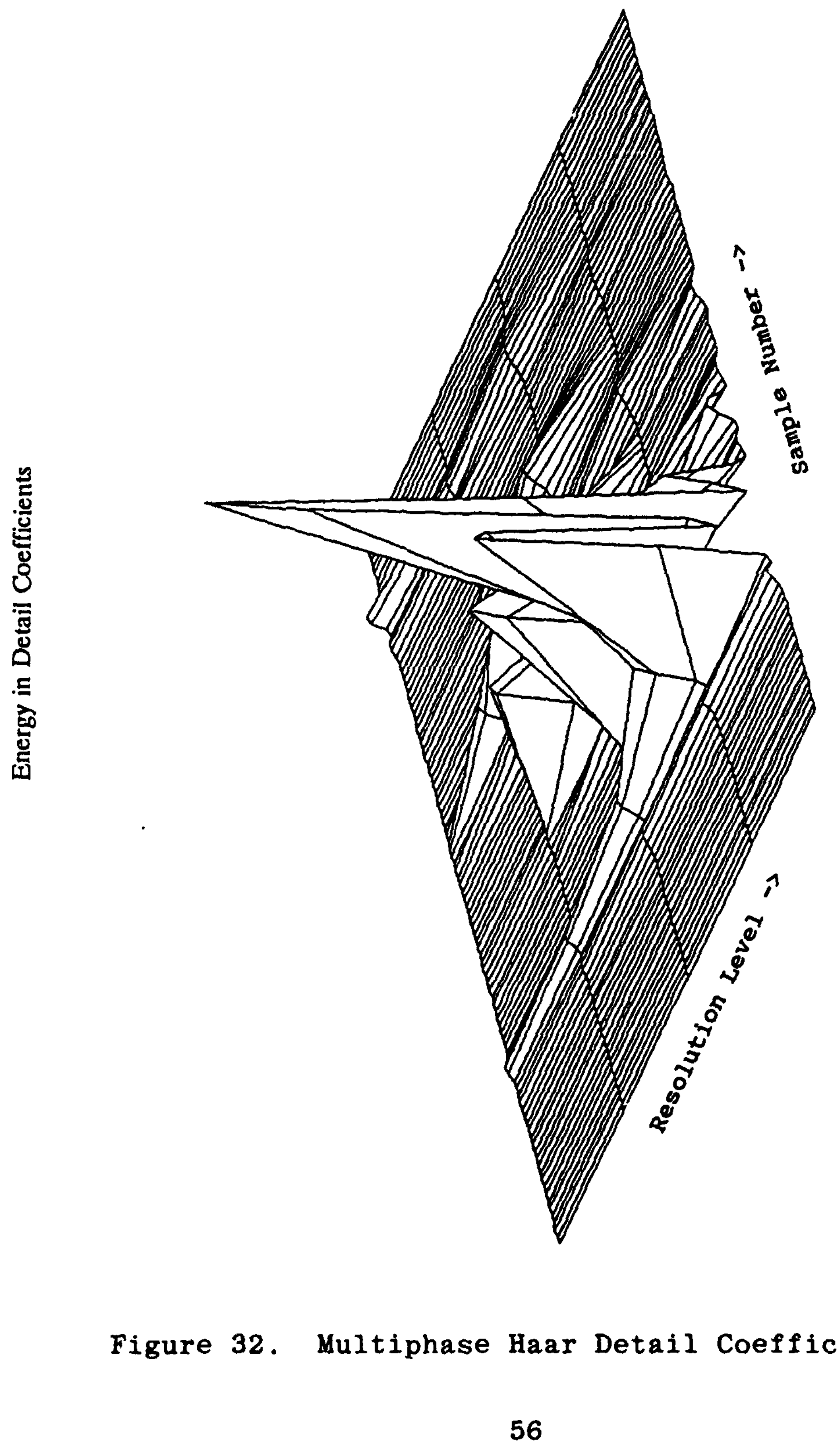

Figure 32. Multiphase Haar Detail Coefficients

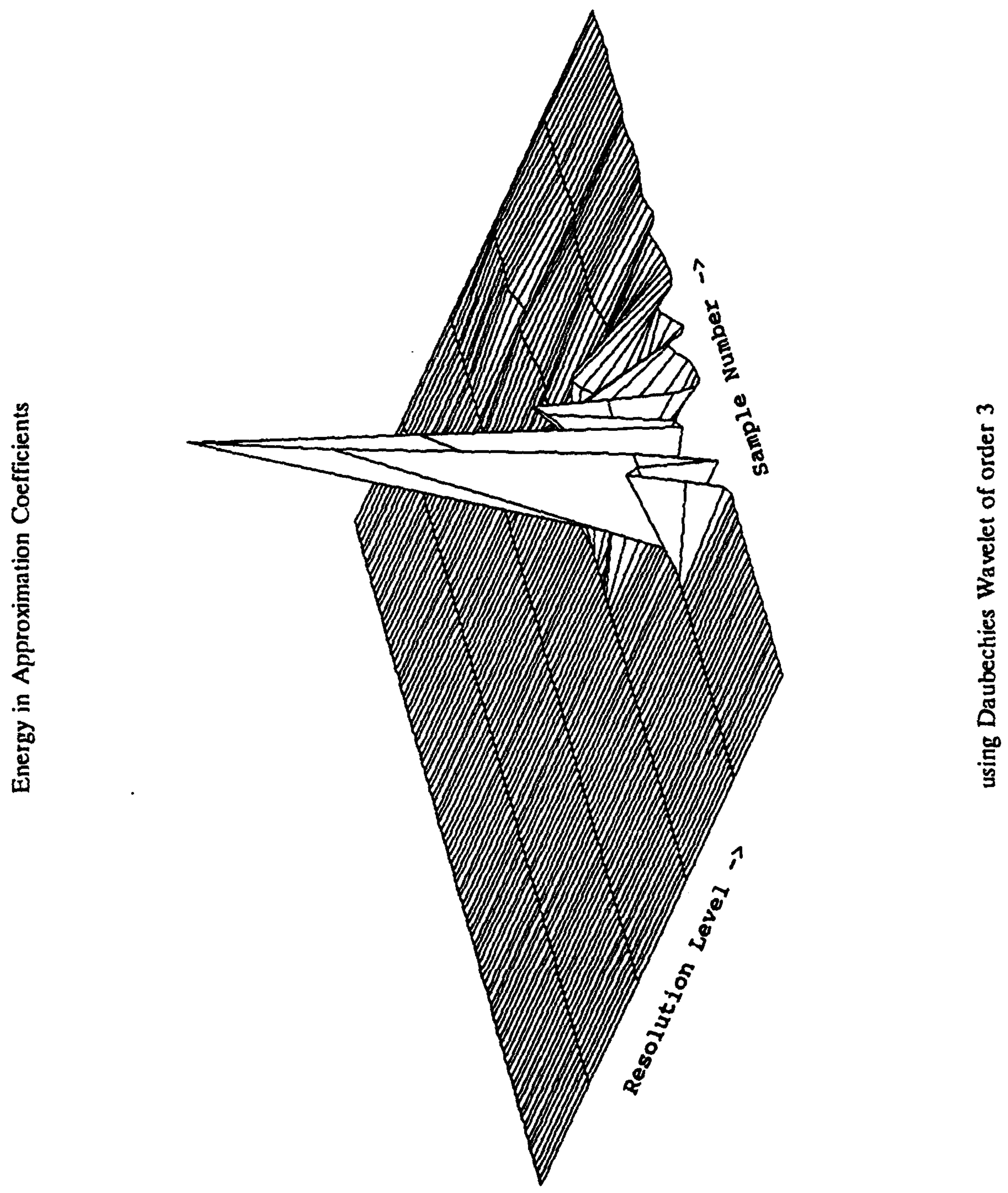

Figure 33. Multiphase Daubechies Approximation Coefficients

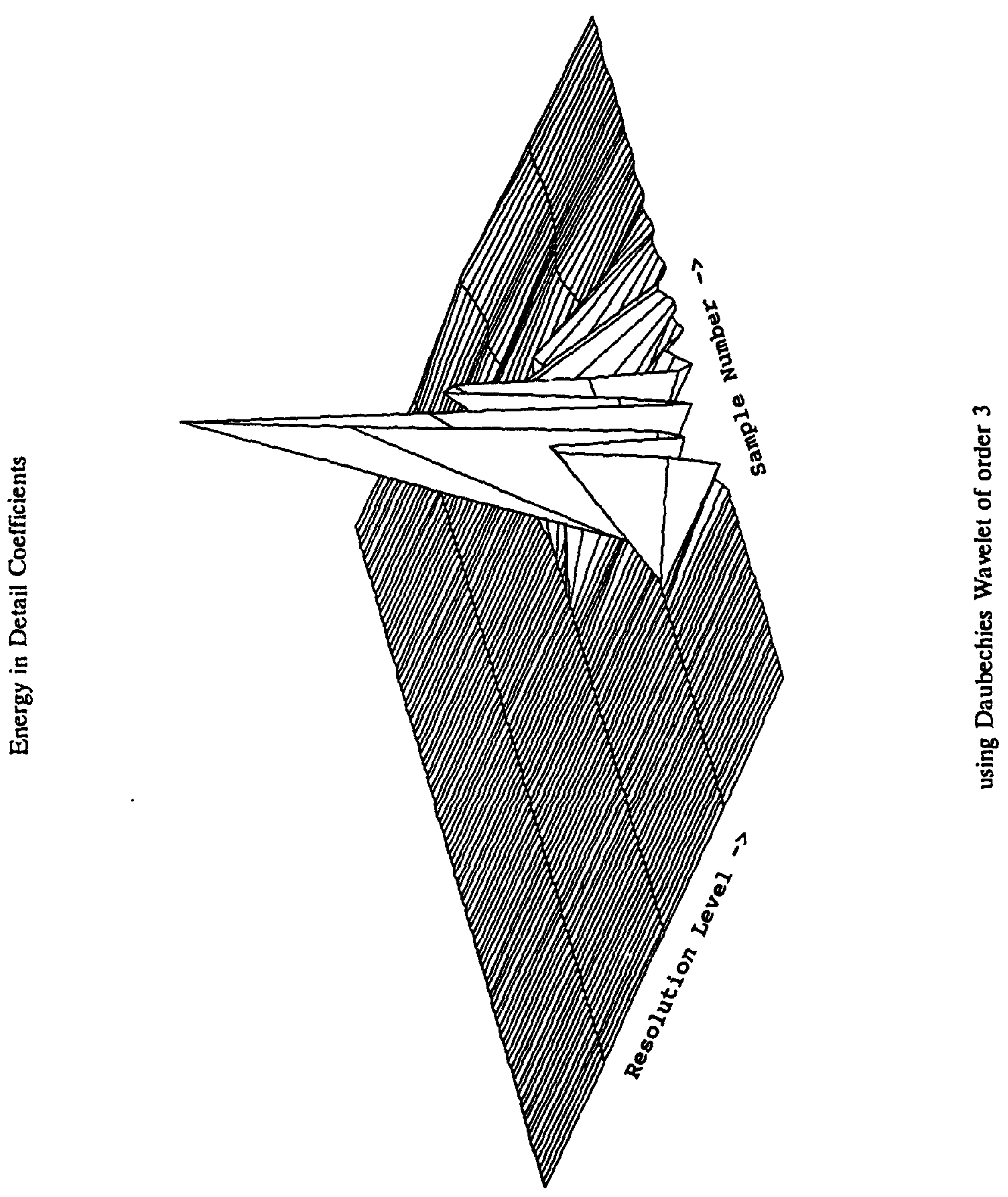

Figure 34. Multiphase Daubechies Detail Coefficients

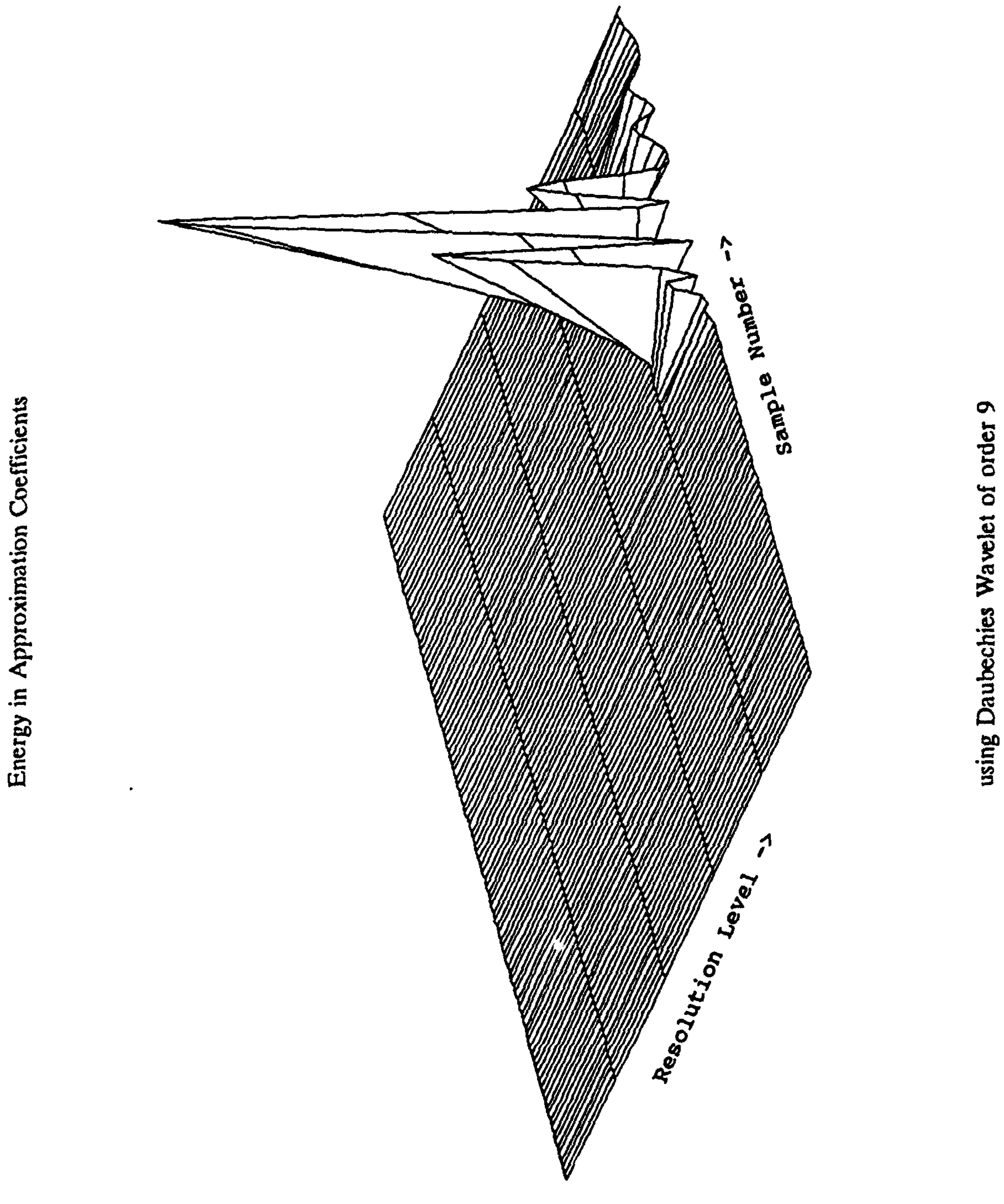

Figure 35. Multiphase Daubechies (Order 9) Approx. Coefficients

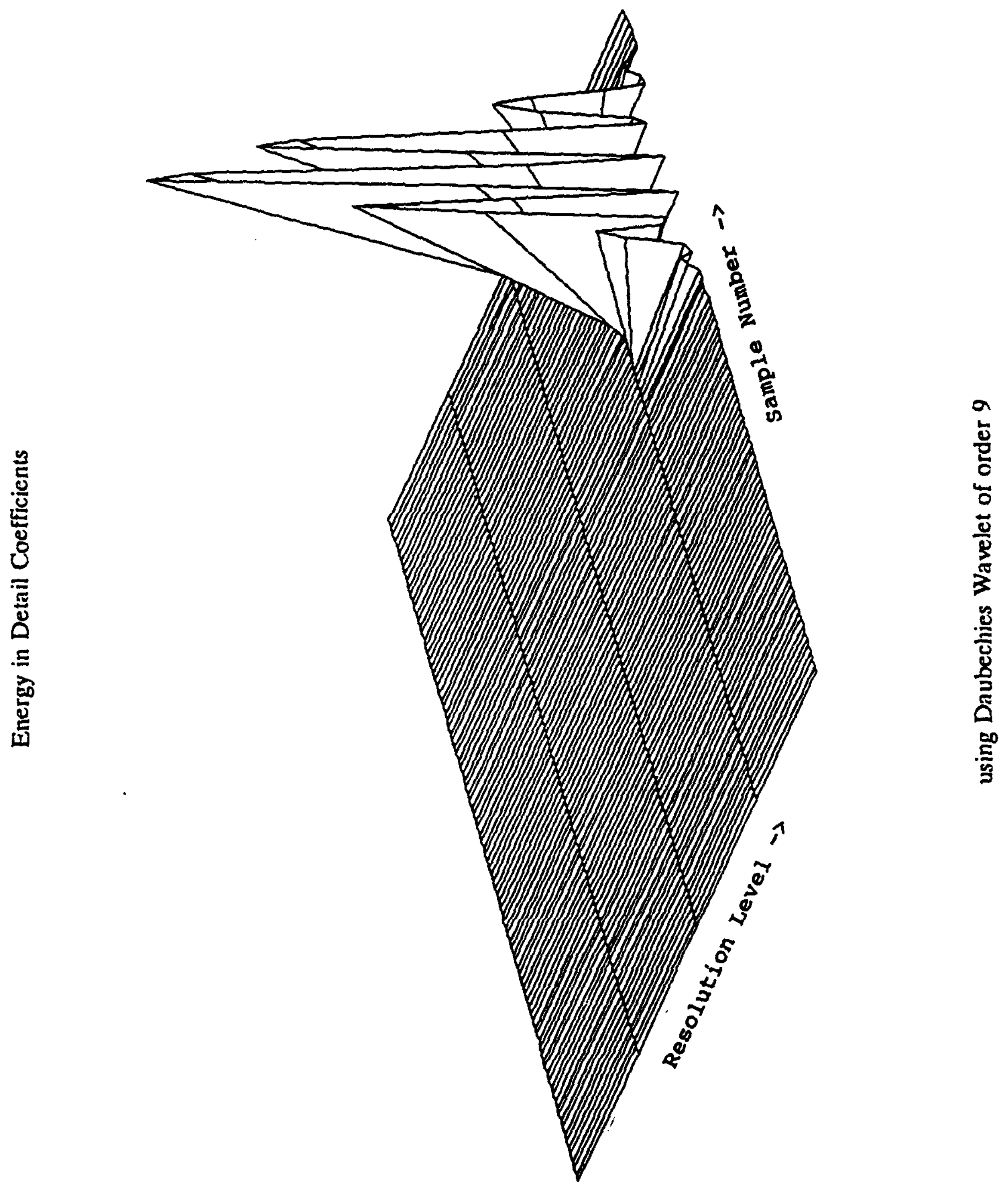

Figure 36. Multiphase Daubechies (Order 9) Detail Coefficients

APPENDIX B

CALLING PROGRAM FOR ACCESSING MULTIRESOLUTION ANALYSES PROGRAMS AND FUNCTIONS

```
%  Calling program for wavelet multiresolution analysis.  The capabilities
%  and options of the entire set of programs are presented, in general terms,
%  in the comments below.

b=[' '];
N1 = ['   This program will perform multiresolution analysis on input data'
      ' through the use of various orthonormal bases of compact support.  '
      ' It is interactive in nature and will allow you the opportunity    '
      ' to select among several different options to best support your   '
      ' specific needs.  Use the Return key to control movement through   '
      ' the program (after prompts and questions, to view next plot, etc.)'
      '                                                                   '
      ' Program options include:                                          '
      '                                                                   '
      '   1. Selection of Basis Functions                                 '
      '     a. Haar wavelet                                               '
      '     b. Daubechies group of compactly supported wavelets           '
      '     c. User-input wavelet ("h" coefficients)                      '
      '                                                                   '
      '   2. Generation of Scaling Function/Wavelet plots                 '
      '                                                                   '
      '   3. Selection of Input Data                                      '
      '     a. User-defined data vector (with optional noise background): '
      '       1) Sine wave                                                '
      '       2) Pseudo-noise (PN) sequence                               '
      '       3) Sine wave modulated by PN sequence                       '
      '     b. User-input data vector                                     '
      '                                          <Return> for more ...'];

N2 = ['   4. Decomposition algorithm based on:                              '
      '     a. Data "snapshot" - Fixed start point; one decomposition;      '
      '                          single phase approach                      '
      '     b. "Sliding" Data window - Floating start point; decomposition '
      '                                for each start point; multiphase    '
      '                                approach; discrete approximation to '
      '                                continuous wavelet transform        '
      '                                                                     '
      '                                                                     '
      '                                                                     '
      '                                                                     '
      '  Do you want to see:                                                '
      '                                                                     '
      '    1. Haar wavelet data "snapshot" decomposition,                   '
      '         (input data treated as a piecewise constant function,       '
      '           allowing inner products to be calculated and displayed)   '
      '                                                                     '
      '    2. Daubechies or user wavelet data "snapshot" decomposition, or '
      '                                                                     '
      '    3. "Sliding" data window wavelet (any type) decomposition?      '];

clc
disp(b)
disp(N1)
pause
clc
disp(b)
disp(N2)
disp(b)
desire = input('Answer 1, 2, or 3. ');
```

```
if desire == 1,
  haarwave
elseif desire == 2,
  daubwave
else
  multiphs
end
```

APPENDIX C

PROGRAM FOR SINGLE PHASE ANALYSIS USING HAAR WAVELET

```
% This program performs single-phase Haar wavelet decomposition on a
% user-supplied or program-generated input waveform.  Representation is
% made with bar graphs to show actual inner product calculation (input
% data is taken as a piecewise constant function).  The input data sequence
% will be zero-padded to accomodate the calculation of all possible non-
% zero coefficients and approximation ("c") and detail ("d") coefficients
% may be computed for sample numbers beyond the data endpoint.  The user
% should consider these "edge effects" in his analysis.

%---------------------------------------------------------------------------
% Determine if user wants to input an external data vector or desires to
% build one through the program.

clc

b=[' '];
Q1 = ['Do you wish to use:                                        '
      '  1. your own MATLAB formatted row vector, or              '
      '  2. a program generated vector from the following menu?'
      '       - Sine wave                                         '
      '       - Pseudo-noise (PN) sequence                        '
      '       - Sine wave modulated by PN sequence                '];

disp(b)
disp(Q1)
Pick = input('Answer 1 or 2: ');

if Pick == 1,
     % Read in user's input vector
     N1 = [' Note: If the number of data points is a power of 2, the'
           ' results are easier to interpret because there are not '
           ' any "edge effects".                                    '];
     disp(b)
     disp(N1)
     Wavedata = input('What is the name of your input vector?  ');
     sampfreq = input('What was the sampling frequency (Hz)? ');
     Tsample = 1/sampfreq;
else
     [Wavedata,Tsample] = wvinput(Pick);
end

%---------------------------------------------------------------------------
% Decomposition algorithm    h(0)=0.5, h(1)=0.5, g(0)=0.5, g(1)=-0.5

datlngth = length(Wavedata);
numrows = ceil(log(datlngth)/log(2));       % find number of resolution levels
numpts = 2^numrows; Newdata = zeros(1,numpts);
Newdata(1:datlngth) = Wavedata;
d = zeros(numrows,numpts); c = zeros(numrows+1,numpts);
detail = zeros(numrows,numpts); approx = zeros(numrows,numpts);
c(numrows+1,:) = Newdata;
top = numpts;
const = 1/sqrt(2);           % normalization constant * coefficient magnitude
ctr = numrows;
while ctr >= 1,
     n = numrows-ctr+1;
     shift = 2^n;
     shift1 = shift/2;
     top = ceil(top/2);                 % number of points at this level
     for k = 1:top                          % get "c" and "d" coefficients
        jump =shift*k;
        index1 = jump-(shift-1);
        index2 = jump-(shift1-1);
        firsttrm = c(ctr+1,index1);
        if index2 <= numpts,
           secndtrm = c(ctr+1,index2);
        else
           secndtrm = 0;                    % zero padding
        end
        d(ctr,index1) = firsttrm - secndtrm;
        c(ctr,index1) = firsttrm + secndtrm;
        for j = 0:(shift-1)                 % build matrices for display purposes
          if j < shift1,
            detail(ctr,index1+j) = d(ctr,index1);
          else
            detail(ctr,index1+j) = -d(ctr,index1);
          end
          approx(ctr,index1+j) = c(ctr,index1);
        end
      end
      d(ctr,:) = const*d(ctr,:);
      c(ctr,:) = const*c(ctr,:);
      normlize = sqrt(2*shift);            % normalization constant for this level
      detail(ctr,:) = detail(ctr,:)/normlize;
      approx(ctr,:) = approx(ctr,:)/normlize;
      ctr = ctr-1;
end

%--------------------------------------------------------------------------------
% Plot "c" and "d" coefficients by resolution level

indxto0 = [0.5:numpts-0.5];
plotmin = 1.2*min(Wavedata); plotmax = 1.2*max(Wavedata);
if ((plotmin==0)&(plotmax==0)), plotmax = 0.5; end
if plotmin > 0, plotmin = 0; end
if plotmax < 0, plotmax = 0; end
v=[0,numpts,plotmin,plotmax];
axis(v);
bar(indxto0,c(numrows+1,:))
title('Approximation function to Input Signal at resolution level 0')
xlabel(['Sample number "n"  (Time = n * ',num2str(Tsample),' sec)'])
pause
clg

outptdet = zeros(1,1); outptapp = zeros(1,1);
for k = numrows:-1:1
     level = k-numrows-1;
     step = (2^(-level))/2;
     for j = 1:numpts/step
        outptdet(j) = detail(k,(j-1)*step+1);
     end
     stepa = 2*step;
     indxto0a = [stepa/2:stepa:numpts-stepa/2];
     for j = 1:numpts/stepa
        outptapp(j) = approx(k,(j-1)*stepa+1);
     end
     axis(v);
     if k > 1,
        subplot(211),bar(indxto0a,outptapp)
      else
         indxto0a = [0:numpts];
         approx1 = approx(1,:);approx1(numpts+1) = approx(1,numpts);
         subplot(211),plot(indxto0a,approx1)
      end
      title(['Approximation function at resolution level ',num2str(level)])
      xlabel(['Sample number "n"  (Number of samples = ',num2str(numpts),')'])
      axis(v);
      subplot(212),bar(indxto0,outptdet)
      title(['Detail function at resolution level ',num2str(level)])
      xlabel(['Sample number "n"  (Time = n * ',num2str(Tsample),' sec)'])
      pause
      clg
      clear outptapp outptdet indxto0
      indxto0 = indxto0a;
      clear indxto0a
end
subplot

%-------------------------------------------------------------------------------
% Reconstruction algorithm

clc
recmpout = zeros(1,1);
disp(b)
N2 = [' Level-by-level Recomposition can be observed if desired.  '
      ' The Recomposition algorithm starts with the lowest level  '
      ' Approximation Function and successively adds in the Detail'
      ' Function to obtain the next higher level Approximation.   '];
disp(N2)
disp(b)
ckalgthm = input('Do you want to see the Recomposition (Y or N)? ','s');
if ckalgthm == 'Y',
  level = -numrows;
  axis(v);
  plot(indxto0,approx1)
  title(['Level ',num2str(level),' Recomposition'])
  xlabel('Sample number')
  pause
  clg
  recomp = approx(1,:);
  for k = 1:numrows
     level = level+1;
     recomp = recomp+detail(k,:);
     step = 2^(-level);
     indxto0 = [step/2:step:numpts-step/2];
     for j = 1:numpts/step;
        recmpout(j) = recomp((j-1)*step+1);
     end
     axis(v);
     bar(indxto0,recmpout)
     title(['Level ',num2str(level),' Recomposition'])
     xlabel('Sample number')
     pause
     clg
  end
  bar(indxto0,Newdata-recmpout)
  title('Recomposition Error')
  xlabel('Sample number')
  pause
  clg
end

%-------------------------------------------------------------------------------
% Coefficient energies are used as a measure of response

c = c.^2;
d = d.^2;

Enrgytot = sum(c(numrows+1,:));
Enrgycro = zeros(1,numrows+1); Enrgydro = zeros(1,numrows+1);
for j = 1:numrows                        % find energy in each resolution level
   Enrgycro(j) = (sum(c(j,:)))/Enrgytot;
   Enrgydro(j) = (sum(d(j,:)))/Enrgytot;
end
if Enrgytot == 0,
  Enrgycro(numrows+1) = 0;
else
  Enrgycro(numrows+1) = 1;
end
Enrgydro(numrows+1) = 0;
lvl = [-(numrows):1:0];
v = [-(numrows+1),1,0,1.2];
axis(v);
subplot(211),bar(lvl,Enrgycro)
title('Normalized Energy of Approximation Function vs. Resolution Level')
xlabel('Resolution Level')
axis(v);
subplot(212),bar(lvl,Enrgydro)
title('Normalized Energy of Detail Function vs. Resolution level')
xlabel('Resolution Level')
pause
subplot
clc

%-------------------------------------------------------------------------------
% Display mesh and contour plots of coefficient energy with optional "zoom"
% capability to aid analysis

strtsamp = 1; endsamp = numpts; strtrow = 1; endrow = numrows; zoom = 1;
highres = -1; lowres = -numrows;
while zoom == 1,
  rangea = [-highres-1:-lowres]; ranged = [-highres:-lowres];
  indxto0 = [strtsamp-1:endsamp-1];
  mesh(c(strtrow:endrow+1,strtsamp:endsamp))
  title('Energy in Approximation Coefficients using Haar basis')
  pause
  contour(c(strtrow:endrow+1,strtsamp:endsamp),10,indxto0,rangea)
  title('Contour Map of Approximation Coefficient Energy Distribution')
  xlabel(['Sample number "n"  (Time = n * ',num2str(Tsample),' sec)'])
  ylabel('Decomposition Number  (= -Resolution Level)')
  pause
  mesh(d(strtrow:endrow,strtsamp:endsamp))
  title('Energy in Difference Coefficients using Haar basis')
  pause
  contour(d(strtrow:endrow,strtsamp:endsamp),10,indxto0,ranged)
  title('Contour Map of Difference Coefficient Energy Distribution')
  xlabel(['Sample number "n"  (Time = n * ',num2str(Tsample),' sec)'])
  ylabel('Decomposition Number  (= -Resolution Level)')
  pause
  clc
  showmore = input('Would you like to "zoom" in on a section (Y or N)? ','s');
  if showmore == 'Y',
    disp(b)
    disp(' You will set the display sample number and resolution level limits.')
    disp(b)
    disp(['   Sample range         0:',num2str(numpts-1)])
    disp(['   Resolution range    -1:',num2str(-numrows)])
    revu = input('Need to see the original contour plots again (Y or N)? ','s');
    if revu == 'Y',
      contour(c,10,[0:numpts-1],[0:numrows])
      title('Contour Map of Approximation Coefficient Energy Distribution')
      xlabel(['Sample number "n"    (Time = n * ',num2str(Tsample),' sec)'])
      ylabel('Decomposition Number  (= -Resolution Level)')
      pause
      contour(d,10,[0:numpts-1],[1:numrows])
      title('Contour Map of Difference Coefficient Energy Distribution')
      xlabel(['Sample number "n"  (Time = n * ',num2str(Tsample),' sec)'])
      ylabel('Decomposition Number  (= -Resolution Level)')
      pause
    end
    strtsamp = input('Sample start point? ')+1;
    endsamp = input('Sample endpoint? ')+1;
    highres = input('Highest resolution level (least negative)? ');
    lowres = input('Lowest resolution level (most negative)? ');
    endrow = numrows+1+highres;
    strtrow = numrows+1+lowres;
  else
    zoom = 0;
  end
  clg
end
clc
```

APPENDIX D

PROGRAM FOR SINGLE PHASE ANALYSIS USING DAUBECHIES OR USER WAVELET

```
% This program performs single phase decomposition of the input waveform
% with either a Daubechies wavelet of user-defined order or with a wavelet
% provided by the user. The input data sequence will be zero-padded to
% accomodate the calculation of all possible non-zero coefficients and
% approximation ("c") and detail ("d") coefficients may be computed for
% sample numbers beyond the data endpoint. The user should consider these
% "edge effects" in his analysis.


%--------------------------------------------------------------------------
% Input the "h" coefficients

clc
b = [' '];
disp(b)
D1 = [' Would you like to use:                                          '
      '   1. your own decomposition coefficients (use wvhaar.m if you desire'
      '                                           to use the Haar basis set)'
      '   2. those associated with Daubechies compactly supported wavelets? '];
disp(D1)
hpick = input('Answer 1 or 2: ');
disp(b)

if hpick == 1,
  disp(' Input as a row vector with sum of coefficients normalized to "1".')
  hcoeff = input('What is the name of your decomposition coefficient vector? ');
  choice = 1;
else
  choice = input('What is the desired wavelet order (2-10 are available)? ');
  [hcoeff] = daubdata(choice);
end

hlength = length(hcoeff);

if hlength == 0
  disp(b)
  disp('ERROR: Wavelet order selected is outside allowable range.')
  break
end

%--------------------------------------------------------------------------
% Determine if user wants to see plots of basis functions

ckplt = input('Do you want to see the Scaling Function/Wavelet (Y or N)? ','s');
if ckplt == 'Y'
  basisplt
end

%--------------------------------------------------------------------------
% Get the input data vector

clc
disp(b)
Q1 = ['Do you wish to use:                                  '
      ' 1.  your own MATLAB formatted row vector, or         '
      ' 2.  a program generated vector from the following menu?'
      '       - Sine wave                                    '
      '       - Pseudo-noise (PN) sequence                   '
      '       - Sine wave modulated by PN sequence           '];
```

```
disp(Q1)
Pick = input('Answer 1 or 2: ');

if Pick == 1,
  % Read in user's input vector
  Wavedata = input('What is the name of your input vector? ');
  sampfreq = input('What was the sampling frequency (Hz)? ');
  Tsample = 1/sampfreq;
else
  [Wavedata,Tsample] = wvinput(Pick);
end

%--------------------------------------------------------------------------------
% "g" coefficients are derived from the "h" coefficients

hcoeff = sqrt(2)*hcoeff;
gcoeff = fliplr(hcoeff);
for j = 2:2:hlength
   gcoeff(j) = -gcoeff(j);
end

%--------------------------------------------------------------------------------
% Decomposition algorithm

datlngth = length(Wavedata);
numrows = ceil(log(datlngth)/log(2));       % find number of resolution levels

Lastpts = zeros(1,numrows+1); Shift = ones(1,numrows+1);
Lastpts(numrows+1) = datlngth; lastpt = datlngth;
for k = numrows:-1:1,      % find number of points required at each level
  evnorodd = rem(lastpt,2);
  if evnorodd == 0,
    lastpt = lastpt/2+ceil((hlength-2)/2);
  else
    lastpt = (lastpt-1)/2+fix(hlength/2);
  end
  Lastpts(k) = lastpt;
  Shift(k) = 2^(numrows-k+1);
end

hhalf = ceil(hlength/2);
dprime = zeros(numrows,Lastpts(numrows)+hhalf);
cprime = zeros(numrows,Lastpts(numrows)+hhalf);

clastvct = zeros(1,datlngth+2*hlength-3);
clastvct(hlength-1:datlngth+hlength-2) = Wavedata;
ctr = numrows;
while ctr > 0,
  lastpt = Lastpts(ctr);
  nwcvctr = zeros(1,lastpt);
  nwdvctr = zeros(1,lastpt);
  for k = 1:lastpt            % coefficients calculated, by resolution level,
     startpt = 2*k-1;         %  using convolution operation with shifts of 2
     endpt = 2*k+hlength-2;                               % (downsampling)
     nwcvctr(k) = hcoeff*clastvct(startpt:endpt)';
     nwdvctr(k) = gcoeff*clastvct(startpt:endpt)';
  end
  clastvct = [zeros(1,hlength-2) nwcvctr zeros(1,hlength-1)];
  cprime(ctr,1:lastpt) = nwcvctr;
  dprime(ctr,1:lastpt) = nwdvctr;
  ctr = ctr-1;
end

%-------------------------------------------------------------------------------
% Coefficient energies are used as a measure of response

cenergy = cprime.^2;
denergy = dprime.^2;

N1 = ['    The next plot will show the normalized energy distributions of the'
      ' approximation and detail coefficients as a function of resolution    '
      ' level.  Based on this information, you will select the minimum       '
      ' resolution level to be displayed on all future plots.                '
      '                                                                      '
      '    "Edge effects" can cause coefficients to be generated for sample  '
      ' numbers beyond the endpoints of your data sequence (the lower the    '
      ' resolution level, the greater the sample number required; therefore, '
      ' zero padding of the original data sequence is required).             '
      '                                                                      '
      '    This program translates the indices of the "h" and "g" vectors to '
      ' [ -(length of vector-2),1 ].  As a result, coefficients will exist   '
      ' for sample numbers beyond the last data point (n=N), but will not    '
      ' exist for sample numbers prior to the first data point (n=0).        '
      ' These coefficients are necessary for completeness, but may be        '
      ' difficult to interpret because the number of original data points    '
      ' used in their calculation can be a small percentage of the total     '
      ' considered (the rest are "0").  Similarly, coefficients calculated   '
      ' for sample numbers near the beginning of the sequence will also be   '
      ' affected by leading zeros affixed to the data sequence.              '
      '                                                                      '
      '    Limiting the sample numbers required for display may assist inter-'
      ' pretation; select the lowest resolution level containing significant '
      ' energy for this effect.  <Return>                                    '];

clc
disp(b)
disp(N1)
pause

originlE = zeros(1,datlngth); originlE = Wavedata.^2;
Enrgytot = sum(originlE);
Enrgycro = zeros(1,numrows+1); Enrgydro = zeros(1,numrows+1);
ckerr = zeros(1,numrows);
for k = 1:numrows                        % find energy in each resolution level
   Enrgycro(k) = sum(cenergy(k,:));
   Enrgydro(k) = sum(denergy(k,:));
end
if Enrgytot == 0,
  Enrgycro(numrows+1) = 0;
else
  Enrgycro(numrows+1) = Enrgytot;
  Enrgycro = Enrgycro/Enrgytot; Enrgydro = Enrgydro/Enrgytot;      % normalize
end
Enrgydro(numrows+1) = 0;

ckenergy = Enrgycro(1:numrows)+Enrgydro(1:numrows);
% Energy balance check.  The total energy at any level should equal the
% energy in the "c" coefficients in the next higher level.
for k = 1:numrows
```

```
  ckerr(k) = abs(Enrgycro(k+1)-ckenergy(k));
  if ckenergy(k) > 10^(-6)*Enrgytot,
   errenrgy = ckerr(k)/max(Enrgycro(k+1),ckenergy(k));
   if errenrgy > 10^(-6),
    clc
    disp(b)
    disp(' ERROR: Energy check between levels does not balance.')
    disp('        If you entered your own "h" vector, recheck its validity.')
    break
   end
  end
end

lvl = [-(numrows):1:0];
v = [-(numrows+1),1,0,1.2];
axis(v);
subplot(211),bar(lvl,Enrgycro)
title('Normalized Energy of Approximation Coefficients')
xlabel('Resolution Level')
axis(v);
subplot(212),bar(lvl,Enrgydro)
title('Normalized Energy of Detail Coefficients')
xlabel('Resolution Level')
subplot
pause

%---------------------------------------------------------------------------
% Output the number of points required to properly display coefficients at
% each resolution level (determined by "edge effects")

clc
disp(b)
N2 = [' Listed below are the number of samples required to properly display'
      ' each Resolution Level.                                             '];
disp(N2)
disp(b)
disp(' Resolution Level   Number of samples')
for k = 1:numrows
   vctrindx = numrows-k+1;
   numsamps = (Lastpts(vctrindx)-1)*Shift(vctrindx)+1;
   disp(['        ',num2str(-k),'                  ',num2str(numsamps)])
end
nmbrlvl = -input('What is the lowest Resolution Level you desire to see? ');

%---------------------------------------------------------------------------
% Plot "c" and "d" coefficients by resolution level

indxto0 = [0:datlngth-1];
plotmin = 1.2*min(Wavedata); plotmax = 1.2*max(Wavedata);
if ((plotmin==0)&(plotmax==0)), plotmax = 0.5; end
if plotmin > 0, plotmin = 0; end
if plotmax < 0, plotmax = 0; end
v = [0,datlngth-1,plotmin,plotmax];
axis(v);
plot(indxto0,Wavedata,'*')
title('Approximation Coefficients at resolution 0')
xlabel(['Sample number "n"   (Time = n * ',num2str(Tsample),' sec)'])
pause

lastrow = numrows-nmbrlvl+1;
```

```
clmnsize = (Lastpts(lastrow)-1)*Shift(lastrow)+1;
c = zeros(nmbrlvl+1,clmnsize);
c(nmbrlvl+1,1:datlngth) = originlE;
d = zeros(nmbrlvl,clmnsize);
ctr = numrows; ctrl = nmbrlvl;
while ctr >= lastrow,
   sampval = zeros(1,Lastpts(ctr));
   for k = 1:Lastpts(ctr)
      index = Shift(ctr)*(k-1)+1;
      c(ctrl,index) = cenergy(ctr,k);
      d(ctrl,index) = denergy(ctr,k);
      sampval(k) = index-1;
   end

   plotmin = 1.2*min(min(cprime(ctr,:)),min(dprime(ctr,:)));
   plotmax = 1.2*max(max(cprime(ctr,:)),max(dprime(ctr,:)));
   if ((plotmin==0)&(plotmax==0)), plotmax = 0.5; end
   if plotmin > 0, plotmin = 0; end
   if plotmax < 0, plotmax = 0; end
   v = [0,max(sampval),plotmin,plotmax];
   n = numrows-ctr+1;
   axis(v);
   subplot(211),plot(sampval,cprime(ctr,1:Lastpts(ctr)),'*')
   title(['Approximation Coefficients at resolution level ',num2str(-n)])
   xlabel(['Sample number "n"   (Last sample = ',num2str(max(sampval)),')'])
   subplot(212),plot(sampval,dprime(ctr,1:Lastpts(ctr)),'*')
   title(['Detail Coefficients at resolution level ',num2str(-n)])
   xlabel(['Sample number "n"   (Time = n * ',num2str(Tsample),' sec)'])
   pause
   clg
   ctr = ctr-1; ctrl = ctrl-1;
end
subplot

%-------------------------------------------------------------------------------
% Reconstruction algorithm

N3 = [' Recomposition of the original data sequence can be checked if        '
      ' desired.  The recomposition algorithm starts with the approximation'
     .' coefficients at the lowest resolution level previously selected      '
      ' and successively "adds" in the detail coefficients of each level    '
      ' until the original resolution of the input sequence is obtained.    '];
clc
disp(b)
disp(N3)
ckalgthm = input('Do you want to see the Recomposition (Y or N)? ','s');
if ckalgthm == 'Y',
  hrecomp = fliplr(hcoeff); grecomp = fliplr(gcoeff);          % invert in time
  cstart = cprime(lastrow,:);
  ctr = lastrow;
  while ctr <= numrows;
    lastpt = Lastpts(ctr);
    ckvctr = zeros(2,lastpt);
    for k = 1:fix(hlength/2)              % compute higher level "c" coefficients
       ckvctr(1,:) = ckvctr(1,:)+hrecomp(2*k)*cstart(k:lastpt+k-1)+..
                         grecomp(2*k)*dprime(ctr,k:lastpt+k-1);
       ckvctr(2,:) = ckvctr(2,:)+hrecomp(2*k-1)*cstart(k:lastpt+k-1)+..
                         grecomp(2*k-1)*dprime(ctr,k:lastpt+k-1);
    end
    if rem(hlength,2) ~= 0,
        ckvctr(2,:)=ckvctr(2,:)+hrecomp(hlength)*cstart(hhalf:lastpt+hhalf-1)+..
                       grecomp(hlength)*dprime(hhalf:lastpt+hhalf-1);
      end
      ckvctr = reshape(ckvctr,1,2*Lastpts(ctr));
      cstart = ckvctr;
      lastsamp = (Lastpts(ctr+1)-1)*Shift(ctr+1);
      indxto0 = [0:Shift(ctr+1):lastsamp];
      plotmin = 1.2*min(ckvctr); plotmax = 1.2*max(ckvctr);
      if ((plotmin==0)&(plotmax==0)), plotmax = 0.5; end
      if plotmin > 0, plotmin = 0; end
      if plotmax < 0, plotmax = 0; end
      v = [0,lastsamp,plotmin,plotmax];
      axis(v);
      plot(indxto0,ckvctr(1:Lastpts(ctr+1)),'*');
      title(['Level ',num2str(ctr-numrows),' Recomposition'])
      xlabel(['Sample number "n"   (Last sample = ',num2str(lastsamp),')'])
      pause
      clg
      ctr = ctr+1;
    end
    axis;
    plot(indxto0,ckvctr(1:datlngth)-Wavedata,'*')
    title('Recomposition Error')
    xlabel(['Sample number "n"    (Last sample = ',num2str(lastsamp),')'])
    pause
    clg
  else
    axis;
  end

  %-----------------------------------------------------------------------------
  % Display mesh and contour plots of coefficient energy with optional "zoom"
  % capability to aid analysis

  strtsamp = 1; endsamp = clmnsize; strtrow = 1; endrow = nmbrlvl; zoom = 1;
  highres = -1; lowres = -nmbrlvl;
  while zoom == 1,
    rangea = [-highres-1:-lowres]; ranged = [-highres:-lowres];
    indxto0 = [strtsamp-1:endsamp-1];
    mesh(c(strtrow:endrow+1,strtsamp:endsamp))
    title('Energy in Approximation Coefficients')
    if choice ~= 1,
      xlabel(['using Daubechies Wavelet of order ',num2str(choice)])
    end
    pause
    contour(c(strtrow:endrow+1,strtsamp:endsamp),10,indxto0,rangea)
    title('Contour Map of Approximation Coefficient Energy Distribution')
    xlabel(['Sample number "n"   (Time = n * ',num2str(Tsample),' sec)'])
    ylabel('Decomposition Number  (= -Resolution Level)')
    pause
    mesh(d(strtrow:endrow,strtsamp:endsamp))
    title('Energy in Detail Coefficients')
    if choice ~= 1,
      xlabel(['using Daubechies Wavelet of order ',num2str(choice)])
    end
    pause
    contour(d(strtrow:endrow,strtsamp:endsamp),10,indxto0,ranged)
    title('Contour Map of Detail Coefficient Energy Distribution')
    xlabel(['Sample number "n"   (Time = n * ',num2str(Tsample),' sec)'])
    ylabel('Decomposition Number  (= -Resolution Level)')
    pause
    clc
    showmore = input('Would you like to "zoom" in on a section (Y or N)? ','s');
    if showmore == 'Y',
      disp(b)
      disp(' You will set the display sample number and resolution level limits.')
      disp(b)
      disp(['   Sample range         0:',num2str(clmnsize-1)])
      disp(['   Resolution range    -1:',num2str(-nmbrlvl)])
      revu = input('Need to see the original contour plots again (Y or N)? ','s');
      if revu == 'Y',
        contour(c,10,[0:clmnsize-1],[0:nmbrlvl])
        title('Contour Map of Approximation Coefficient Energy Distribution')
        xlabel(['Sample number "n"   (Time = n * ',num2str(Tsample),' sec)'])
        ylabel('Decomposition Number  (= -Resolution Level)')
        pause
        contour(d,10,[0:clmnsize-1],[1:nmbrlvl])
        title('Contour Map of Detail Coefficient Energy Distribution')
        xlabel(['Sample number "n"   (Time = n * ',num2str(Tsample),' sec)'])
        ylabel('Decomposition Number  (= -Resolution Level)')
        pause
      end
      strtsamp = input('Sample start point? ')+1;
      endsamp = input('Sample endpoint? ')+1;
      highres = input('Highest resolution level (least negative)? ');
      lowres = input('Lowest resolution level (most negative)? ');
      endrow = nmbrlvl+1+highres;
      strtrow = nmbrlvl+1+lowres;
    else
      zoom = 0;
    end
    clg
  end
  clc
```

APPENDIX E

PROGRAM FOR MULTIPHASE ANALYSIS WITH ANY WAVELET

```
%   This program will decompose the input waveform with a Haar wavelet, a
%  Daubechies compactly supported wavelet of user-defined order, or with a
%  wavelet provided by the user.
%   A "sliding" data window is used to maximize signal analysis features
%  of the decomposition by negating the effects of random signal time of
%  arrival.  Approximation and detail coefficients may therefore be
%  computed at all resolution levels for every "n" (sample number).
%  Because of finite input data length, however, some of the coefficients
%  will suffer from "edge effects", i.e., the sample values beyond the
%  end of the data sequence are treated as "0"s.
%   Output graphs consist of "3-D" MATLAB mesh plots and "waterfall"
%  contour maps of the approximation and detail coefficients, in addition
%  to plots of the coefficients at each level (if desired).

%--------------------------------------------------------------------------------
% Input the "h" coefficients

clc
b = [' '];
disp(b)
D1 = [' Would you like to use:                              '
      '   1. your own set of decomposition coefficients, or '
      '   2. those associated with the Haar wavelet, or     '
      '   3. those derived from Daubechies group of wavelets?'];
disp(D1)
hpick = input('Answer 1, 2, or 3: ');
disp(b)
disp(b)

if hpick == 1,
  disp(' Note: Sum of coefficients must be normalized to equal 1.')
  disp(b)
  hcoeff=input('What is the name of your decomposition coefficient vector? ');
  choice = 1;
elseif hpick == 2,
  hcoeff = [0.5 0.5];
  choice = 1;
else
  choice = input('What is the desired wavelet order (2-10 are available)? ');
  [hcoeff] = daubdata(choice);
end

hlength = length(hcoeff);

if hlength == 0
  disp(b)
  disp('ERROR: Wavelet order selected is outside allowable range.')
  break
end

%--------------------------------------------------------------------------------
% Determine if user wants to see plots of the basis functions

picture=input('Do you want to see the Scaling Function/Wavelet (Y or N)? ','s');
if picture == 'Y'
  basisplt
end

%--------------------------------------------------------------------------------
```

```
% Get the input data vector

clc
disp(b)
Q1 = ['Do you wish to use:                                    '
      '  1.  your own MATLAB formatted row vector, or          '
      '  2.  a program generated vector from the following menu?'
      '        - Sine wave                                     '
      '        - Pseudo-noise (PN) sequence                    '
      '        - Sine wave modulated by PN sequence            '];

disp(Q1)
Pick = input('Answer 1 or 2: ');

if Pick == 1,
  % Read in user's input vector
  Wavedata = input('What is the name of your input vector? ');
  sampfreq = input('What was the sampling frequency (Hz)? ');
  Tsample = 1/sampfreq;
else
  [Wavedata,Tsample] = wvinput(Pick);
end

%--------------------------------------------------------------------------------
% "g" coefficients are derived from the "h" coefficients

hcoeff = sqrt(2)*hcoeff;
gcoeff = fliplr(hcoeff);
for j = 2:2:hlength
   gcoeff(j) = -gcoeff(j);
end

%--------------------------------------------------------------------------------
% Decomposition algorithm

datlngth = length(Wavedata);
numrows = ceil(log(datlngth)/log(2));    % determine number of resolution levels
Lastpts = zeros(1,numrows+1);
for k = 1:numrows             % determine number of points required for each level
   Lastpts(numrows-k+1) = datlngth+(2^(k)-1)*(hlength-1);
end
Lastpts(numrows+1) = datlngth; numpts = Lastpts(1);
d = zeros(numrows,numpts); c = zeros(numrows+1,numpts);
c(numrows+1,1:datlngth) = Wavedata; cworkvct = Wavedata;
ctr = numrows;
while ctr >= 1,
  n = numrows-ctr;
  shift = 2^n; oldendpt = Lastpts(ctr+1); newendpt = Lastpts(ctr);
  dnewrow = zeros(1,newendpt); cnewrow = zeros(1,newendpt);
  for k = 0:hlength-1          % coefficient vectors are calculated for each
                               % resolution level by adding shifted, weighted
                               % versions of the next higher level "c" vector
                               %    (same effect as direct convolution)
     shift1 = k*shift;
     cnewrow(1+shift1:oldendpt+shift1) = cnewrow(1+shift1:oldendpt+shift1)+..
                                         hcoeff(hlength-k)*cworkvct;
     dnewrow(1+shift1:oldendpt+shift1) = dnewrow(1+shift1:oldendpt+shift1)+..
                                         gcoeff(hlength-k)*cworkvct;
  end
  d(ctr,1:newendpt) = dnewrow;
  c(ctr,1:newendpt) = cnewrow;
  cworkvct = cnewrow;
  ctr = ctr-1;
end

%-------------------------------------------------------------------------------
% User determines output format

clc
disp(b)
D3 = [' Do you want to see:                                                   '
      '    1. separate plots of the coefficients at each Resolution Level'
      '        in addition to the "3-D" and contour plots, or               '
      '    2. only the "3-D" and contour plots?                              '];
disp(D3)
pickplot = input('Answer 1 or 2: ');

%-------------------------------------------------------------------------------
% Plot "c" and "d" coefficients by resolution level, if desired.

if pickplot == 1,
  indxto0 = [0:datlngth-1];
  plotmin = 1.2*min(Wavedata); plotmax = 1.2*max(Wavedata);
  if ((plotmin==0)&(plotmax==0)), plotmax = 0.5; end
  if plotmin > 0, plotmin = 0; end
  if plotmax < 0, plotmax = 0; end
  v = [0,datlngth-1,plotmin,plotmax];
  axis(v);
  plot(indxto0,Wavedata,'*')
  title('Approximation Coefficients at resolution 0')
  xlabel(['Sample number "n"   (Time = n * ',num2str(Tsample),' sec)'])
  pause

  for k = numrows:-1:1
    n = numrows-k+1;
    endpt = Lastpts(k);
    indxto0 = [0:endpt-1];
    plotmin = 1.2*min(min(c(k,:)),min(d(k,:)));
    plotmax = 1.2*max(max(c(k,:)),max(d(k,:)));
    if ((plotmin==0)&(plotmax==0)), plotmax = 0.5; end
    if plotmin > 0, plotmin = 0; end
    if plotmax < 0, plotmax = 0; end
    v = [0,endpt-1,plotmin,plotmax];
    axis(v);
    subplot(211),plot(indxto0,c(k,1:endpt),'*')
    title(['Approximation Coefficients at resolution level ',num2str(-n)])
    xlabel(['Sample number "n"   (Number of points = ',num2str(endpt),')'])
    subplot(212),plot(indxto0,d(k,1:endpt),'*')
    title(['Detail Coefficients at resolution level ',num2str(-n)])
    xlabel(['Sample number "n"   (Time = n * ',num2str(Tsample),' sec)'])
    pause
    clg
  end
  subplot
  axis;
end

%-------------------------------------------------------------------------------
% Coefficient energies are used as a measure of response.  Display mesh and
% contour plots of coefficient energy with optimal "zoom" capability to aid
% signal analysis

c = c.^2;
d = d.^2;

strtsamp = 1; endsamp = numpts; strtrow = 1; endrow = numrows; zoom = 1;
highres = -1; lowres = -numrows;
while zoom == 1,
  rangea = [-highres-1:-lowres]; ranged = [-highres:-lowres];
  indxto0 = [strtsamp-1:endsamp-1];
  mesh(c(strtrow:endrow+1,strtsamp:endsamp))
  title('Energy in Approximation Coefficients')
  if choice ~= 1,
    xlabel(['using Daubechies Wavelet of order ',num2str(choice)])
  end
  pause
  contour(c(strtrow:endrow+1,strtsamp:endsamp),10,indxto0,rangea)
  title('Contour Map of Approximation Coefficient Energy Distribution')
  xlabel(['Sample number "n"    (Time = n * ',num2str(Tsample),' sec)'])
  ylabel('Decomposition Number  (= - Resolution Level)')
  pause
  mesh(d(strtrow:endrow,strtsamp:endsamp))
  title('Energy in Detail Coefficients')
  if choice ~= 1,
  xlabel(['using Daubechies Wavelet of order ',num2str(choice)])
  end
  pause
  contour(d(strtrow:endrow,strtsamp:endsamp),10,indxto0,ranged)
  title('Contour Map of Detail Coefficient Energy Distribution')
  xlabel(['Sample number "n"    (Time = n * ',num2str(Tsample),' sec)'])
  ylabel('Decomposition Number  (= - Resolution Level)')
  pause
  clc
  showmore = input('Would you like to "zoom" in on a section (Y or N)? ','s');
  if showmore == 'Y',
    disp(b)
    disp(' You will set the display sample number and resolution level limits.')
    disp(b)
    disp(['   Sample range         0:',num2str(numpts-1)])
    disp(['   Resolution range    -1:',num2str(-numrows)])
    revu = input('Need to see the original contour plots again (Y or N)? ','s');
    if revu == 'Y',
      contour(c,10,[0:numpts-1],[0:numrows])
      title('Contour Map of Approximation Coefficient Energy Distribution')
      xlabel(['Sample number "n"    (Time = n * ',num2str(Tsample),' sec)'])
      ylabel('Decomposition Number  (= - Resolution Level)')
      pause
      contour(d,10,[0:numpts-1],[1:numrows])
      title('Contour Map of Detail Coefficient Energy Distribution')
      xlabel(['Sample number "n"    (Time = n * ',num2str(Tsample),' sec)'])
      ylabel('Decomposition Number  (= - Resolution Level)')
      pause
    end
    strtsamp = input('Sample start point? ')+1;
    endsamp = input('Sample endpoint? ')+1;
    highres = input('Highest resolution level (least negative)? ');
    lowres = input('Lowest resolution level (most negative)? ');
    endrow = numrows+1+highres;
    strtrow = numrows+1+lowres;
  else
```

```
        zoom = 0;
      end
      clg
    end
    clc
```

APPENDIX F

PROGRAM FOR GENERATING ITERATIVE PLOTS OF WAVELETS

```
% This program creates and plots the desired iterative approximation
% to the scaling function and the associated wavelet as determined by
% the input "h" vector.  The construction is based on the "graphical"
% recursion method.


hcoefnew = 2*hcoeff;
m = length(hcoefnew);

%------------------------------------------------------------------------
% The "g" vector is determined from the "h" vector.

gcoefnew = fliplr(hcoefnew);
for j = 2:2:m
  gcoefnew(j) = -gcoefnew(j);
end

%------------------------------------------------------------------------
% Recursively build basis functions

numbrits = input('How many iterations shall we run for the approximation? ');

size = 1;
hpast = ones(1,m);
newsize = m;

for i = 1:numbrits

 % Build scaling function approximation

 hnew = zeros(1,newsize);
 for k = 0:size-1
    hnew(2*k+1:2*k+m)=hnew(2*k+1:2*k+m)+hpast(k+1)*hcoefnew;
 end

 % Get wavelet from scaling function approximation and also build time vector

 wv = zeros(1,2*newsize);wvlet = zeros(1,newsize);timevctr = zeros(1,newsize);
 shift = newsize/(m-1);
 for k = 0:m-1
    shift1 = round(k*shift+1);
    shift2 = round(newsize+k*shift);
    wv(shift1:shift2) = wv(shift1:shift2)+gcoefnew(k+1)*hnew;
 end
 for k = 1:newsize
    wvlet(k) = wv(2*k-1);
    timevctr(k) = k-1;
 end

 interval = (1/2)^i;
 timevctr = interval*timevctr;          % scale time axis
 s = interval*newsize;

 % Plot basis functions and check calculations with areas and inner products

 if i <= 5,
  plot(timevctr,hnew,'+',timevctr,wvlet,'*')
  xlabel(['Support = ',num2str(s),'   (Scaling Function: ++++, Wavelet: ****)'])
 else
```

```
  plot(timevctr,hnew,'-',timevctr,wvlet,'--')
  xlabel(['Support = ',num2str(s),'   (Scaling Function: line, Wavelet: dash)'])
 end
 title(['Approximations for Iteration number ',num2str(i)])
 pause

 hpast = hnew;
 size = newsize;
 newsize = 2*size+m-2;

 phisum = sum(hpast);
 wvsum = sum(wvlet);
 sp(i) = phisum*interval;    % find area under scaling function
 sw(i) = wvsum*interval;     % find area under wavelet
 ip(i)= hpast*wvlet';        % find scaling function/wavelet inner product

end

%------------------------------------------------------------------------------
% Output calculation checks

disp(' ')
disp('   Area under     Area under   Inner')
disp('Scaling Function   Wavelet    Product')
disp(' ')
for i = 1:numbrits
disp(['       ',num2str(sp(i)),'         ',num2str(sw(i)),'   ',num2str(ip(i))])
end
disp(' ')
disp('<Return>')
pause

clear hcoefnew gcoefnew hpast timevctr wv
```

APPENDIX G

FUNCTION FOR CONSTRUCTION OF INPUT DATA

```
 function [Data,Tsample] = wvinput(x)

%  This function supports the various multiresolution programs by building
%  the input test vector desired by the user from the following choices:
%  Sine wave, Pseudo-noise sequence, and Sine wave modulated by Pseudo-
%  noise sequence with optional additive Gaussian noise.

%-------------------------------------------------------------------------------

%  Determine desired input signal type.

b = [' '];
Q2 = [' Select the desired type of waveform:   '
      '   1. Sine wave                         '
      '   2. Pseudo-noise (PN) sequence        '
      '   3. Sine wave modulated by PN sequence'];
disp(b)
disp(Q2)
Make = input('Answer 1, 2 or 3: ');

%-------------------------------------------------------------------------------

% Generate the desired sinusoidal signal.

if Make ~= 2,
   clc
   % Determine desired sine wave characteristics.
   N2 = [' You will determine the following Sine wave parameters:'
         '   - frequency                                          '
         '   - phase                                              '
         '   - amplitude                                          '
         '   - number of samples                                  '
         '   - sampling frequency or data length                  '];
   disp(b)
   disp(N2)
   fc = input('Sinewave frequency ("fc") in Hertz? ');
   theta = input('Phase (degrees)? ');
   A = input('Amplitude? ');
   N = input('Number of Samples ("N", with N = a power of 2)? ');
   Q3 = [' Do you wish to set:            '
         '   1. the sampling frequency, or'
         '   2. the data length?          '];
   disp(b)
   disp(Q3)
   S5 = input('Answer 1 or 2. ');

   if S5 == 1,
       N3 = [' Note: Sampling frequency (fs) must be selected so that   '
             ' fs/fc >= 2; Data length will be set to allow "N" samples.'];
       disp(b)
       disp(N3)
       fs2fc = input('Desired sampling-to-signal frequency ratio? ');
       const = 2*pi/fs2fc;
       fs = fs2fc*fc;
   else
       N4 = [' Note: Sampling frequency will be chosen to give "N" samples'
             ' so observation time must be <= "N"/(2*"fc").              '];
       disp(b)
       disp(N4)
```

```
        Tobserv = input('Desired observation time (seconds)? ');
        fs = N/Tobserv;
        const = 2*pi*fc/fs;
        Tsample = 1/fs;
    end
    Tsample = 1/fs;

    % Construct sine wave
    theta = theta*pi/180;
    for k = 1:N
        Data(k) = sin(const*(k-1)+theta);
    end
    Data = A*Data;

    if Make == 3,                     % used if PN modulation is desired
       Sinedata = Data;
       disp(b)
       disp(b)
       N4pt5 = [' You will now determine the PN sequence characteristics.'];
       disp(N4pt5)
    end
end

%------------------------------------------------------------------------------

% This section generates the desired pseudo-noise signal by building the
% appropriate feedback shift register (FSR).

if Make ~= 1,
   clc
   %  Determine desired PN sequence characteristics.
   N5 = [' Note: PN sequence = (+1 or -1,...,2^m - 1)              '
         ' Feedback shift register initial state is (1,1,...,1)'];
   disp(b)
   disp(N5)
   N6 = [' You will determine the following parameters:'
         '    - number of stages                        '
         '    - number of and position of taps          '
         '    - chip rate                               '
         '    - time delay                              '
         '    - number of samples                       '
         '    - sampling frequency or data length       '];
   disp(b)
   disp(N6)
   mstages = input('Desired number (m) of stages (2-10)? ');
   numbrtap = input('Number of taps? ');
   for k = 1:numbrtap
     Tap(k)=input(['What is the position of Tap number ',num2str(k),'?  ']);
   end
   chiprate = input('Chip rate (chips/sec)? ');
   delay = input('Sequence delay in chips (positive real number)? ');
   if Make == 2,

     % If the PN signal alone is desired the user must input the number
     % of samples and sampling rate: otherwise, they are dictated by
     % the choices made for the sinusoid.

     A = sqrt(2);
     N = input('Number of samples? ');
     Q4 = [' Do you wish to set:                          '
```

```
        '   1. the sampling frequency, or'
        '   2. the data length?         '];
  disp(b)
  disp(Q4)
  P6 = input('Answer 1 or 2. ');

  if P6 == 1,
    N7 = [' Note: Sampling frequency ("fs") must be selected so that  '
          ' fs/chiprate >= 2; data length will be set for "N" samples.'];
    disp(b)
    disp(N7)
    fs2cr = input('Desired sampling frequency to chip rate ratio? ');
    Tsample = 1/(chiprate*fs2cr);
  else
    N8 = [' Note: Sampling frequency will be chosen to give "N" samples'
          ' so observation time must be <= "N"/(2*Chip rate).         '];
    disp(b)
    disp(N8)
    Tobserv = input('Desired observation time (seconds)? ');
    fs = N/Tobserv;
    fs2cr = fs/chiprate;
    Tsample = 1/fs;
  end
else
  disp(b)
  N9 = [' Same number of samples (',num2str(N),') and sampling '];
  N10 = [' frequency (',num2str(fs),' Hz) that were selected for'];
  N11 = [' the sine wave will be used.  <Return>'];
  disp(N9)
  disp(N10)
  disp(N11)
  pause
  fs2cr = fs/chiprate;
end

%  Generate base PN sequence
FSR = ones(1,mstages);
L = 2^mstages - 1;
cklength = N/(fs2cr*L);
repeat = ceil(cklength);     % Determine how many periods are necessary
V = zeros(1,(repeat+1)*L);
for k = 1:L                  % build one period of the sequence

   V(k) = FSR(mstages);
   sigma = 0;
   for m = 1:numbrtap          % modulo-2 tap adder
      sigma = sigma+FSR(Tap(m));
   end
   FSR(2:mstages) = FSR(1:mstages-1);
   FSR(1) = rem(sigma,2);
end

% Ensure enough periods are available for the desired number of samples
   for n = 1:repeat
      V(n*L+1:(n+1)*L) = V(1:L);
   end

delay = rem(delay,L);
for n = 1:N
   Data(n) = V(fix((n-1)/fs2cr + delay)+1);
```

```
      if Data(n) == 0, Data(n) = -1; end         % make sequence bipolar
   end
end

%-------------------------------------------------------------------------------

% If selected, modulate the sine wave with the PN sequence.

if Make == 3
  Data = Sinedata.*Data;
end

%-------------------------------------------------------------------------------

% Plot the constructed signal and add noise, if desired.

indxto0 = [0:N-1];
v = [0,N-1,-1.2*max(Data),1.2*max(Data)];
axis(v);
plot(indxto0,Data),title('MATLAB Plot of Input Signal Vector')
xlabel('Sample number'),ylabel('Volts')
pause
axis;

clc
disp(b)
disp(' White Gaussian Noise is generated by the MATLAB "random" function.')
noise = input('Do you want noise added to your signal (Y or N)? ','s');
if noise == 'Y',

  % SNR based on "continous" signal energy and the Gaussian noise variance

  SNRdB = input('What is the desired Signal-to-Noise Ratio (dB)? ');
  SNR = 10^(SNRdB/10);
  Noisepwr = ((A^2)/2)/SNR;
  rand('seed',0); rand('normal');
  noisvctr = Noisepwr*rand(1,N);
  Data = Data+noisvctr;
  v = [0,N-1,-1.2*max(Data),1.2*max(Data)];
  axis(v);
  plot(indxto0,Data),title('MATLAB Plot of Input Data Vector')
  xlabel('Sample number'),ylabel('Volts')
  pause
  axis;
end

return
```

APPENDIX H

FUNCTION FOR DAUBECHIES' H-COEFFICIENTS

```
function[hcoeff] = daubdata(x)

%  This function returns the proper "h" coefficients for the specified order
%  Daubechies wavelet. The input value of "x" is the specified order.

if x == 2
  hcoeff = [0.482962913145;0.836516303738;0.224143868042;-0.129409522551];
elseif x == 3
  hcoeff = [0.332670552950;0.806891509311;0.459877502118;-0.135011020010;
            -0.085441273882;0.035226291882];
elseif x == 4
  hcoeff = [0.230377813309;0.714846570553;0.630880767930;-0.027983769417;
            -0.187034811719;0.030841381836;0.032883011667;-0.010597401785];
elseif x == 5
  hcoeff = [0.160102397974;0.603829269797;0.724308528438;0.138428145901;
            -0.242294887066;-0.032244869585;0.077571493840;-0.006241490213;
            -0.012580751999;0.003335725285];
elseif x == 6
  hcoeff = [0.111540743350;0.494623890398;0.751133908021;0.315250351709;
            -0.226264693965;-0.129766867567;0.097501605587;0.027522865530;
            -0.031582039318;0.000553842201;0.004777257511;-0.001077301085];
elseif x == 7
  hcoeff = [0.077852054085;0.396539319482;0.729132090846;0.469782287405;
            -0.143906003929;-0.224036184994;0.071309219267;0.080612609151;
            -0.038029936935;-0.016574541631;0.012550998556;0.000429577973;
            -0.001801640704;0.000353713800];
elseif x == 8
  hcoeff = [0.054415842243;0.312871590914;0.675630736297;0.585354683654;
            -0.015829105256;-0.284015542962;0.000472484574;0.128747426620;
            -0.017369301002;-0.044088253931;0.013981027917;0.008746094047;
            -0.004870352993;-0.000391740373;0.000675449406;-0.000117476784];
elseif x == 9
  hcoeff = [0.038077947364;0.243834674613;0.604823123690;0.657288078051;
            0.133197385825;-0.293273783279;-0.096840783223;0.148540749338;
            0.030725681479;-0.067632829061;0.000250947115;0.022361662124;
            -0.004723204758;-0.004281503682;0.001847646883;0.000230385764;
            -0.000251963189;0.000039347320];
elseif x == 10
  hcoeff = [0.026670057901;0.188176800078;0.527201188932;0.688459039454;
            0.281172343661;-0.249846424327;-0.195946274377;0.127369340336;
            0.093057364604;-0.071394147166;-0.029457536822;0.033212674059;
            0.003606553567;-0.010733175483;0.001395351747;0.001992405295;
            -0.000685856695;-0.000116466855;0.000093588670;-0.000013264203];
else
  hcoeff = [];
  break
end

hcoeff = hcoeff'/sqrt(2);                    % normalize the "h" vector

return
```

LIST OF REFERENCES

1. Kay, S.M., *Modern Spectral Estimation*, Prentice-Hall, 1988.

2. Daubechies, I., "Orthonormal Bases of Compactly Supported Wavelets," *Communications on Pure and Applied Mathematics*, Vol. XLI, pp. 909-996, November 1988.

3. Mallat, S.G., "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 11, No. 7, pp. 674-693, July 1989.

4. Franks, L.E., Unpublished wavelet notes, 1991.

INITIAL DISTRIBUTION LIST

No. copies

1. Defense Technical Information Center — 2
   Cameron Station
   Alexandria, VA 22304-6145

2. Library, Code 52 — 2
   Naval Postgraduate School
   Monterey, CA 93943-5002

3. Chairman, Code EC — 1
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, CA 93943-5000

4. Professor Alex W. Lam, Code EC/La — 5
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, CA 93943-5000

5. Professor Roberto W. Cristi, Code EC/Cx — 1
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, CA 93943-5000

6. Professor Herschel H. Loomis, Code EC/Lm — 1
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, CA 93943-5000

7. Mark R. Kalmbach — 2
   908 East Applegate Dr.
   Austin, TX 78753