

Cranfield Institute of Technology

College of Aeronautics

Department of Avionics and Electronic System Design

PhD Thesis

Academic Year 1992-1993

Ko , Sung-Yuan

**Investigation and Development of DSP Algorithms/Hardware for Real
Time Power Spectral Density Estimation**

Supervisor:

Dr. D. D. Russell

To my

Wife: Ya-Lin,

Son: Bee-Wen,

and

Daughter: Bee-Hwa

Acknowledgements

The author would like to express his gratitude to Dr. D. D. Russell for his valuable suggestions and continuous guidance during the supervision of this work. Thanks are also due to all the staff of Electronic System Design and Avionic Department, especially Mr. J. Wyatt, for the facilities and support which they provided him with.

I also thank my wife and my son for the joy they brought about during their stay in the U.K.

Abstract

This research is concerned with the Power Spectrum Density Estimation with emphasize on the high-resolution algorithms and their real-time implementations.

The classical PSD estimation methods are fast and robust, but their resolutions may not be adequate when the record length is short. On the other hand when the record length is short the autoregressive parametric methods have higher resolution capability, but they may have spurious peaks if the order of the model is chosen too high in the attempt to increase the resolution when the SNR is low. An algorithm is proposed to combine the spectrum of the classical method and the autoregressive model. This allows the overestimation of the order of the autoregressive model. The spurious peaks that result are then suppressed by the low values in the spectrum of the classical methods. The wide spectral mainlobe of the classical method, on the other hand, serves to indicate the area where the true signals are located. This alleviates the difficult order selection problem of the parametric methods. An adaptive version of this method is also proposed. It is based on the adaptive autoregressive and adaptive maximum eigenvector concept. It can track a slowly changing environment. With the combination of these two methods, it is shown that it has the high-resolution performance of AR method with improved performance in the noisy environment.

Contents

1	Introduction	1
1.1	Overview of Power Spectral Density Estimation Problems	1
1.2	Objective and Contributions	2
1.3	Classification of PSD Estimators	3
1.4	Previous Work	7
1.5	Structure of The Thesis	10
2	Block PSD Estimation Methods	12
2.1	Introduction	12
2.2	Nonparametric Methods	15
2.2.1	Classical Methods	15
2.2.2	Minimum Variance Spectral Estimation Method	19
2.3	Parametric Methods	21
2.3.1	Autoregressive PSD Methods	22
2.3.2	Moving Average Method	32
2.3.3	Autoregressive Moving Average Method	35

2.3.4	Eigenanalysis-based Methods	38
2.4	A Proposed Algorithm for PSD Estimation	43
3	Sequential PSD Estimation Methods	47
3.1	Introduction	47
3.2	Sequential Periodogram Method	48
3.3	Adaptive Autoregressive PSD Estimator	49
3.3.1	The LMS Adaptive Spectrum Estimator	50
3.3.2	The Effect of The Filter Length	52
3.3.3	The Effect of Step Size μ	53
3.4	Adaptive Eigenanalysis-based PSD Estimators	54
3.4.1	The Adaptive Algorithms	54
3.4.2	The Adaptive Pisarenko Harmonic Decomposition Algorithm .	56
3.5	The Proposed Adaptive Frequency Estimator	59
4	Performance Test of PSD Estimators	62
4.1	Introduction	62
4.2	Performance Test of Block PSD Estimators	63
4.2.1	The Effect of Record Length	63
4.2.2	The Effect of SNR	64

4.2.3	The Effect of Phase Difference	65
4.2.4	The Effect of Relative Power	65
4.2.5	The Effect of A Nearby Signal	66
4.3	Performance Test of Sequential PSD Estimators	84
4.3.1	Introduction	84
4.3.2	The Detectability Test	85
4.3.3	The Frequency Resolution Test	85
5	Detection of Visual Evoked Potential Using PSD Techniques	101
5.1	Introduction	101
5.2	The Description of Experimental Data	103
5.3	The Snap Shot Implementation of MUSIC Algorithm	104
5.4	Comparison of Periodogram and MUSIC Frequency Tracking Performance	105
5.5	Some Comments	105
6	The Real-Time Implementation of PSD Estimators	109
6.1	Introduction	109
6.2	The Development System Approach	109
6.2.1	The Real-Time Sequential Periodogram Implemented on The Development System	110

6.3	The Design of A Digital Signal Processing Board	113
6.3.1	The Analog to Digital Converter	113
6.3.2	The Digital Signal Processor DSP56001	114
6.3.3	EPROMS	115
6.3.4	RAMs	116
6.3.5	The Digital to Analog Conversion	117
6.3.6	The Timer	117
6.3.7	The Address Decoder	118
6.4	The Proposed Multiprocessor Architecture for PSD Estimation Appli- cation	120
6.4.1	The PSD Estimation Using The TMS320C4X Parallel Process- ing Features	121
7	Conclusion and Suggestion for Further Work	125
7.1	Conclusion	125
7.2	Suggestion for Further Work	126
A	FORTTRAN Program for VEP PSD Estimation	128
B	Assembly Program for Sequential Periodogram	153

List of Figures

2.1	The flow chart of PSDE	14
2.2	The unmodified periodogram with rectangular window	16
2.3	The periodogram with Hamming window	17
2.4	The Welch periodogram with $N = 32$, $L = 20$, $S = 10$	18
2.5	The Blackman-Tukey spectrum with Hamming window	19
2.6	The MVSE spectrum	21
2.7	AR process	22
2.8	The AR process corrupted by additive noise	23
2.9	The prediction error filter	25
2.10	The orthogonality principle of the least square algorithm	26
2.11	The autocorrelation method spectrum	27
2.12	The covariance method spectrum	28
2.13	The modified covariance method spectrum	30
2.14	The Burg spectrum	31
2.15	The MA process	33
2.16	The Durbin MA spectrum	34

2.17	The ARMA Process	35
2.18	The Least Square Modified Yule-Walker ARMA spectrum	37
2.19	The MUSIC spectrum	42
2.20	The EigenVector method spectrum	43
2.21	The combined PSD estimation method	44
2.22	The combined periodogram and modified covariance method	45
2.23	The combined periodogram and Burg method	46
3.1	The cyclic buffer	48
3.2	The ping-pong memory configuration	49
3.3	The Adaptive Line Enhancer	51
3.4	The spectral analysis based on the output of ALE	52
3.5	The spectral analysis based on the weight of the ALE	53
3.6	The Adaptive Pisarenko Harmonic Decomposition filter	57
3.7	Two examples of the adaptive combined PSD estimation	61
4.1	Record length of 64	67
4.2	Record length of 30	68
4.3	SNR of 30 dB	69
4.4	SNR of 20 dB	70

4.5	SNR of 10 dB	71
4.6	SNR of 0 dB	72
4.7	Phase difference of 0 degree	73
4.8	Phase difference of 45 degree	74
4.9	Phase difference of 90 degree	75
4.10	Phase difference of 135 degree	76
4.11	Phase difference of 180 degree	77
4.12	Relative amplitude of 0.5	78
4.13	Relative amplitude of 0.25	79
4.14	Relative amplitude of 0.1	80
4.15	The third signal at normalized frequency of 0.4	81
4.16	The third signal at normalized frequency of 0.3	82
4.17	The third signal at normalized frequency of 0.25	83
4.18	The detectability test: SNR = 5dB	87
4.19	The detectability test: SNR = 0dB	88
4.20	The detectability test: SNR = -5 dB	89
4.21	The frequency resolution test: SNR = 20 dB	90
4.22	The frequency resolution test: SNR = 0 dB	91
4.23	The frequency resolution test: SNR = -5 dB	92

4.24	The frequency resolution test: relative amplitude = 0.5	93
4.25	The frequency resolution test: relative amplitude = 0.25	94
4.26	The frequency resolution test: relative amplitude = 0.1	95
4.27	The frequency resolution test: relative phase = 0 degree	96
4.28	The frequency resolution test: relative phase = 45 degree	97
4.29	The frequency resolution test: relative phase = 90 degree	98
4.30	The frequency resolution test: relative phase = 135 degree	99
4.31	The frequency resolution test: relative phase = 180 degree	100
5.1	The VEP response of a pilot	103
5.2	The partition of data to form matrix	104
5.3	The flow chart of program VEP	106
5.4	The 3D spectrogram display of periodogram method	107
5.5	The 3D spectrogram display of MUSIC method	108
6.1	The DSP56000 ADS	110
6.2	Flow chart of the main program of the sequential periodogram implemented on DSP56000ADS	111
6.2	Flow chart of the interrupt service subroutine of the sequential periodogram Implemented on DSP56000ADS (cont.)	112
6.3	Block diagram of the Digital Signal Processing Board	113

6.4 The analog to digital converter 114

6.5 Block diagram of DSP56001 115

6.6 The EPROM connections 116

6.7 The digital to analog converter 117

6.8 The timer 118

6.9 The PAL as address decoder 120

6.10 The multiprocessor configuration using TMS320C4X 122

6.11 The program flow chart of processor 1 123

6.12 The program flow chart of processor 2 124

List of Abbreviations

ACF	Autocorrelation Function
ALE	Adaptive Line Enhancer
ALEO	Adaptive LINE Enhancer Output
ALEW	Adaptive Line Enhancer Weights
AMAXEV	Adaptive Maximum EigenVector
APHD	Adaptive Pisarenko Harmonic Decomposition
AR	Autoregressive
ARMA	Autoregressive Moving Average
CUPL	Universal Compiler for Programmable Logic
DFT	Discrete Fourier Transform
DMA	Direct Memory Access
DSP	Digital Signal Processing (Processor)
DSPB	Digital Signal Processing Board
EEG	Electroencephalography
EV	Eigenvector Method
EVD	Eigenvalue Decomposition
FFT	Fast Fourier Transform
FT	Fourier Transform
LMS	Least Mean Square
MA	Moving Average
MCOV	Modified Covariance
MLM	Maximum Likelihood Method
MN	Minimum Norm
MUSIC	Multiple signal Classification
MVSE	Minimum Variance Spectral Estimation
PAL	Programmable Array Logic
PSD	Power Spectral Density
SVD	Singular Value Decomposition
VEP	Visual Evoked Potential
WSS	Wide Sense Stationary

Chapter 1

Introduction

1.1 Overview of Power Spectral Density Estimation Problems

The Power Spectral Density (PSD) of a Wide Sense Stationary (WSS) process is formally defined as

$$P_{xx}(f) = \sum_{k=-\infty}^{\infty} r_{xx}[k] \exp(-j2\pi f k) \quad (1.1)$$

Where $r_{xx}[k]$ is the AutoCorrelation Function (ACF) of $x[n]$ defined as

$$r_{xx}[n] = \xi(x^*[n]x[n+k]) \quad (1.2)$$

and ξ is the expectation operator. A random process is WSS if its mean is independent of time and its autocorrelation depends only on the time index difference between samples. The PSD function describes how the power of a measured random process is distributed with respect to frequency. It can be seen from the definition that it is impossible to know exactly the PSD, since an infinitely long autocorrelation sequence is not possible. The objective of spectral estimation is then to estimate the spectral content of a random process based on a finite set of observations from that process. There are many PSD estimators available and they may give different PSD estimations. If there is no prior information about the input, it is not obvious which estimator gives the correct estimation. One estimator may have sharp spectral appearance resembling high resolution, but it may be an artifact. The common approach is to use a known input signal as a test case to test the characteristic of different estimators. However, when applying these estimators to calculate the spectrum of real data instead of simulated data, it is better to apply several of them and then

extract the common features of their spectrum. This will give more confidence that the true spectrum has been estimated. Also the bias and variance of the estimation have to be taken into account since the spectral estimator is a statistical function of the data. It is not sufficient to calculate only one PSD, many are needed. The average of these gives the bias information and the variability of different PSDs gives the variance information. In this research we use a known scenario as a test case to test the characteristic of different estimators.

1.2 Objective and Contributions

The objective of this research is to investigate the available PSD estimators and apply them to the detection of VEP (Visual Evoked Potential). It consists of three stages:

1. **Theoretical Study:** As mentioned in the last section the true PSD is unknown hence there is no “best” spectral estimator. A multitude of spectral estimators have been proposed. The derivation of those PSD estimators are investigated. Also it is necessary to understand the assumptions made in order to apply them correctly.
2. **Software Simulation:** The spectral estimator is a statistical function of the data hence its performance ideally should be described by statistical terms, e.g. bias, variance. But there is no exact statistical description of PSD estimators available for finite data record length. A common practice to compare the performance of the PSD estimators visually is to use the Monte Carlo computer simulations based on a known test case, i.e. the performance between PSD estimators is compared qualitatively rather than quantitatively.
3. **Hardware Implementation:** The ultimate goal of this research is to implement the PSD estimator in real-time. It is described in Chapter 6 that a 512 points sequential periodogram is implemented in real-time for the detection of the VEP.

In response to the objective there are three contributions in this research:

1. We developed a new algorithm which combines the best features of the non-parametric and parametric methods. The nonparametric methods have low resolution when the data samples are small but they are robust in indicating the true location of the signals. The parametric methods have higher resolution but they may have spurious peaks. By multiplying their spectra together the spurious peaks in the parametric spectrum are suppressed and resolution capability of the nonparametric methods is improved.
2. A real-time periodogram is designed to investigate the VEP signal. The FFT is triggered after a certain amount of samples are acquired. These samples are stored in a short temporary buffer first. When enough samples are accumulated they are then appended to a cyclic buffer and the FFT is triggered. This cyclic buffer is the first stage of the FFT input data buffer. This technique gives the DSP (Digital Signal Processor) more time for number crunching and allows the signal to build up gradually in the input data buffer. This will give a smooth appearance of the VEP signal and the required resolution is achieved by choosing a suitable size of the input data buffer.
3. A DSPB (Digital Signal Processing Board) is designed. It has A/D, D/A, timer, RAMs, EPROMs and PAL (Programmable Array Logic) as the address decoder. It has the stand-alone capability. When the program is developed and debugged on the development system, the program is transferred to EPROMs. After power-up it automatically takes the data from A/D, executes the sequential periodogram and displays the output on the oscilloscope. It doesn't need a host computer.

1.3 Classification of PSD Estimators

The PSD estimators can be classified into two categories:

1. Nonparametric PSD estimators
2. Parametric PSD estimators

There are several PSD estimators in each category. The popular ones are as follows

1. Nonparametric Methods

(a) Classical Methods

- i. Periodogram
- ii. Blackman-Tukey

(b) MVSE (Minimum Variance Spectral Estimation)

2. Parametric Methods

(a) Rational Transfer Function Model

- i. AR (Autoregressive)
- ii. MA (Moving Average)
- iii. ARMA (AutoRegressive Moving Average)

(b) Eigenanalysis-based Methods

- i. PC (Principal Component)
- ii. MUSIC (MUltiple Signal Classification)
- iii. MN (Minimun Norm)
- iv. EV (EigenVector)

The Classical PSD estimators which belong to the category of nonparametric methods are robust. Because they only make the assumption of wide sense stationarity. Due to the usage of the Fast Fourier Transform (FFT) the classical methods are also fast. However, the windowing effect (i. e. the implication that the data are zero outside the observation period) set the resolution capability of the classical methods to be $\frac{1}{NT}$. Where N is the number of samples and $\frac{1}{T}$ is the sampling rate. The advantages of this class are:

- They are robust.
- They are fast.

The disadvantages are:

- Their resolutions are limited to $\frac{1}{NT}$. Many samples may be needed to get an acceptable resolution, this is sometimes difficult to achieve.
- An Implicit assumption made by the classical methods is that the data of ACF values outside the observation interval are zero. In the time domain this can be seen as multiplying the data record with a rectangular observation window. The windowing effect in the frequency domain is equivalent to convolving the data spectrum with the sinc function (i.e. the Fourier transform of the rectangular window). This creates sidelobes. If the data consist of sinusoidal or narrowband signals. The sidelobes of a high level signal may mask the low level signal close to it in frequency. This may cause difficulty in detecting weak wanted signals.
- In a severely nonstationary situation the number of samples are constrained to be small in order to consider it as locally stationary. In this case the resolution performance of classical estimators are degraded.

The rational transfer function parametric methods assume a model for the random process. When the model is correct it can extrapolate the data outside the observation interval. So it eliminates the windowing effect and can achieve resolution higher than $\frac{1}{NT}$ which is the limit of classical methods. There are three models suitable for random processes:

1. Autoregressive (AR)
2. Moving Average (MA)
3. Autoregressive and Moving Average (ARMA)

AR PSD estimators will display sharp peaks. MA PSD estimators will display sharp nulls. While the ARMA is the combination of these two, it will display sharp peaks and nulls. When there is prior knowledge of the random process, e.g. narrowband or broadband, sharp peaks or sharp nulls, it should be taken into account in choosing a

suitable model. In general the AR model is the most popular because it only needs to solve simultaneous linear equations to obtain the parameters. The MA and ARMA techniques require the solution of nonlinear equations which is more difficult. The advantages of this class are:

- They have high resolution.
- They need less data than the classical methods.
- Many statistical parameter estimation algorithms are already available.

The disadvantages are:

- When the model differs from the random process an incorrect PSD is obtained.
- It is difficult to choose the model order. When a high order model is chosen it may have spurious peaks or nulls.
- The AR model is the most popular parametric model. However, the additive noise (e. g. measurement noise) is not part of the model. When the all pole AR model is added with the noise, it will have zeros as well as poles. Those zeros will flatten the AR spectral estimate. Therefore the detection and resolution performance of the AR spectral estimator deteriorates in the presence of additive noise.

The eigenanalysis-based parametric methods are based on spectral decomposition of the correlation or data matrix of the random process. They separate the eigenvectors into two subspaces, *signal subspace* and *noise subspace*. By using the orthogonality between these two subspaces several algorithms are developed. Theoretically these methods can achieve very high resolution and currently attract many research interests. The advantages of this class are

- They can achieve high resolution.

- They need less data than classical methods.
- They can cope with the additive noise situation.

The disadvantages are

- It is difficult to separate the noise subspace and signal subspace. When the estimation of the number of sinusoids is wrong it will produce an incorrect PSD estimation.
- They are computationally intensive.

1.4 Previous Work

Sir Isaac Newton first introduced the word *spectrum* as a scientific term to describe the band of colours in the seventeenth-century. Daniel Bernoulli developed the general solution for the displacement $y(x,t)$ of the string at time t and position x (with the end points at $x = 0$ and $x = \pi$ fixed) in the *wave equation* to be

$$y(x, t) = \sum_{n=1}^{\infty} \sin nx (A_n \cos nct + B_n \sin nct) \quad (1.3)$$

Where c is a constant representing the speed of the traveling wave on the string. Later in the nineteenth-century Jean Baptiste Joseph Fourier extended the wave equation solutions by asserting that any arbitrary function $y(x)$, satisfying Dirichlet conditions, could be represented as an infinite summation of sine and cosine terms,

$$y(x) = \sum_{n=1}^{\infty} (A_n \cos n\alpha x + B_k \sin n\alpha x) \quad (1.4)$$

The process of determining A_n and B_n of a function $y(x)$, or its sample, is known as *harmonic analysis*, due to the harmonic relationship between cosine and sine terms. Of the many scientists who used harmonic analysis, Schuster promoted the *periodogram* method which now known as the classical PSD estimation. He suggested the plot of the squared envelop of $P_n = A_n^2 + B_n^2$ of the Fourier transform coefficients [41]. Norbert Wiener published his classic paper *Generalized Harmonic Analysis in*

1930 [49]. This paper established spectral analysis as a fundamental statistical tool to analyze random processes. He related the *autocorrelation function* (ACF) and *power spectral density* as a Fourier transform pair which is known today as *Wiener-Khintchine theorem*. Blackman and Tukey noticed that the poorer estimates of the ACF at higher lags is a result of the fewer number of lag products averaged. They therefore suggested weighting the ACF estimates at higher lags less or use the spectral estimator

$$\hat{P}_{BT}(f) = \sum_{k=-(N-1)}^{N-1} w[k] \hat{r}_{xx}[k] \exp(-j2\pi f k) \quad (1.5)$$

This is known as the Blackman-Tukey method. A classic treatise describing this procedure in detail was published in 1958 [4]. The FFT (Fast Fourier Transform) described in the paper of J. Cooley and J. Tukey in 1965 [11] computes the discrete Fourier transform efficiently. This computational algorithm has led to an increased usage of digital spectral analysis as a signal processing tool.

G. Yule introduced in 1927 [53] a notable alternative analysis tool. He modelled a time series as a linear regression process driven by a white noise. Thus Yule established the basis of the *parametric* approach to spectral analysis. But not until Burg's work in 1967 [5] did interest in this method receive significant attention. A PSD estimator is a parametric estimator when it employs *a priori* information about the random process. In this case the system is modelled as a process which has a rational transfer function. Burg described his high-resolution spectral estimate in the context of a *maximum entropy* mathematical formulation which is very closely related to autoregressive spectral analysis. Techniques are called *high-resolution* PSD estimation when their frequency resolution can exceed the $\frac{1}{NT}$ limit of the classical methods. The model of a random process is called an *autoregressive* model when the resulting transfer function of that process has only poles. The high resolution property of the AR spectral estimator is due to an implied extrapolation of the ACF samples outside the observation period by the assumed AR model. Burg argued that the extrapolation should be chosen so that the time series characterized by extrapolated ACF has maximum entropy. In practice there is no difference between the MESE (Maximum Entropy Spectral Estimation) and the AR spectral estimator in the one-dimensional case. There are several algorithms to estimate the AR parameters. A software package, which is believed to be

the first of its kind available for use in personal computer, called MESA (Maximum Entropy Spectral Analysis) [8], [9] implements those algorithms on the PC (Personal Computer). It allows an interactive and data-adaptive spectral analysis based on the maximum entropy principle. Recently, due to the advances in VLSI, the real time implementation of the AR spectrum analyzer is also available [40]. The ARMA model in theory can be more accurate than the AR model for spectral analysis. Because it has more freedom in adjusting poles and zeros of the rational transfer function. However, the estimation of the parameters of the ARMA model is far more difficult than the AR model and the result of ARMA model spectral estimation is sensitive to parameter adjustment and order selection. Kaveh [24] studied the fundamental problem of determining the minimal set of lags of an estimate of the autocorrelation needed to achieve adequate statistical efficiency for ARMA parameter estimation.

The *eigenanalysis-based technique* which belongs to the parametric methods uses the eigenstructure of the signal correlation matrix. It separates the correlation matrix into two vector subspaces namely *signal* and *noise* subspaces. Techniques which use only the signal subspace are called *Principal Component* methods. The noise subspace techniques on the other hand employ the orthogonal property between the noise eigenvectors and signal subspace. Pisarenko used only one noise eigenvector (i. e. the eigenvector corresponding to the minimum eigenvalue) to extract the signal frequencies from the correlation matrix of a linear array which is known as *Pisarenko Harmonic Decomposition* (PHD) method [35]. This set the foundation of the eigenanalysis-based techniques. Schmidt extended it to the general array case and developed the *Multiple Signal Classification* (MUSIC) algorithm [38] [39] which is based on the complete noise subspace eigenstructure of the correlation matrix. Independently, Bienvenu and Kopp [2] developed a frequency domain technique based on the eigenstructure of the spectral density matrix of received signals. Kumaresan and Tufts [26] developed the *Minimum Norm* (MN) method which just used one vector to represent the noise subspace. Johnson and DeGraaf [20] weighted each noise subspace eigenvector by the inverse of its associated eigenvalue which is known as the *EigenVector* (EV) frequency estimator.

To calculate the eigenstructure of the correlation matrix is very computationally

intensive. Adaptive algorithms were devised to reduce the computation. Thompson [44] proposed an adaptive algorithm to implement the Pisarenko's method. Yang [50] extended the idea to estimate the whole signal or noise subspace adaptively. Kahunen [22], Sharman [43], Schreiber [42], DeGraf [12], Ortigueira [32] and Yu [51] also proposed adaptive implementation of eigenanalysis-based methods. In general, these estimators estimate the frequencies as the peaks of the fraction

$$\frac{1}{g(|\mathbf{e}^H \hat{\mathbf{v}}_{p+1}|^2, |\mathbf{e}^H \hat{\mathbf{v}}_{p+2}|^2, \dots, |\mathbf{e}^H \hat{\mathbf{v}}_M|^2)} \quad (1.6)$$

where $\mathbf{e}$ is the signal vector and $\hat{\mathbf{v}}_i$ are eigenvectors corresponding to the noise eigenvalues.

With such a long list of PSD estimators, an *expert system* will be very helpful in choosing an appropriate estimator for the problem at hand. Gaby and Hayes developed such a system [14]. The idea is to combine the best features of different approaches. An approach based on this idea is proposed in chapter 2 and 3.

1.5 Structure of The Thesis

Chapter 1 discusses general PSD estimation problems. The PSD estimators are classified into two categories. A general comment about those estimators are given. The mentioned PSD estimators are then described in more detail in chapter 2 and 3. Previous works are also described in this chapter.

Chapter 2 gives a survey of block PSD estimation algorithms. There are many spectral estimators to be found in the literature. Only the more promising approaches are discussed in this thesis.

Chapter 3 presents a survey of the PSD estimators in the context of sequential nature. This enables the PSD estimators to track a slowly changing environment. Also due to the reduction of computation, the sequential PSD estimators are more viable in real-time applications.

Chapter 4 presents software simulation results. An interactive program is described which tests the performance of block and sequential PSD estimators. The pro-

gram is written in FORTRAN 77 language and implemented on a VAX computer. UNIRAS graphic subroutines are used for display purposes. The graphical representation is an important part in the comparison of PSD estimators.

Chapter 5 presents the results of applying periodogram and MUSIC PSD algorithms to the detection of the VEP (Visual Evoked Potential) signal. This is a very small signal (few microvolts) measured on the scalp by sophisticated data acquisition systems. The 3D spectrogram shows that the MUSIC algorithm is capable of detecting the VEP signal by using just 32 samples despite being heavily embedded in the noise.

Chapter 6 discusses the real-time implementation of PSD estimators. Two approaches are taken to implement the PSD estimator in real time. One is based on the development system. The other is based on a stand-alone digital signal processing board. The design of this board is described in detail. A multiprocessor system suitable for parallel processing is also proposed in this chapter.

Chapter 7 is the conclusion and suggestion for further work.

Chapter 2

Block PSD Estimation Methods

2.1 Introduction

In this chapter block PSD estimators are described together with plots to indicate the characteristics of each of these. The plot is generated from a data set according to the formula

$$x[n] = \sum_{i=1}^k A_i \exp[j(2\pi f_i n + \phi_i)] + w[n] \quad (2.1)$$

where $n = 1, 2, \dots, N$, and N is number of samples, k is number of signals. The parameters used are $N = 32, k = 2, A_1 = 1., A_2 = 1., f_1 = 0.2, f_2 = 0.22, \phi_1 = 0., \phi_2 = 0., SNR = 10dB$. $w[n]$ is zero mean white Gaussian noise with variance of σ_w^2 . SNR is defined as

$$SNR = 10 \log \frac{A_i^2}{2\sigma_w^2} \quad (2.2)$$

therefore

$$\sigma_w^2 = \frac{A_i^2}{2 \times 10^{SNR/10}} \quad (2.3)$$

The normalized frequency separation of these two signals is chosen to be about $\frac{T}{NT} = \frac{1}{N} = \frac{1}{32} = 0.03125$. Since this value is generally accepted as the critical value to distinguish an estimator being high-resolution or not. The plots are normalized to make the maximum value 0 dB. A program called PSDE is written for this purpose. The flow chart is shown in figure 2.1. This program is used in this chapter and the following chapters. It has the following functions:

1. Generate software simulated data.
2. Store the software simulated data into a file for later use.

3. Read the data from a file.
4. Calculate various PSD estimations.
5. Display the calculated spectrum.
6. Store the calculated spectral data into a file for hardware printout.
7. Allow the user to repeat the test to check the effect of different parameters.

The AR model is the most popular parametric method. Because it only needs to solve a set of linear equations and it tends to have sharp peaks. However, the main difficulty of using this model is the choice of model order. When the order of the model is chosen too low the resulting spectral estimation will be highly smoothed thus the resolution is low. On the other hand, when the order of the model is chosen too high, although the resolution is increased the spurious peaks may be introduced into the spectrum. The dilemma is that the best choice of the model order is generally not known *a priori*. There are several criteria to choose the model order, e. g. FPE (Final Prediction Error), AIC (Akaike Information Criterion), MDL (Minimum Description Length), and CAT (Criterion Autoregressive Transform). However, it has been found that for short data segments no criteria work very well ([29], page 231), particularly when applied to the actual data rather than simulated data. These criteria generally just serve as a guideline for an initial order selection. The subjective judgement of choosing the model order, rather than science, is still needed to fine tune the estimation. One of the original contributions of this work is described fully at the end of this chapter on using a conceptually simple idea. The order is selected between $\frac{1}{3} \sim \frac{1}{2}$ of the data record length. It has been found analytically and experimentally that the order selection of this range produces satisfactory results [29]. The general guideline is to overestimate the order of the model to make the resolution as good as possible. This generally will introduce some spurious peaks. The spectrum is then multiplied with that of the nonparametric method, e. g. periodogram. The spurious peaks will be suppressed since the spectrum of the latter doesn't depend on the model order and we can control its sidelobes. The result is quite effective, see figure 2.22 and figure 2.23. The cost of this method is the increase of computational

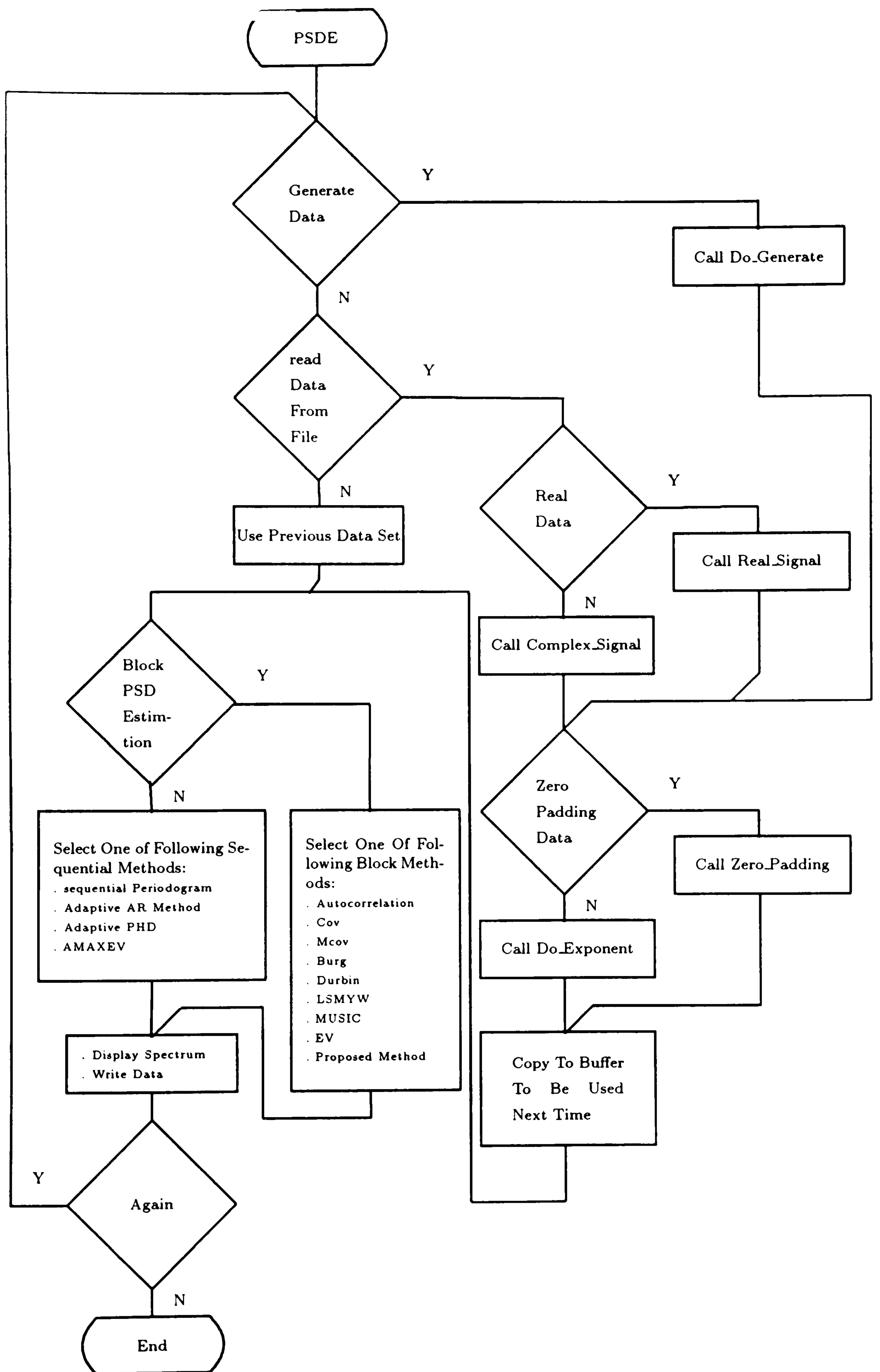


Figure 2.1: The flow chart of PSDE

2.2 Nonparametric Methods

The methods described in this section are called nonparametric methods because they do not assume a model of the data and hence they do not have parameters to fit the data. The only assumption is wide sense stationarity. It can be seen from the accompanying plots that the periodogram and Blackman-Tukey methods are unable to resolve two signals in the test data while the MVSE can.

2.2.1 Classical Methods

The periodogram and Blackman-Tukey (Correlogram) methods are called classical methods to distinguish them from the modern high-resolution techniques. The periodogram method estimates the spectrum directly from the data, whereas the Blackman-Tukey method estimates the spectrum indirectly from the autocorrelation function of the data.

2.2.1.1 Periodogram Method One definition of the PSD, based on ergodicity, has the discrete time form

$$P_{xx}(f) = \lim_{N \rightarrow \infty} \xi \left\{ \frac{1}{(2N+1)T} \left| T \sum_{n=-N}^N x[n] \exp(-j2\pi f n T) \right|^2 \right\} \quad (2.4)$$

Ignoring the expectation operation and assuming a finite data set $x[0], x[1], \dots, x[N-1]$ of N samples, the sample spectrum

$$\begin{aligned} \hat{P}_{xx}(f) &= \frac{1}{NT} \left| T \sum_{n=0}^{N-1} x[n] \exp(-j2\pi f n T) \right|^2 \\ &= \frac{T}{N} \left| \sum_{n=0}^{N-1} x[n] \exp(-j2\pi n f T) \right|^2 \end{aligned} \quad (2.5)$$

may be computed from the finite data sequence. This unmodified periodogram will yield statistically inconsistent PSD estimation because the expectation has been ignored. The variance does not decrease even if N is increased. See figure 2.2 for the unmodified periodogram of the test data. The sidelobes of the periodogram can be suppressed by applying a data window to the data prior to the transform. Figure 2.3 shows the periodogram with a Hamming window. To decrease the variance one

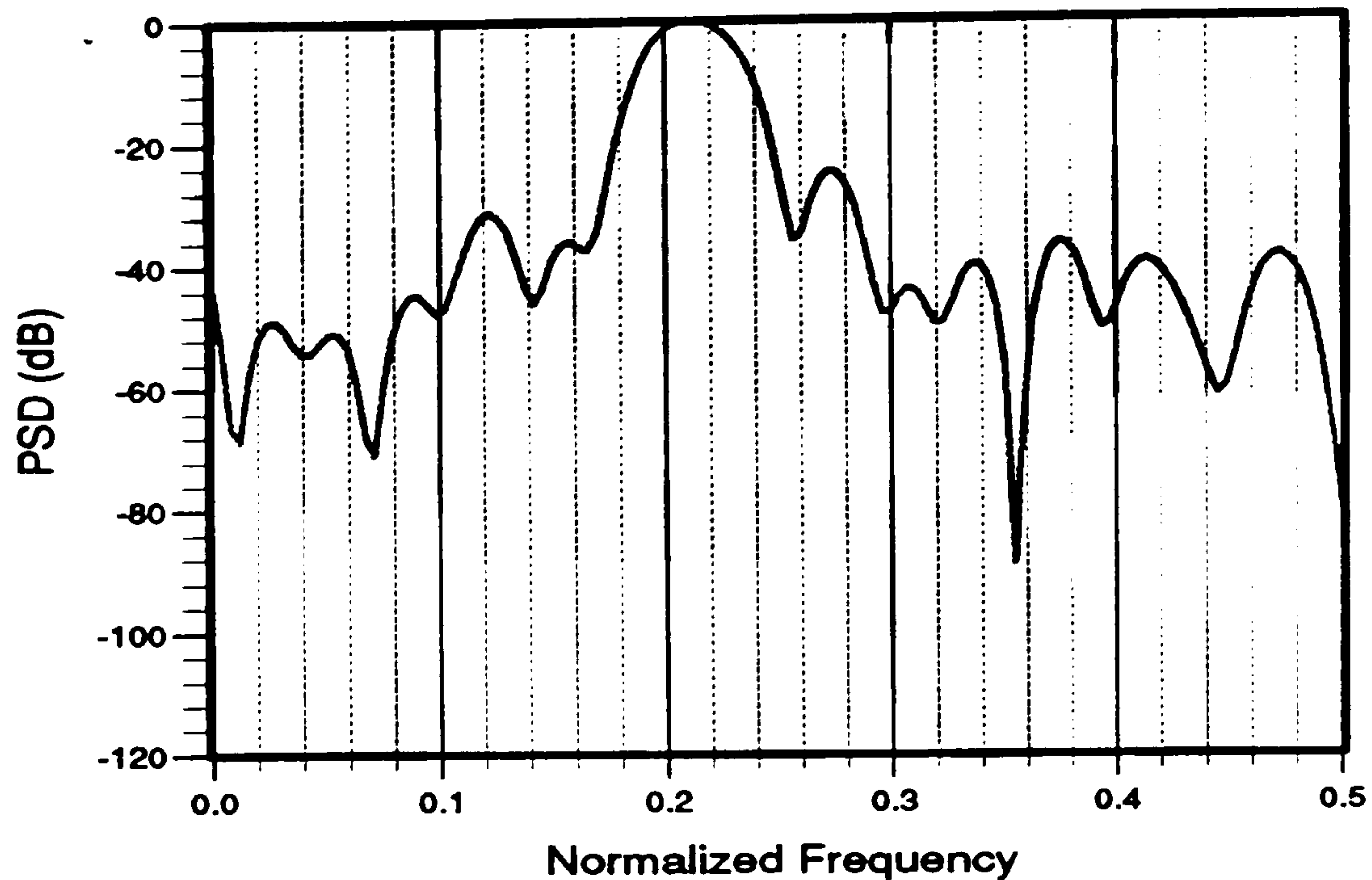


Figure 2.2: The unmodified periodogram with rectangular window

can use pseudo ensemble averaging to smooth the periodogram PSD estimate (i. e. modifying the periodogram). Three averaging techniques are considered as follows.

1. Daniel Periodogram

Daniel suggested to average over adjacent spectral frequencies

$$\hat{P}_D[f_i] = \frac{1}{2p+1} \sum_{n=i-p}^{i+p} \hat{P}_{xx}[f_n] \quad (2.6)$$

i. e. Daniel periodogram estimation at frequency f_i is obtained by averaging the p points on either side of this frequency.

2. Bartlett Periodogram

Bartlett created a pseudo ensemble of the sample spectrum by dividing the data sequence of N samples into M nonoverlapping segments of L samples so that $ML \leq N$. The m^{th} segment consists of the samples $x^m[n] = x[mL + n]$ for $0 \leq n \leq L - 1$. Where the superscript m denotes the segment number. An independent sample spectrum

$$\hat{P}_{xx}^m[f] = \frac{1}{LT} \left| T \sum_{k=0}^{L-1} x^m[k] \exp(-j2\pi f k T) \right|^2 \quad (2.7)$$

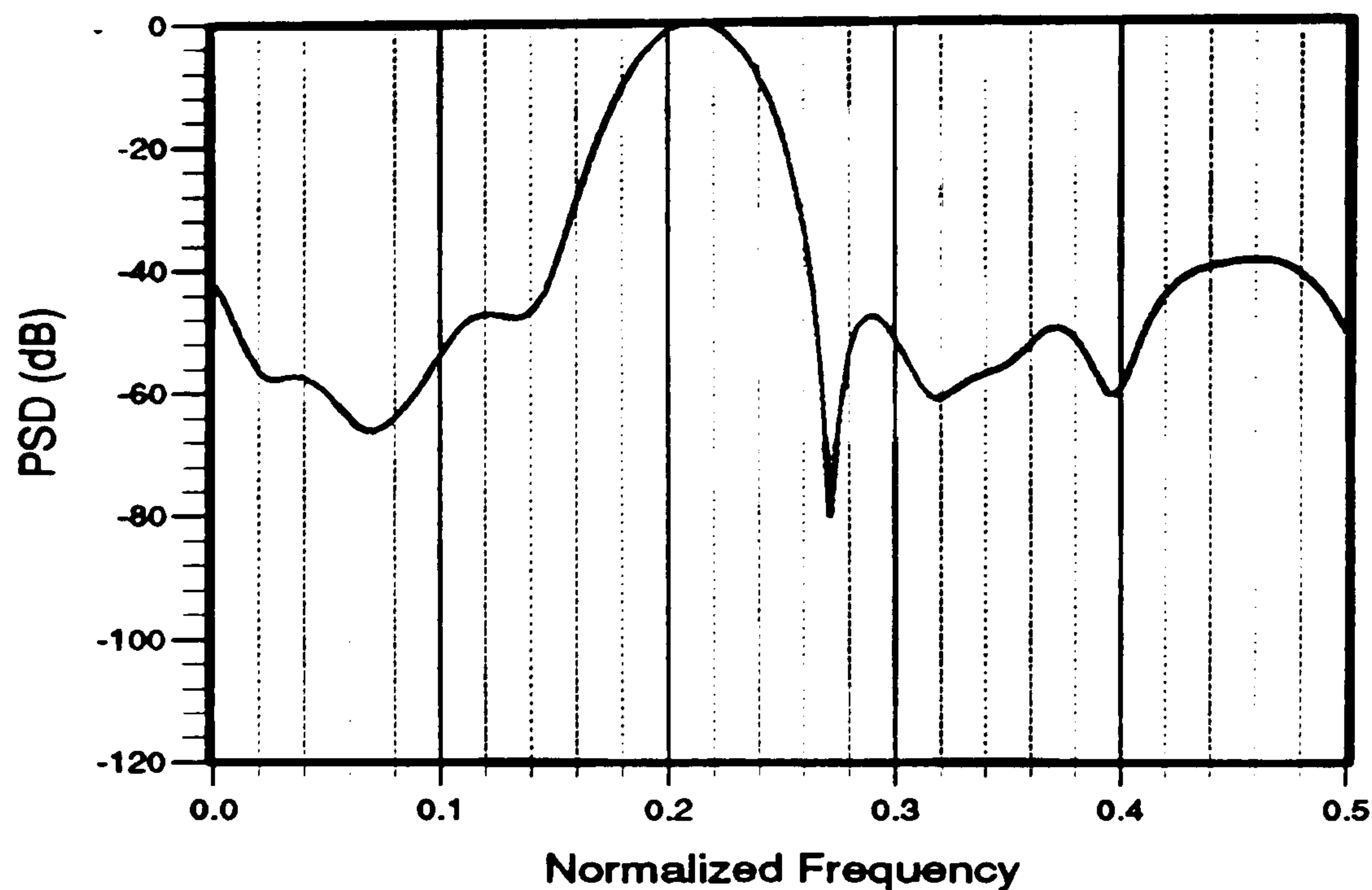


Figure 2.3: The periodogram with Hamming window

for each segment $0 \leq m \leq M - 1$ is computed over the range $-\frac{1}{2T} \leq f \leq \frac{1}{2T}$. The Bartlett averaged periodogram is then calculated by

$$\hat{P}_B[f] = \frac{1}{M} \sum_{m=0}^{M-1} \hat{P}_{xx}^m[f] \quad (2.8)$$

3. Welch Periodogram

Basically this is a variation of Bartlett algorithm by using data windows prior to the computation of the segment periodogram and allowing the data segments to overlap. The average of the windowed segment periodograms yields the Welch periodogram estimate

$$\hat{P}_W(f) = \frac{1}{M} \sum_{m=0}^{M-1} \check{P}^m(f) \quad (2.9)$$

where $\check{P}^m(f)$ is the sample spectrum of the weighted m th segment

$$\check{P}^m = \frac{1}{LWT} \left| T \sum_{n=0}^{L-1} x^m[n] \exp(-j2\pi f n T) \right|^2 \quad (2.10)$$

and L is the data length of each segment. $\frac{1}{T}$ is the sampling rate. W is the discrete-time window energy

$$W = T \sum_{n=0}^{L-1} w^2[n] \quad (2.11)$$

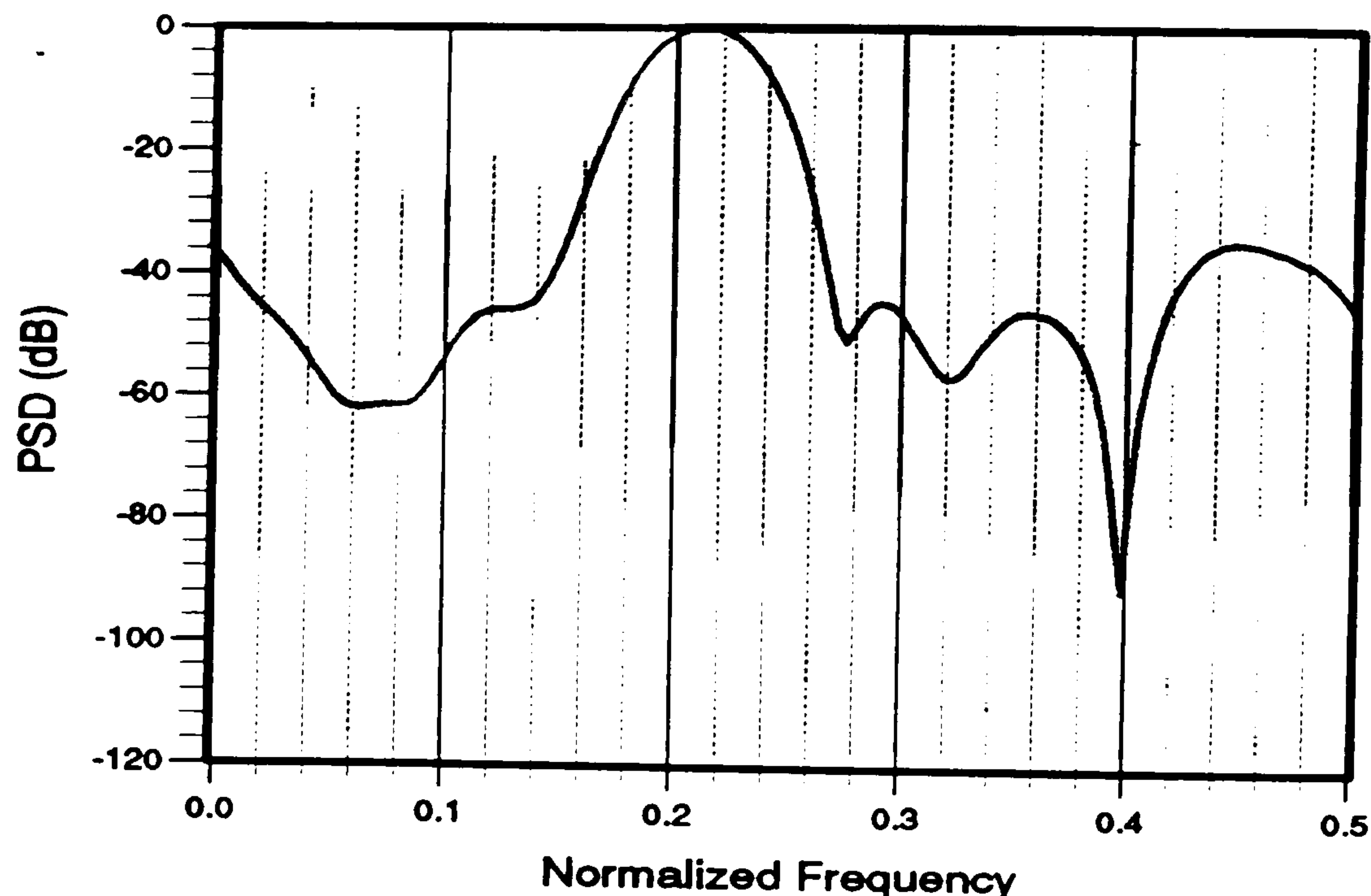


Figure 2.4: The Welch periodogram with $N = 32$, $L = 20$, $S = 10$

x^m is the weighted m th segment samples

$$x^m[n] = w[n]x[n + mS] \quad (2.12)$$

where S is the shift between each segment. If there are N samples then the maximum number of segments M is given by the integer part of $\frac{N-L}{S} + 1$

2.2.1.2 Blackman-Tukey Method The power spectrum density (PSD) is formally defined as the discrete time Fourier transform of the autocorrelation sequence:

$$P_{xx}(f) = T \sum_{m=-\infty}^{\infty} r_{xx}[m] \exp(-j2\pi f m T) \quad (2.13)$$

The Blackman-Tukey method simply substitutes the infinite sequence of unknown true autocorrelation values by a finite sequence of autocorrelation estimations and applies a real lag window to the estimated autocorrelation sequence:

$$\hat{P}_{xx}(f) = T \sum_{m=-L}^L w[m] \hat{r}_{xx}[m] \exp(-j2\pi f m T) \quad (2.14)$$

defined for $-\frac{1}{2T} \leq f \leq \frac{1}{2T}$ where the estimation of the autocorrelation sequence can be an unbiased estimate:

$$\hat{r}_{ubxx}[m] = \frac{1}{(N-m)} \sum_{n=0}^{N-m-1} x[n+m]x^*[n] \quad (2.15)$$

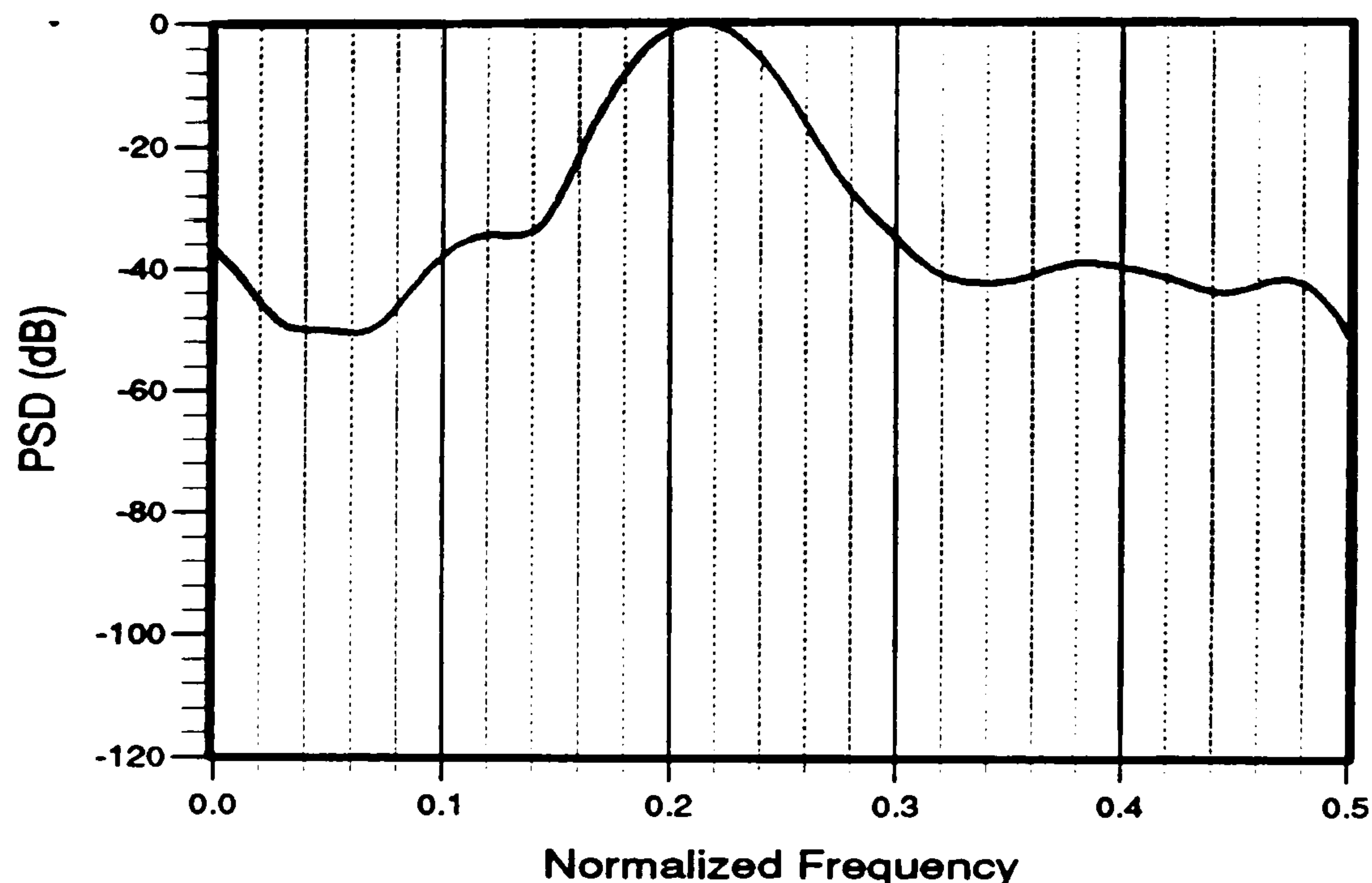


Figure 2.5: The Blackman-Tukey spectrum with Hamming window

or a biased estimate:

$$\hat{r}_{bxx}[m] = \frac{1}{N} \sum_{n=0}^{N-m-1} x[n+m]x^*[n] \quad (2.16)$$

for $0 \leq m \leq N - 1$. Where $*$ denotes complex conjugate.

The autocorrelation matrix formed from the biased autocorrelation is always positive definite, whereas the unbiased autocorrelation can not guarantee to produce a positive definite autocorrelation matrix which means certain matrix equations have no solutions. So the biased autocorrelation estimator is often the preferred estimator. PSD estimators that are based on forming correlation estimates first from the data, instead of working directly on the data like periodogram, are collectively termed *correlogram*.

2.2.2 Minimum Variance Spectral Estimation Method

The MVSE is also called Capon spectral estimator, or *Maximum Likelihood Method* (MLM) spectral estimator. It is originally developed for seismic array frequency-wavenumber analysis. The difference between the MVSE and periodogram is that the shape of narrowband filters in the periodogram is the same for all frequencies, whereas the the shape of the narrowband filters in the MVSE is dependent on the frequency under consideration, i. e. the filter sidelobes can be adjusted to reduce the

response to PSD components outside the band of interest. It has more resolution than Fourier estimators but less than that of an AR spectral estimator.

Consider a transversal filter with $p+1$ coefficients $a[0], a[1], a[2], \dots, a[p]$. The filter output is given by the convolution

$$y[n] = \sum_{k=0}^p a[k]x[n-k] = \mathbf{x}^T[n]\mathbf{a} \quad (2.17)$$

Where $\mathbf{x}$ is the input data vector $\mathbf{x} = (x[n], x[n-1], \dots, x[n-p])^T$. $\mathbf{a}$ is the tap-weight vector $\mathbf{a} = (a[0], a[1], \dots, a[p])^T$. The variance of the filter output is

$$\begin{aligned} \rho &= \xi(|y[n]|^2) \\ &= \xi(\mathbf{a}^H \mathbf{x}^*[n] \mathbf{x}^T[n] \mathbf{a}) \\ &= \mathbf{a}^H \mathbf{R}_p \mathbf{a} \end{aligned} \quad (2.18)$$

where $\mathbf{R}$ is a $(p+1) \times (p+1)$ Toeplitz autocorrelation matrix

$$\mathbf{R}_p = \begin{bmatrix} \hat{r}_{xx}[0] & \hat{r}_{xx}^*[1] & \cdots & \hat{r}_{xx}^*[p] \\ \hat{r}_{xx}[1] & \hat{r}_{xx}[0] & \cdots & \hat{r}_{xx}^*[p-1] \\ \vdots & \vdots & \ddots & \vdots \\ \hat{r}_{xx}[p] & \hat{r}_{xx}[p-1] & \cdots & \hat{r}_{xx}[0] \end{bmatrix} \quad (2.19)$$

The coefficients are constrained so that the magnitude of the frequency response of the filter at the frequency f_i under consideration is unity, i. e. $\mathbf{e}^H(f_i)\mathbf{a} = 1$, and at the same time the variance of the output is minimum, where $\mathbf{e}(f_i) = (1, \exp(j2\pi f_i T), \dots, \exp(j2\pi f_i pT))^T$ is the complex sinusoid vector. Define a cost function as $P = \mathbf{a}^H \mathbf{R}_p \mathbf{a} + \lambda(1 - \mathbf{e}^H(f_i)\mathbf{a})$. Where λ is the Lagrange multiplier. The optimum weight $\mathbf{a}$ is obtained by taking the derivative of P with respect to $\mathbf{a}$ and set the result to zero. It can be shown [23] that the $\mathbf{a}_{MV}$ thus obtained is given by

$$\mathbf{a} = \frac{\mathbf{R}_p^{-1} \mathbf{e}(f_i)}{\mathbf{e}^H(f_i) \mathbf{R}_p^{-1} \mathbf{e}(f_i)} \quad (2.20)$$

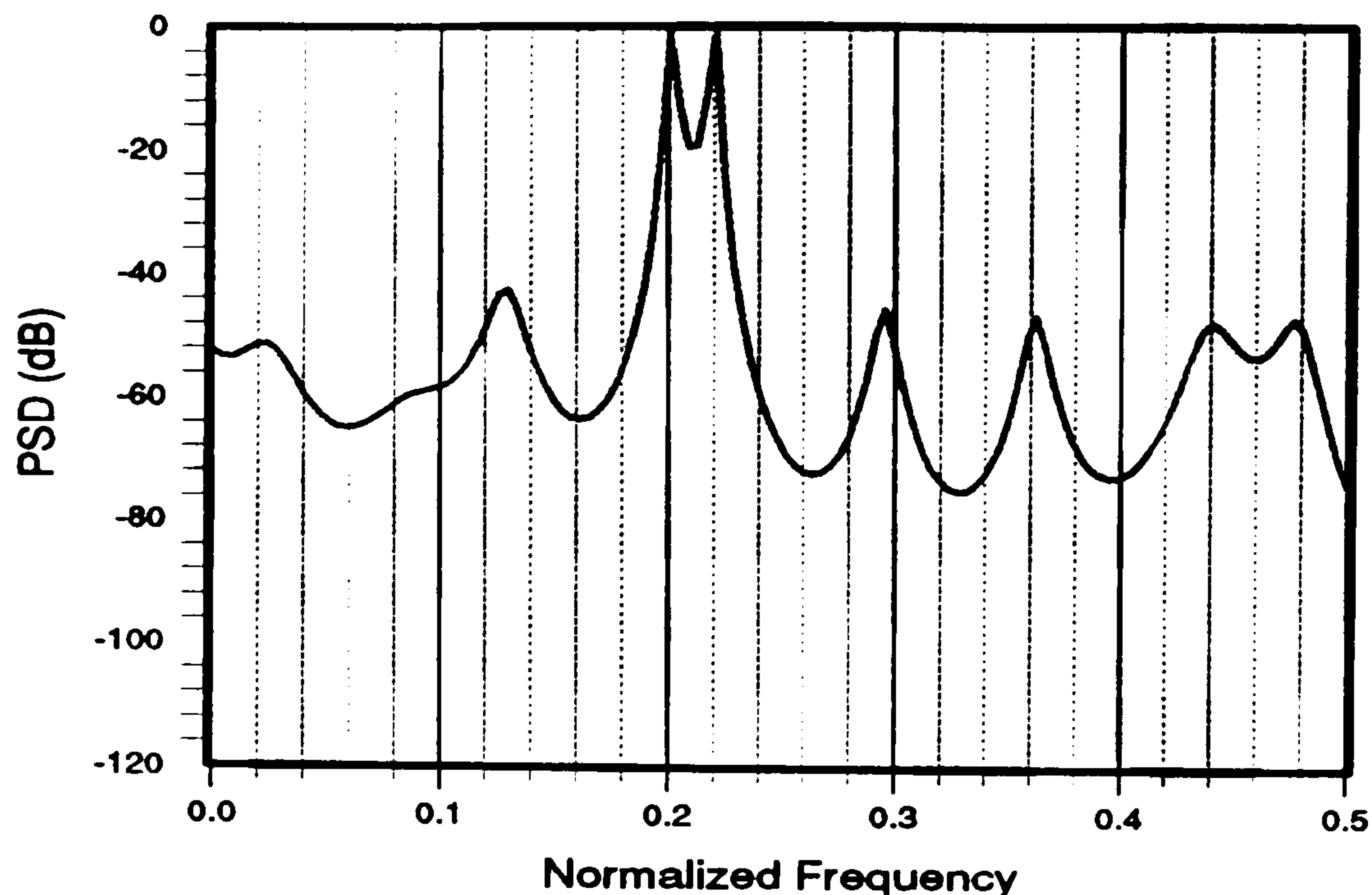


Figure 2.6: The MVSE spectrum

Substitute (2.20) into (2.18) gives the minimum variance

$$\rho_{MV} = \frac{1}{\mathbf{e}^H(f_i) \mathbf{R}_p^{-1} \mathbf{e}(f_i)} \quad (2.21)$$

The minimum variance spectral estimator is defined as

$$P_{MV}(f) = T \rho_{MV} = \frac{T}{\mathbf{e}^H(f) \mathbf{R}_p^{-1} \mathbf{e}(f)} \quad (2.22)$$

When input data consists of sinusoids embedded in white noise and the SNR is high, peaks in the MVSE are approximately linearly proportional to the power of the sinusoids. This is in contrast to the AR PSD estimators that are proportional to the square of the power.

2.3 Parametric Methods

The methods described in this section are called parametric methods because they assume a model of the data. If the data are correctly modeled then it can be described by just a few parameters. The eigenanalysis-based methods which model the data as sinusoids corrupted by noise are currently attracting very much research interest. Because these methods can give very good frequency estimation in temporal data analysis [6] or direction of arrival estimation in the spatial-temporal data analysis [7],[13],[21]. The performance of these estimators, however, are dependent on the validity of the model .

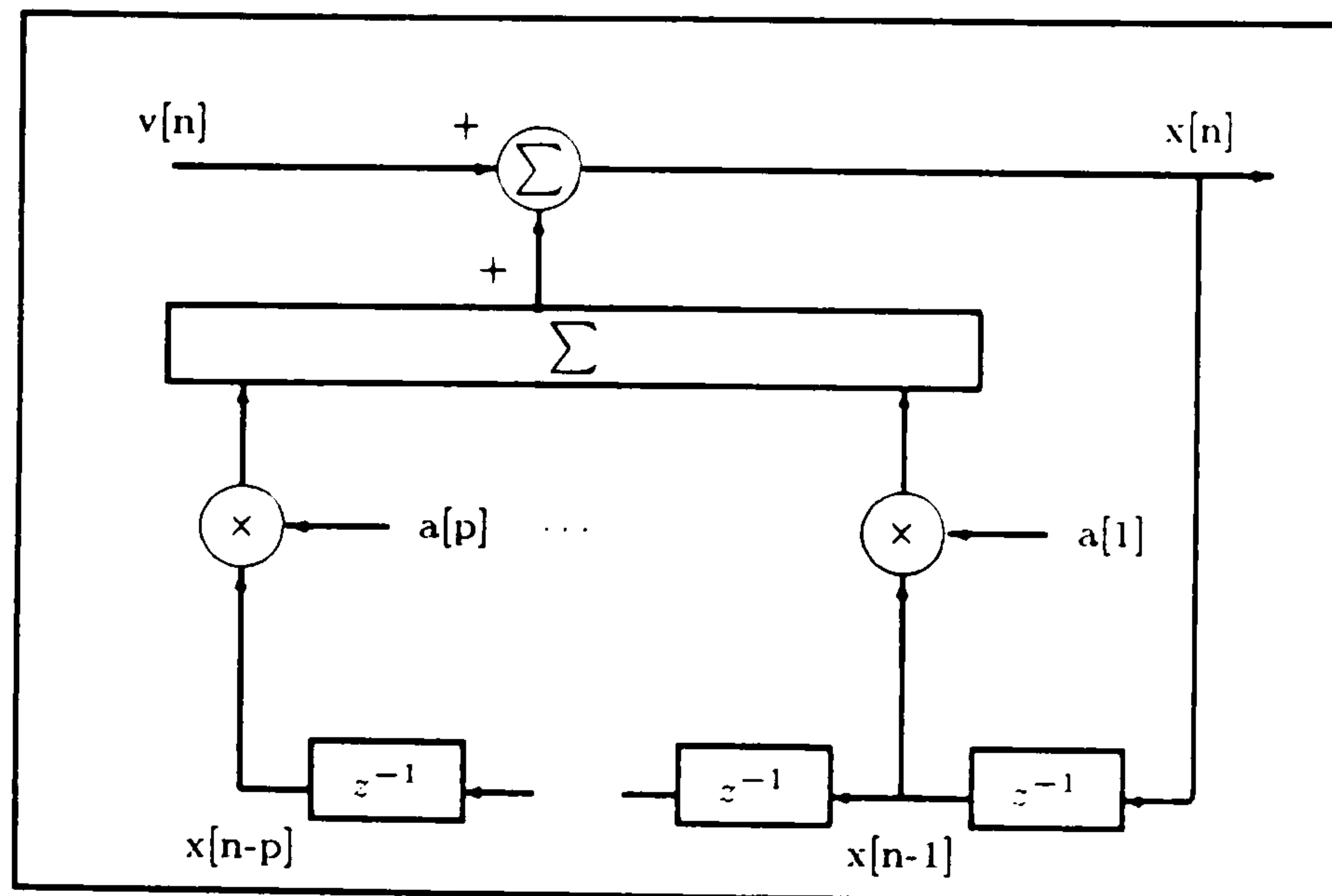


Figure 2.7: AR process

2.3.1 Autoregressive PSD Methods

The most popular rational transfer function model is the autoregressive model, see figure 2.7. To find the parameters of a AR model is numerically easier than to find the parameters of a MA or ARMA models. The AR process of order p is described by the equation:

$$x[n] = \sum_{k=0}^p a[k]x[n-k] + v[n] \quad (2.23)$$

The system function in this case is

$$H(z) = \frac{1}{A(z)} \quad (2.24)$$

Where $A(z)$ is the z -transform of the coefficients:

$$A(z) = \sum_{k=0}^p a[k]z^{-k}$$

The z -transform of the autocorrelation sequence at the output of a linear filter is related to the z -transform of the autocorrelation sequence at the input by

$$\begin{aligned} P_{xx}(z) &= P_{vv}H(z)H^*(1/z^*) \\ &= \frac{P_{vv}}{A(z)A^*(1/z^*)} \end{aligned} \quad (2.25)$$

The PSD of the output is obtained by substituting $z = \exp(j2\pi f)$ into (2.25). The driving process $v[n]$ is assumed to be a white noise sequence of zero-mean with variance

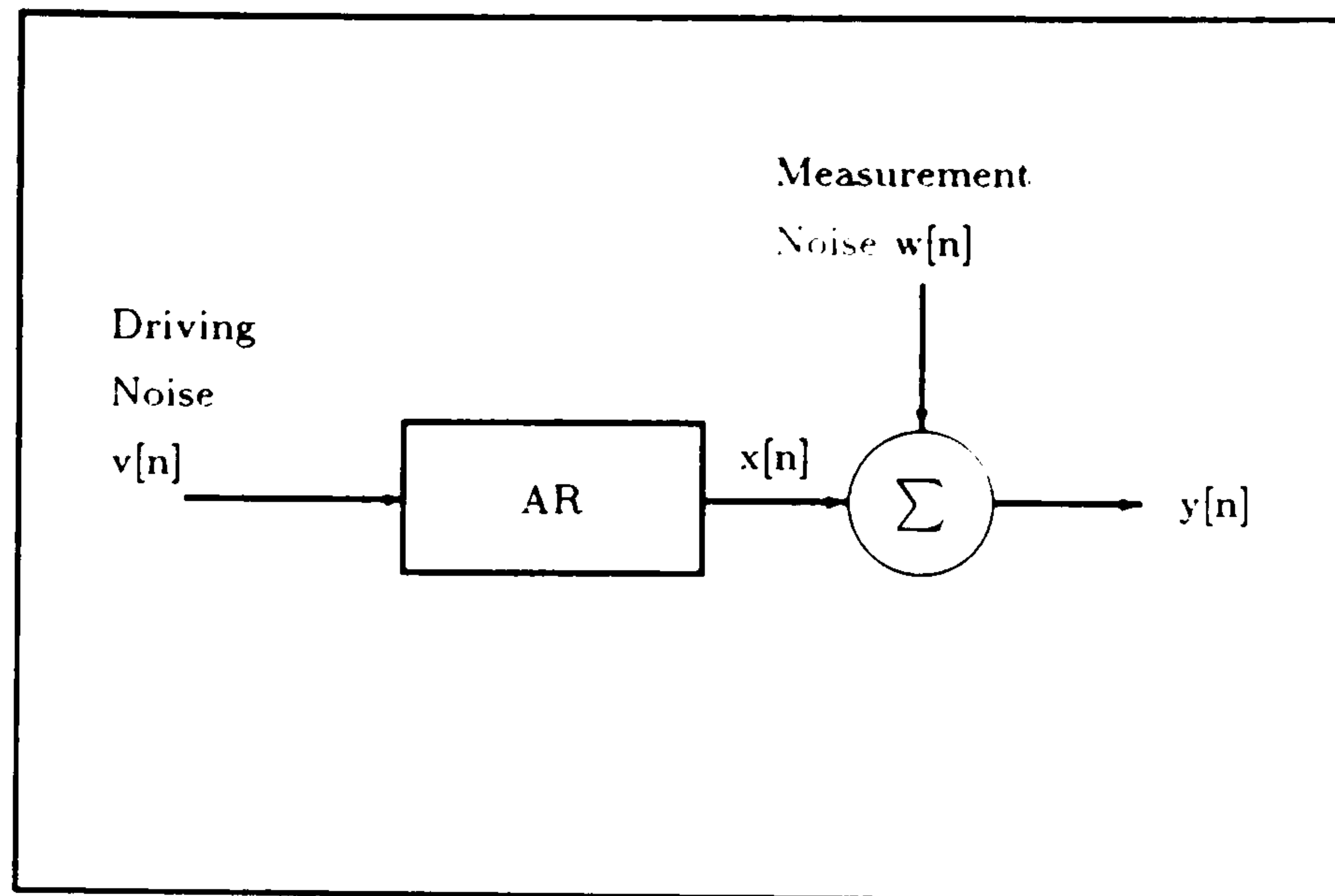


Figure 2.8: The AR process corrupted by additive noise

of σ_v^2 . The PSD of the input is then σ_v^2 . i. e.

$$P_{AR}(f) = \frac{\sigma_v^2}{|A(f)|^2} \quad (2.26)$$

Note that $v[n]$ is an inherent part of the model which is different from the additive noise (e. g. measurement noise). When an AR process is corrupted by additive noise (e. g. measurement noise) ,see figure 2.8, it is better described by the ARMA. Since the transfer function

$$\frac{\sigma^2}{A(z)} + w[n] = \frac{\sigma^2 + w[n]A[z]}{A(z)} \quad (2.27)$$

is a ARMA process. The detection and resolution performance of AR PSD estimators degrade in this situation.

To find the relationship between the AR parameters and the autocorrelation function of the data we can multiply (2.23) by $x^*[n - k]$ and take the expectation as follows;

$$\xi\{x^*[n - k]x[n]\} = \xi\{x^*[n - k](-\sum_{i=1}^p a[i]x[n - i] + v[n])\} \quad (2.28)$$

$$r_{xx}[k] = \begin{cases} -\sum_{i=1}^p a[i]r_{xx}[k - i] & \text{for } k \geq 1 \\ -\sum_{i=1}^p a[i]r_{xx}[-i] + \sigma^2 & \text{for } k = 0 \end{cases} \quad (2.29)$$

This is the Yule-Walker equation for the AR process. The AR parameters, in this case, are derived from the autocorrelation sequence. From the Yule-Walker equation we can

see the reason for the AR PSD estimators high resolution capability. The autocorrelation lags from 0 to p uniquely describe the autoregressive process of order p . After the coefficients $a[1], a[2], \dots, a[p]$ have been determined from $\hat{r}_{xx}[0], \hat{r}_{xx}[1], \dots, \hat{r}_{xx}[p]$, the lags for $|m| > p$ can be obtained recursively as shown by the Yule-Walker equation:

$$\hat{r}_{xx}[m] = - \sum_{k=1}^p a[k] \hat{r}_{xx}[m-k] \quad (2.30)$$

This implies that an alternative expression for the AR PSD is

$$\begin{aligned} P_{AR}(f) &= \frac{\rho^2}{|A(f)|^2} \\ &= \sum_{m=-\infty}^{\infty} \hat{r}_{xx}[m] \exp(j2\pi fm) \end{aligned} \quad (2.31)$$

The implied extrapolation of the ACF after $|m| > p$ is the basis for the high-resolution capability of the AR PSD estimator. There is no windowing of the autocorrelation sequence and thus no accompanying sidelobes as there are for the classical PSD estimators.

The other way to find the AR parameters is based directly on the data without calculating the autocorrelation sequence first. The linear prediction method is commonly used to formulate the equation directly from the data samples and the least squares algorithm is then employed to solve the equation for AR parameters. When the amount of data are limited and the estimation of the autocorrelation function is not accurate, the least squares methods usually give better results.

When a transversal filter of length M is used as a linear prediction error filter, see figure 2.9, the salient result of the linear prediction is the *normal equation* which states that, when the transversal filter operates in its optimum condition, the estimation error vector ε is normal to the estimation vector $\hat{\mathbf{d}}$; hence the name “normal equation”. This can be seen graphically from figure 2.10. ε has minimum length only when it is perpendicular to the estimate $\hat{\mathbf{d}}$. Where the tap-weight vector $\mathbf{w}$ is defined as

$$\mathbf{w}^T = (w_1, w_2, \dots, w_M)$$

The input vector is defined as

$$\mathbf{u}^T[i] = (u[i], u[i-1], \dots, u[i-M+1])$$

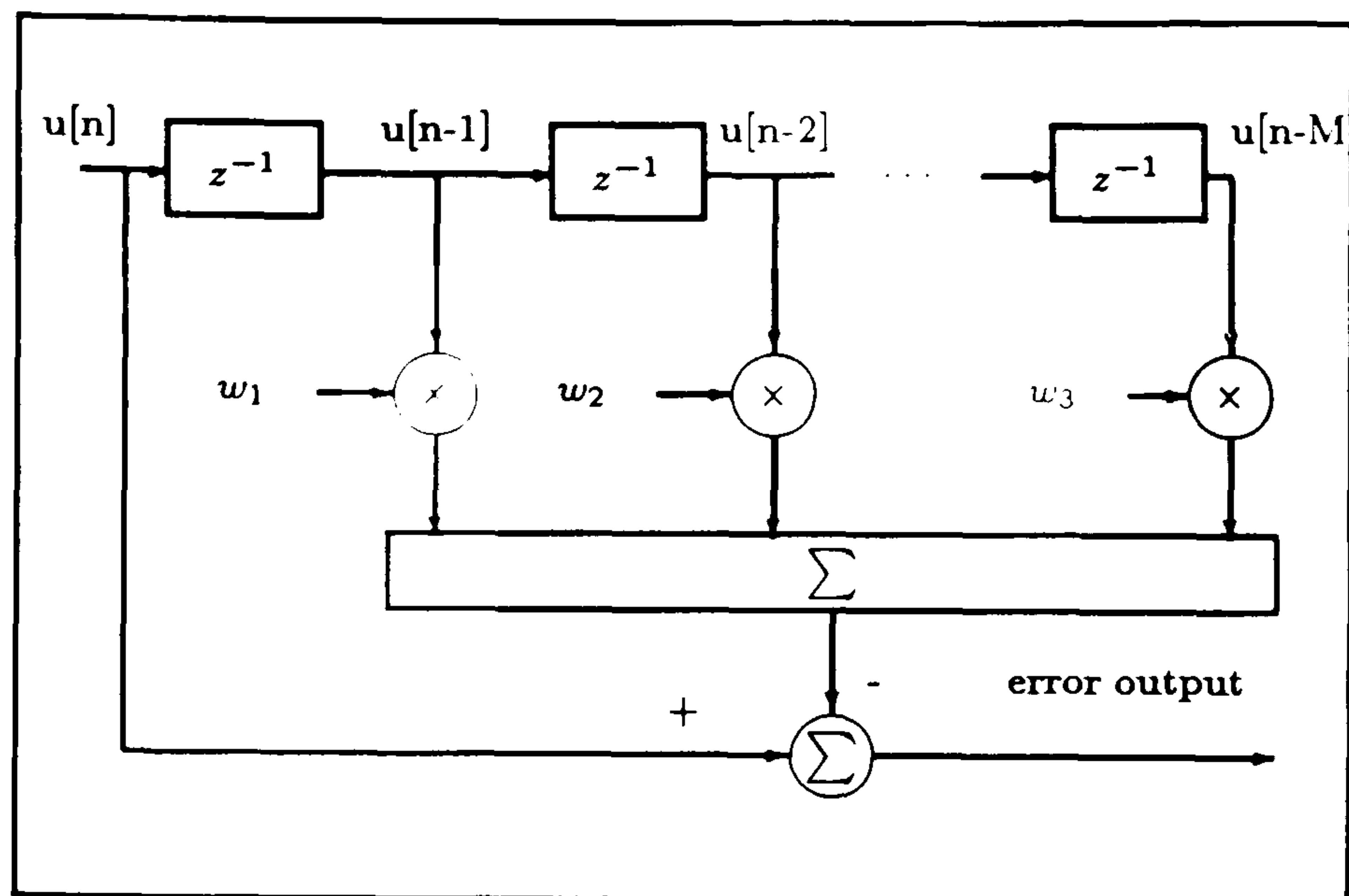


Figure 2.9: The prediction error filter

The desired value at time i is $d[i]$. We can write the estimation error as

$$e[i] = d[i] - \mathbf{w}^H \mathbf{u} \quad M \leq i \leq L \quad (2.32)$$

The estimation error vector $\boldsymbol{\varepsilon}$ is defined as

$$\boldsymbol{\varepsilon}^H = (e[M], e[M+1], \dots, e[L])$$

The limit M and L define the error terms of interest. The desired response vector $\mathbf{d}$ is defined as

$$\mathbf{d}^H = (d[M], d[M+1], \dots, d[L])$$

We can then write (2.32) in matrix form

$$\boldsymbol{\varepsilon} = \mathbf{d} - \mathbf{A}\mathbf{w} \quad (2.33)$$

where

$$\mathbf{A}^H = (\mathbf{u}[M], \mathbf{u}[M+1], \dots, \mathbf{u}[L])$$

Let $\boldsymbol{\varepsilon}_{min}$ denote the minimum error vector when the tap-weight vector is the optimum weight $\mathbf{w}_{opt}$

$$\boldsymbol{\varepsilon}_{min} = \mathbf{d} - \mathbf{A}\mathbf{w}_{opt} \quad (2.34)$$

The normal equation states that

$$\begin{aligned} \hat{\mathbf{d}}^H \boldsymbol{\varepsilon}_{min} &= 0 \\ &= \mathbf{w}^H \mathbf{A}^H (\mathbf{d} - \mathbf{A}\hat{\mathbf{w}}_{opt}) \\ &= \mathbf{w}^H \mathbf{A}^H \mathbf{d} - \mathbf{w}^H \mathbf{A}^H \mathbf{A} \hat{\mathbf{w}}_{opt} \end{aligned} \quad (2.35)$$

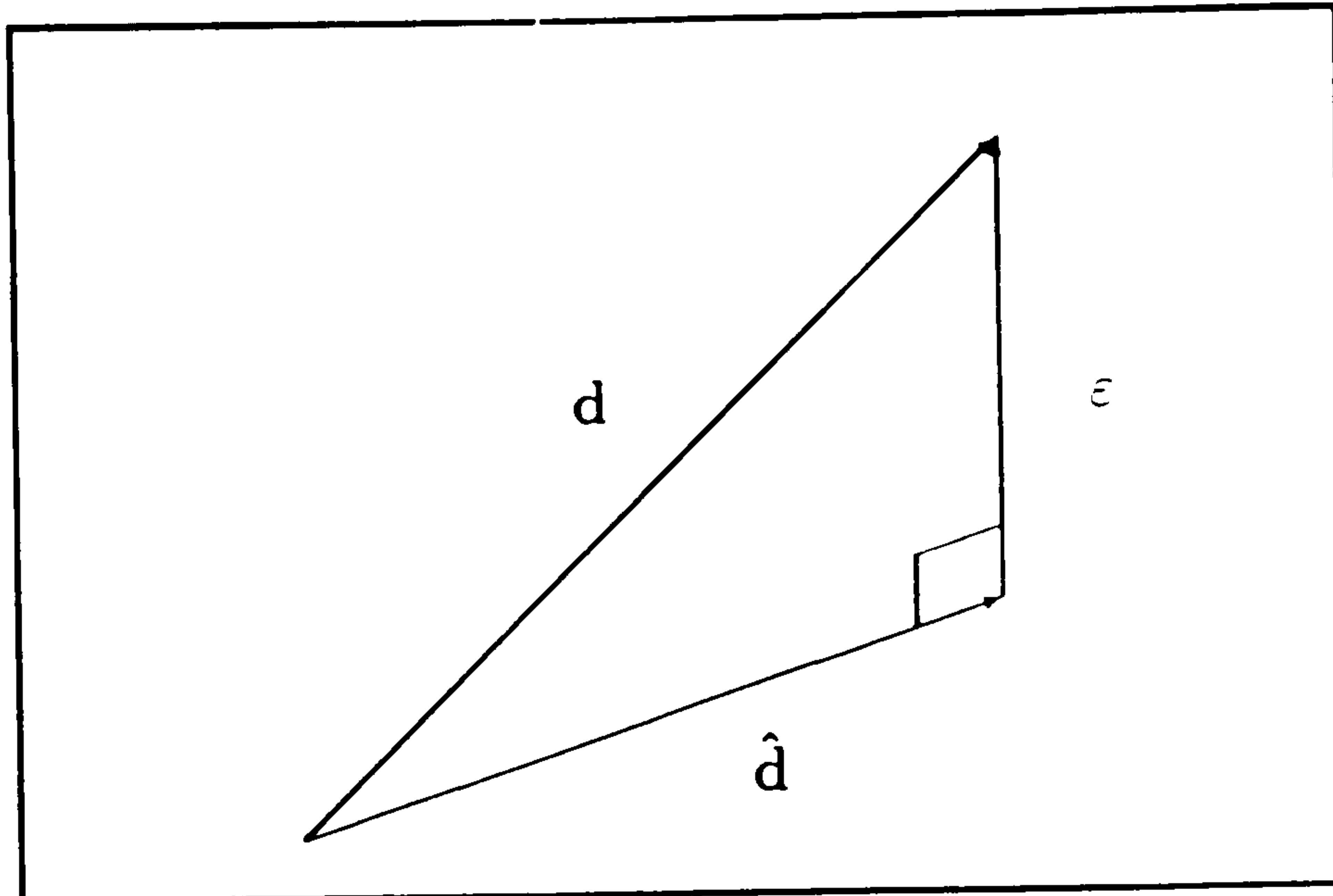


Figure 2.10: The orthogonality principle of the least square algorithm

i. e.

$$\mathbf{A}^H \mathbf{A} \mathbf{w}_{opt} = \mathbf{A}^H \mathbf{d} \quad (2.36)$$

or

$$\mathbf{A}^H \boldsymbol{\varepsilon}_{min} = 0 \quad (2.37)$$

From (2.37) another interpretation of the normal equation is that the error vector is orthogonal to the tap-input vector when it is operating at its optimum condition. This is also known as the *orthogonal principle of the least squares algorithm*. (2.36) will be used by several algorithms described below. The model order of all AR PSD estimators in this section is $p = 16$ which is $\frac{1}{2}$ of the data length.

2.3.1.1 Autocorrelation Method In this method the AR parameters are estimated by minimizing an estimate of the prediction error power

$$\hat{\rho} = \frac{1}{N} \sum_{n=-\infty}^{\infty} |x[n] + \sum_{k=1}^p a[k]x[n-k]|^2 \quad (2.38)$$

It is assumed that the data $\{x[0], x[1], \dots, x[N-1]\}$ are observed. Those not in the range $0 \leq n \leq N-1$ are set equal to zero. The estimated prediction error power is minimized by differentiating (2.38) with respect to the coefficient $a[k]$. From the

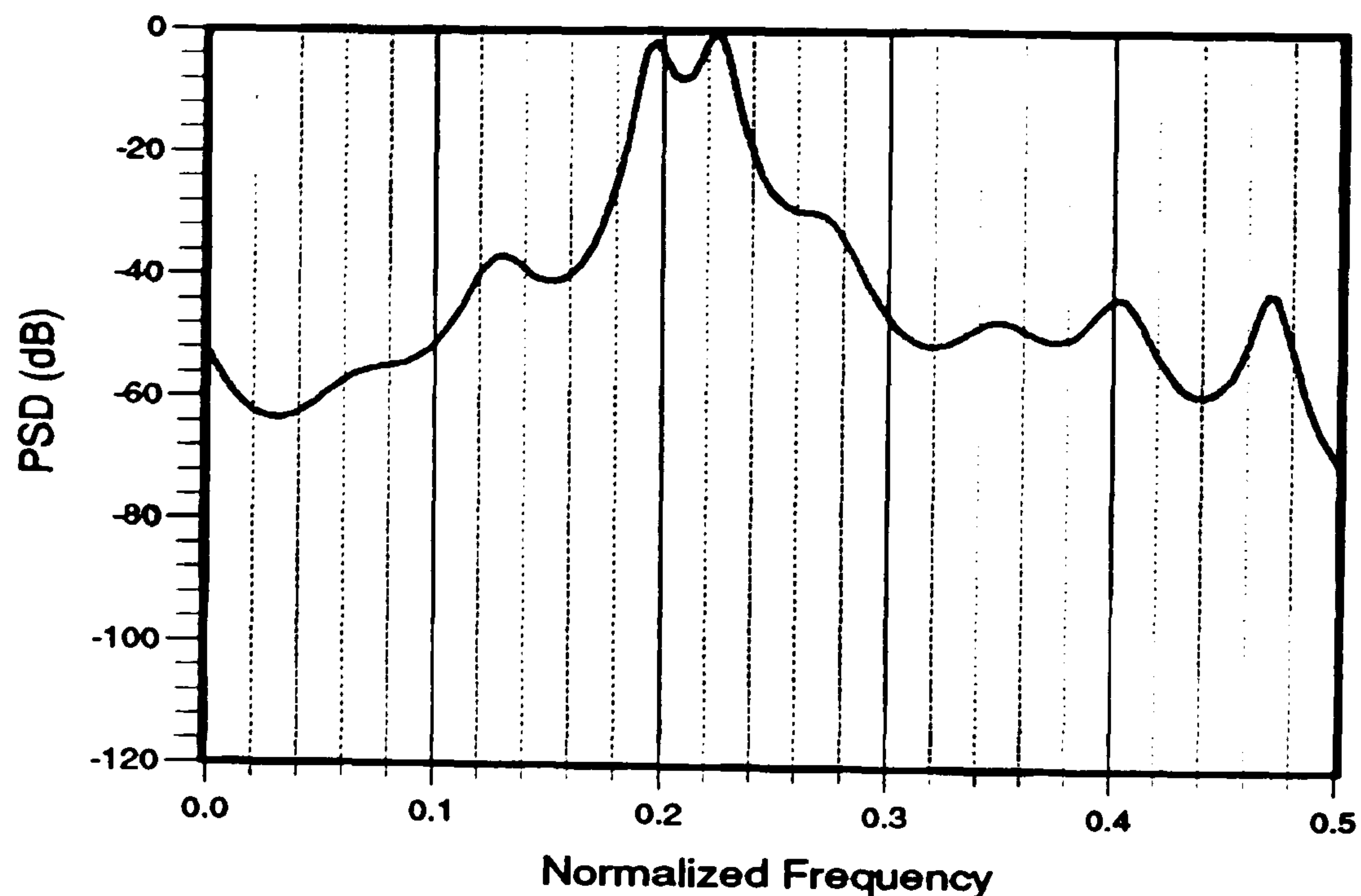


Figure 2.11: The autocorrelation method spectrum

normal equation (2.37)

$$\frac{1}{N} \sum_{n=-\infty}^{\infty} (x[n] + \sum_{k=1}^p a[k]x[n-k])x^*[n-l] = 0, \quad l = 1, 2, \dots, p \quad (2.39)$$

, or in the matrix form

$$\begin{bmatrix} \hat{r}_{xx}[0] & \hat{r}_{xx}[-1] & \cdots & \hat{r}_{xx}[-p+1] \\ \hat{r}_{xx}[1] & \hat{r}_{xx}[0] & \cdots & \hat{r}_{xx}[-p] \\ \vdots & \vdots & \ddots & \vdots \\ \hat{r}_{xx}[p-1] & \hat{r}_{xx}[p-2] & \cdots & \hat{r}_{xx}[0] \end{bmatrix} \begin{bmatrix} a[1] \\ a[2] \\ \vdots \\ a[p] \end{bmatrix} = - \begin{bmatrix} \hat{r}_{xx}[1] \\ \hat{r}_{xx}[2] \\ \vdots \\ \hat{r}_{xx}[p] \end{bmatrix} \quad (2.40)$$

where

$$\hat{r}_{xx}[k] = \begin{cases} \frac{1}{N} \sum_{n=0}^{N-k-1} x^*[n]x[n+k] & \text{for } k = 0, 1, \dots, p \\ \hat{r}_{xx}^*[-k] & \text{for } k = -1, -2, \dots, -(p-1) \end{cases} \quad (2.41)$$

The Levinson recursion may be used to solve (2.40). The estimate of the white noise variance $\hat{\sigma}^2$ is given by

$$\begin{aligned} \hat{\sigma}^2 &= \hat{\rho}_{min} \\ &= \frac{1}{N} \sum_{n=-\infty}^{\infty} |x[n] + \sum_{k=1}^p \hat{a}[k]x[n-k]|^2 \\ &= \frac{1}{N} \sum_{n=-\infty}^{\infty} (x[n] + \sum_{k=1}^p \hat{a}[k]x[n-k])(x[n] + \sum_{k=1}^p \hat{a}[k]x[n-k])^* \end{aligned}$$

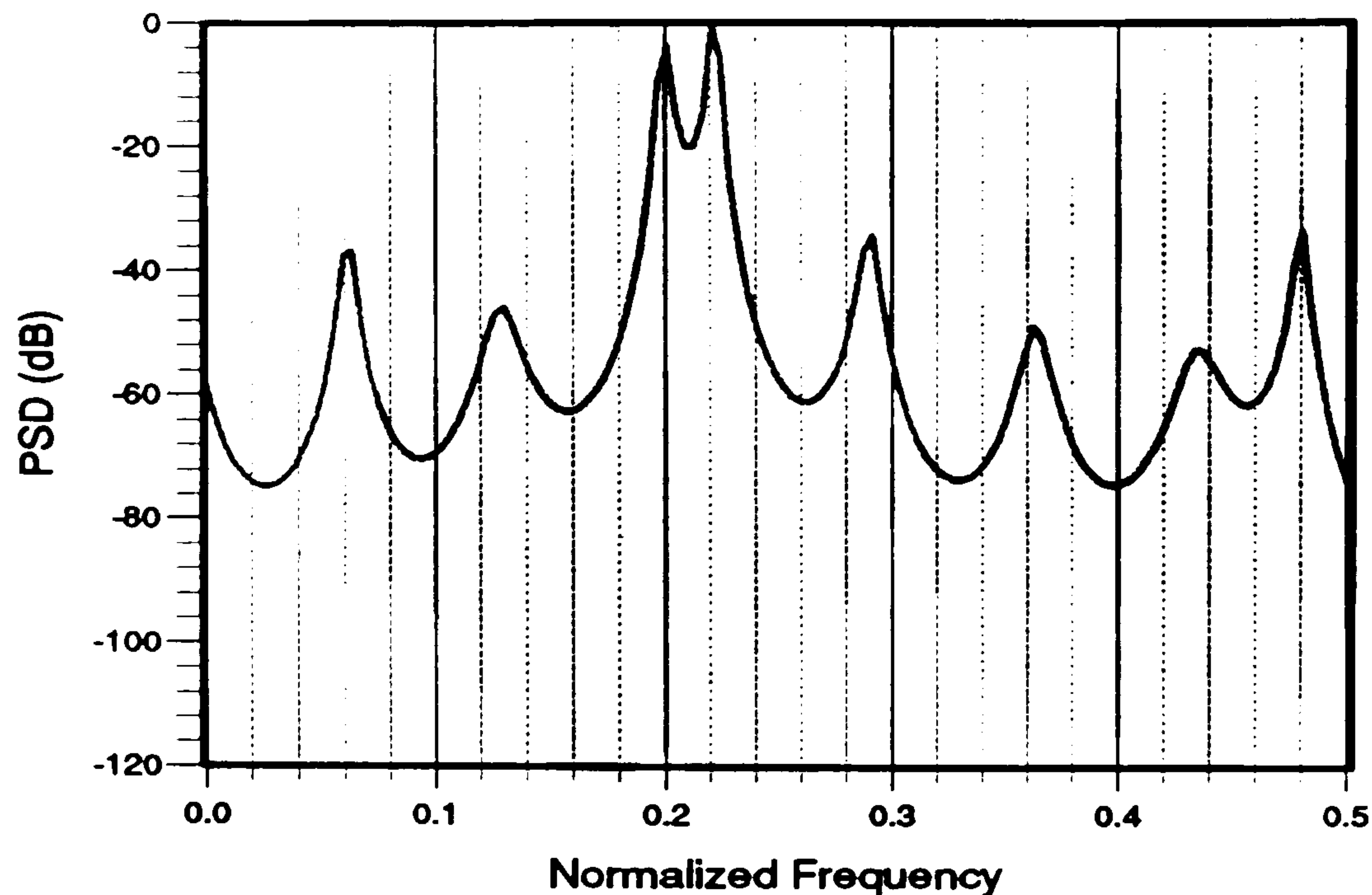


Figure 2.12: The covariance method spectrum

$$\begin{aligned}
 &= \frac{1}{N} \sum_{n=-\infty}^{\infty} [(x[n] + \sum_{k=1}^p \hat{a}[k]x[n-k])x^*[n] + \\
 &\quad (x[n] + \sum_{k=1}^p \hat{a}[k]x[n-k]) \sum_{k=1}^p \hat{a}^*[k]x^*[n-k]]
 \end{aligned} \tag{2.42}$$

The notation $\hat{\rho}_{min}$ is used to indicate when the transversal filter is operating at its optimum condition, the variance of the output is minimum. The second term is zero from the normal equation (2.39). Therefore

$$\hat{\sigma}^2 = \hat{r}_{xx}[0] + \sum_{k=1}^p \hat{a}[k]\hat{r}_{xx}[-k] \tag{2.43}$$

It can be seen from (2.41) that in the estimation of autocorrelation sequence $\hat{r}_{xx}[n]$ the higher index terms will have fewer terms for averaging. This will cause poor resolution. Therefore this method is not recommended for short data records.

2.3.1.2 Covariance Method In the covariance method all the data points to compute $\hat{\rho}$ are observed. No zeroing of the data is necessary. The AR parameters are found by minimizing the estimate of the prediction error:

$$\hat{\rho} = \frac{1}{N-P} \sum_{n=P}^{N-1} |x[n] + \sum_{k=1}^p a[k]x[n-k]|^2 \tag{2.44}$$

From the normal equation (2.36) we can get the following relation:

$$\begin{bmatrix} \hat{c}_{xx}[1,1] & \hat{c}_{xx}[1,2] & \cdots & \hat{c}_{xx}[1,p] \\ \hat{c}_{xx}[2,1] & \hat{c}_{xx}[2,2] & \cdots & \hat{c}_{xx}[2,p] \\ \vdots & \vdots & \vdots & \vdots \\ \hat{c}_{xx}[p,1] & \hat{c}_{xx}[p,2] & \cdots & \hat{c}_{xx}[p,p] \end{bmatrix} \begin{bmatrix} \hat{a}[1] \\ \hat{a}[2] \\ \vdots \\ \hat{a}[p] \end{bmatrix} = - \begin{bmatrix} \hat{c}_{xx}[1,0] \\ \hat{c}_{xx}[2,0] \\ \vdots \\ \hat{c}_{xx}[p,0] \end{bmatrix} \quad (2.45)$$

where

$$\hat{c}_{xx}[j,k] = \frac{1}{N-p} \sum_{n=p}^{N-1} x^*[n-j]x[n-k] \quad (2.46)$$

and the white noise is estimated as

$$\begin{aligned} \hat{\sigma}^2 &= \hat{\rho}_{min} \\ &= \hat{c}_{xx}[0,0] + \sum_{k=1}^p \hat{a}[k] \hat{c}_{xx}[0,k] \end{aligned} \quad (2.47)$$

The left side matrix in (2.45) is Hermitian (i. e. $c_{xx}[j,k] = c_{xx}^*[j,k]$) and positive semidefinite. It may be solved by Cholesky decomposition. The resolution of the covariance method exceeds that of the autocorrelation method.

2.3.1.3 Modified Covariance Method The modified covariance method estimates the AR parameters by minimizing the average of the estimated forward and backward prediction error powers i. e.

$$\hat{\rho} = \frac{1}{2}(\hat{\rho}^f + \hat{\rho}^b) \quad (2.48)$$

where

$$\hat{\rho}^f = \frac{1}{N-p} \sum_{n=p}^{N-1} |x[n] + \sum_{k=1}^p a[k]x[n-k]|^2 \quad (2.49)$$

$$\hat{\rho}^b = \frac{1}{N-p} \sum_{n=0}^{N-p-1} |x[n] + \sum_{k=1}^p a^*[k]x[n+k]|^2 \quad (2.50)$$

As in the covariance method the summations only involve observed data. To minimize the prediction error power $\hat{\rho}$ is differentiated with respect to $a[k]$ and the result is set

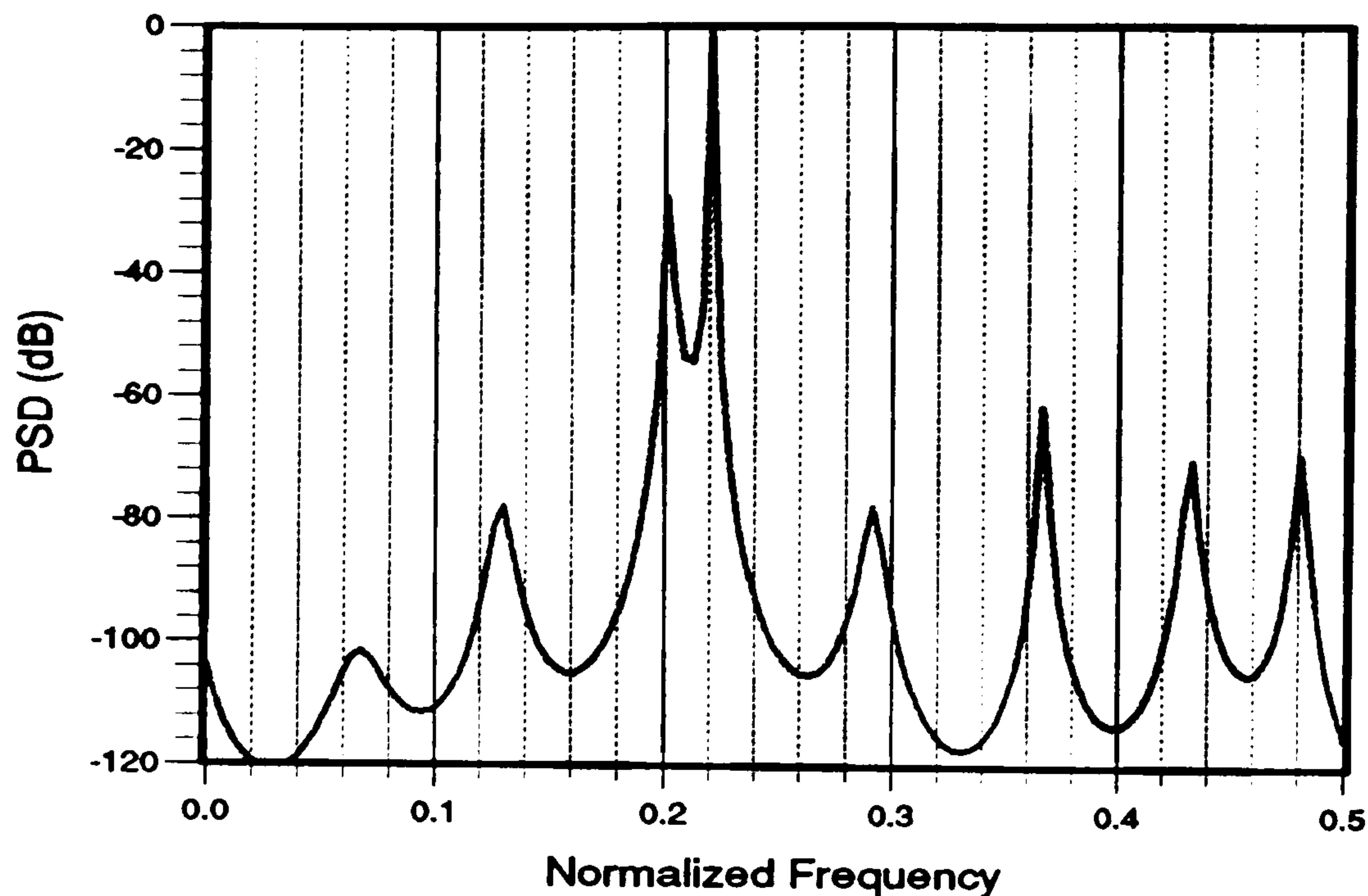


Figure 2.13: The modified covariance method spectrum

to zero. The result can be put in matrix form

$$\begin{bmatrix} \hat{c}_{xx}[1, 1] & \hat{c}_{xx}[1, 2] & \cdots & \hat{c}_{xx}[1, p] \\ \hat{c}_{xx}[2, 1] & \hat{c}_{xx}[2, 2] & \cdots & \hat{c}_{xx}[2, p] \\ \vdots & \vdots & \ddots & \vdots \\ \hat{c}_{xx}[p, 1] & \hat{c}_{xx}[p, 2] & \cdots & \hat{c}_{xx}[p, p] \end{bmatrix} \begin{bmatrix} \hat{a}[1] \\ \hat{a}[2] \\ \vdots \\ \hat{a}[p] \end{bmatrix} = - \begin{bmatrix} \hat{c}_{xx}[1, 0] \\ \hat{c}_{xx}[2, 0] \\ \vdots \\ \hat{c}_{xx}[p, 0] \end{bmatrix} \quad (2.51)$$

where

$$\hat{c}_{xx}[j, k] = \frac{1}{2(N-p)} \left(\sum_{n=p}^{N-1} x^*[n-j]x[n-k] + \sum_{n=0}^{N-p-1} x^*[n+j]x[n+k] \right) \quad (2.52)$$

and the white noise is estimated as

$$\hat{\sigma}^2 = \hat{c}_{xx}[0, 0] + \sum_{k=1}^p \hat{a}[k] \hat{c}_{xx}[0, k] \quad (2.53)$$

Due to the combination of forward and backward prediction error power which means more points to average, this method avoids the spectral line splitting problem and gives statistically stable estimates with high resolution.

2.3.1.4 Burg Method The Burg method estimates the reflection coefficients (k_k) and then uses the Levinson recursion to obtain the AR parameters. Burg proposed to estimate k_k by minimizing the average of the estimates of the forward and backward prediction error powers subject to the constraint that the $(k-1)^{st}$ order prediction error filter coefficients had already been obtained. i. e. to obtain the estimate

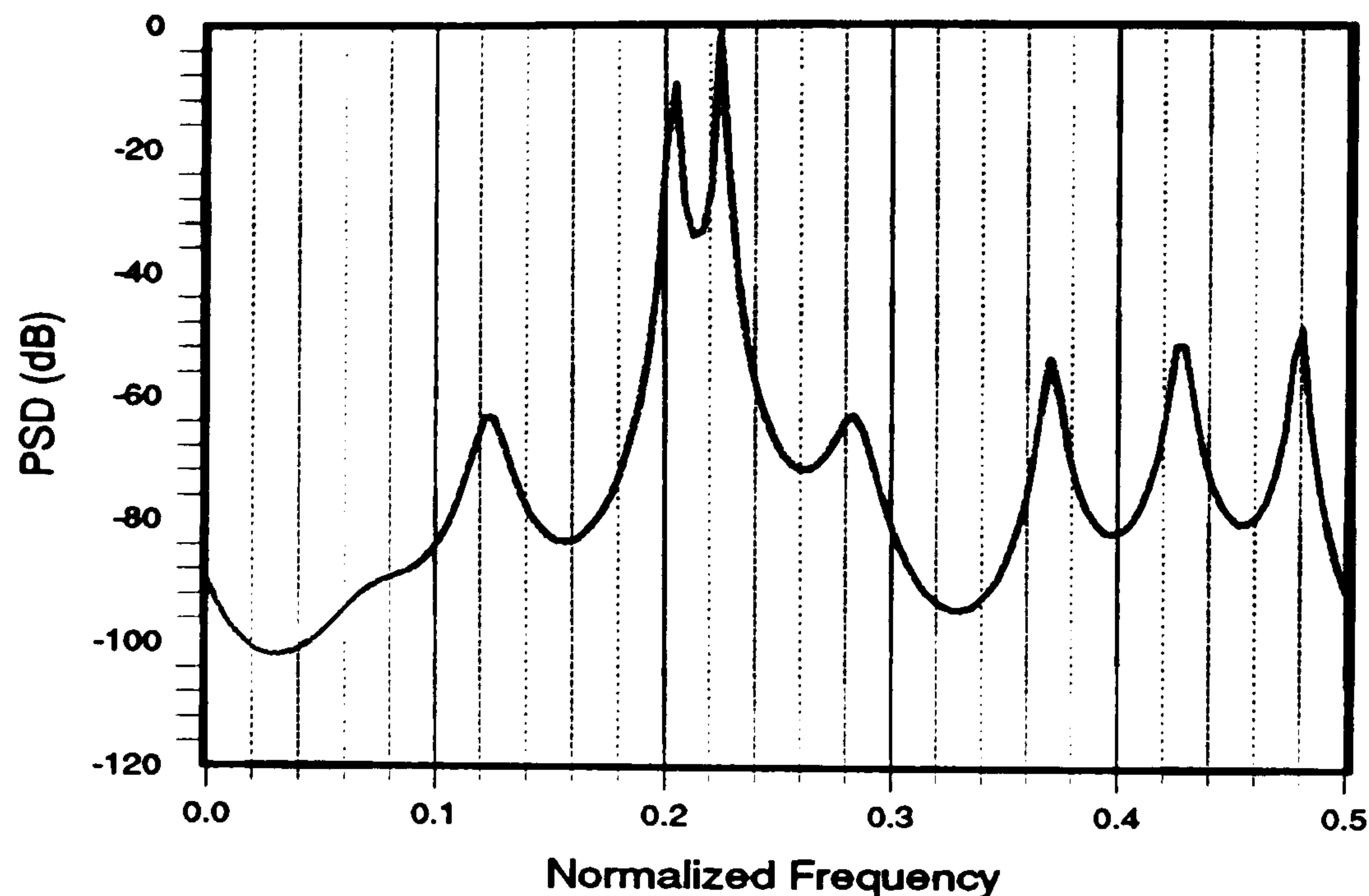


Figure 2.14: The Burg spectrum

of k_k we minimize

$$\hat{\rho}_k = \frac{1}{2}(\hat{\rho}_k^f + \hat{\rho}_k^b) \quad (2.54)$$

where

$$\hat{\rho}_k^f = \frac{1}{N-k} \sum_{n=k}^{N-1} |x[n] + \sum_{i=1}^k a_k[i]x[n-i]|^2 \quad (2.55)$$

$$\hat{\rho}_k^b = \frac{1}{N-k} \sum_{n=0}^{N-k-1} |x[n] + \sum_{i=1}^k a_k^*[i]x[n+i]|^2 \quad (2.56)$$

and

$$a_k[i] = \begin{cases} \hat{a}_{k-1}[i] + k_k \hat{a}_{k-1}^*[k-i] & \text{for } i = 1, 2, \dots, k-1 \\ \hat{k}_k & \text{for } i = k \end{cases} \quad (2.57)$$

$$\hat{\rho}_k = (1 - |\hat{a}_k[k]|^2) \hat{\rho}_{k-1} \quad (2.58)$$

Defining the prediction error as

$$\hat{e}_k^f[n] = x[n] + \sum_{i=1}^k a_k[i]x[n-i] \quad (2.59)$$

$$\hat{e}_k^b[n] = x[n-k] + \sum_{i=1}^k a_k[i]x[n-k+i] \quad (2.60)$$

then

$$\hat{\rho}_k^f = \frac{1}{N-k} \sum_{n=k}^{N-1} |\hat{e}_k^f[n]|^2 \quad (2.61)$$

$$\hat{\rho}_k^b = \frac{1}{N-k} \sum_{n=k}^{N-1} |\hat{e}_k^b[n]|^2 \quad (2.62)$$

From the lattice filter relation

$$\hat{e}_k^f[n] = \hat{e}_{k-1}^f[n] + k_k \hat{e}_{k-1}^b[n-1] \quad (2.63)$$

$$\hat{e}_k^b[n] = \hat{e}_{k-1}^b[n-1] + k_k^* \hat{e}_{k-1}^f[n] \quad (2.64)$$

and the averaged estimated prediction error power becomes

$$\hat{\rho}_k = \frac{1}{2(N-k)} \sum_{n=k}^{N-1} \left[|\hat{e}_{k-1}^f[n] + k_k \hat{e}_{k-1}^b[n-1]|^2 + |\hat{e}_{k-1}^b[n-1] + k_k^* \hat{e}_{k-1}^f[n]|^2 \right] \quad (2.65)$$

To find the optimum reflection coefficient we set the derivative of $\hat{\rho}_k$ with respect to k_k to zero

$$\begin{aligned} \frac{\partial \hat{\rho}_k}{\partial k_k} &= \frac{1}{N-k} \sum_{n=k}^{N-1} (\hat{e}_{k-1}^f[n] + k_k \hat{e}_{k-1}^b[n-1]) \hat{e}_{k-1}^{b*}[n-1] \\ &\quad + (\hat{e}_{k-1}^{b*}[n-1] + k_k^* \hat{e}_{k-1}^f[n]) \hat{e}_{k-1}^f[n] \\ &= 0 \end{aligned} \quad (2.66)$$

Solving for k_k we have

$$\hat{k}_k = \frac{-2 \sum_{n=k}^{N-1} \hat{e}_{k-1}^f[n] \hat{e}_{k-1}^{b*}[n-1]}{\sum_{n=k}^{N-1} (|\hat{e}_{k-1}^f[n]|^2 + |\hat{e}_{k-1}^b[n-1]|^2)} \quad (2.67)$$

(2.67) (2.58) and (2.57) constitute the Burg algorithm. For sinusoidal data, it has been observed that the Burg algorithm is subject to line splitting [18] and the peak locations are strongly dependent on phase. The modified covariance method can eliminate these deficiencies.

2.3.2 Moving Average Method

A moving average process of order q , see figure 2.15, is defined as

$$x[n] = \sum_{k=0}^q b[k] v[n-k] \quad (2.68)$$

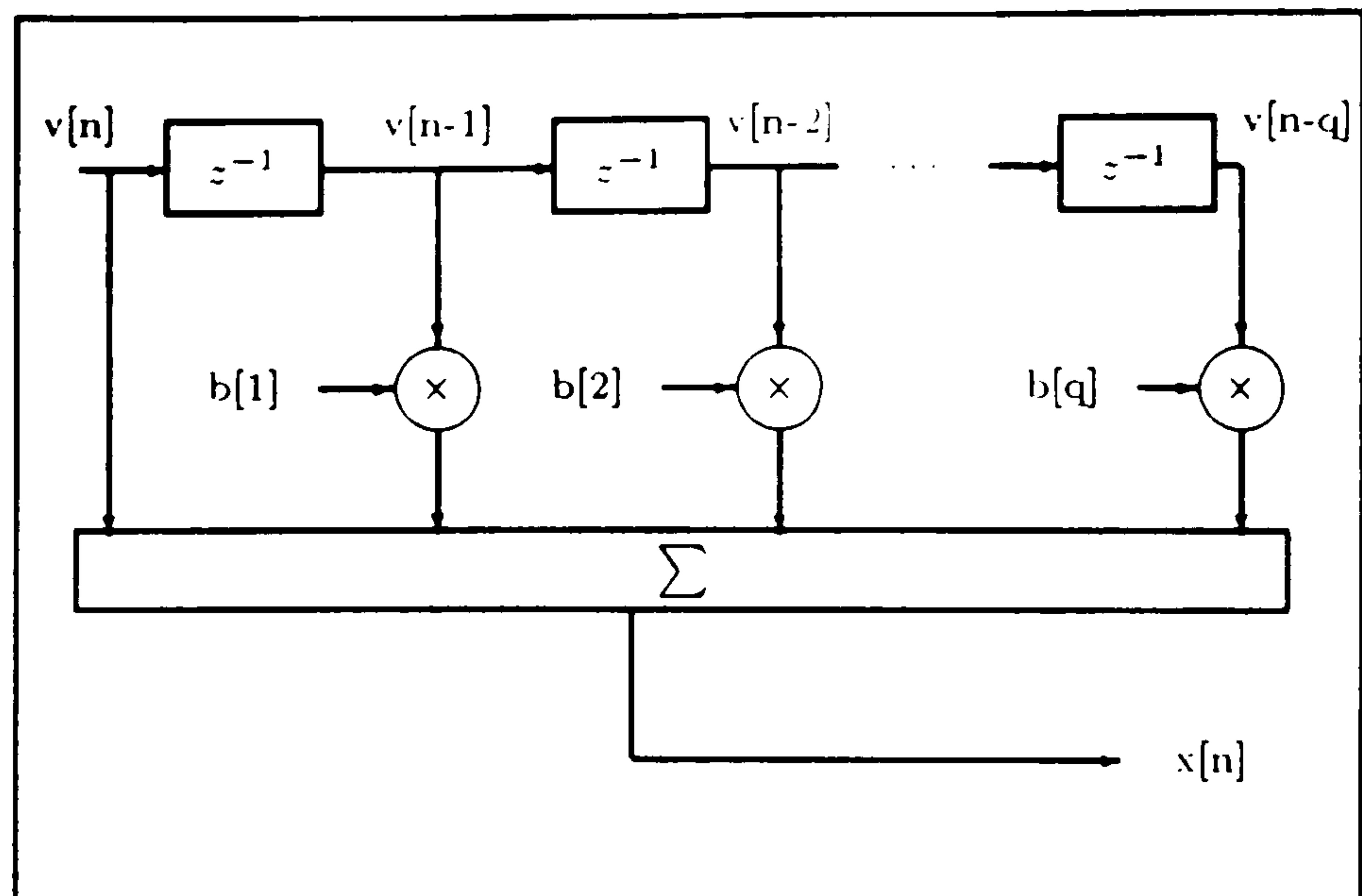


Figure 2.15: The MA process

The system function $H(z)$ is

$$\begin{aligned} H(z) &= B(z) \\ &= 1 + \sum_{k=1}^q b[k]z^{-k} \end{aligned} \quad (2.69)$$

The z-transform of the ACF at the output is related to the z-transform of the ACF at input by

$$\begin{aligned} P_{xx}(z) &= P_{vv}(z)H(z)H^*(1/z^*) \\ &= P_{vv}B(z)B^*(1/z^*) \end{aligned} \quad (2.70)$$

The MA PSD estimator does not have sharp peaks therefore it is not considered a high-resolution method, e. g. see figure 2.16. The PSD of a MA process is obtained by substituting $z = \exp(j2\pi f)$ into (2.70) yielding

$$P_{MA}(f) = \sigma_v^2 |B(f)|^2 \quad (2.71)$$

To find the coefficients $b[k]$ requires the solutions of nonlinear equations. This can be seen by multiplying (2.68) by $x^*[n - k]$ and take the expectation:

$$\xi\{x[n]x^*[n - k]\} = \xi\left\{\sum_{l=0}^q b[l]v[n - l]x^*[n - k]\right\} \quad (2.72)$$

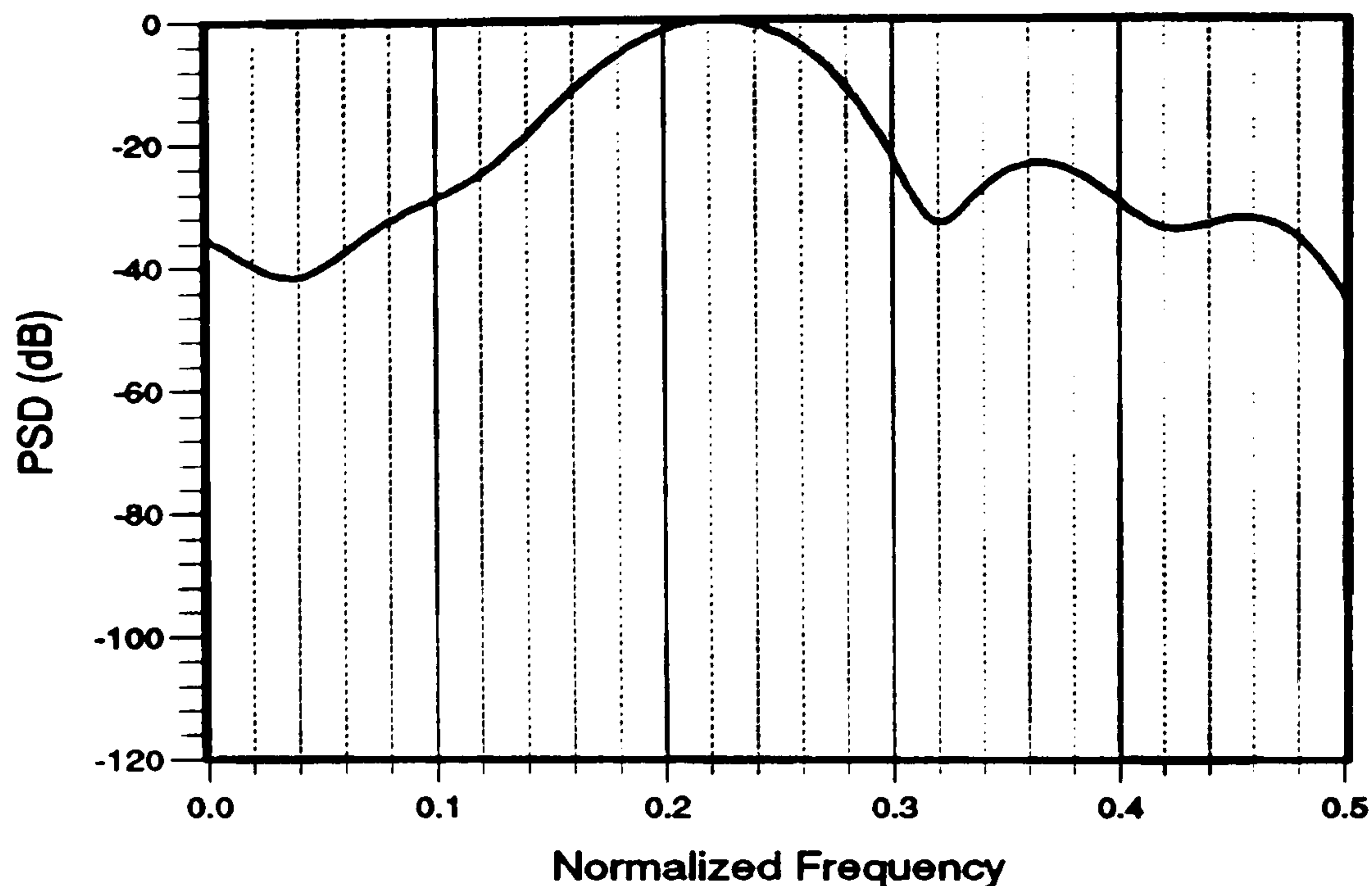


Figure 2.16: The Durbin MA spectrum

or

$$r_{xx}[k] = \sum_{l=0}^q b[l]r_{xv}[k-l] \quad (2.73)$$

Where the unknowns $b[l]$ and $r_{xv}[k-l]$ are coupled together. This is a nonlinear equation even the ACF (i. e. $r_{xx}[k]$) can be calculated from the data. Durbin proposed the following method to avoid this difficulty, let $\frac{1}{A(\infty)}$, where $A(\infty) = 1 + \sum_{k=1}^{\infty} a[k]z^{-k}$, be the system function of an AR process that is equivalent to the MA(q) process. i. e.

$$B(z) = \frac{1}{A(\infty)} \quad (2.74)$$

or

$$B(z)A(\infty) = 1 \quad (2.75)$$

Take inverse z transform of (2.75) and make use of the convolution theorem. We can get the following result

$$a[m] + \sum_{n=1}^q b[n]a[m-n] = \begin{cases} 1 & \text{if } m = 0 \\ 0 & \text{if } m \neq 0 \end{cases} \quad (2.76)$$

In practice a high order AR process is used to approximate the infinite order AR process. Based on the high-order AR parameter estimations $1, a_M[1], \dots, a[M]$. A set of equations of the form

$$e_{MA}[m] = \hat{a}_M[m] + \sum_{n=1}^q b[n]\hat{a}_M[n-m] \quad (2.77)$$

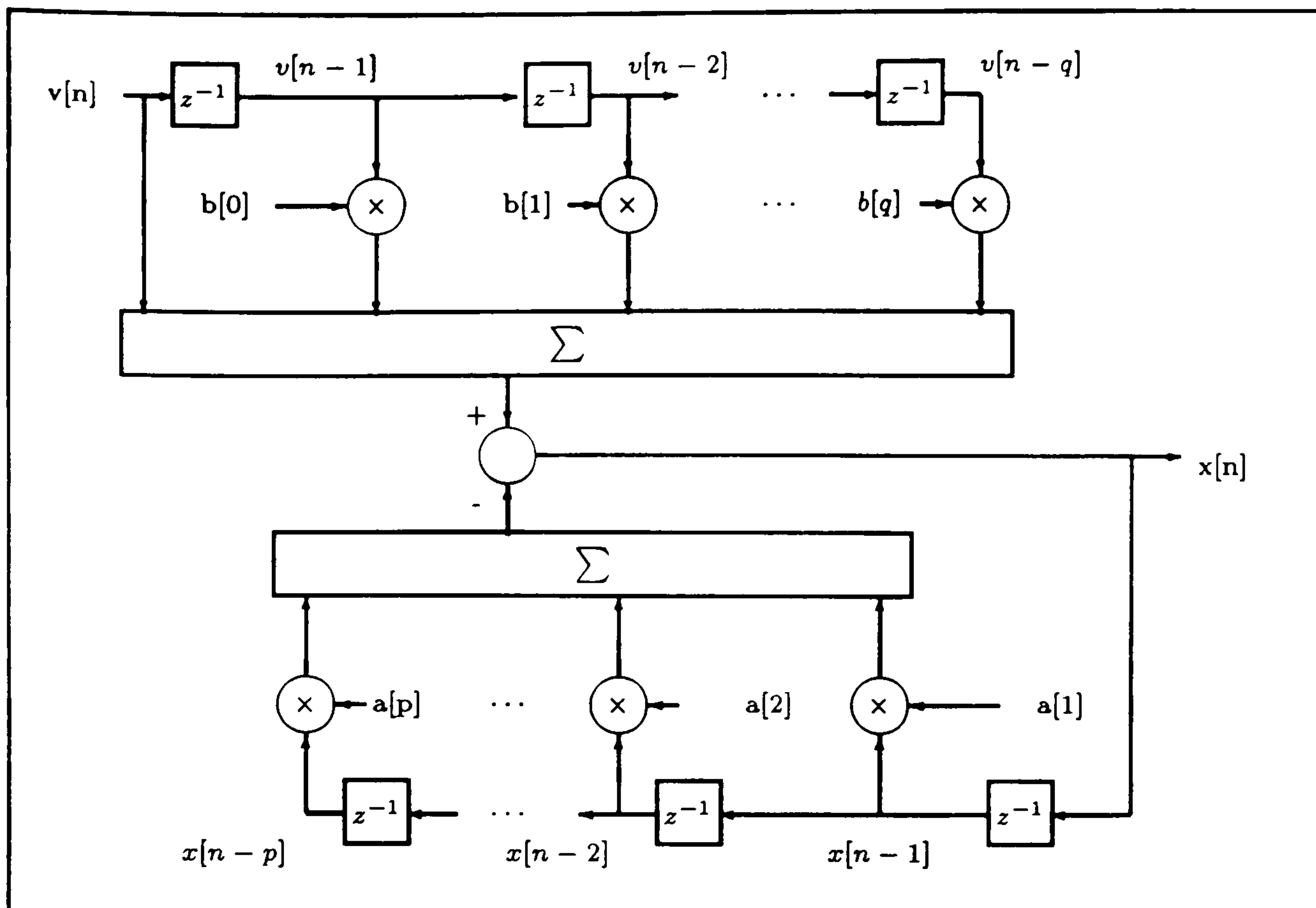


Figure 2.17: The ARMA Process

can be set up. $e_{MA}[m]$ will not be zero due to only using finite data records. Estimates of the MA parameters are obtained by the minimization of the squared error variance

$$\hat{\rho} = \frac{|\sum_m e_{MA}[m]|^2}{M} \quad (2.78)$$

(1.77) is identical to a forward prediction error filter, with the $\hat{a}[k]$ parameter taking the role of data samples. Figure 1.16 is obtained by the Durbin method with the AR order $p = 16$ and the MA order $q = 10$.

2.3.3 Autoregressive Moving Average Method

The ARMA process, see figure 1.17, is defined as

$$x[n] = - \sum_{k=1}^p a[k]x[n-k] + \sum_{k=0}^q b[k]v[n-k] \quad (2.79)$$

The system function $H(z)$ is

$$H(z) = \frac{B(z)}{A(z)} \quad (2.80)$$

Where

$$A(z) = 1 + \sum_{k=1}^p a[k]z^{-k}$$

$$B(z) = 1 + \sum_{k=1}^q b[k]z^{-k}$$

and the z-transform of the ACF at the output is related to the z-transform of ACF at the input by

$$\begin{aligned} P_{xx}(z) &= P_{vv}(z)H(z)H^*(1/z^*) \\ &= P_{vv}(z)\frac{B(z)B^*(1/z^*)}{A(z)A^*(1/z^*)} \end{aligned} \quad (2.81)$$

The PSD of the ARMA is obtained by substituting $z = \exp(j2\pi f)$ into (2.81) yielding

$$P_{ARMA}(f) = \sigma_v^2 \left| \frac{B(f)}{A(f)} \right|^2 \quad (2.82)$$

The relationship between the coefficients and the ACF is nonlinear. This can be seen by multiplying (2.79) by $x^*[n-k]$ and take the expectation:

$$\xi\{x[n]x^*[n-k]\} = \xi\left\{\left(-\sum_{l=1}^p a[l]x[n-l] + \sum_{l=0}^q b[l]v[n-l]\right)x^*[n-k]\right\} \quad (2.83)$$

or

$$r_{xx}[k] = -\sum_{l=1}^p a[l]r_{xx}[k-l] + \sum_{l=0}^q b[l]r_{xv}[k-l] \quad (2.84)$$

The equation is nonlinear due to the two unknowns $b[l]$ and $r_{xv}[k-l]$ being coupled together. Because of the nonlinear nature of the algorithm to solve for the coefficients there have been a few algorithms produced for ARMA PSD estimators. Optimal techniques are computationally intensive. Suboptimal techniques have been developed to reduce the computational complexity. These techniques estimate the AR and MA parameters separately, rather than jointly as required for optimal techniques. Typically, the AR parameters are estimated first, and then used to construct an inverse filter to apply to the original data. The output of this filter is the approximation of the MA process. i. e.

$$\frac{B(z)}{A(z)}\hat{A}(z) \approx B(z) \quad (2.85)$$

The least square modified Yule-Walker technique [29] for ARMA PSD estimation takes this approach. It is a 3 steps procedure

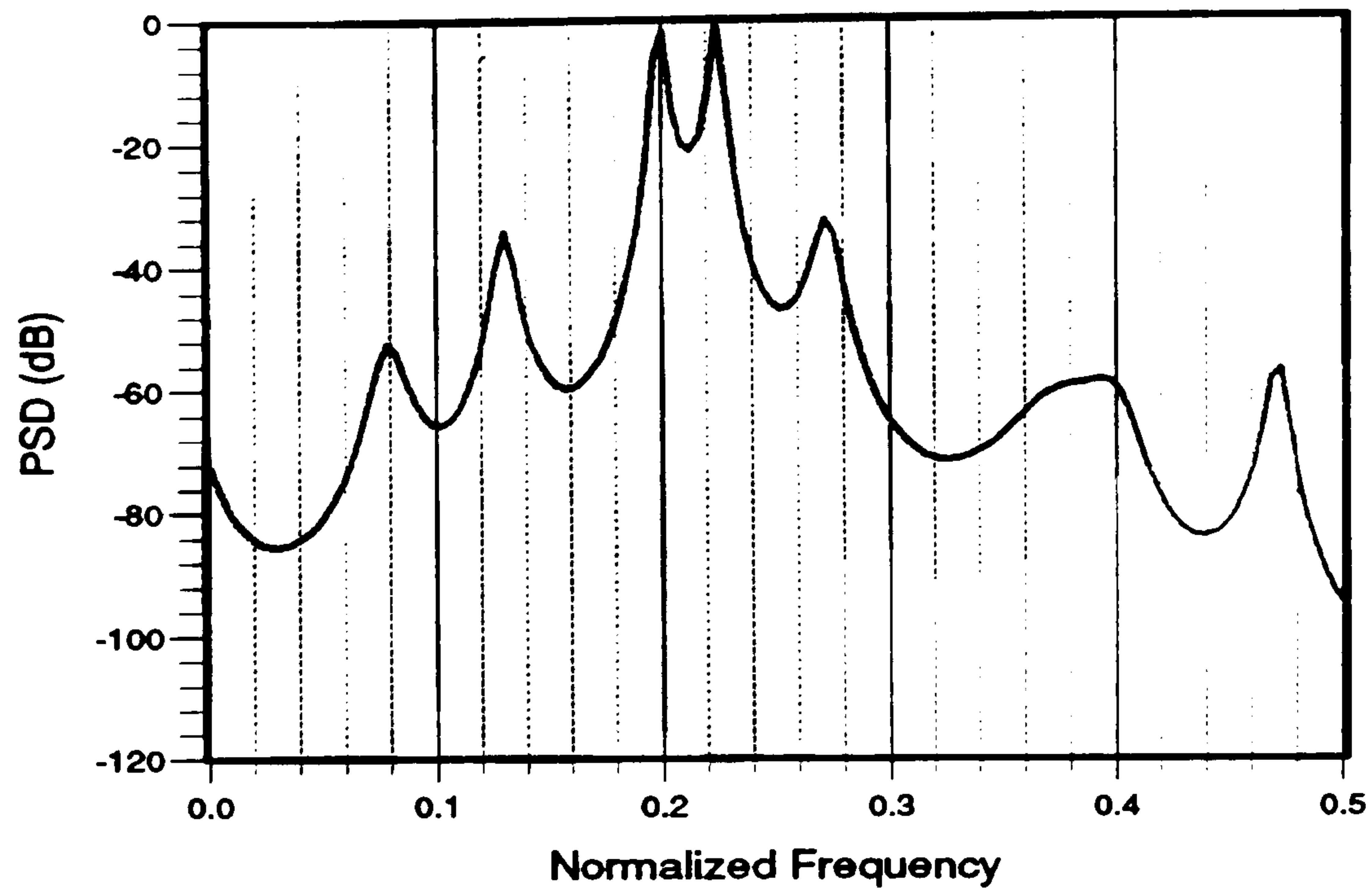


Figure 2.18: The Least Square Modified Yule-Walker ARMA spectrum

1. Estimate the AR parameters by the least square modified Yule-Walker Method.

The normal equations that result are

$$\begin{bmatrix} \mathbf{R}^H \mathbf{R} \end{bmatrix} \begin{bmatrix} 1 \\ a_p[1] \\ \vdots \\ a_p[p] \end{bmatrix} = \begin{bmatrix} \rho \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad (2.86)$$

where $\mathbf{R}$ is a rectangular Toeplitz matrix autocorrelation estimates

$$\mathbf{R} = \begin{bmatrix} \hat{r}_{xx}[q+1] & \hat{r}_{xx}[q] & \cdots & \hat{r}_{xx}[q-p+1] \\ \hat{r}_{xx}[q+2] & \hat{r}_{xx}[q+1] & \cdots & \hat{r}_{xx}[q-p+2] \\ \vdots & \vdots & \vdots & \vdots \\ \hat{r}_{xx}[M] & \hat{r}_{xx}[M-1] & \cdots & \hat{r}_{xx}[M-p] \end{bmatrix} \quad (2.87)$$

and the $\hat{r}_{xx}$ is the unbiased autocorrelation estimation.

2. Filter the original data with the filter which has the system function

$$\hat{A}(z) = 1 + \sum_{k=1}^p \hat{a}[k]z^{-k} \quad (2.88)$$

3. Estimate the MA parameters from the filter residuals. The Durbin algorithm described above for the MA process can be used to estimate the MA parameters $b[k]$.

Figure 2.18 is obtained with the order of the AR part $p = 16$ and the order of the MA part $q = 10$ which are the same as that of figure 2.16. It can be seen that the ARMA model has higher resolution than that of the MA model.

2.3.4 Eigenanalysis-based Methods

2.3.4.1 Introduction of Subspace Method Signal Processing based on the eigenstructure of subspace is the subject of intensive research. The eigendecomposition based algorithms can be grouped into two classes:

signal subspace algorithms : The noise subspace is eliminated to effectively increased the SNR and thereby enhance the performance.

noise subspace algorithms : The orthogonal property between the noise and signal subspace is exploited to extract the signal frequency information.

In recent years, eigenspace decomposition approaches have been applied to array problems, e. g. Direction Of Arrival(DOA) estimation [7] and array beamforming [44]. Eigendecomposition approaches have also been applied to time series spectral estimation [6]. It has been shown that it has better resolution than rational transfer model parametric techniques especially at lower SNR [29]. Where these latter techniques often fail to resolve close sinusoids or other narrowband spectral components. The basic principle is to divide the original large dimension eigenspace into two small orthogonal subspace, i. e. ‘signal’ and ‘noise’ subspace. Due to this partition the resolution is no longer limited by the SNR. Bienvenu [3] claimed that the resolution increases with the observation time up to infinity if latter can be unlimited.(i. e. if the true covariance matrix can be obtained)

2.3.4.2 Orthogonal Property of The Autocorrelation Matrix Implicit in the use of an autocorrelation matrix is the modeling of the L sinusoids as a WSS (Wide Sense Stationary) random process which is only possible if the phases are assumed to be random. If the process consists of L real signals

$$x[n] = \sum_{i=1}^L A_i \sin(2\pi f_i nT + \theta_i) \quad (2.89)$$

and each signal has a phase which is uniformly distributed in the interval $[0, 2\pi]$ independent of the other phases then the mean of the L signals is zero and the ACS is

$$r_{xx}[m] = \sum_{i=1}^L \frac{A_i^2}{2} \cos(2\pi f_i m T) \quad (2.90)$$

If the process consists of L complex sinusoids

$$x[n] = \sum_{i=1}^L A_i \exp[j(2\pi f_i n T + \theta_i)] \quad (2.91)$$

then the ACS will have the form

$$r_{xx}[m] = \sum_{i=1}^L A_i \exp(j2\pi f_i m T) \quad (2.92)$$

When independent white noise $w[n]$, with variance σ^2 is added to the above mentioned complex sinusoids then the combined process $y[n] = x[n] + w[n]$ will have an ACS

$$\begin{aligned} r_{yy}[m] &= r_{xx}[m] + r_{ww}[m] \\ &= \sum_{i=1}^L A_i^2 \exp(j2\pi f_i m T) + \sigma^2 \delta[m] \end{aligned} \quad (2.93)$$

In matrix form it can be expressed as

$$\begin{aligned} \mathbf{R}_{yy} &= \sum_{i=1}^L \mathbf{A}_i^2 \mathbf{e}_M(f_i) \mathbf{e}_M^H(f_i) + \sigma^2 \mathbf{I} \\ &= \mathbf{R}_{ss} + \mathbf{R}_{ww} \end{aligned} \quad (2.94)$$

Where $\mathbf{I}$ is an $(M+1) \times (M+1)$ identity matrix and

$$\mathbf{e}_M(f_i) = \begin{bmatrix} 1 \\ \exp(j2\pi f_i T) \\ \vdots \\ \exp(j2\pi f_i M T) \end{bmatrix} \quad (2.95)$$

Let the eigendecomposition of $\mathbf{R}_{ss}$ be denoted as

$$\mathbf{R}_{ss} = \sum_{j=1}^p \lambda_j \mathbf{v}_j \mathbf{v}_j^H \quad (2.96)$$

in which the eigenvalues have been arranged in decending value

$$\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_{L+1} \quad (2.97)$$

and the eigenvectors (known as principal eigenvectors) are orthonormal. i. e.

$$\mathbf{v}_i^H \mathbf{v}_j = \begin{cases} 0 & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases} \quad (2.98)$$

The principal eigenvectors of the signal autocorrelation matrix span the same space as the signal vectors. i. e. $\text{span}\{\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_L\} = \text{span}\{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_L\}$ [23]. An alternative representation of identity matrix in terms of the orthonormal eigenvectors is

$$\mathbf{I} = \sum_{i=1}^M \mathbf{v}_i \mathbf{v}_i^H. \quad (2.99)$$

and (2.94) can be rewritten as

$$\begin{aligned} \mathbf{R}_{yy} &= \sum_{i=1}^L \lambda_i \mathbf{v}_i \mathbf{v}_i^H + \sigma^2 \mathbf{I} \\ &= \sum_{i=1}^L \lambda_i \mathbf{v}_i \mathbf{v}_i^H + \sum_{i=1}^M \sigma^2 \mathbf{v}_i \mathbf{v}_i^H \\ &= \sum_{i=1}^L (\lambda_i + \sigma^2) \mathbf{v}_i \mathbf{v}_i^H + \sum_{i=L+1}^M \sigma^2 \mathbf{v}_i \mathbf{v}_i^H \end{aligned} \quad (2.100)$$

Where eigenvectors $\{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_L\}$ correspond to the L largest eigenvalues span the signal subspace and the remaining eigenvectors $\{\mathbf{v}_{L+1}, \dots, \mathbf{v}_M\}$ span the noise subspace. Since the eigenvectors of a correlation matrix are orthogonal and the signal vectors span the same space as the principal eigenvectors then the signal vector is orthogonal to the noise eigenvectors including any linear combinations. i. e.

$$\mathbf{s}_i \left(\sum_{k=L+1}^M \alpha_k \mathbf{v}_k = 0 \right) \quad (2.101)$$

This property forms the basis of all the eigen-based frequency estimation algorithms. In practice, however, the true autocorrelation sequence normally is not available. either the estimation of the correlation matrix is used or we can perform the singular value decomposition on the data matrix directly. The singular value is simply the positive square roots of the eigenvalues [17].

2.3.4.3 The Decomposition Approaches Basically there are two approaches to achieve the decomposition.

1. **EigenValue Decomposition (EVD) approach** : The EVD of an estimated data covariance matrix is used to obtain the spectral modes of the data which are divided into signal and noise spaces. Each spectral mode is described by an eigenvalue/eigenvector pair and the power of the mode is equal to its eigenvalue. This is a well known subject in linear algebra and there are many algorithms available for the calculation of the eigenvalues and eigenvectors. The main disadvantage, however, of this approach is when the matrices become singular (e.g. coherent interference or SNR is high in array processing) then the decomposition will become ill-conditioned and fail. Another disadvantage is that it is necessary to form the estimated covariance matrix which adds to the computational costs.
2. **Singular Value Decomposition (SVD) Approach** : The SVD of the data matrix $\mathbf{X}$ instead of the estimated covariance matrix $\mathbf{X}^T \mathbf{X}$ is performed in this approach. The primary advantage of this approach is numerical accuracy. It is now generally acknowledged that the SVD is the most reliable method of determining the rank numerically and hence the most reliable decomposition [25]. But the disadvantage is that it is computationally intensive.

2.3.4.4 Pisarenko Harmonic Decomposition (PHD) In the PHD algorithm the eigenvector corresponding to the minimum eigenvalue of the estimated autocorrelation matrix $\mathbf{R}_{xx}$ is used for frequency estimation. For the special case of $k = M - 1$ in (1.101) the noise subspace will have a single eigenvector $\mathbf{v}_L$. The components of this smallest eigenvector is then used as the coefficients of the polynomial. The roots of the polynomial correspond to the sinusoidal frequencies [35], [44].

2.3.4.5 MUSIC Method The multiple signal classification algorithm [39] is also based on the orthogonality relation of

$$\mathbf{e}_i \sum_{j=L+1}^M \alpha_j \mathbf{v}_j = 0 \quad (2.102)$$

$$i = 1, 2, \dots, L$$

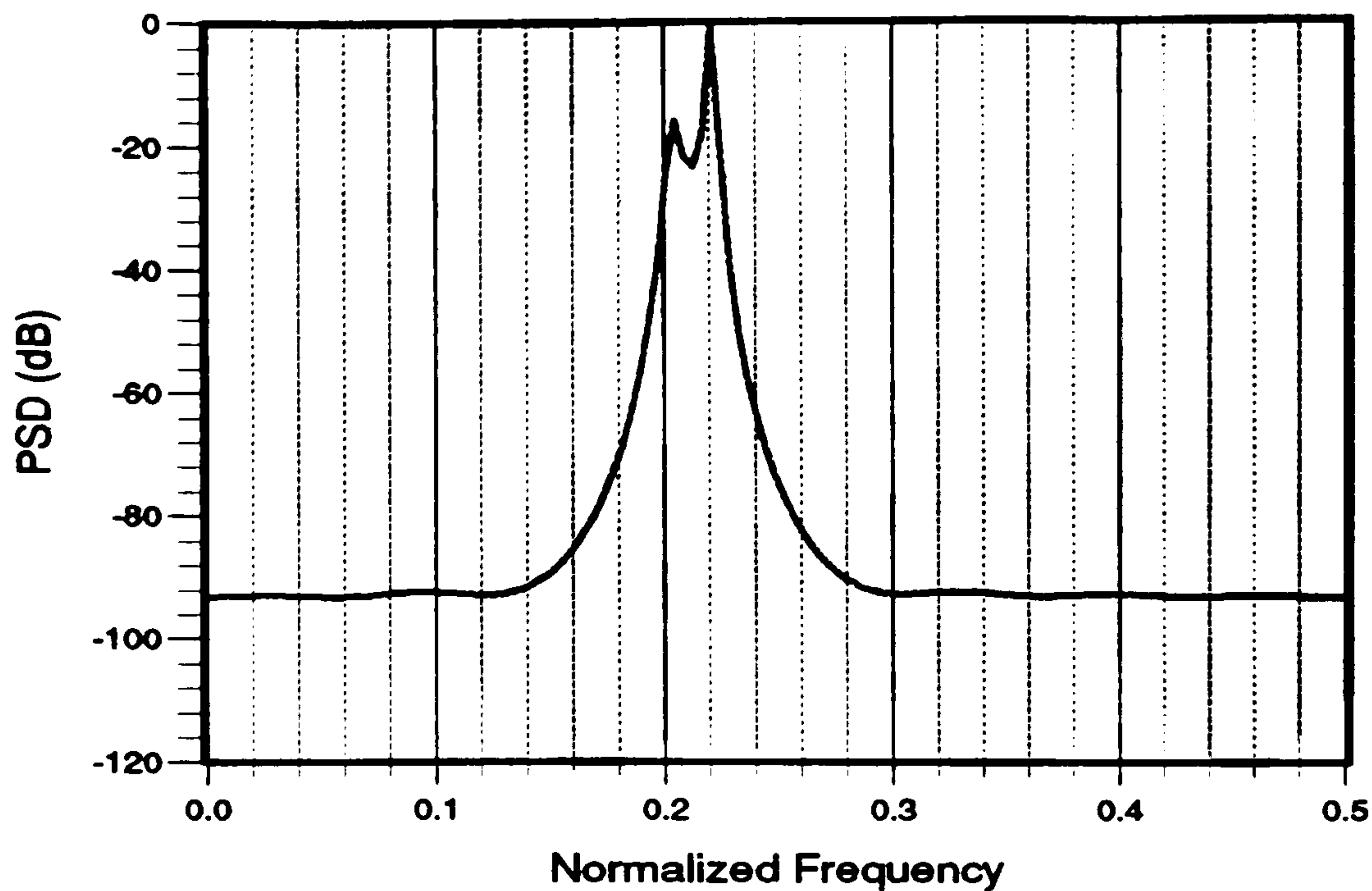


Figure 2.19: The MUSIC spectrum

It estimates the sinusoidal frequencies as the peaks of the spectral estimator

$$\hat{P}_{MUSIC}(f) = \frac{1}{\sum_{i=L+1}^M |\mathbf{e}^H \hat{\mathbf{v}}_i|^2} \quad (2.103)$$

Theoretically when $\mathbf{e} = \mathbf{e}_i$, the i th sinusoidal frequency, $\hat{P}_{MUSIC}(f) \rightarrow \infty$. In practice due to the estimation error a peak will be displayed at or near the sinusoidal frequency.

2.3.4.6 EigenVector Method Johnson and DeGraaf [20] weighted the noise subspace eigenvectors by their corresponding eigenvalues yielding the EigenVector(EV) algorithm frequency estimator

$$\hat{P}_{EV}(f) = \frac{1}{\sum_{i=L+1}^M \frac{1}{\lambda_i} |\mathbf{e}^H \hat{\mathbf{v}}_i|^2} \quad (2.104)$$

The frequency estimation performance of MUSIC and EV are quite similar. Johnson and Degraaf claim that the EV method tends to shape the noise spectrum better than the MUSIC.

2.3.4.7 Minimum Norm Method In MUSIC and EV method the noise subspace is represented by several eigenvectors. Kumaresan and Tufts [26] use only one vector $\mathbf{a}$ to represent the noise subspace. This vector satisfies following three requirements:

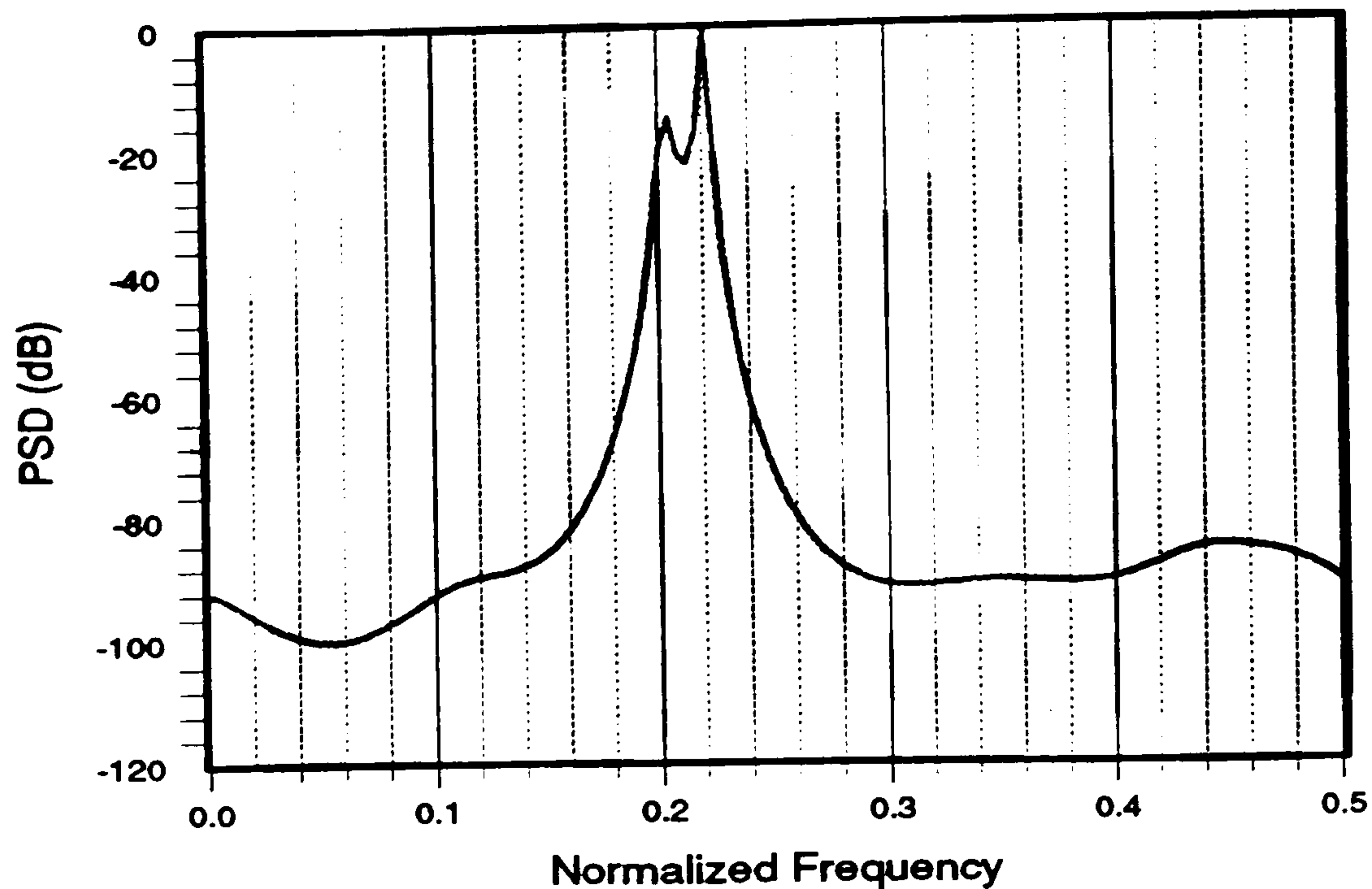


Figure 2.20: The EigenVector method spectrum

1. The vector $\mathbf{a}$ lies in the noise subspace so that it is orthogonal to the signal subspace.
2. The first element of $\mathbf{a}$ is set to 1
3. The Euclidean norm of $\mathbf{a}$ is minimum

The minimum norm spectrum is then calculated as

$$\hat{P}_{MN} = \frac{1}{|\mathbf{e}(f)^H \mathbf{a}|^2} \quad (2.105)$$

2.4 A Proposed Algorithm for PSD Estimation

From the above discussion, it can be seen that the parametric methods have higher resolution but they can produce spurious peaks. The Nonparametric methods, on the other hand, have lower resolution but they are robust. They can indicate the region where signals are located. If the spectrum produced by a nonparametric method, e. g. periodogram, and parametric method, e. g. Burg method, are multiplied together, see figure 1.21, then the true signal spectrum will be more clearly estimated and the spurious peaks will be suppressed. This approach suggests that various PSD algorithms should work in tandem, and make use of the best properties of each individual PSD estimation algorithm. Several PSD estimations using this technique are shown

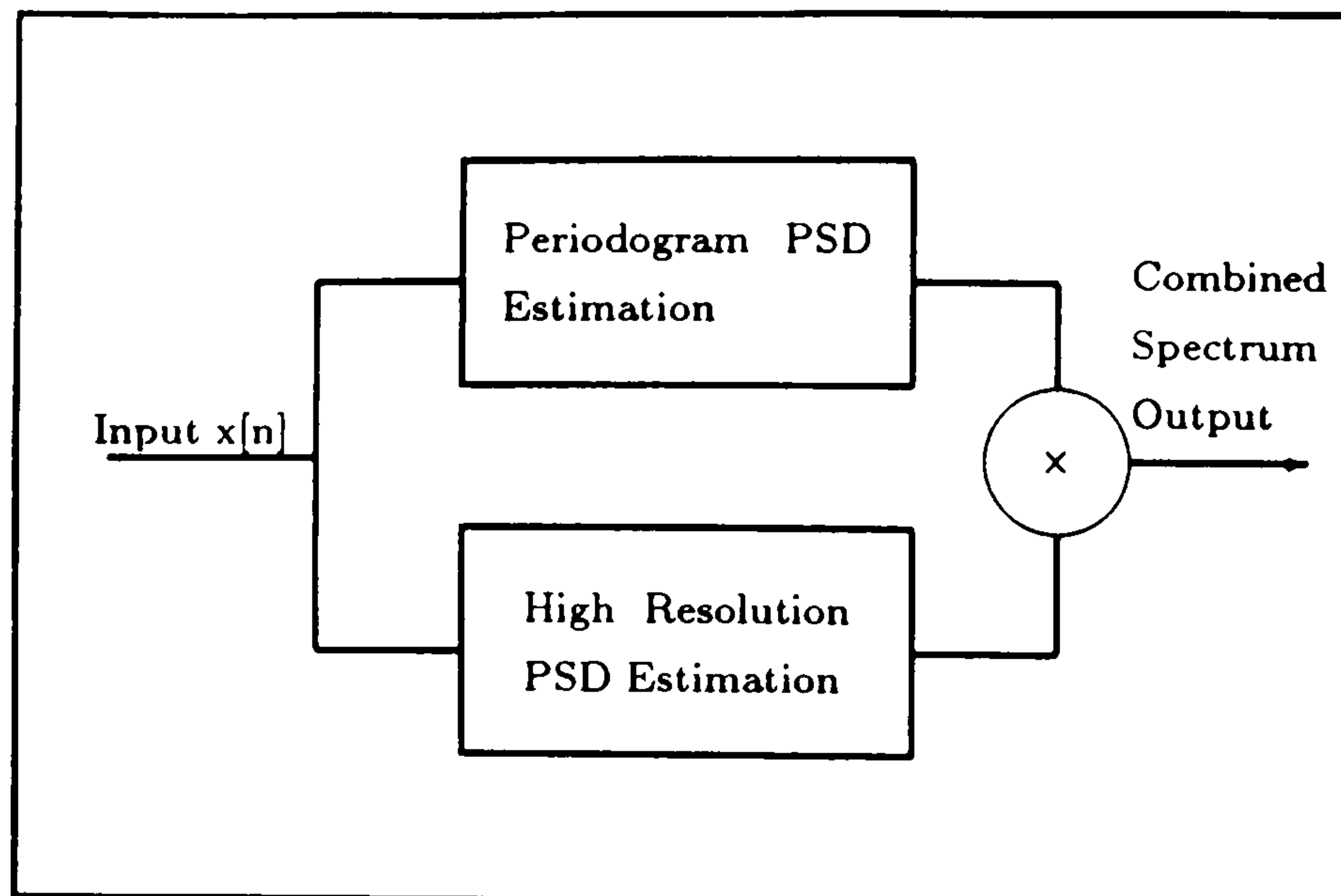
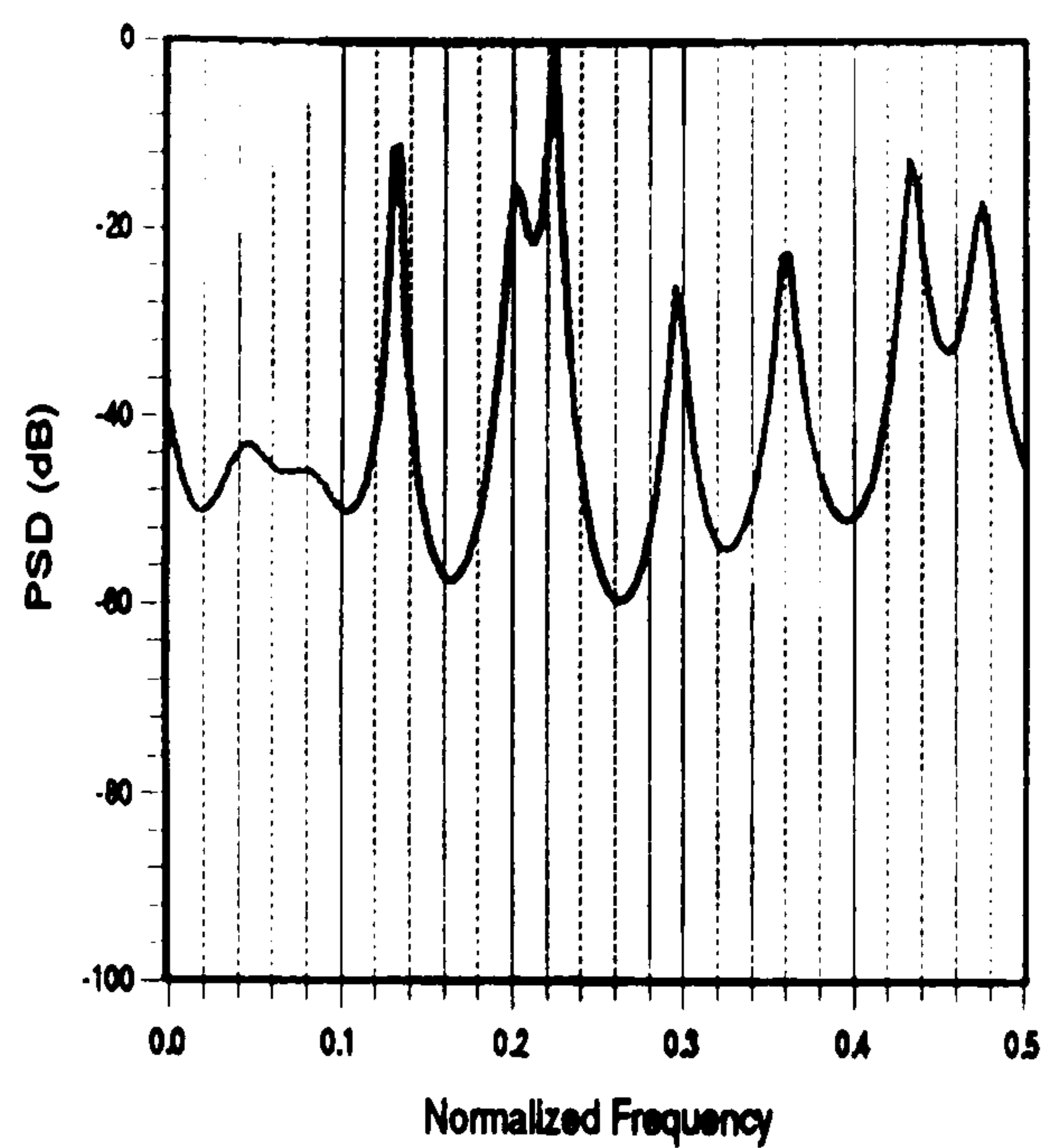
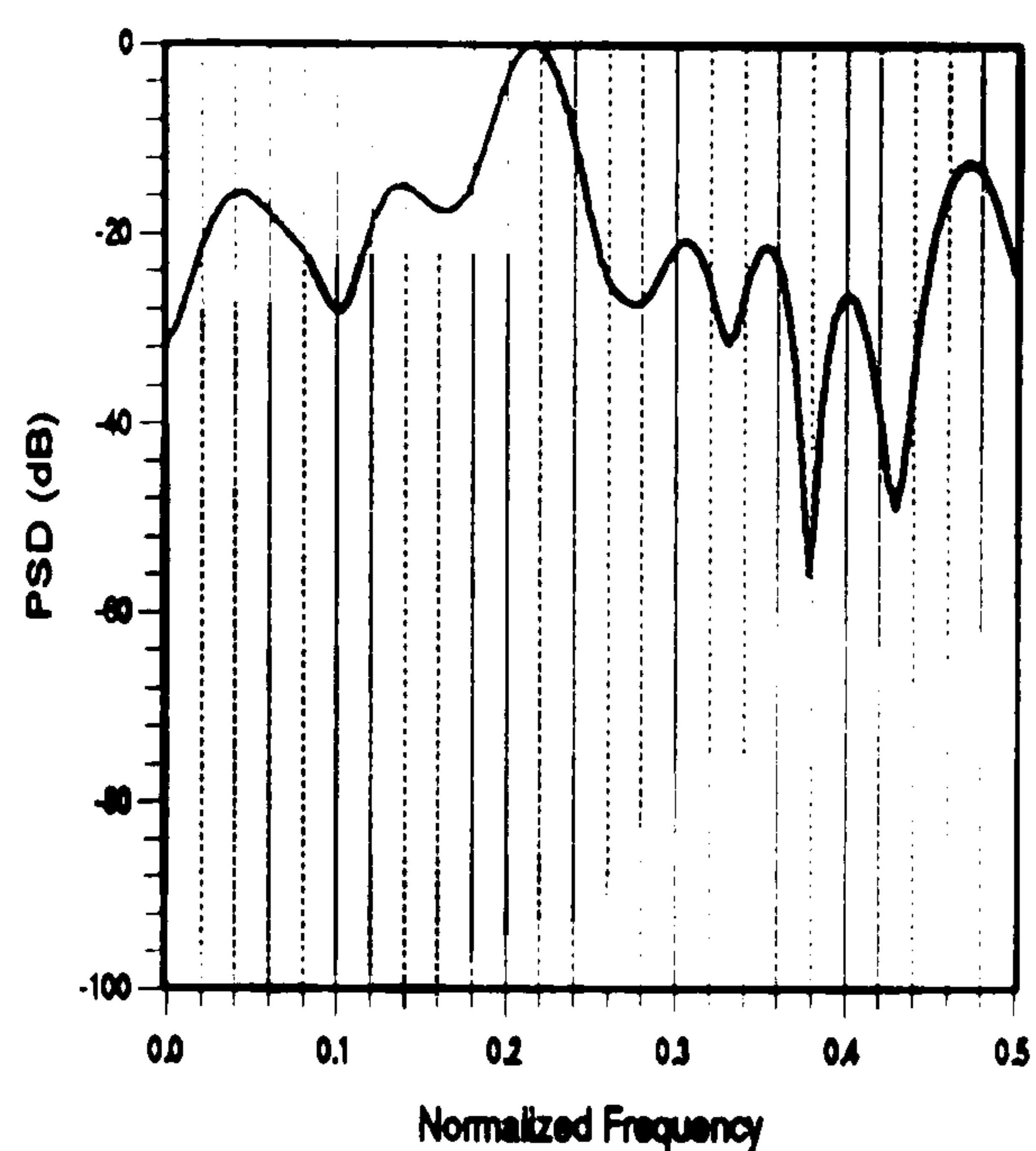


Figure 2.21: The combined PSD estimation method

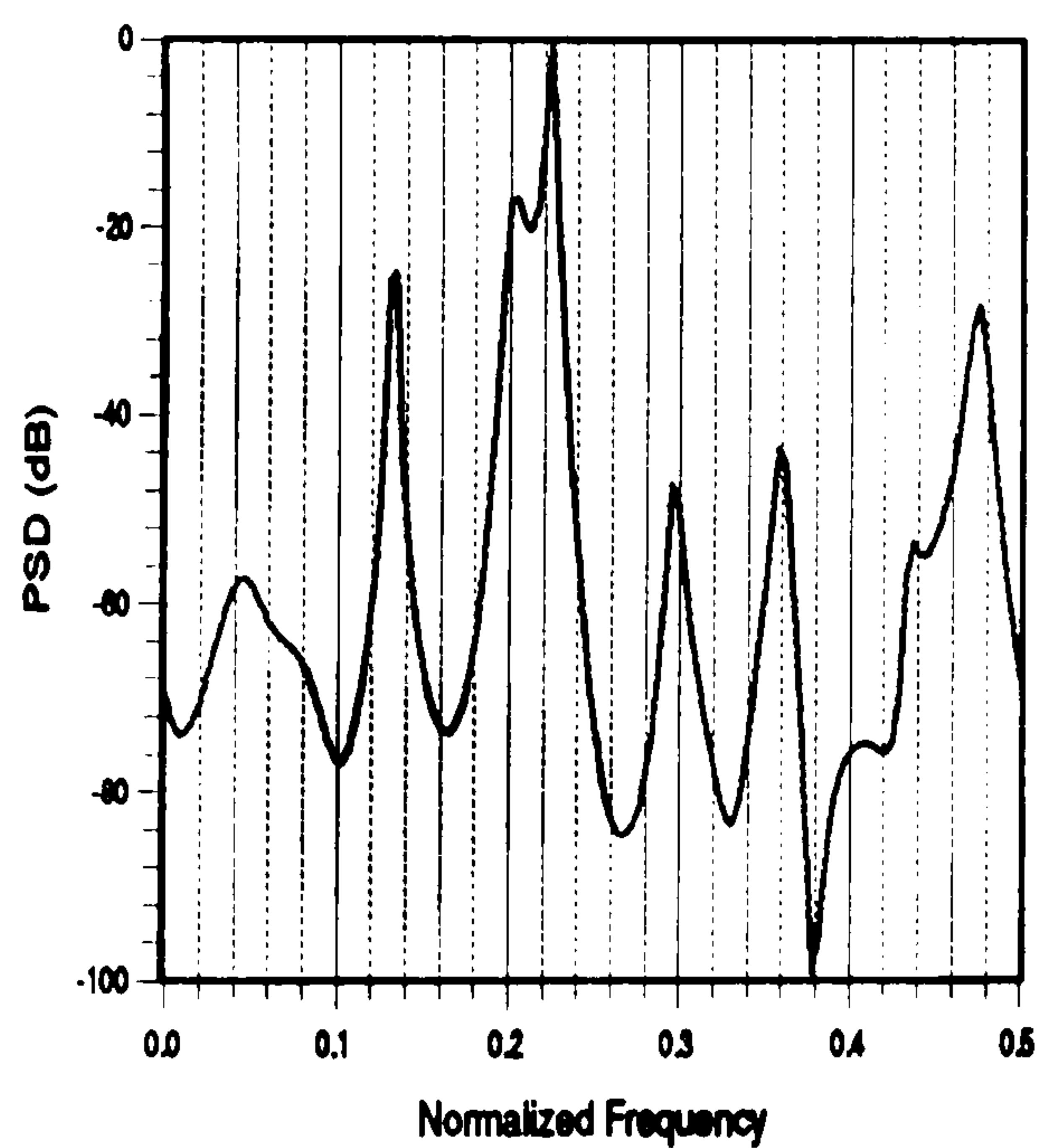
An adaptive algorithm using this technique will be shown in Chapter 3. Using this approach, the order of the rational transfer function model and the number of signal in the eigenanalysis method can be overestimated, the spurious peaks that result will be suppressed by the low value in the corresponding part of the nonparametric spectrum. Some results using this technique are shown in figure 2.22 ~ figure 2.23. In this test case the signal is highly embedded in the noise, the SNR is -10 dB. The model order is 21 which is about $\frac{1}{3}$ of the record length ($N = 64$). In figure 2.22(a) and figure 2.23(a) the spurious peaks cause difficulty in deciding the true number of signals. One of the spurious peaks is even higher than the true signal. In figure 2.22(b) and figure 2.23(b) the periodogram alone is unable to resolve these two signals. However, when the spectra of these two are multiplied together as shown in figure 2.22(c) and figure 2.23(c), the spurious peaks are suppressed. Two nearby signals are now clearly shown.



(a) *Modified Covariance Method*



(b) *Periodogram*



(c) *The Combined Spectrum*

$F1 = 0.2$

$F2 = 0.22$

$SNR = -10\text{ DB}$

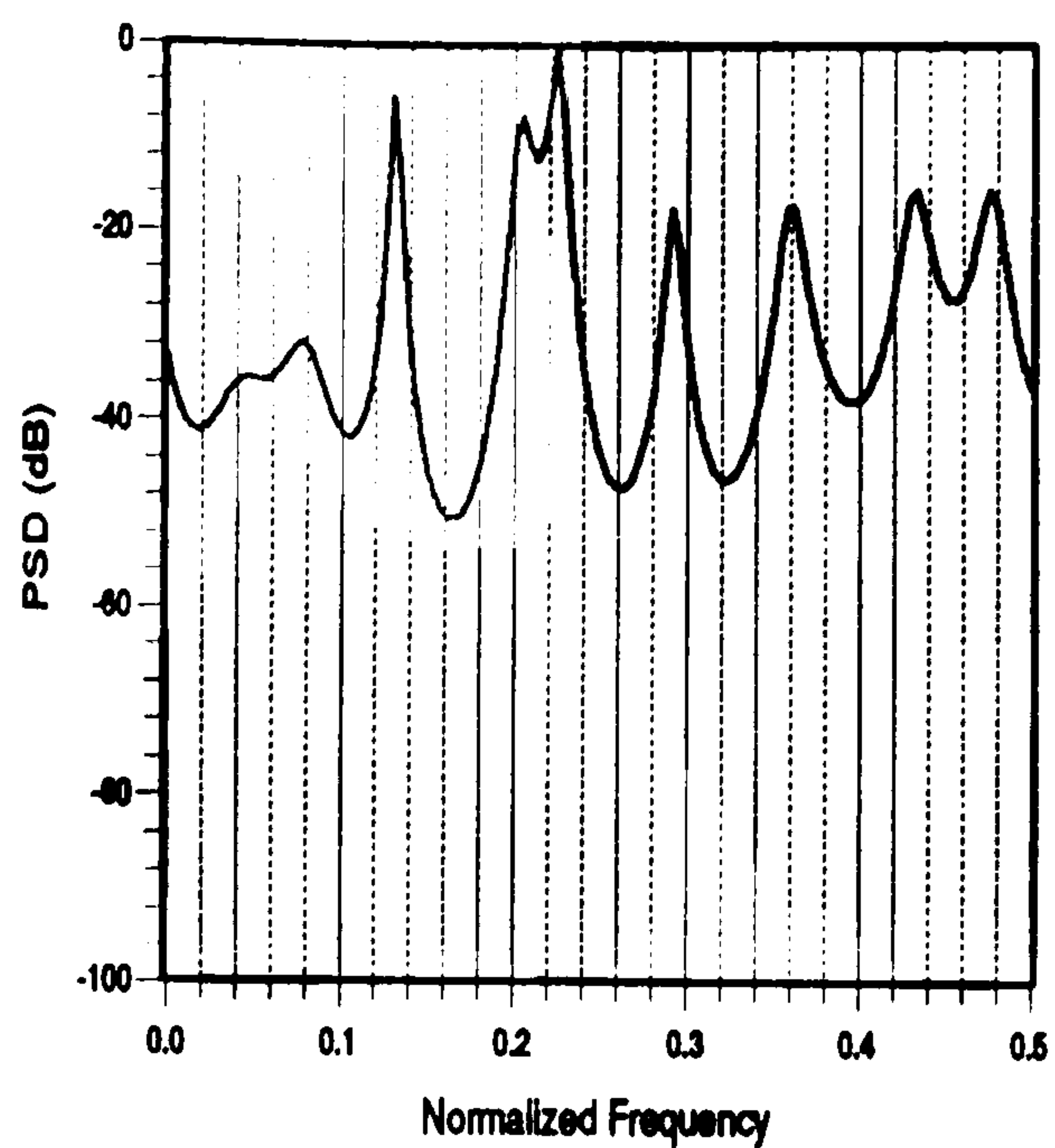
Record Length = 64 Points

Note: The periodogram is smoothed by the welch method with:

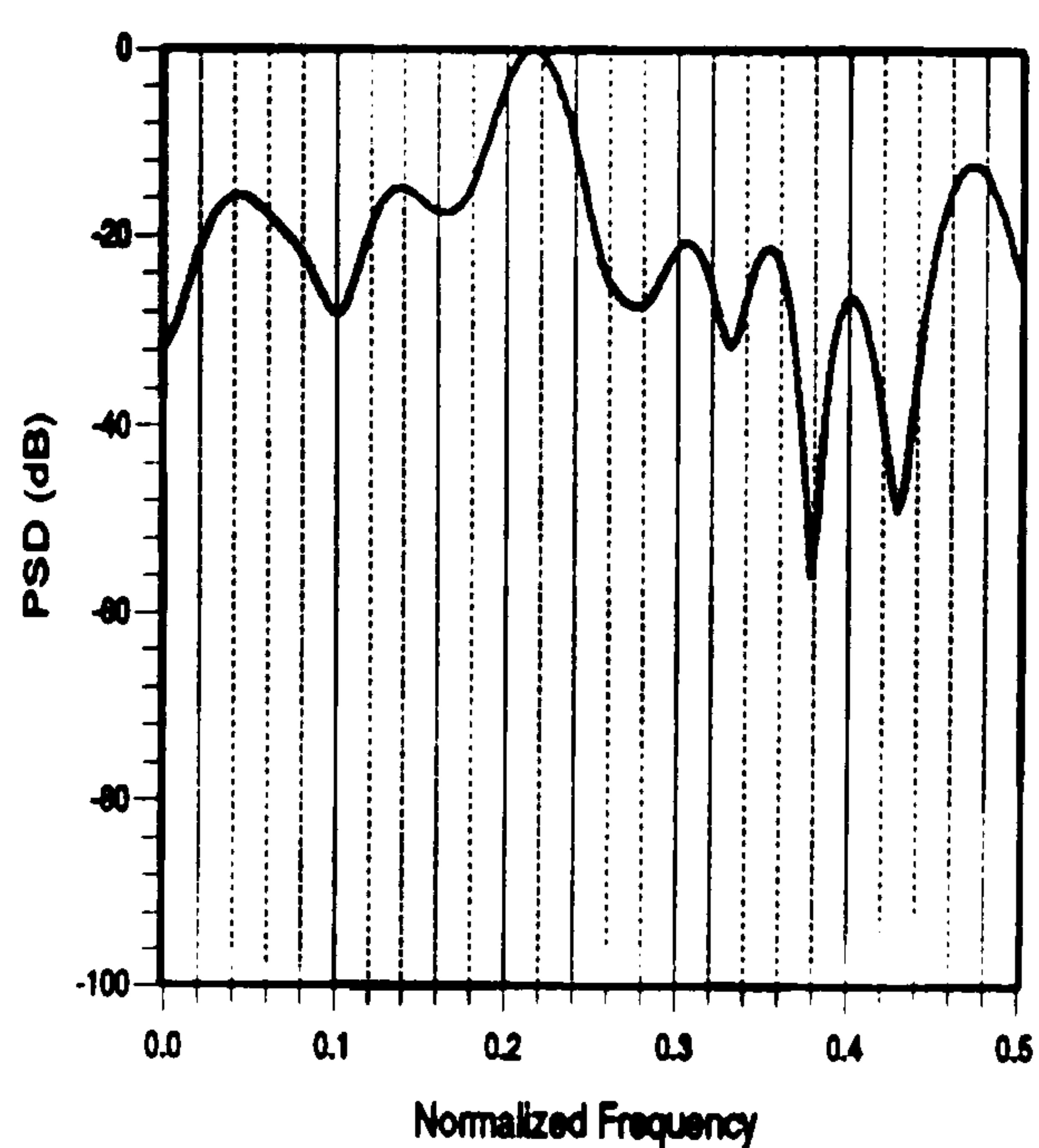
Number per segment = 32

Shift per segment = 16

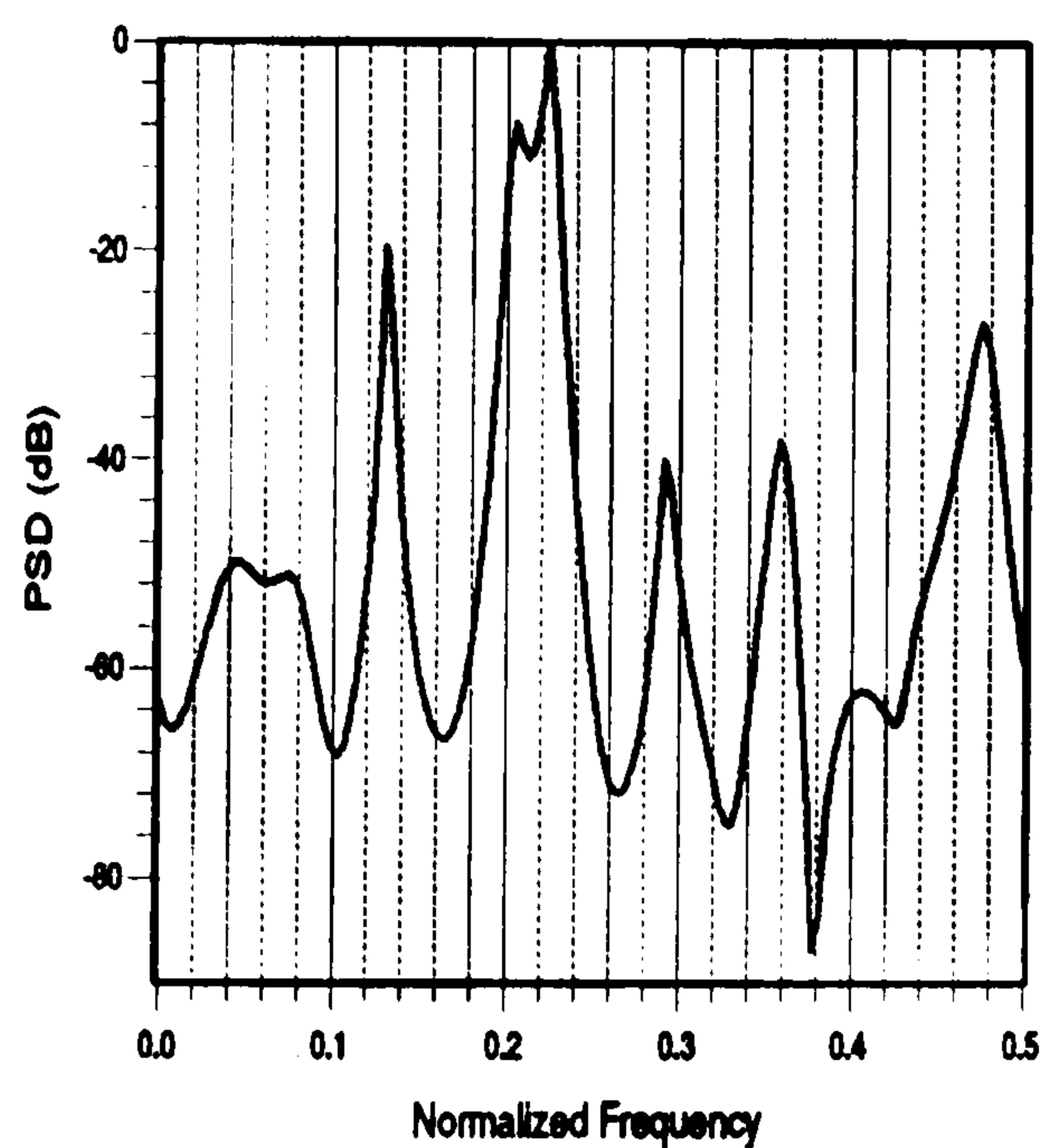
Figure 2.22: The combined periodogram and modified covariance method



(a) Burg Method



(b) Periodogram



(c) The Combined Spectrum

$F1 = 0.2$

$F2 = 0.22$

$SNR = -10 \text{ DB}$

Record Length = 64 Points

Note: The periodogram is smoothed by the welch method with:

Number per segment = 32

Shift per segment = 16

Figure 2.23: The combined periodogram and Burg method

Chapter 3

Sequential PSD Estimation Methods

3.1 Introduction

Block PSD estimation methods described in chapter 2 are suitable for applications where data is limited and the best performance with the available data is desired. In real-time applications longer data are usually available, i. e. data are continuously arriving. The adaptive PSD estimators can track the characteristic of the signal, i. e. they can adapt to the changing environment. The data can be processed sequentially either as each new piece of data is received or as a block of data is available. The choice, in practice, depends on how fast the digital signal processor (or general purpose digital computer) can compute a single spectrum. If the DSP can compute the spectrum very fast then we can process the data as every new piece of data arrives. Otherwise we can process the signal after a segment of data is received. No matter which technique is used, the signal must be partitioned into frames to be sequentially analyzed. The optimum length of each frame depends on the stationarity of the signal. The quasi-stationary situation suggests the use of shorter data segment. On the other hand the use of too short data segments can lead to a biased estimation, because low frequencies do not develop sufficiently. These frames can be sequential, or overlapped, or with a gap between them. Again the choice mainly depends on how fast the processor can compute a single spectrum.

In this chapter sequential implementation techniques for nonparametric and parametric PSD estimators are presented. The same data parameters as chapter 2 are used except that the number of samples is now 256. As mentioned in chapter 2 the AR estimators can achieve high-resolution. But they may produce spurious peaks if

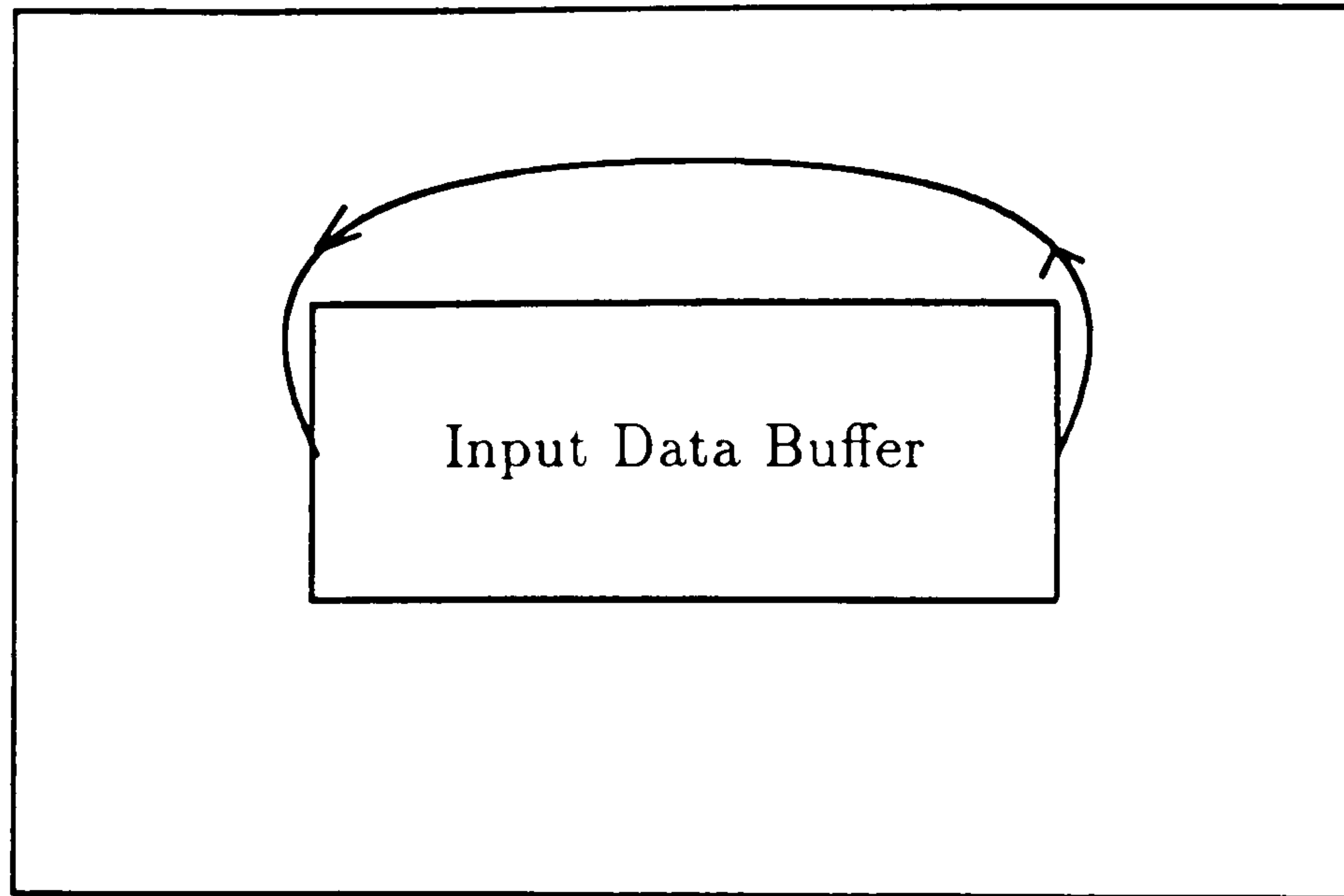


Figure 3.1: The cyclic buffer

the signal is corrupted by noise. A new adaptive method is proposed. This method uses adaptive PHD (APHD) or adaptive line enhancer (ALE) to achieve the high-resolution performance, and uses adaptive maximum eigenvector method to enhance the signal part and suppress the spurious part of the APHD (or ALE) spectrum due to noise.

3.2 Sequential Periodogram Method

To implement the periodogram sequentially we can do the FFT transformation after a certain number of samples are acquired. This is called hopped FFT. These samples are put in a short temporary buffer. When enough samples are accumulated they are then appended to a cyclic buffer as shown in figure 3.1. This cyclic buffer is the first stage of the FFT input data. The number of samples updated each time depends on the application. This technique alleviates the computational burden of the DSP (digital signal processor) and allows the signal to build up gradually in the input buffer. Since the resolution depends on the size of the input buffer, by choosing the size of the input buffer we can get the resolution wanted. A 512 points sequential periodogram using this technique is implemented on the MOTOROLA Application Development System in real-time. The FFT is triggered after eight samples are acquired. The detail is described in chapter 6.

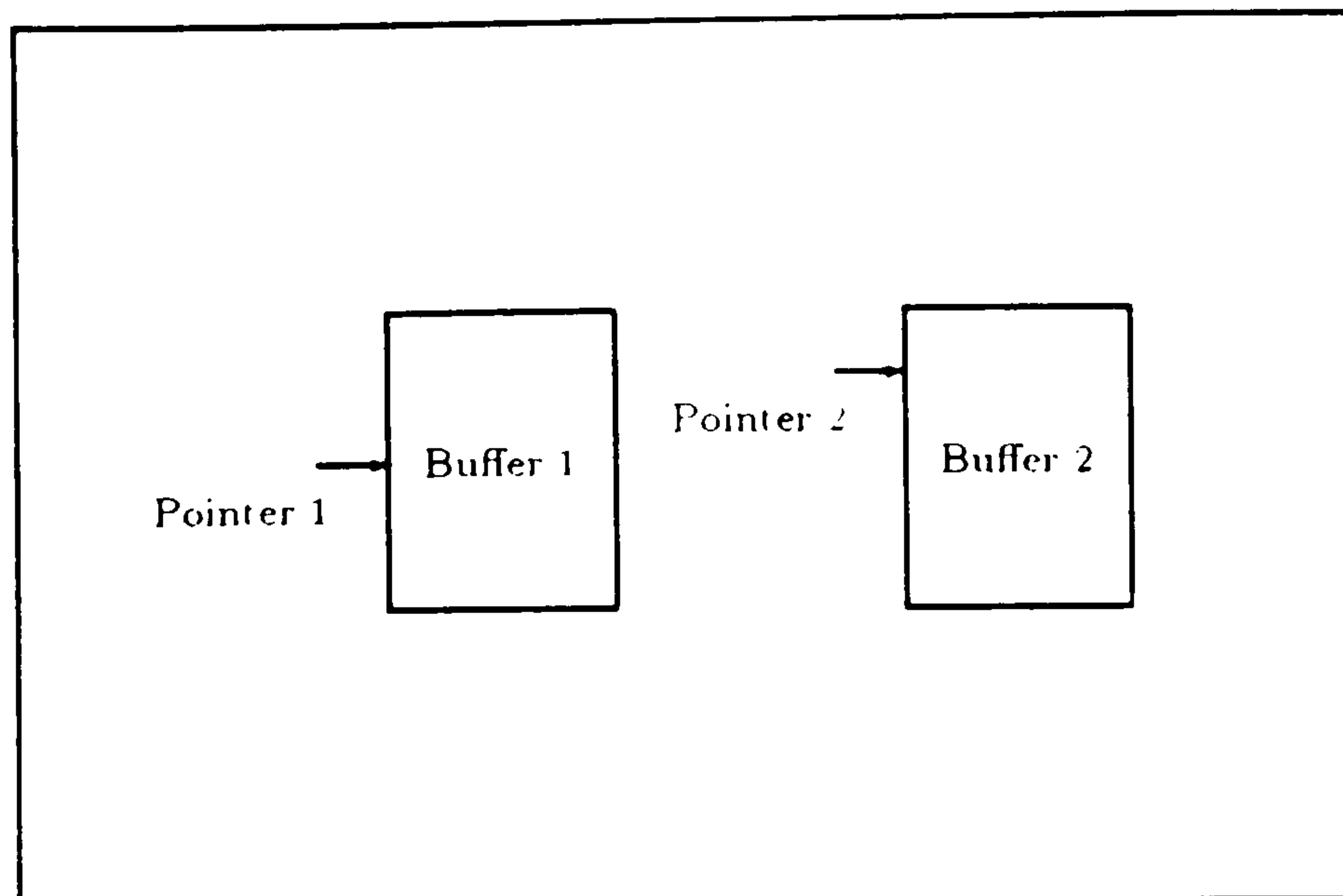


Figure 3.2: The ping-pong memory configuration

The restriction of the above method is that the sampling rate has to be fixed. For those applications where the sampling rate might change, e. g. a pulsed doppler tracking radar might change its PRF (pulse repetition frequency) in different bursts to avoid a clutter area, in this case a ping-pong memory configuration as figure 3.2 can be used. The samples which are acquired by the same sampling rate are processed as a block. At the same time the newly acquired samples which might be sampled using a different rate are stored in another block of memory. When enough samples are acquired then the FFT is triggered to process it. Thus when one block of data is processed the other block is used for storing currently sampled data, and vice versa.

3.3 Adaptive Autoregressive PSD Estimator

The sequential algorithms for adaptive AR parameter estimation fall into two categories :

- Gradient approximation approach
- recursive least square approach

The LMS (Least Mean Square) algorithm is the most commonly used gradient type adaptive algorithm. It is robust and simple therefore it is suitable for real-time ap-

plications. The RLS (Recursive Least Square) type adaptive algorithms have a faster convergent rate at the expense of more computation, however, it is more sensitive to finite-word-length effects than the LMS algorithm. For applications in which a slower convergent rate is tolerable, the LMS method is appropriate. In this project we only consider the LMS adaptive filter for spectral estimation application.

3.3.1 The LMS Adaptive Spectrum Estimator

The LMS adaptive filter is computationally simple and gives good result. There are numerous research papers, e. g. see [54], about the LMS adaptive linear prediction filters since its first introduction by Widrow et. al. in 1975 [47]. It was designated an Adaptive Line Enhancer (ALE) due to the initial application of separating the line spectrum of the sinusoidal signal from broadband noise. Griff [15] applied it to the instantaneous frequency estimation. This is an adaptive AR model of the input signal. Consider the adaptive line enhancer of figure 3.3, where

$$\hat{x}[n] = \sum_{k=1}^P a[k]^* x[n-k] \quad (3.1)$$

is the prediction of x at time n .

$$e[n] = x[n] - \hat{x}[n] \quad (3.2)$$

is the estimation error at time n . The mean square error is

$$J(\mathbf{a}) = \xi\{e[n]e^*[n]\} \quad (3.3)$$

The mean square error is a real and positive scalar. When $J(\mathbf{a})$ is minimum, the filter is said to be operating in the optimum condition in the minimum mean-square error sense. Let $\mathbf{a}^T = (a[1], a[2], \dots, a[p])$ be the weighting vector and $\mathbf{x}^T = (x[n-1], x[n-2], \dots, x[n-p])$ be the input vector. $x[n]$ serves as the desired value at time n . The estimation error can be written as

$$e[n] = x[n] - \mathbf{a}^H \mathbf{x} \quad (3.4)$$

taking hermitian transposition on both side and recognizing $\mathbf{a}$ is constant during the filtering period.

$$e^*[n] = x^* - \mathbf{x}^H \mathbf{a} \quad (3.5)$$

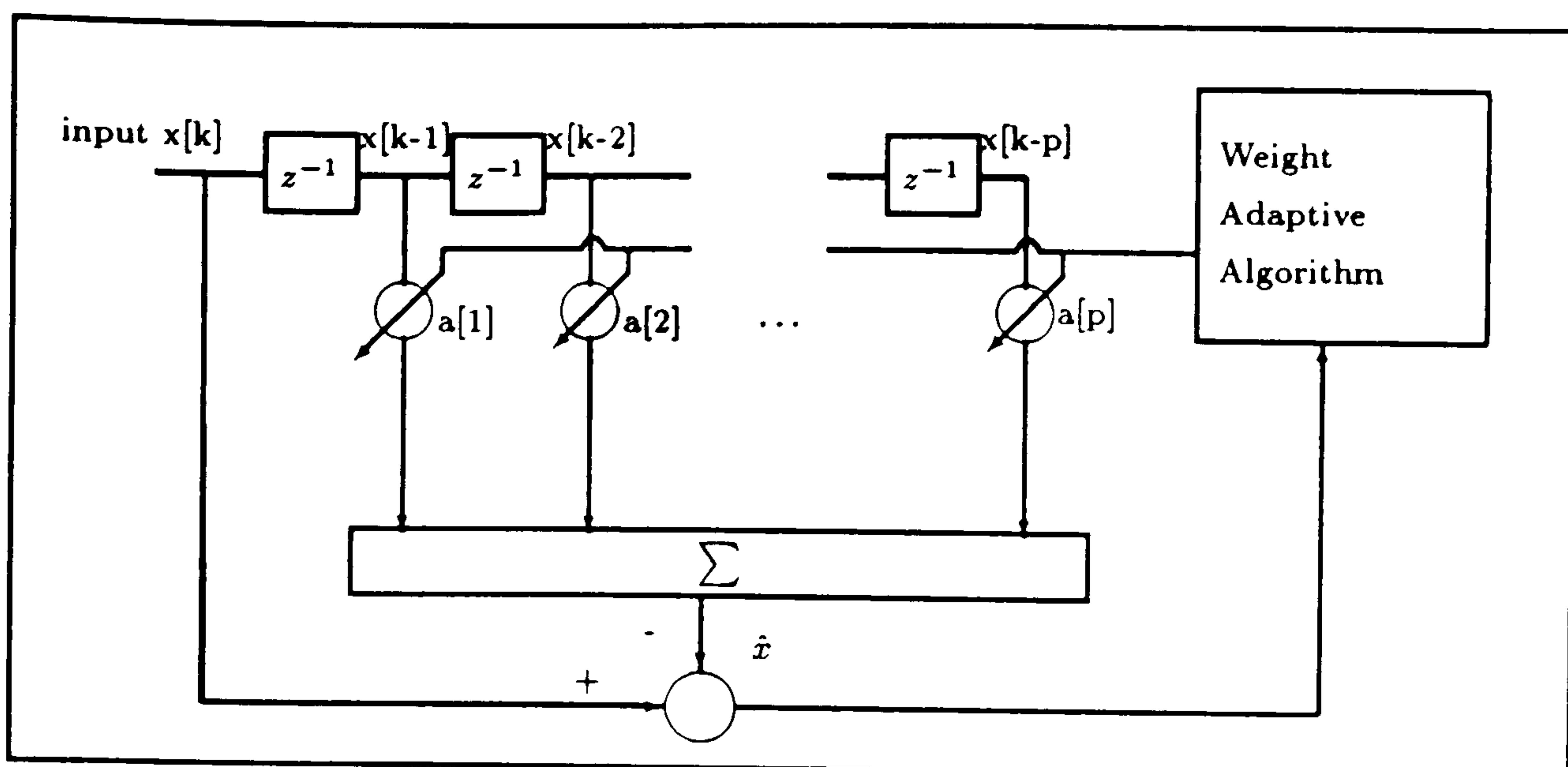


Figure 3.3: The Adaptive Line Enhancer

Substitute (3.4) and (3.5) into (3.3)

$$\begin{aligned}
 J(\mathbf{a}) &= \xi\{e[n]e^*[n]\} \\
 &= \xi\{(x[n] - \mathbf{a}^H \mathbf{x})(x^*[n] - \mathbf{x}^H \mathbf{a})\} \\
 &= \xi\{x[n]x^*[n]\} - \mathbf{a}^H \xi\{\mathbf{x}x^*[n]\} - \xi\{x[n]\mathbf{x}^H\}\mathbf{a} + \mathbf{a}^H \xi\{\mathbf{x}\mathbf{x}^H\}\mathbf{a} \\
 &= \sigma_d^2 - \mathbf{a}^H \mathbf{p} - \mathbf{p}^H \mathbf{a} + \mathbf{a}^H \mathbf{R} \mathbf{a}
 \end{aligned} \tag{3.6}$$

Where $\mathbf{p} = \xi\{\mathbf{x}x^*[n]\}$ and $\mathbf{R} = \xi\{\mathbf{x}\mathbf{x}^H\}$. Therefore $J(\mathbf{a})$ is a second-order function of the tap weight $\mathbf{a}$. It has a bowl-shape surface with a unique minimum point. To find the optimum weight vector at this point we differentiate $J(\mathbf{a})$ with respect to $\mathbf{a}$ and set the result to zero.

$$\begin{aligned}
 \frac{\partial J(\mathbf{a})}{\partial \mathbf{a}} &= \frac{\partial}{\partial \mathbf{a}}(\sigma_d^2 - \mathbf{a}^H \mathbf{p} - \mathbf{p}^H \mathbf{a} + \mathbf{a}^H \mathbf{R} \mathbf{a}) \\
 &= -2\mathbf{p} + 2\mathbf{R}\mathbf{a} \\
 &= 0
 \end{aligned} \tag{3.7}$$

i. e.

$$\mathbf{R}\mathbf{a}_{opt} = \mathbf{P} \tag{3.8}$$

$$\mathbf{a}_{opt} = \mathbf{R}^{-1}\mathbf{P} \tag{3.9}$$

The gradient type adaptive algorithms use the formula:

$$\begin{aligned}
 \hat{\mathbf{a}}[n+1] &= \hat{\mathbf{a}}[n] - \mu \nabla J(\mathbf{a}) \\
 &= \hat{\mathbf{a}} + 2\mu(\mathbf{p} - \mathbf{R}\mathbf{a})
 \end{aligned} \tag{3.10}$$

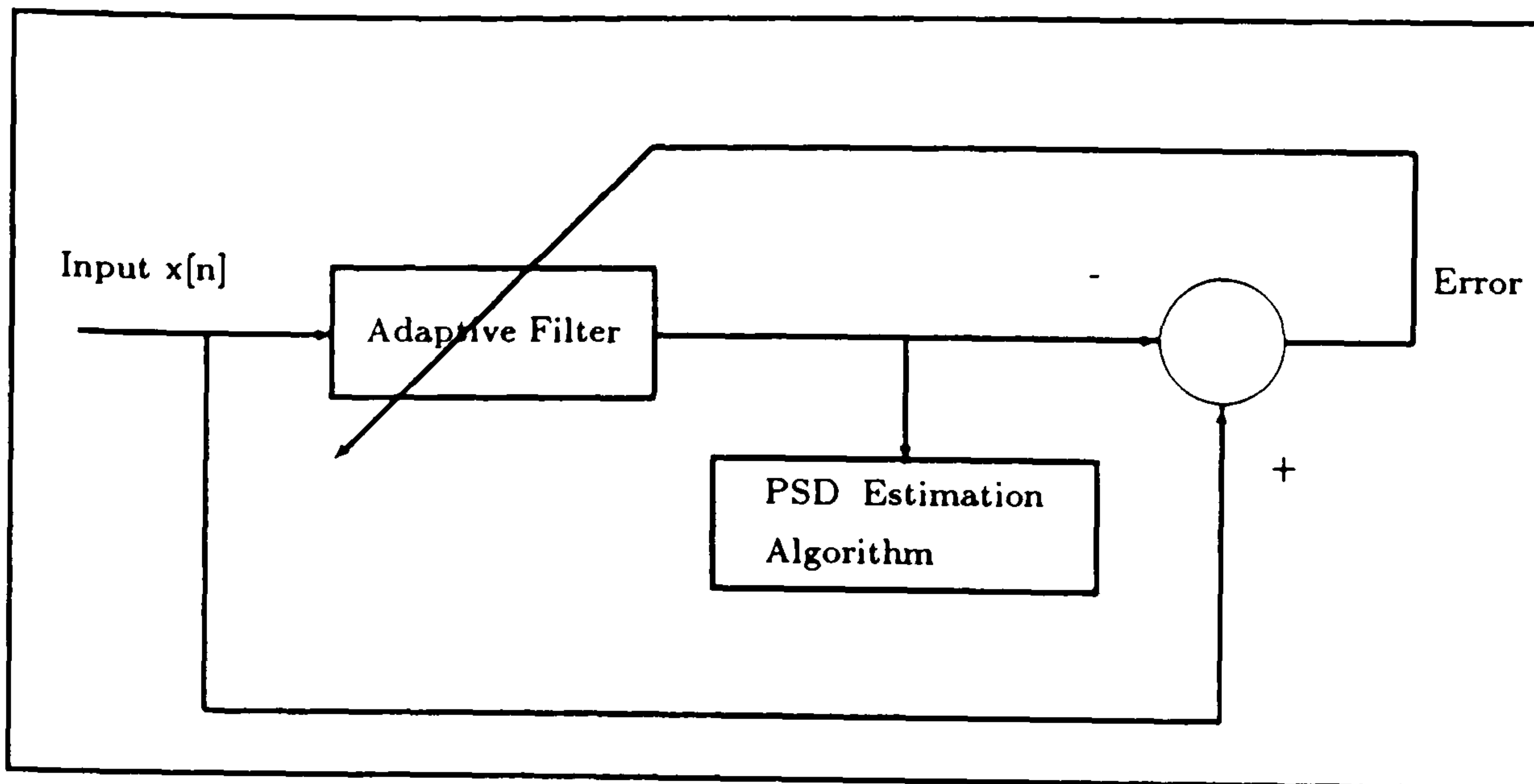


Figure 3.4: The spectral analysis based on the output of ALE

to estimate $\mathbf{a}$ at time index $n+1$ from the previous estimate at time index n . (3.8) is the Wiener-Hopf equation for the optimum linear prediction filter. To find $\mathbf{a}_{opt}$, $\mathbf{R}$ and $\mathbf{P}$ have to be known exactly. In practice, however, they are unknown. The LMS algorithm uses the following approximate for $\mathbf{p}$ and $\mathbf{R}$

$$\mathbf{p} = \mathbf{x}[n]\mathbf{x}^*[n]$$

$$\mathbf{R} = \mathbf{x}\mathbf{x}^H$$

This yields

$$\begin{aligned} \hat{\mathbf{a}}[n+1] &= \hat{\mathbf{a}}[n] + 2\mu(\mathbf{x}\mathbf{x}^*[n] - \mathbf{x}\mathbf{x}^H\mathbf{a}) \\ &= \hat{\mathbf{a}}[n] + 2\mu\mathbf{x}(\mathbf{x}^*[n] - \mathbf{x}^H\mathbf{a}) \\ &= \hat{\mathbf{a}}[n] + 2\mu e^*[n]\mathbf{x} \end{aligned} \quad (3.11)$$

There are two ways of getting the PSD from the ALE. One is from the output of the ALE (ALEO) as shown in figure 3.4. The other is from the weights of the ALE (ALEW) as shown in figure 3.5.

3.3.2 The Effect of The Filter Length

The choice of the prediction filter length is perhaps the most difficult because it depends on the structure of the input states [15]. Zeidler [53] finds a optimum prediction

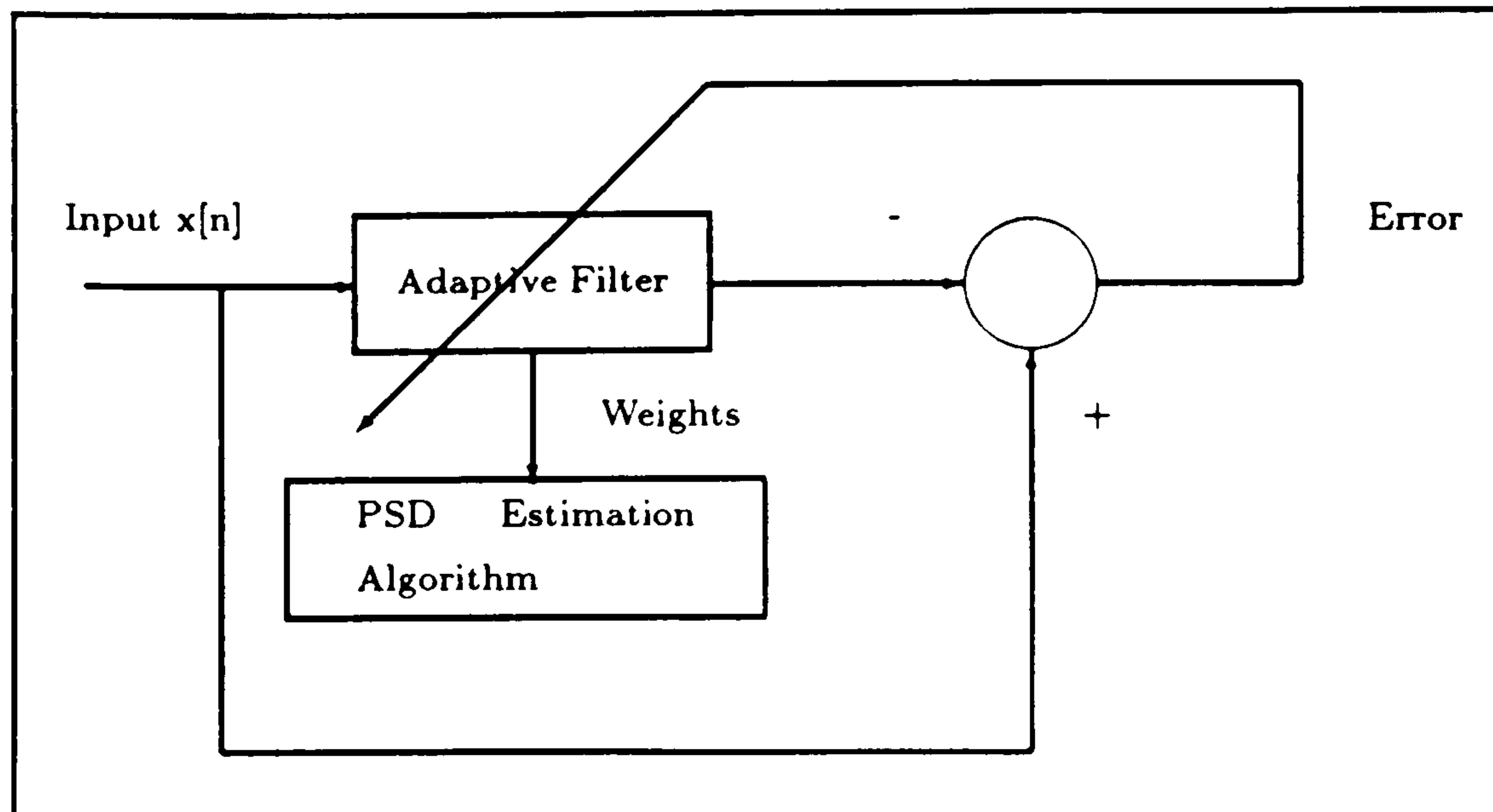


Figure 3.5: The spectral analysis based on the weight of the ALE

filter length for a narrow band signal by extensive computer simulation to be

$$L_{opt} = \frac{f_s}{\alpha} \quad (3.12)$$

Where f_s is the sampling frequency and α is the bandwidth of the input signal. When the SNR of the data is high then a shorter filter length can be used. When the SNR is decreased, it requires a longer filter length to resolve close by spaced sinusoidal signals. Since the signal corrupted by noise is no longer an autoregressive process but an ARMA process, i. e. the signal is generated equivalent to passing a white noise sequence through a linear filter having transfer function $H(\omega)$ and $H(\omega)$ has both poles and zeros, and it needs an infinite order AR process to perfectly match a ARMA process. Therefore the presence of signal zero requires an infinite long prediction filter length for perfect spectral representation.

3.3.3 The Effect of Step Size μ

The choice of step size μ determines both the rate of convergence and the steady state mean-squared error. The convergence is assured if

$$0 < \mu < \frac{2}{Lr_x[0]} \quad (3.13)$$

Where L is the length of the prediction filter and $r_x[0] = \xi x[n]^2$ is the signal power level [15]. Bershad found that the optimum selection of μ is a complicated function

of the filter length, the initial weight setting, the Wiener weights, and the number of learning samples [1]. Widrow [48] found an approximation relationship between the time constant of learning curve, filter length L , and the total power of the input to be

$$\tau \approx \frac{L}{4\mu \text{tr}[\mathbf{R}]} \quad (3.14)$$

Where $\text{tr}[\mathbf{R}]$ is the total power of input, i. e. the quantity $L r_x[0]$ in (3.12), which is generally known. Widrow also found the relationship between misadjustment M , time constant of learning curve, and number of weights to be

$$M \approx \frac{L}{4\tau} \quad (3.15)$$

In general the transient will die out after 4 time constant (i. e. 4τ). If we call the 4τ as the settling time then a rule of thumb is that the misadjustment equals to the number of weights divided by the settling time. Where the misadjustment is defined as the ratio of the excess mean-squared error to the minimum mean-squared error. From the required M we can find τ from (3.15) given L . In turn from (3.14) we can find μ given the total power of input $Lr_x[0]$. This will give a starting point of choosing μ . It needs fine tuning to find a appropriate μ for the application at hand. When the SNR of the input is high, in general increasing μ will increase the resolution capability of the ALE.

3.4 Adaptive Eigenanalysis-based PSD Estimators

3.4.1 The Adaptive Algorithms

To implement eigenanalysis-based algorithms for real-time applications, it is necessary to make them adapt to the changing conditions in the signal or environment. There is intensive research about adaptive algorithms. They can be summarized as follows.

1. Adaptive Eigenvalue Decomposition Approach:

- Time varying estimation of the data correlation matrix :

$$\mathbf{R}_{k+1} = \alpha \mathbf{R}_k + (1 - \alpha) \mathbf{x}_{k+1} \mathbf{x}_{k+1}^t \quad (3.16)$$

- Rank-one EVD updating. Suppose $\mathbf{R} = \mathbf{U}\mathbf{D}\mathbf{U}^T$ then

$$\begin{aligned}\mathbf{R}_{k+1} &= \alpha\mathbf{U}\mathbf{D}\mathbf{U}^T + (1 - \alpha)\mathbf{x}_{k+1}\mathbf{x}_{k+1}^t \\ &= \mathbf{U}(\alpha\mathbf{D} + (1 - \alpha)\beta\beta^t)\mathbf{U}^T\end{aligned}\quad (3.17)$$

where $\beta = \mathbf{U}^T\mathbf{x}_{k+1}$

- solve $\alpha\mathbf{D} + (1 - \alpha)\beta\beta^t$ iteratively. This will avoid the need to do the eigendecomposition from scratch again and will speed up the convergence.

2. Adaptive Singular Value Decomposition Approach:

Instead of working on the estimated correlation matrix $\hat{\mathbf{R}}$, the SVD (Singular Value Decomposition) works directly on the data matrix. If $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ is the SVD of the data matrix, where

$$\begin{aligned}\mathbf{X} &= [\mathbf{x}[1], \mathbf{x}[2], \dots, \mathbf{x}[m]] \\ &= \begin{bmatrix} x[P] & x[P+1] & \dots & x[P+L-1] \\ x[P-1] & x[P] & \dots & x[P+L] \\ \vdots & \vdots & \dots & \vdots \\ x[1] & x[2] & \dots & x[P] \end{bmatrix}\end{aligned}\quad (3.18)$$

is the data matrix, $\mathbf{\Sigma}$ is the diagonal matrix with the singular values on the main diagonal, then the eigenvalue decomposition of $\mathbf{X}\mathbf{X}^T$ is given as

$$\begin{aligned}\mathbf{X}\mathbf{X}^T &= (\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T)(\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T) \\ &= \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T\mathbf{V}\mathbf{\Sigma}\mathbf{U}^T \\ &= \mathbf{U}\mathbf{\Sigma}^2\mathbf{U}^T\end{aligned}\quad (3.19)$$

The singular values of $\mathbf{X}$ are the positive square roots of the eigenvalues of $\mathbf{X}\mathbf{X}^T$ and the left singular vectors, i. e. the column of $\mathbf{U}$, are the eigenvectors of $\mathbf{X}\mathbf{X}^T$. Updating the SVD of $\mathbf{X}$ when a column is appended corresponding to a rank one modification of the symmetric eigenproblem of $\mathbf{X}\mathbf{X}^T$. The recursive SVD procedure is similar to the recursive EVD. i. e. the new SVD is based on the old singular value plus some modification based on the new data. The purpose is to avoid doing the decomposition from the beginning again hence speed up the decomposition process.

3. LMS Approach :

The update algorithm is as follows :

$$\mathbf{w}'(k) = \mathbf{w}(k-1) - \mu \hat{\nabla}(k) \quad (3.20)$$

where $\mathbf{w}(k)$ is the normalization of the transversal filter weight vector $\mathbf{w}'$ and μ is a small positive constant. $\hat{\nabla}(k)$ is an estimator of the gradient of mean square value of the error output respect to $\mathbf{w}$. This adaptive algorithm can be used to track the minimum eigenvector or maximum eigenvector of the correlation matrix. It will be described more fully in the following section.

3.4.2 The Adaptive Pisarenko Harmonic Decomposition Algorithm

Thompson developed an adaptive spectral analysis technique [44] for unbiased frequency estimation in the presence of white noise which is a real-time implementation of Pisarenko Harmonic Decomposition (PHD) method. The main improvement of this method is that it overcomes the frequency bias of the AR spectral estimator when the AR process is observed in the presence of additive noise. Thompson pointed out that the LMS prediction-error filter based on the AR model will give a biased frequency estimation in the presence of noise because the sinusoid-plus-noise input cannot be modeled as an AR process. The original PHD method is an off-line procedure that assumes knowledge of the covariance matrix $\mathbf{R}$. The PHD procedure for matrix $\mathbf{R}$ consists of

1. Finding the minimum eigenvalue λ_L whose multiplicity is denoted by m .
2. Computing an eigenvector $\mathbf{q}_L$ that corresponds to λ_L .
3. Computing the roots $z_i, i = 1, 2, \dots, L$ of the polynomial whose coefficients are the components of $\mathbf{q}_L$

It can be shown [44] that r of the roots, where $r = L - m$, are of unit modulus with the form $z_i = e^{j\theta_i}, i = 1, 2, \dots, r$, where θ_i identify harmonic frequencies contained in $\mathbf{R}$. The PHD technique provides a frequency estimation that is unaffected by white noise

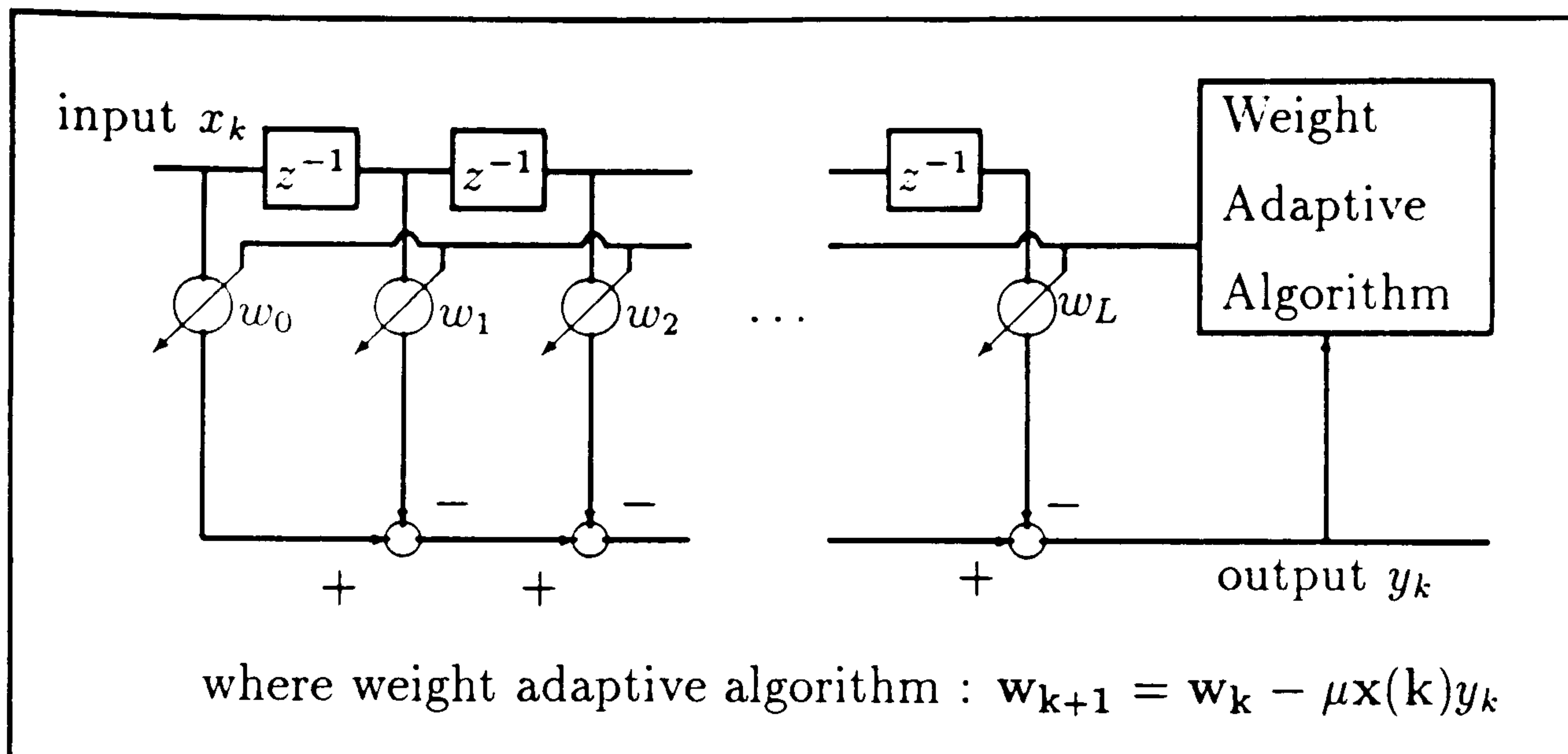


Figure 3.6: The Adaptive Pisarenko Harmonic Decomposition filter

at the expense of considerable computation. The adaptive method of continuously seeking the minimum eigenvector of $\mathbf{R}$ can be obtained by constraining the weight vector to have a unit norm, see figure 3.6 i. e.

$$\mathbf{w}^H \mathbf{w} = \sum_{i=0}^L w_i^2 = 1 \quad (3.21)$$

This is derived as follows: Let the modal matrix of $\mathbf{R}$ be denoted by $\mathbf{Q}$ and be defined as

$$\mathbf{Q} = [\mathbf{q}_0, \mathbf{q}_1, \dots, \mathbf{q}_L] \quad (3.22)$$

Where $\mathbf{q}_L$ is the eigenvector corresponding to the minimum eigenvalue λ_L .

$$\mathbf{Q}\mathbf{Q}^H = \mathbf{Q}^H\mathbf{Q} = \mathbf{I} \quad (3.23)$$

$$\mathbf{Q}^H \mathbf{R} \mathbf{Q} = \mathbf{\Lambda} = \text{diag}[\lambda_0, \lambda_1, \dots, \lambda_L] \quad (3.24)$$

Where $\lambda_0 \geq \lambda_1 \geq \dots \geq \lambda_L$ are the eigenvalues of $\mathbf{R}$. Define

$$\mathbf{v} = \mathbf{Q}^H \mathbf{w} \text{ and } \mathbf{w} = \mathbf{Q} \mathbf{v} \quad (3.25)$$

$\mathbf{v}$ represents the weight vector $\mathbf{w}$ in the modal coordinate system constructed by using the eigenvectors as a basis. The cost function

$$J = \xi\{y_k y_k^H\}$$

$$\begin{aligned}
&= \xi\{\mathbf{w}^H \mathbf{x} \mathbf{x}^H \mathbf{w}\} \\
&= \mathbf{w}^H \mathbf{R} \mathbf{w} \\
&= \mathbf{w}^H \mathbf{Q} \mathbf{\Lambda} \mathbf{Q}^H \mathbf{w} \\
&= \mathbf{v}^H \mathbf{\Lambda} \mathbf{v} \\
&= \sum_{i=0}^L v_i^2 \lambda_i
\end{aligned} \tag{3.26}$$

and the unit norm constraint becomes

$$\begin{aligned}
\mathbf{w}^H \mathbf{w} &= \mathbf{v}^H \mathbf{Q}^H \mathbf{Q} \mathbf{v} \\
&= \mathbf{v}^H \mathbf{v} \\
&= \sum_{i=0}^L v_i^2 = 1
\end{aligned} \tag{3.27}$$

in which v_i denotes the i th component of the vector $\mathbf{v}$. Since $\lambda_0 \geq \lambda_1 \cdots \geq \lambda_L$, J is minimized subject to the constraints (3.27) when $v_L = \pm 1$ and $v_i = 0$ for all $i \neq L$. The corresponding $\mathbf{w}$ obtained from (3.25) is $\mathbf{w} = \pm \mathbf{q}_L$ which is the eigenvector corresponding to the minimum eigenvalue within an arbitrary sign.

Assume that the adaptive filter input consists of white noise with variance σ^2 plus some number of sinusoids. According to the definition of eigenvalue and eigenvector

$$\mathbf{R} \mathbf{q}_L = \lambda_L \mathbf{q}_L = \sigma^2 \mathbf{q}_L \tag{3.28}$$

Take norm on both side

$$\|\mathbf{R} \mathbf{q}_L\| = \|\sigma^2 \mathbf{q}_L\| = \sigma^2 \tag{3.29}$$

The effect of the unit norm constraint in this situation is to fix the noise power at the filter output the same as the input white noise power. Any variation in $\mathbf{w}$ within the constraint varies only the sinusoidal contributions to the output power. These sinusoidal contributions will be made to vanish in order to minimize the output power. Therefore the unit norm constrained filter will seek to place its zeros on the unit circle at the input sinusoid frequencies and will provide unbiased frequency estimates independent of the white noise component.

The spectral estimation procedure for the unit norm constrained adaptive filter can be summarized as follows :

1. Calculate the output

$$y_k = \mathbf{w}^H \mathbf{x}_k$$

2. Calculate the instantaneous gradient estimate

$$\hat{\nabla} = \mathbf{x}_k y_k$$

3. Calculate the weight for next time index

$$\mathbf{w}'_{k+1} = \mathbf{w}_k - \mu \mathbf{x}_k y_k$$

where μ is a small positive scalar constant.

4. Normalization

$$\mathbf{w}_{k+1} = \frac{\mathbf{w}'_{k+1}}{[\mathbf{w}'_{k+1}^H \mathbf{w}'_{k+1}]^{1/2}}$$

3.5 The Proposed Adaptive Frequency Estimator

The adaptive PHD transversal filter continuously tracks the minimum eigenvector of the signal. A variation of it is to track the maximum eigenvector which corresponds to the signal subspace. This is derived as follows.

For stationary signal, the output of the transversal filter is

$$y = \mathbf{w}^H \mathbf{x} \quad (3.30)$$

Update the weight of the transversal filter as

$$\begin{aligned} \mathbf{w}[k+1] &= \mathbf{w}[k] + \mu \hat{\nabla} \\ &= \mathbf{w}[k] + \mu \mathbf{x}[k] y[k] \end{aligned} \quad (3.31)$$

The cost function J is as (3.26). J is maximized subject to the constraints of (3.27) when $v_0 = \pm 1$ and $v_i = 0$ for all $i \neq 0$. The corresponding eigenvector obtained from (3.25) is $\mathbf{w} = \pm \mathbf{q}_0$. Therefore $y[k]$ will converge to λ_{max} , and $\mathbf{w}$ will be the corresponding eigenvector. The effect of unit norm constraint in this case is to place the zeros of the transversal filter corresponding to the signal zeros away from the unit circle at the input sinusoidal frequencies. The eigenvector $\mathbf{q}_0$ will give the spectrum

of the principal components. It has low resolution as expected. However, it faithfully indicates the region where signals are located. If another high-resolution spectrum is multiplied with the principal component spectrum the true signals will be enhanced. Two PSD estimations using this technique are shown in figure 3.7. In figure 3.7(a) ~ (c) the spurious peaks compete with true signals. Without the prior knowledge of the true signals being at the normalized frequency of 0.2 and 0.22, it is not clear which are the true signals. However, when they are multiplied with the spectrum of (d), the true signals are now more clearly estimated as shown in (e) ~ (g) respectively.

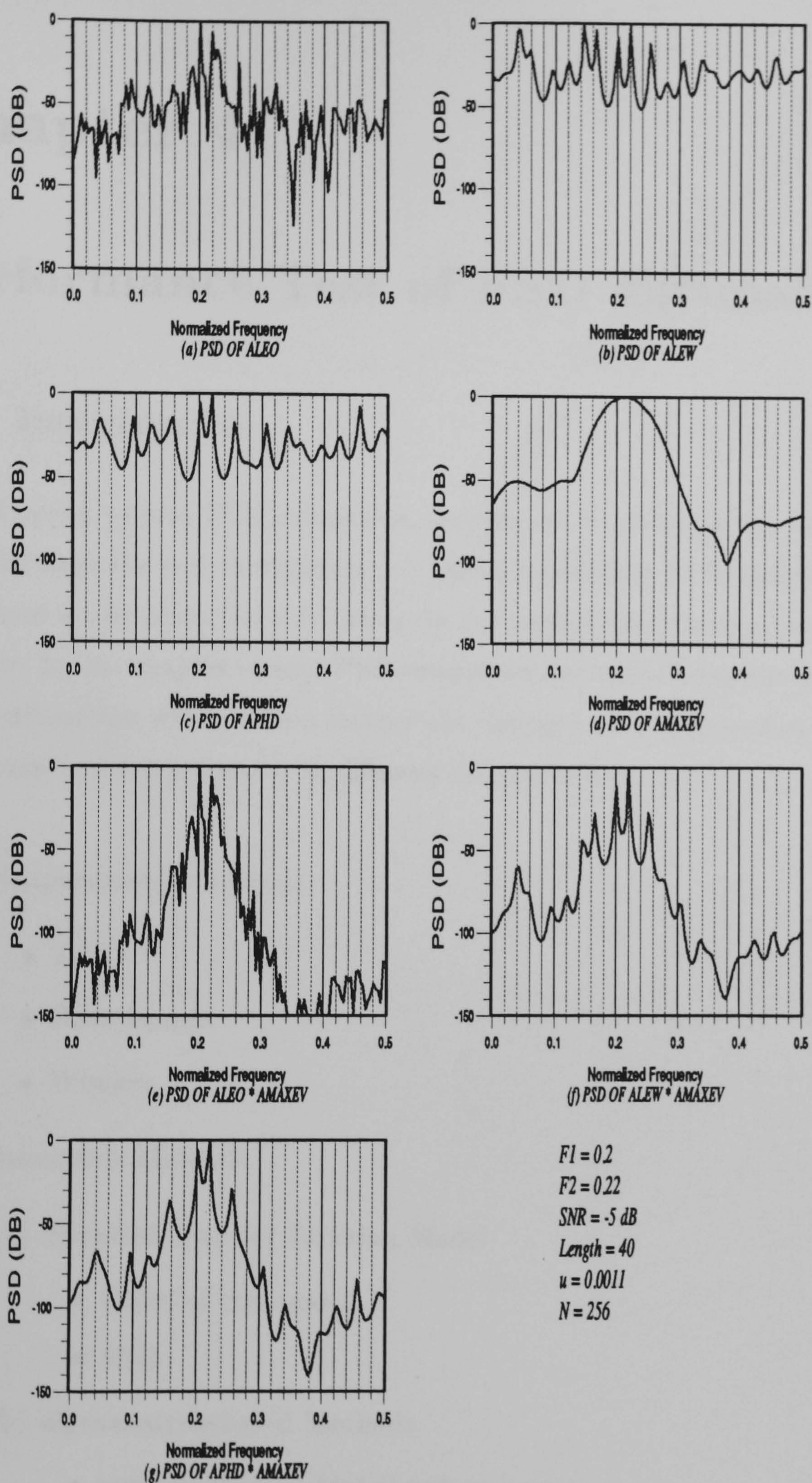


Figure 3.7: Two examples of the adaptive combined PSD estimation

Chapter 4

Performance Test of PSD Estimators

4.1 Introduction

In evaluating various PSD estimators, indexes for a quantitative evaluation are not available since the true spectrum is not normally available as a reference. Thus visual inspection is very often the only means for the analysis and comparison of the spectral features. In this chapter various PSD estimators are tested by investigating the result of the estimation when certain factors are changed. The key factors that affect the performance of the estimator in different categories are

1. Nonparametric Methods

- SNR
- Data Length
- Window

2. Parametric Methods

(a) Rational Transfer Function Model

- Order of the model
- SNR

(b) eigenanalysis-based Methods

- number of sinusoidal signal estimation
- SNR
- order of the correlation matrix

The factor is changed to see its effect on the various performances of the estimator, e. g. frequency resolution capability, frequency bias, spurious peaks etc., This will give some ideas about when the performance of the estimator is degraded and the result is thus not reliable. Knowledge gained from the simulation is also helpful in choosing a suitable estimator when some *a priori* information about the environment is available.

4.2 Performance Test of Block PSD Estimators

4.2.1 The Effect of Record Length

When the frequency separation between two sinusoids is greater than $\frac{1}{NT}$, where N is number of samples and $\frac{1}{T}$ is the sampling rate, then these two signals can be distinguished by all PSD estimators. This is because taking N samples of a sinusoidal signal is equivalent to multiplying an infinitely long data sequence by a rectangular window of width NT . In the frequency domain this is equivalent to convolving the line spectrum of the sinusoidal signal with a sinc function with width of $\frac{1}{NT}$. So two sinusoidal signals closer than $\frac{1}{NT}$ can not be distinguished. This is the resolution limit of the classical nonparametric PSD estimators due to the windowing effect. Parametric methods have higher resolution capability by getting rid of the windowing effect. A test case can show the effect of the record length. The test scenario is

$$f_1 = 0.2$$

$$f_2 = 0.22$$

$$N = 64$$

$$SNR = 10dB$$

Note that T is assumed 1 sec. for simplicity. In this case $\frac{1}{NT} = \frac{1}{64} = 0.016$ so f_1 and f_2 can be resolved. See figure 4.1. when the record is reduced so that Δf is smaller than $\frac{1}{NT}$ than the classical methods are unable to resolve them. These can be seen from figure 4.2. The two equal power test frequencies are

$$f_1 = 0.2$$

$$f_2 = 0.22$$

$$SNR = 10dB$$

$$N = 30$$

The classical methods are unable to resolve these two signals. Since $\frac{1}{30} = 0.33$ is bigger than the frequency separation of the signals. While all the parametric methods are able to resolve these two signals. It is interesting to note that the MVSE method which belongs to the nonparametric method also can resolve these two signals. This justifies the statement made in [23] that MVSE exhibits more resolution than Fourier estimators but less than that of an AR spectral estimator.

4.2.2 The Effect of SNR

In practice, the signal is usually corrupted by noise. Sometimes the strength of the signal can be very small. In this section the resolution capability of PSD estimators is tested against different SNRs. The amplitude of the signals are assumed 1. The noise is assumed to be white gaussian noise with variance σ_n^2 . The SNR is defined as

$$SNR = 10 \log \frac{A^2}{2\sigma_n^2} \quad (4.1)$$

The spectrum of the periodogram can be thought as the output of N equal bandwidth bandpass filters with normalized bandwidth of $\frac{1}{NT}$. The energy of the periodic signal is concentrated into one single filter. While the energy of the white noise is spread into N filters. When the power of the signal and the noise are equal, i. e. $SNR = 0dB$. The signal spectrum will be $10 \log N$ dB above the noise floor. When N is big, the periodogram is a very good technique for detecting a periodic signal buried deeply in noise, e. g. The VEP signal. Also when N is big the resolution limit of the periodogram $\frac{1}{NT}$ is very small, so it also has good resolution capability. The problem comes when the amount of available data are limited. This may be due to genuine lack of data or due to the consideration of stationarity or due to the requirement of fast reaction time. In this situation the parametric high-resolution techniques can be applied. The main concern about these techniques is their performance in the low SNR situation. So it is very important to investigate their performance in different SNR situations. The frequency separation of two test signals is fixed at the critical value of $\frac{1}{NT}$. It is

assumed that T is 1 second. The SNR of these two equal power signals are changed gradually from 30 dB to 0 dB in step of 10 dB. See figure 4.3 ~ figure 4.6.

4.2.3 The Effect of Phase Difference

If the true autocorrelation sequence (i.e. not the estimated one) is used for calculating the spectrum of parameters then the phase difference between signals does not influence the spectrum estimation. This is because signals are assumed uncorrelated and the phase information between them is lost due to the ensemble average. However, in reality we normally use time average to replace the ensemble average and the number of data is usually limited, so it is expected that the phase difference between signals will have some influence on the performance of PSD estimators. e. g. Burg method has been reported to be subject to line splitting and peak positions are strongly dependent on phase [10]. Since the periodogram operates directly on the data, it is most effected by the phase difference between signals. In this test the frequency of two signals are fixed at $f_1 = 0.2$ and $f_2 = 0.22$. The phase difference between them is changed from 0 degree to 180 degree in step of 45 degree. It can be seen that the classical methods are highly affected by the phase difference between signals while the parametric methods are less affected by the phase difference, see figure 4.7 ~ figure 4.11 This is because during the derivation of parametric methods the signals are assumed uncorrelated and the phase is assumed random.

4.2.4 The Effect of Relative Power

In practice the amplitude of different signals are usually different, so it is of interest to see the behaviour of the PSD estimators in a different relative power situation. In applying PSD estimators, the issue of spectral estimation and parameter estimation should not be confused. The spectral amplitude of nonparametric methods and rational transfer function model parametric methods bears a relationship to the power of the input. While the eigenanalysis based parametric methods do not. The latter are actually parameter(i. e. frequency) estimators. The peaks in the spectrum just indicate the frequency of the signals. Therefore it is not surprising to see that the relative power variation has little influence of the appearance of their spectrum.

One particular PSD estimator worth mention is the MVSE. Its peak amplitude is an indication of the power of the sinusoid. Therefore when the relative power between signals is changed it will truly reflect the change. The frequency of two test signals are the same as before and the relative amplitudes between them are 0.5, 0.25 and 0.1, see figure 4.12 ~ figure 4.14. The amplitude of the bigger one being normalized to one.

4.2.5 The Effect of A Nearby Signal

The presence of a third signal doesn't influence the resolution capability of the PSD estimators as long as all components are $\frac{1}{NT}$ apart. When there are multiple signals present the classical methods may have detection problems due to the spectrum leakage, if their amplitudes have big differences. In this test 3 equal power signals are present. Two signals located at $f_1 = 0.2$, $f_2 = 0.22$ and the third signal is moving toward them from 0.4, 0.3 and 0.25, see figure 4.15 ~ figure 4.17.

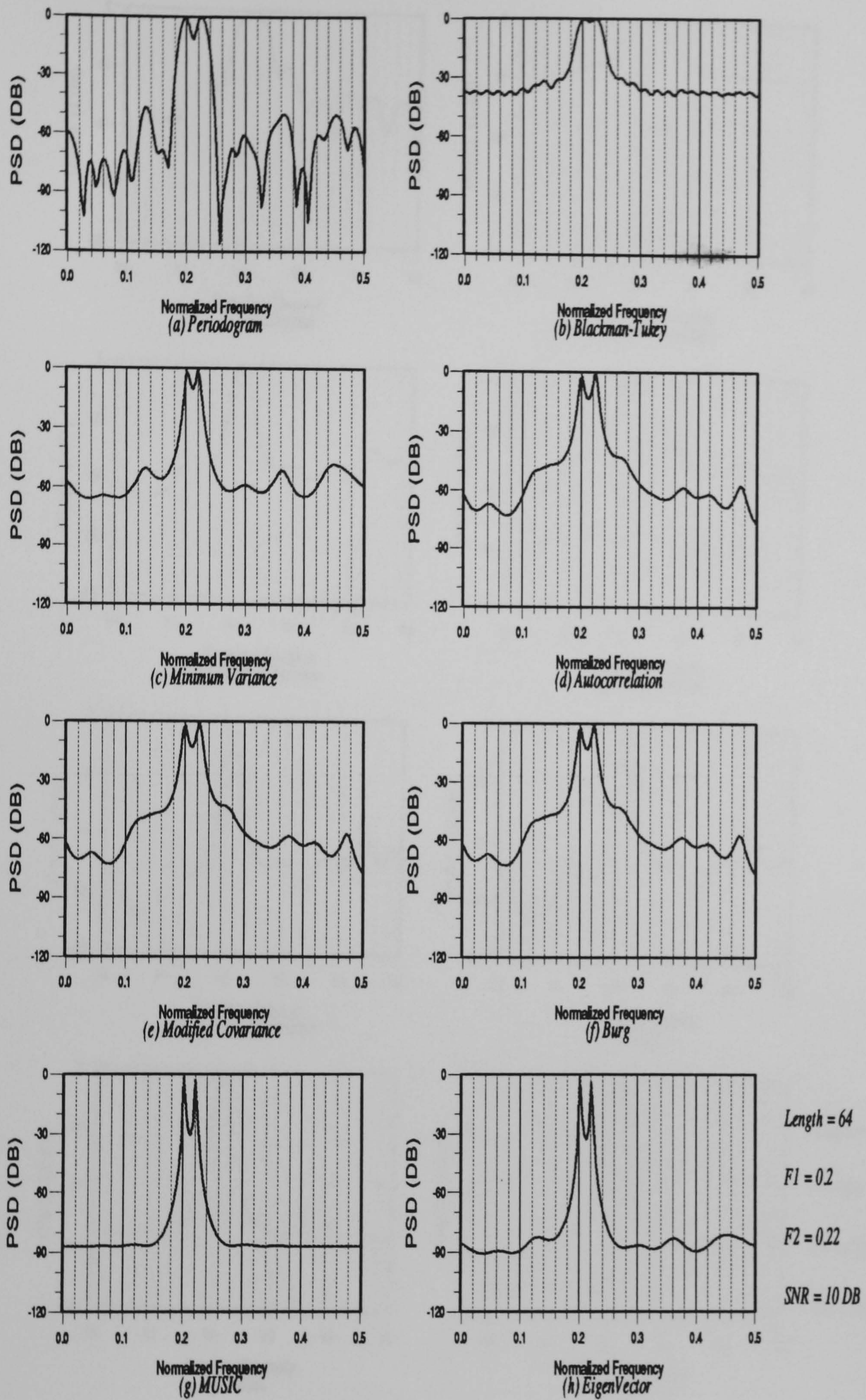


Figure 4.1: Record length of 64

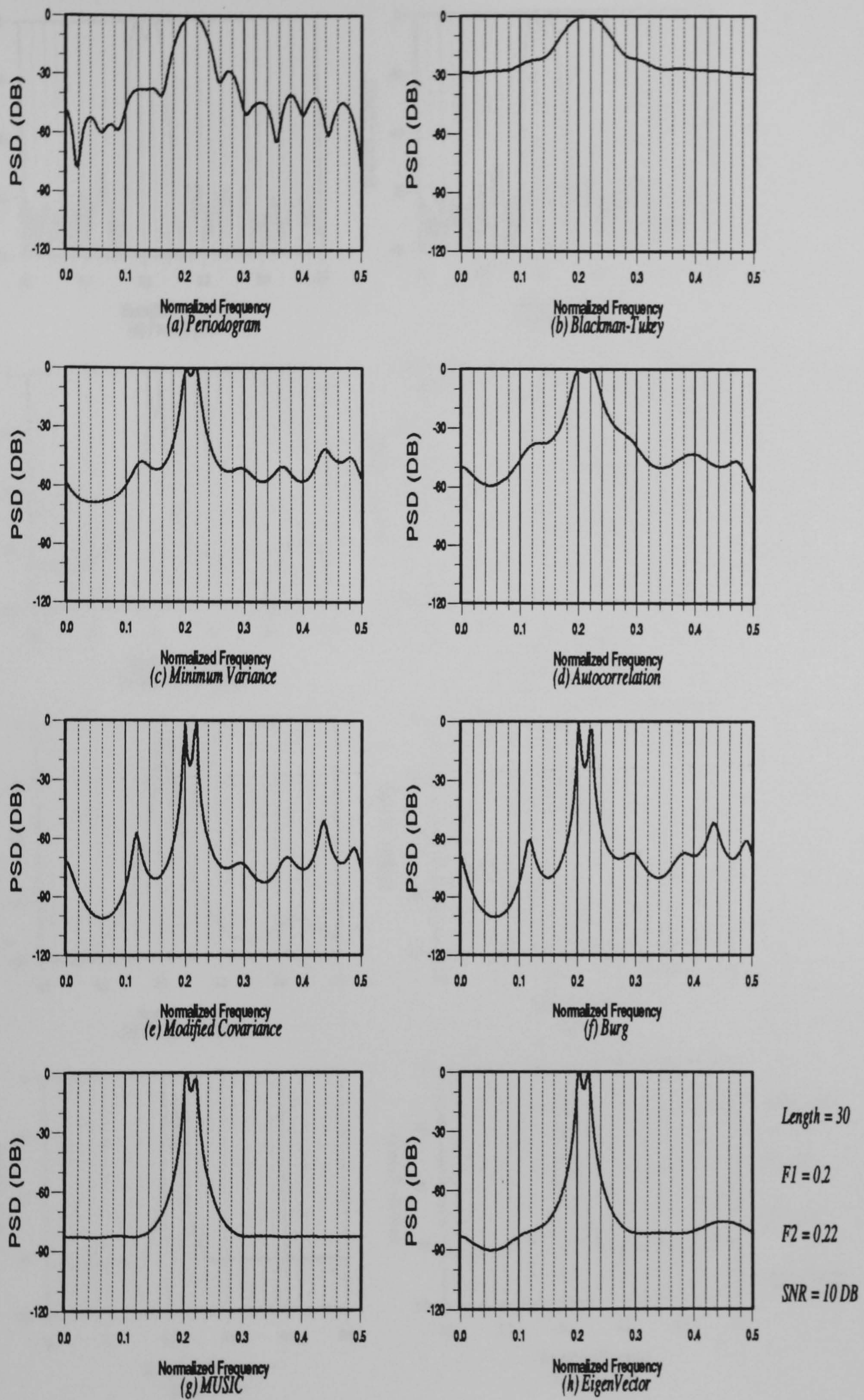


Figure 4.2: Record length of 30

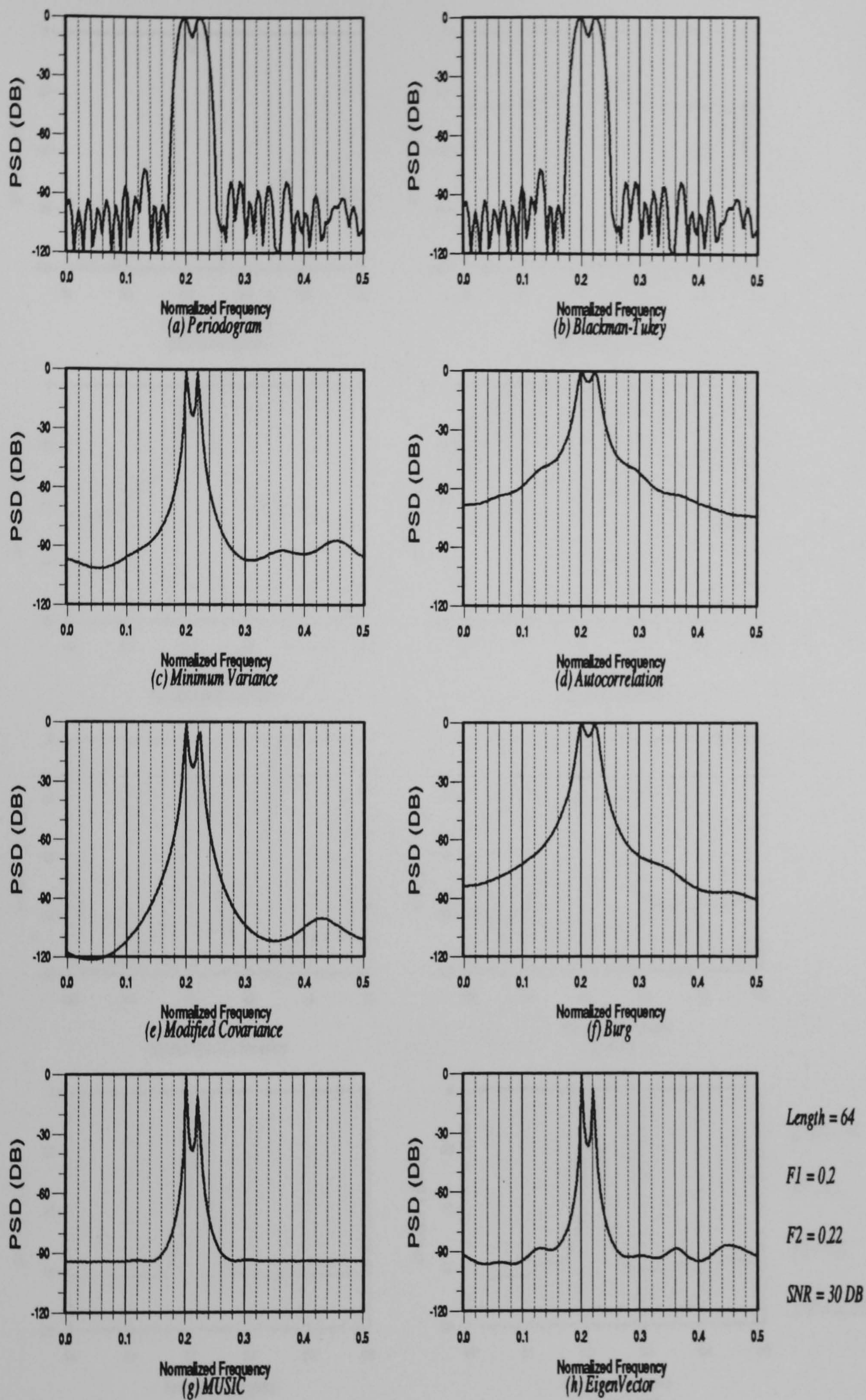


Figure 4.3: SNR of 30 dB

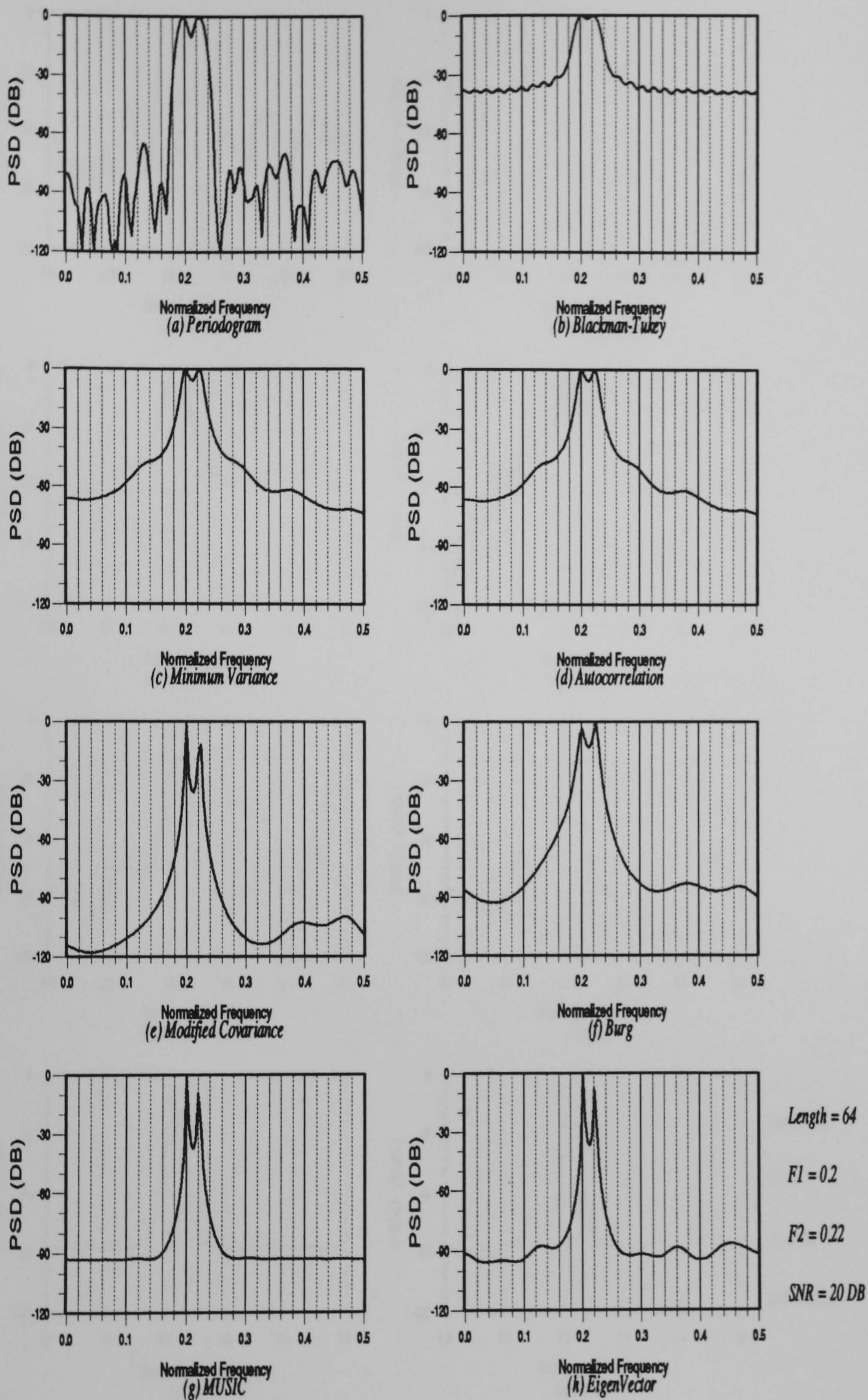


Figure 4.4: SNR of 20 dB

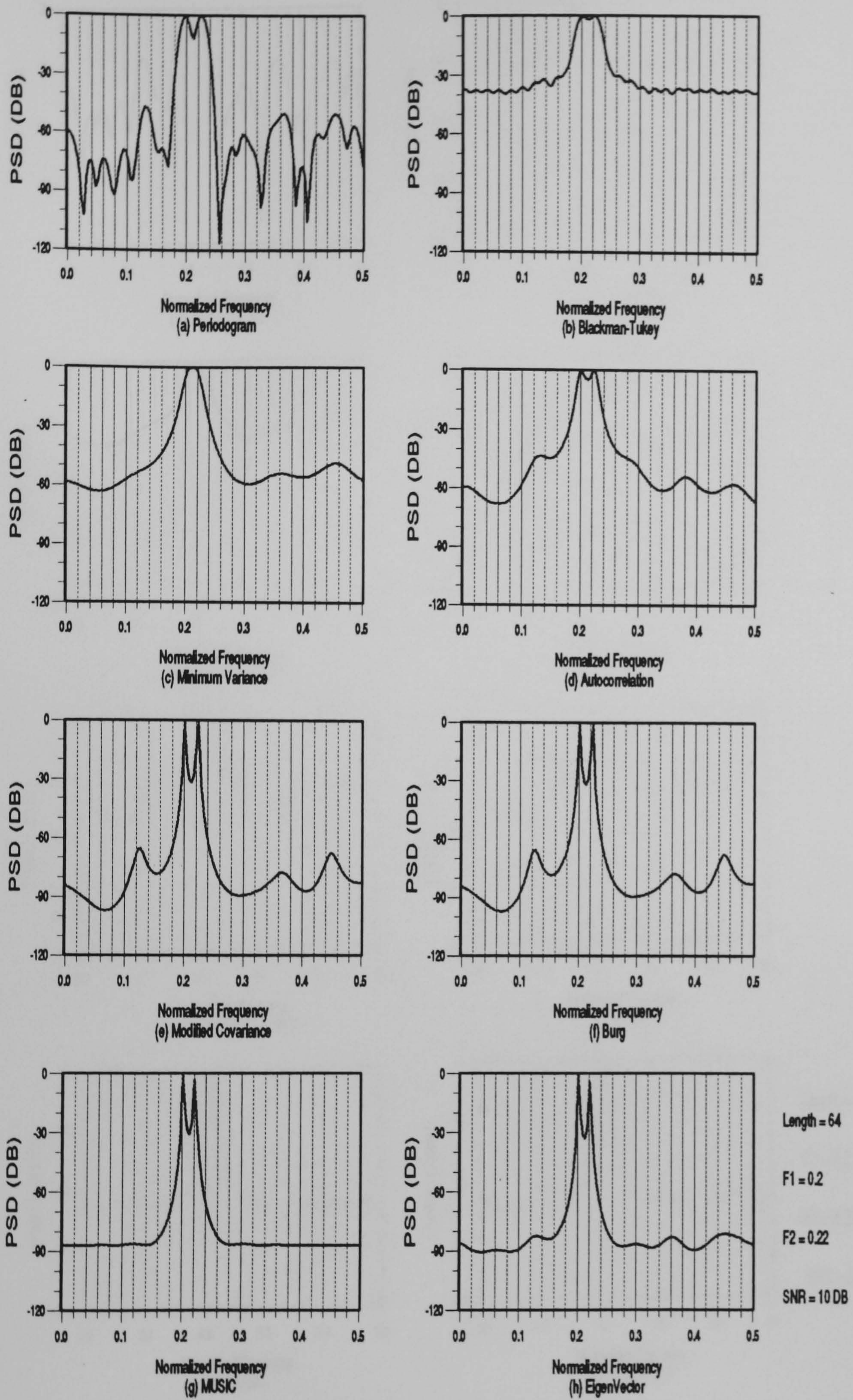


Figure 4.5: SNR of 10 dB

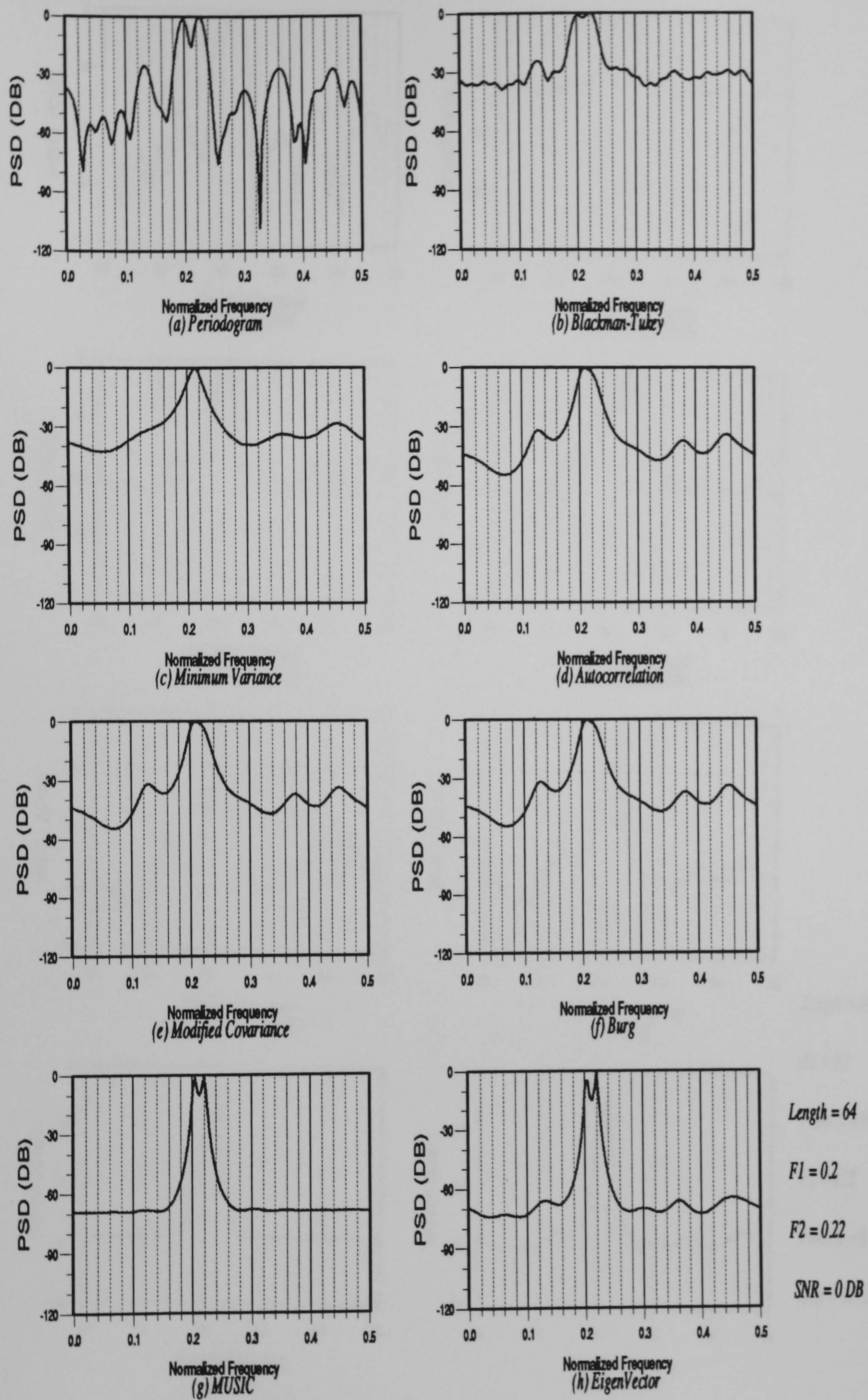


Figure 4.6: SNR of 0 dB

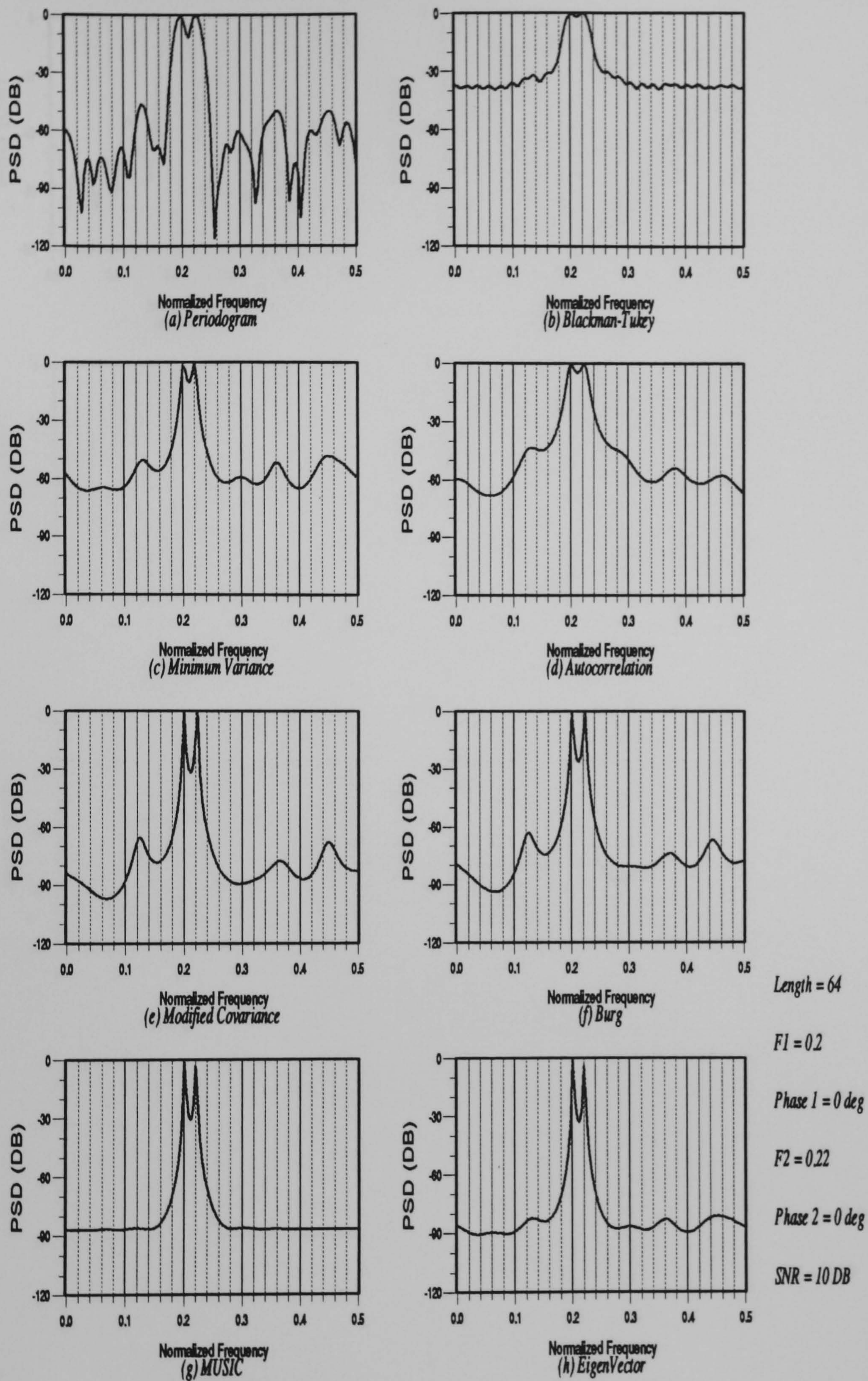
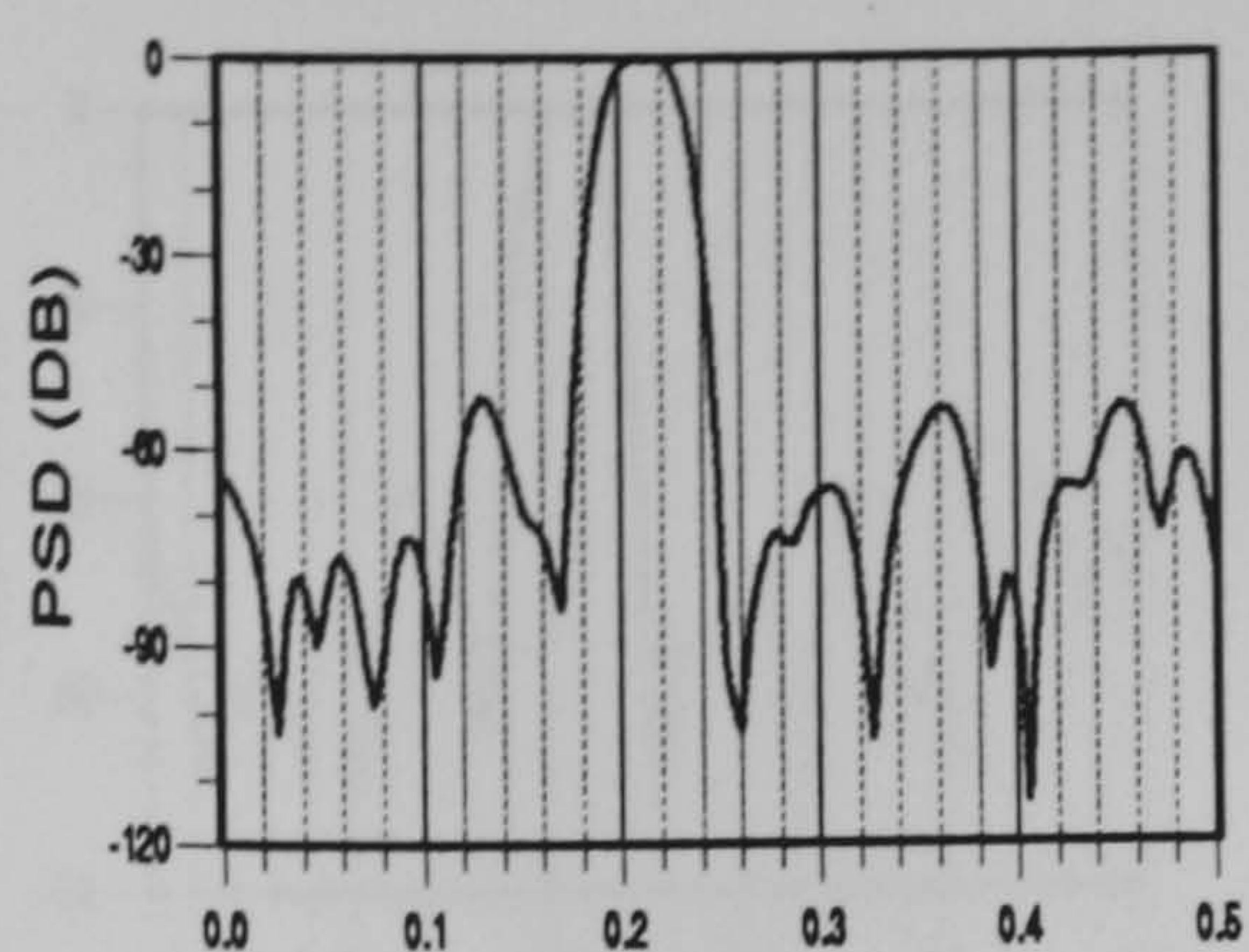
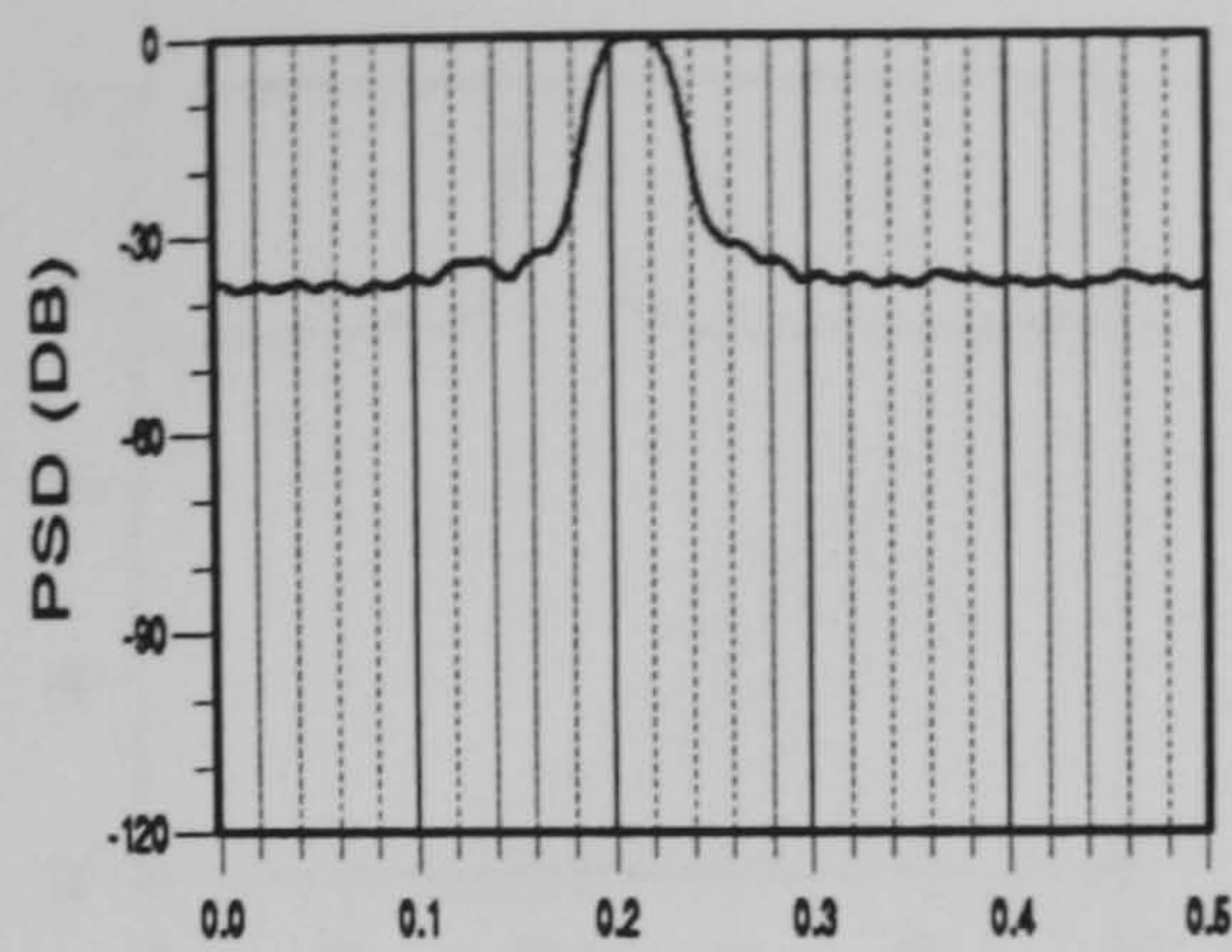


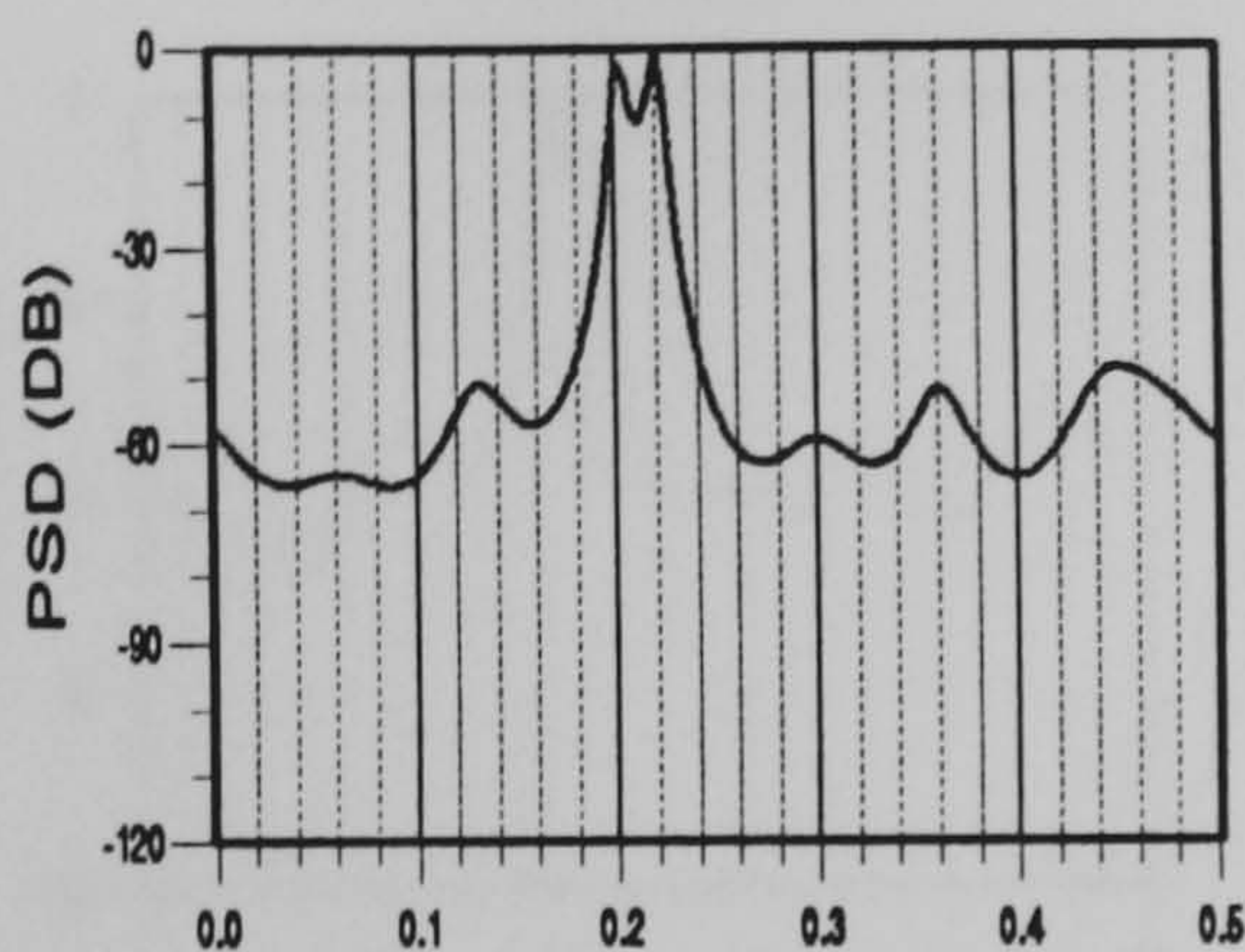
Figure 4.7: Phase difference of 0 degree



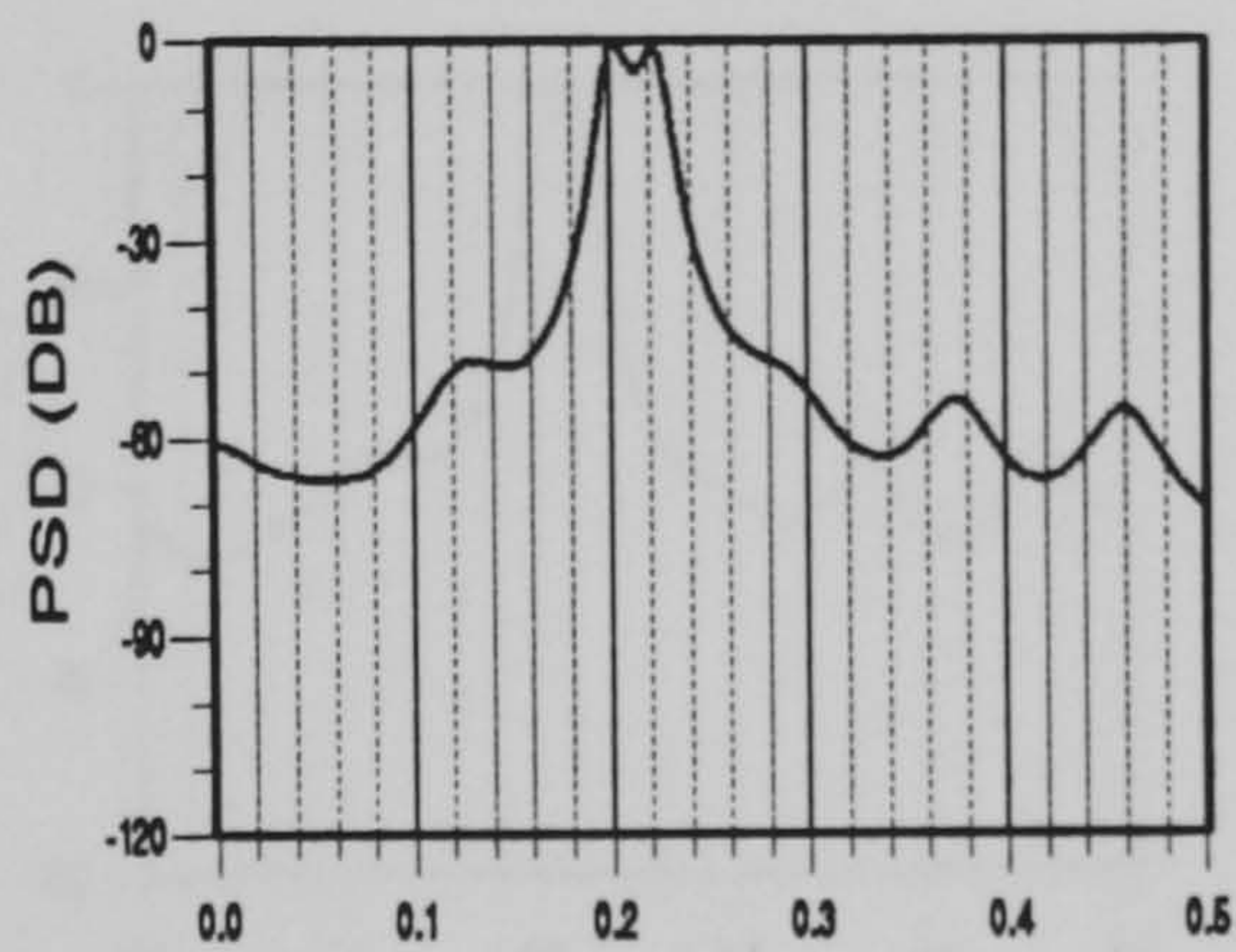
Normalized Frequency
(a) Periodogram



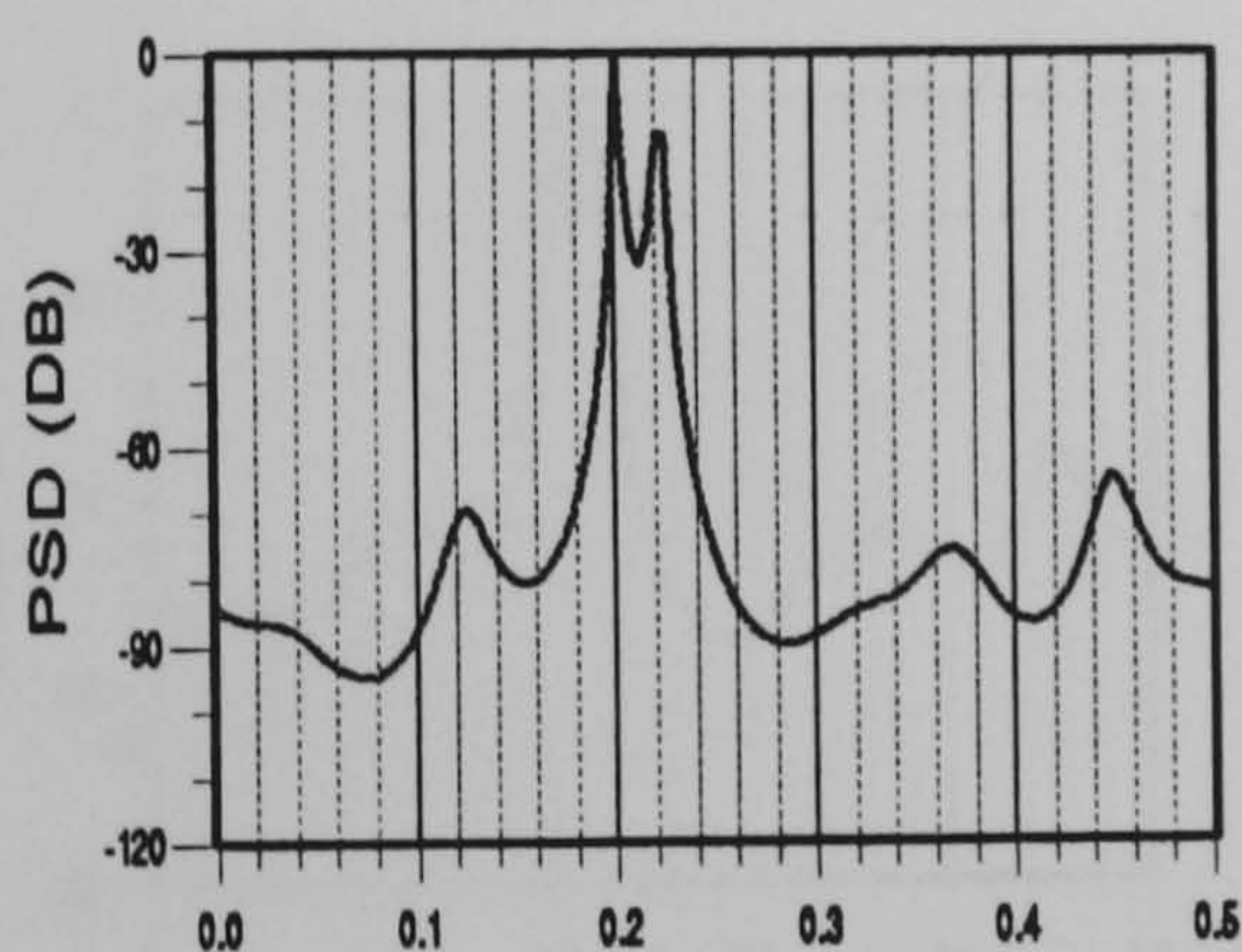
Normalized Frequency
(b) Blackman-Tukey



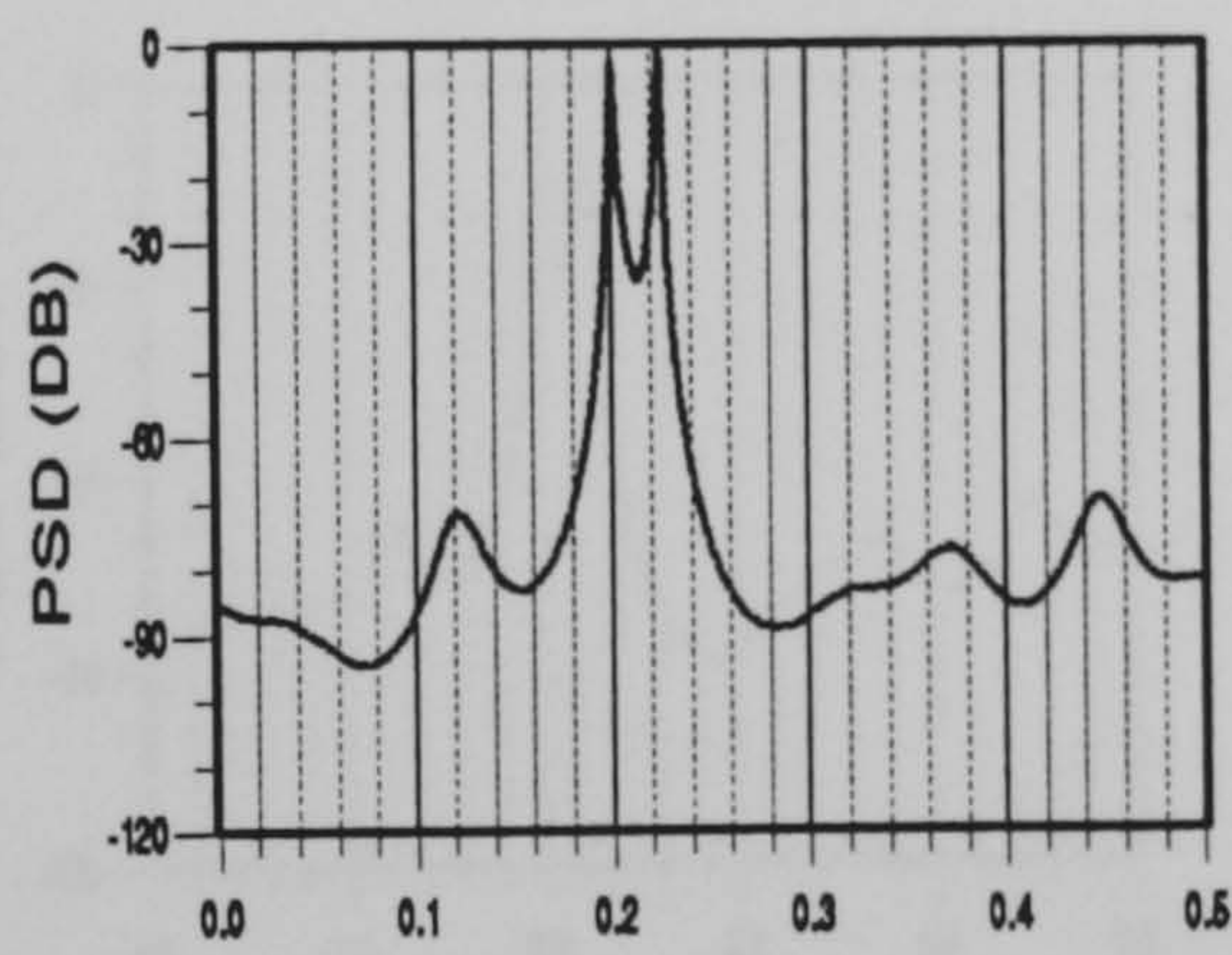
Normalized Frequency
(c) Minimum Variance



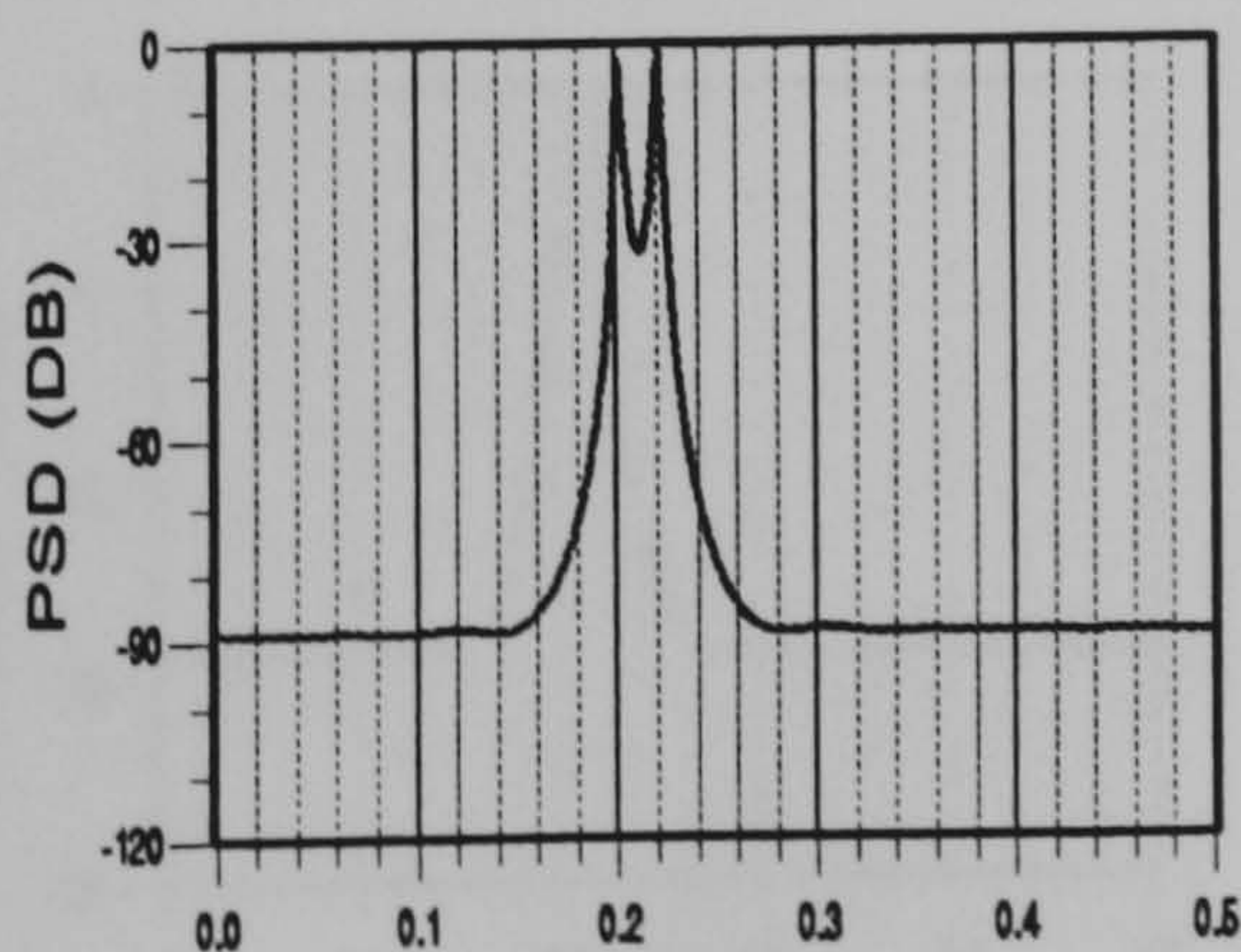
Normalized Frequency
(d) Autocorrelation



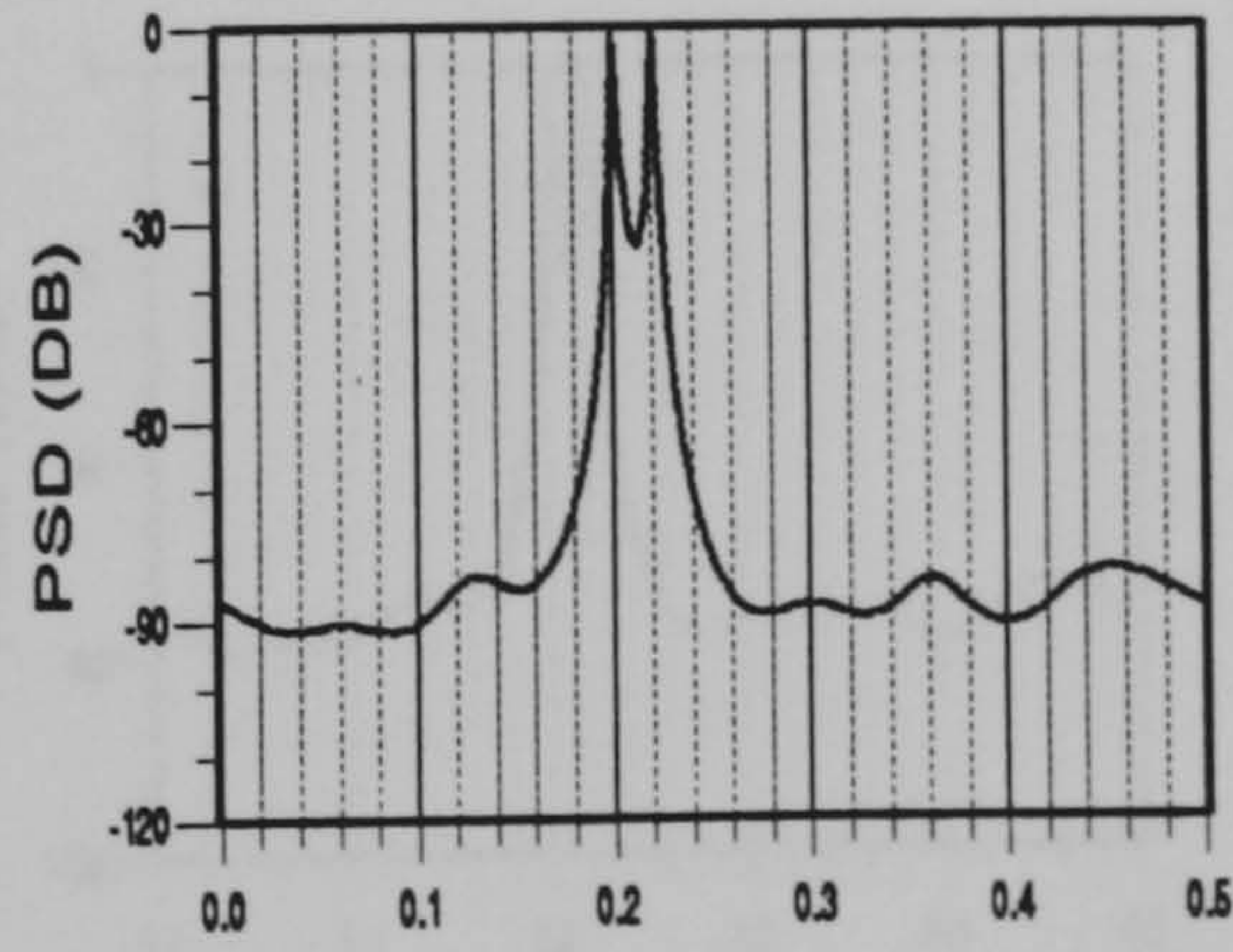
Normalized Frequency
(e) Modified Covariance



Normalized Frequency
(f) Burg



Normalized Frequency
(g) MUSIC



Normalized Frequency
(h) EigenVector

Length = 64

$F1 = 0.2$

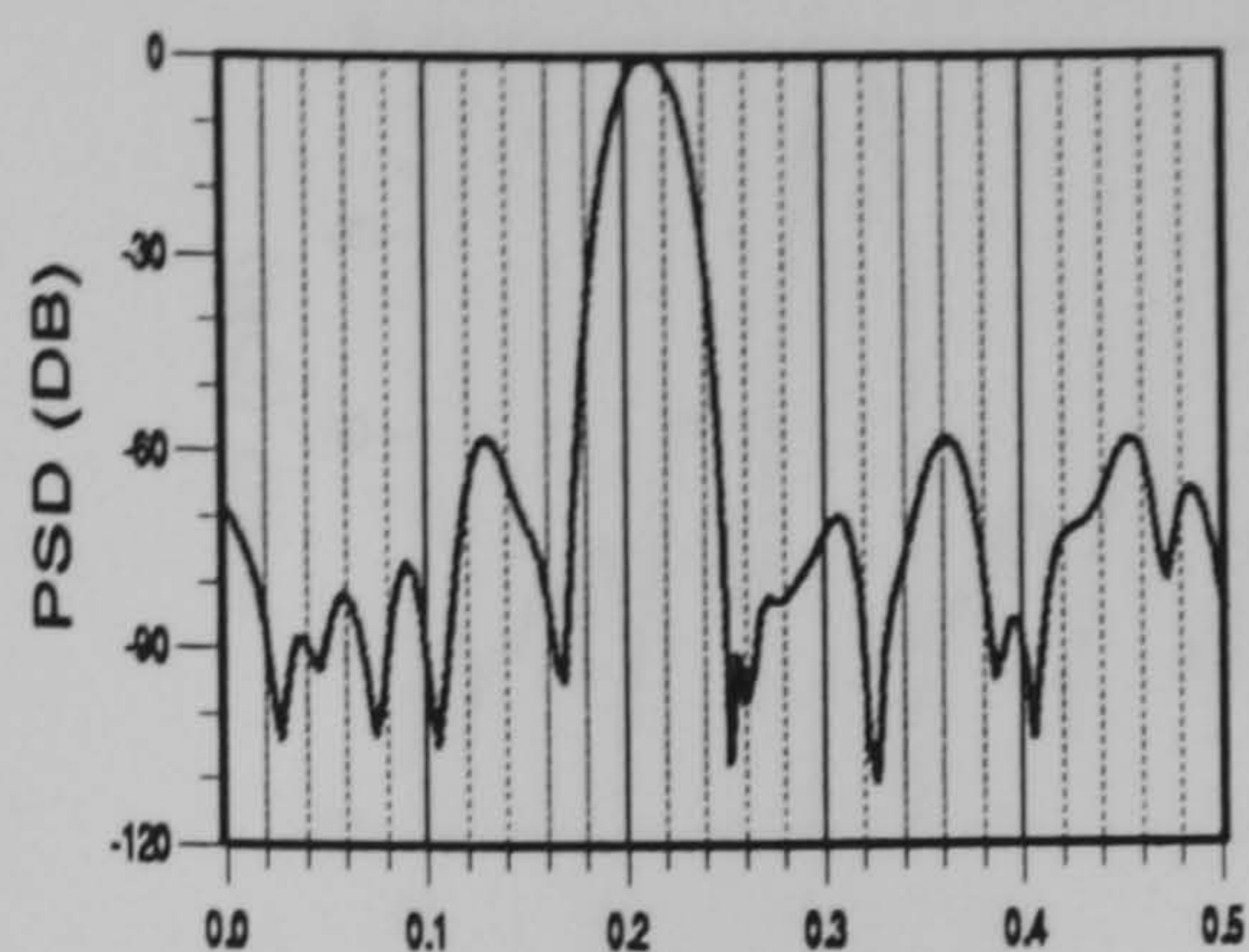
Phase 1 = 0 deg

$F2 = 0.22$

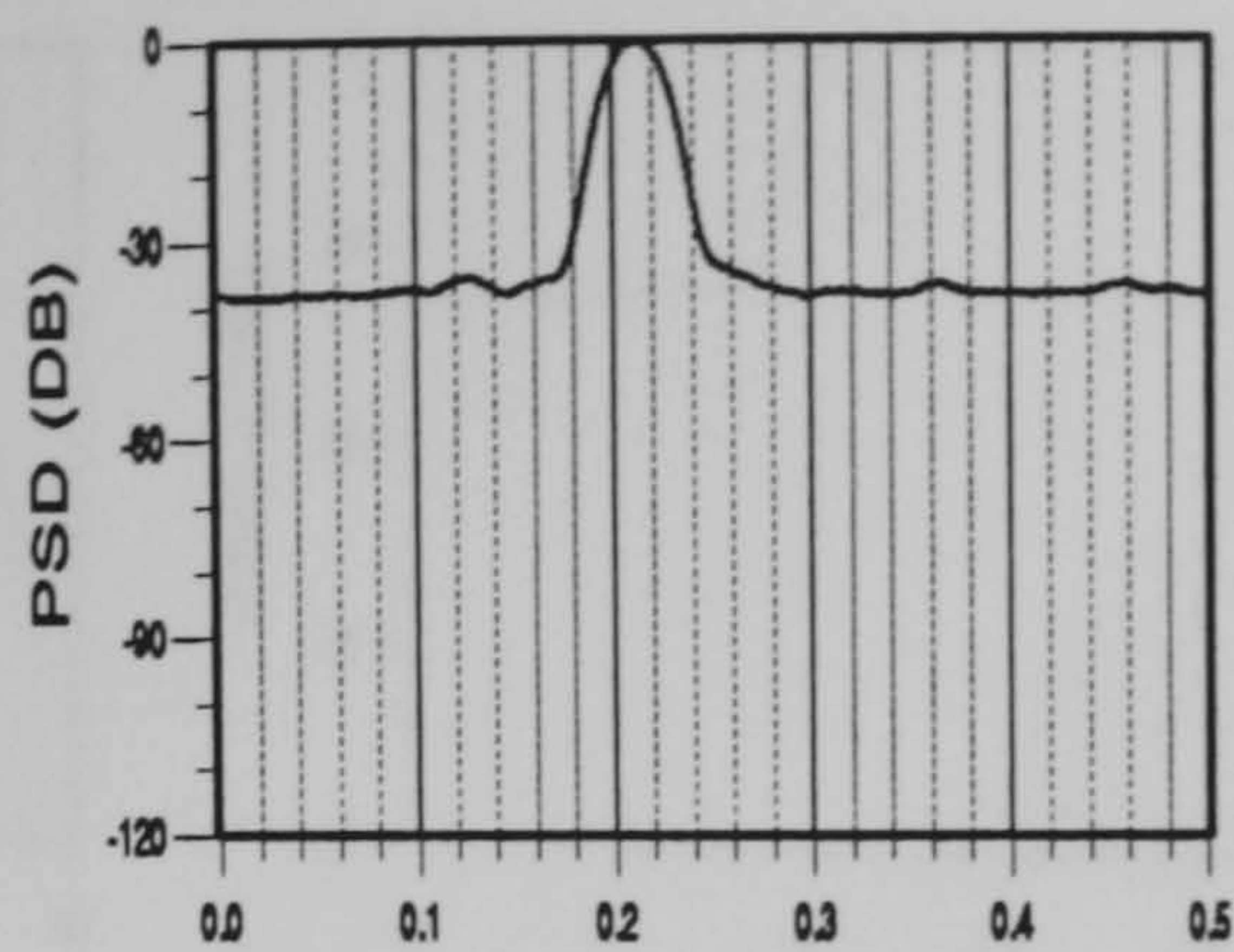
Phase 2 = 45 deg

SNR = 10 DB

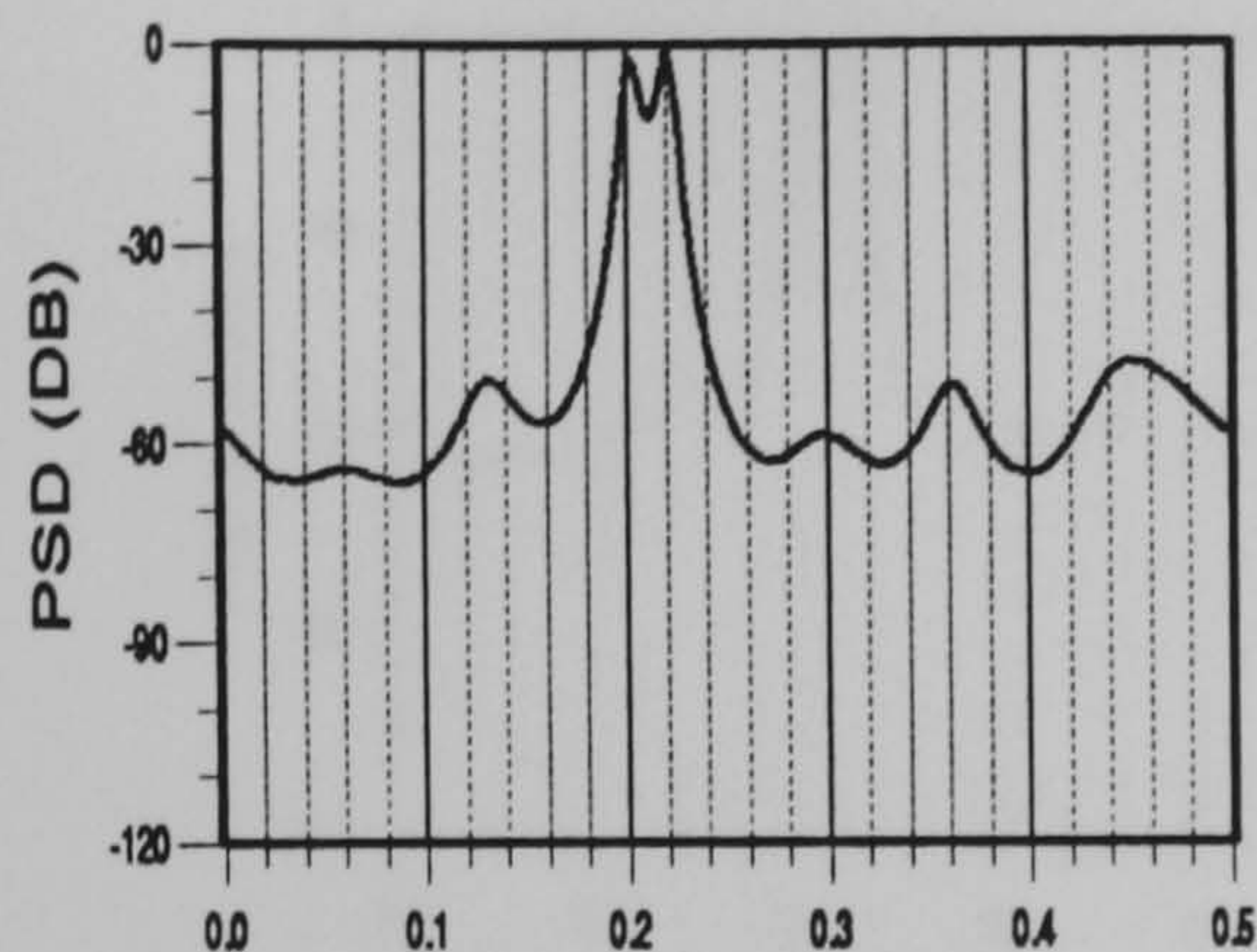
Figure 4.8: Phase difference of 45 degree



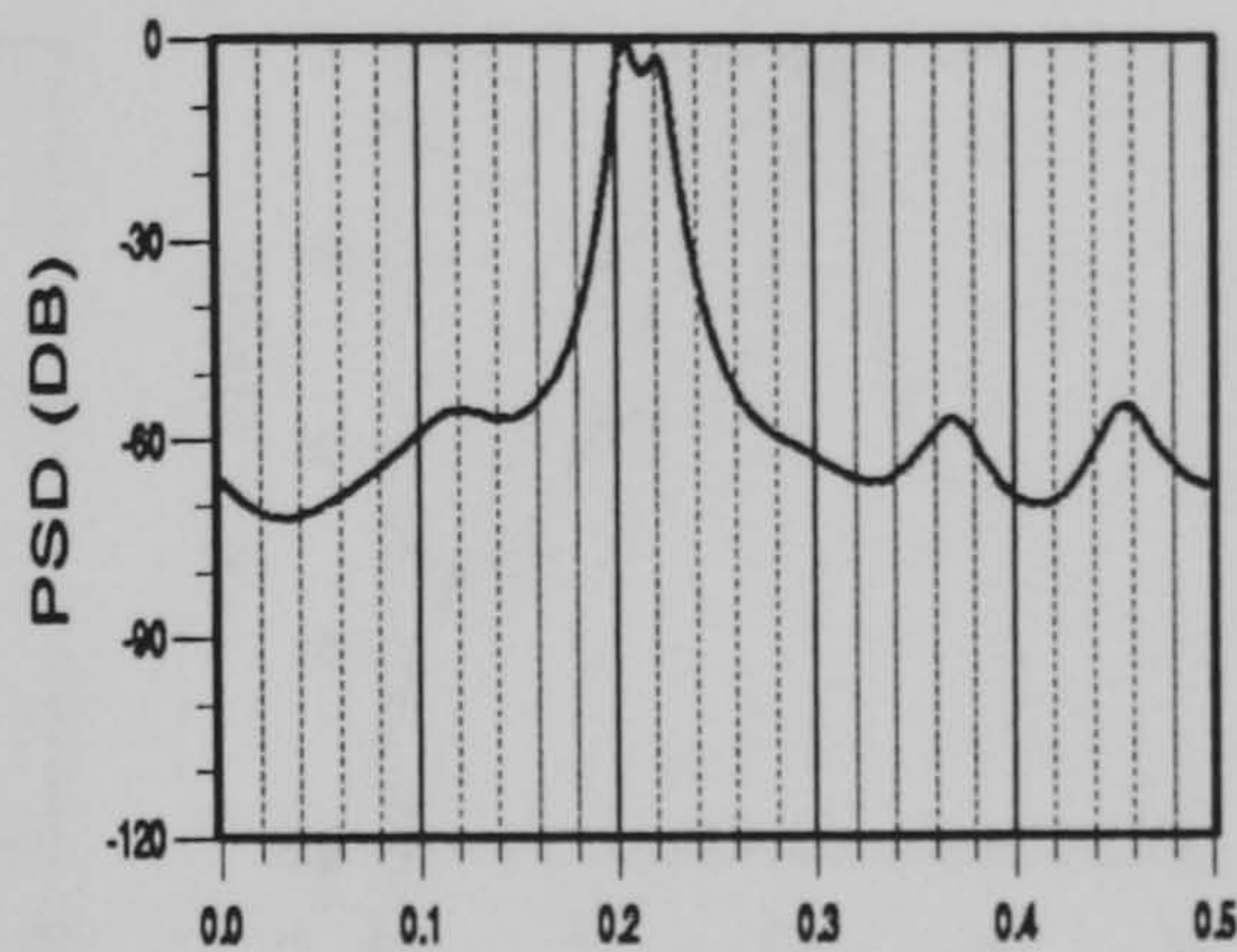
Normalized Frequency
(a) Periodogram



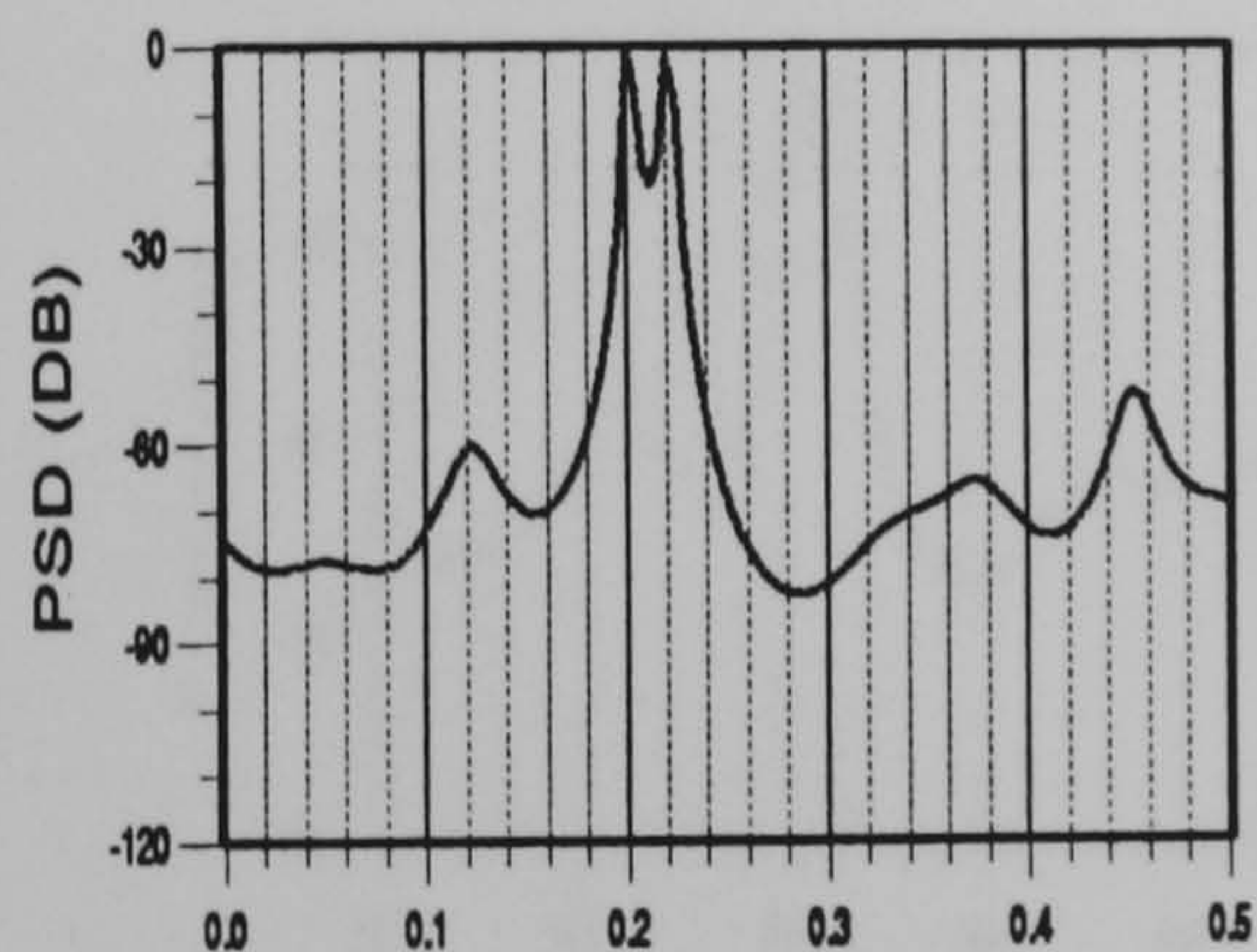
Normalized Frequency
(b) Blackman-Tukey



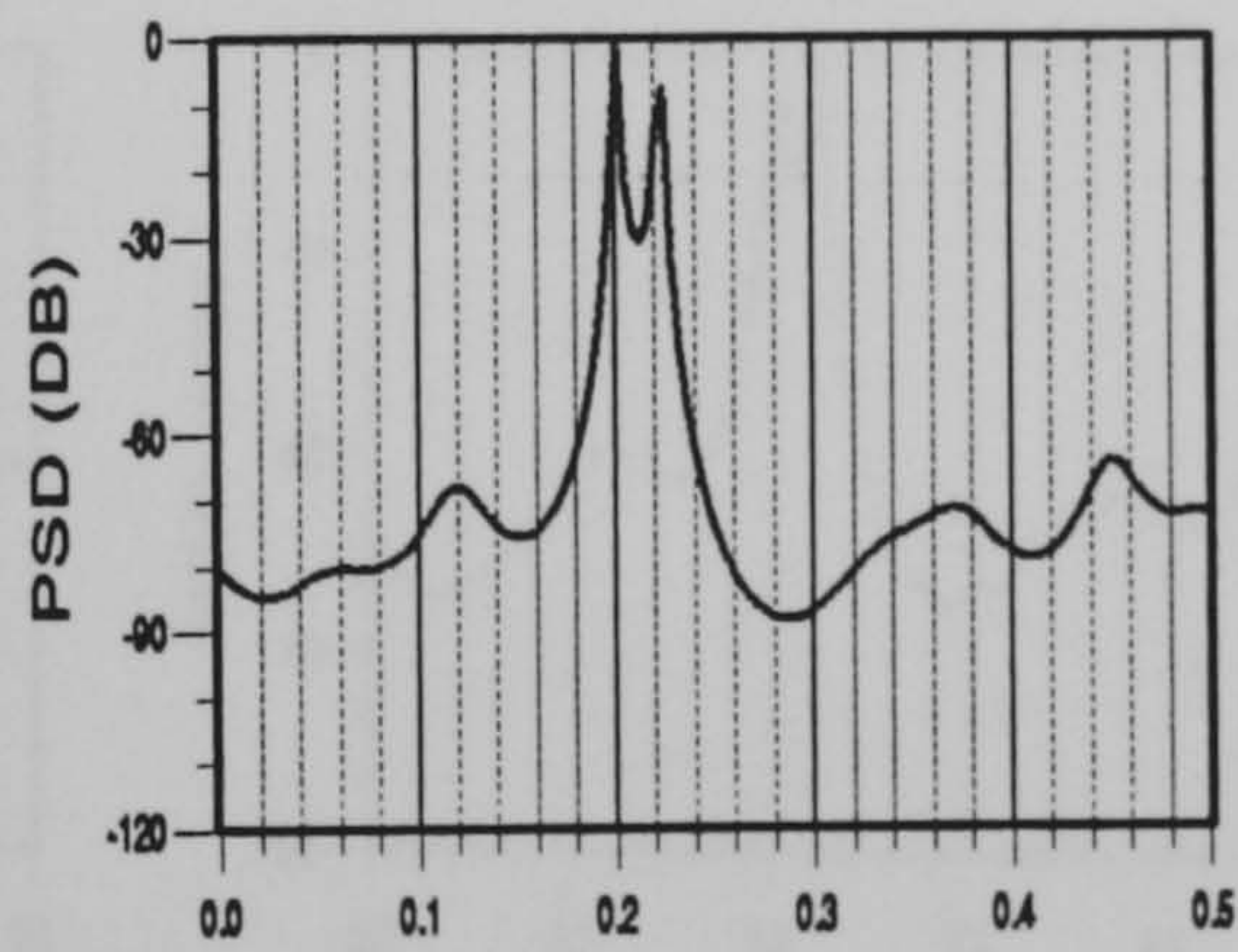
Normalized Frequency
(c) Minimum Variance



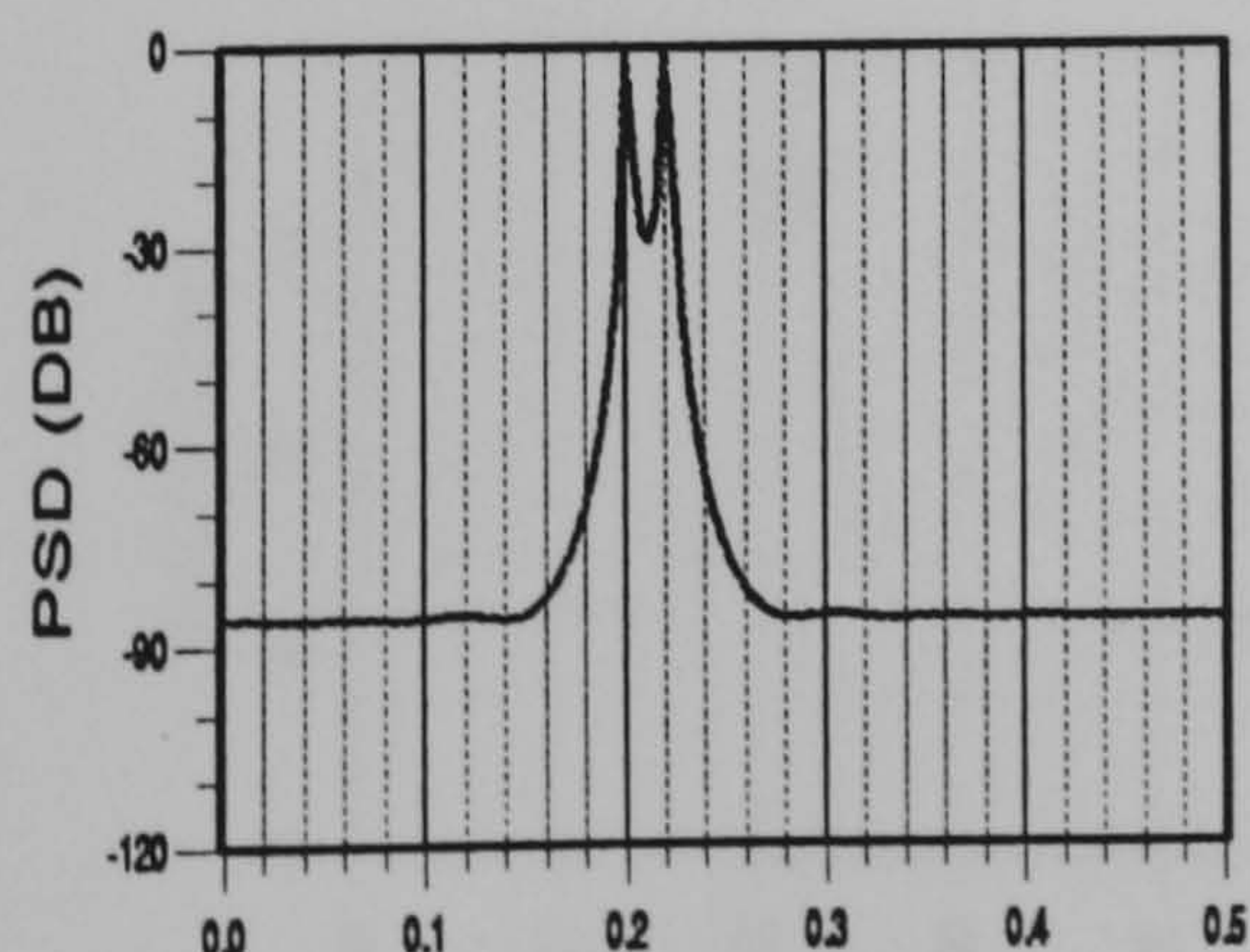
Normalized Frequency
(d) Autocorrelation



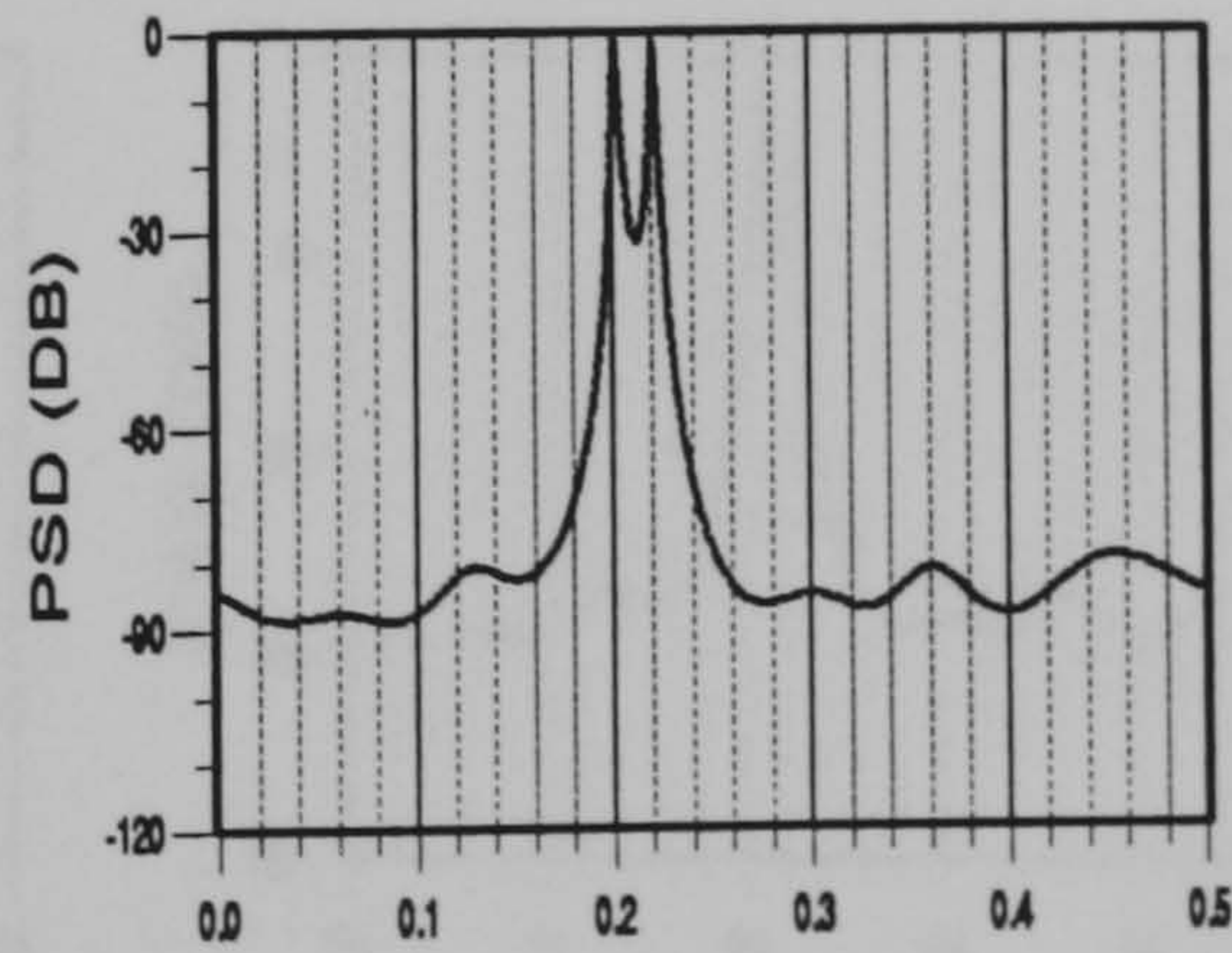
Normalized Frequency
(e) Modified Covariance



Normalized Frequency
(f) Burg



Normalized Frequency
(g) MUSIC



Normalized Frequency
(h) EigenVector

Length = 64

$F1 = 0.2$

Phase 1 = 0 deg

$F2 = 0.22$

Phase 2 = 90 deg

SNR = 10 DB

Figure 4.9: Phase difference of 90 degree

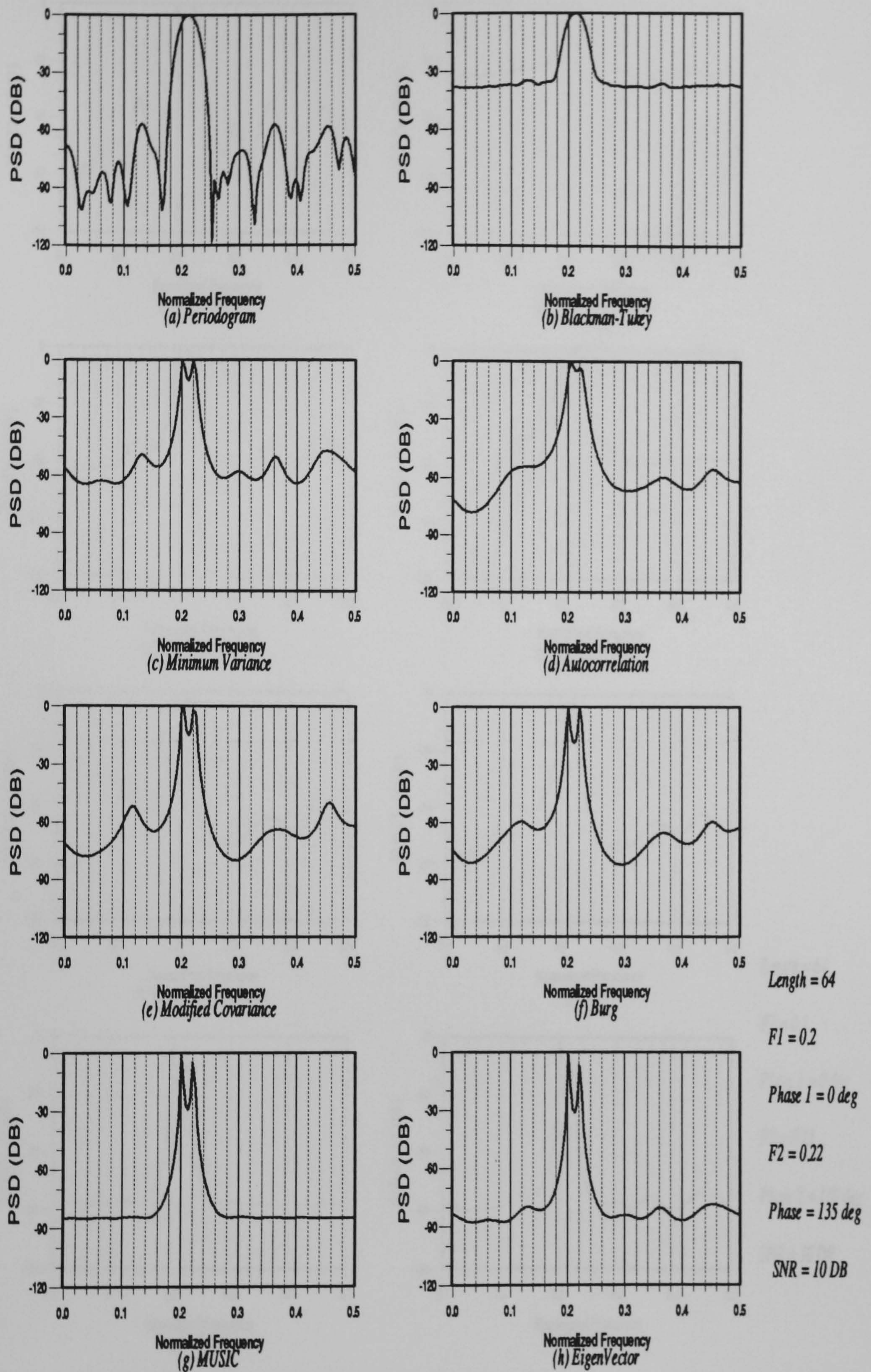


Figure 4.10: Phase difference of 135 degree

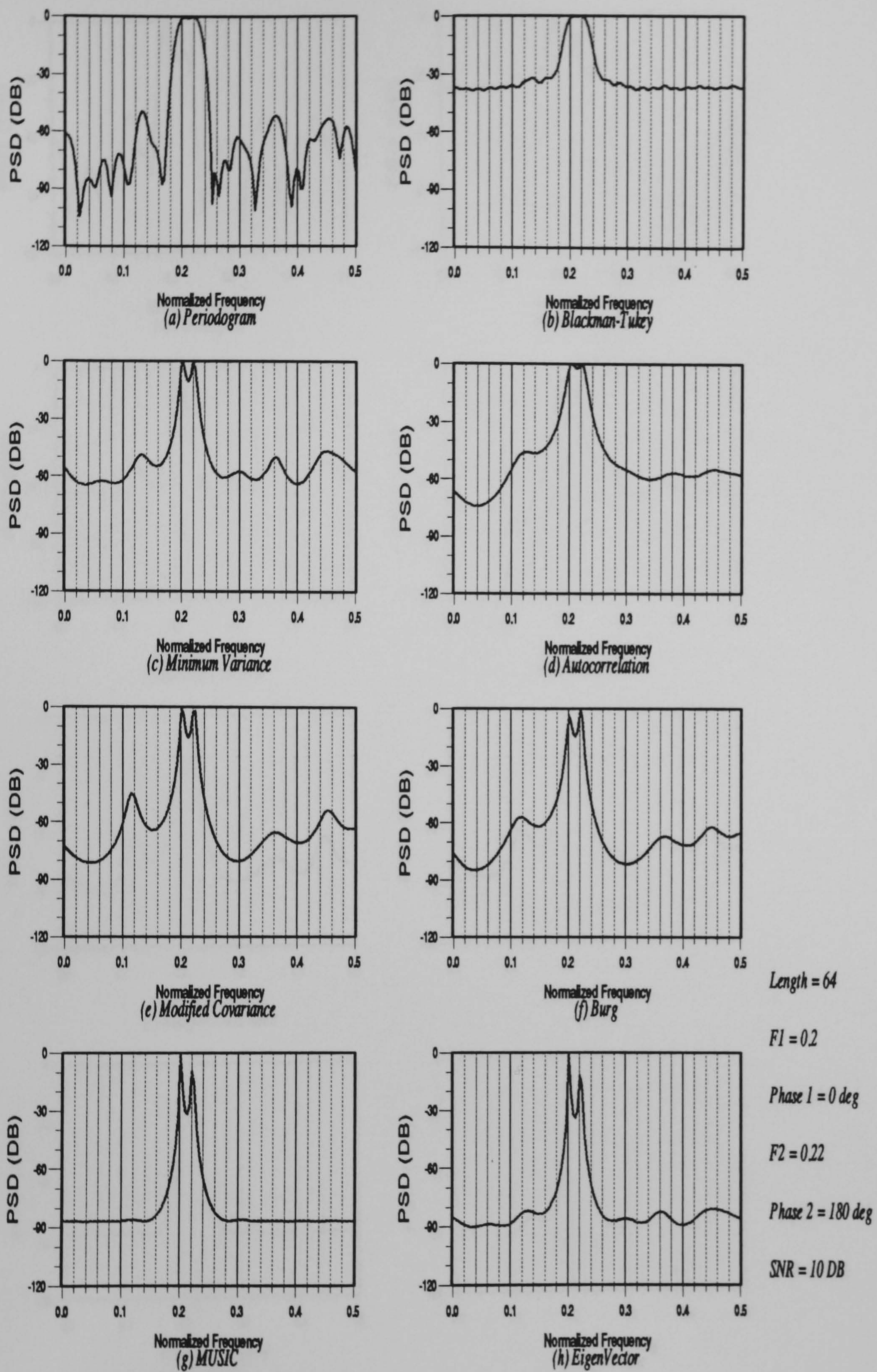
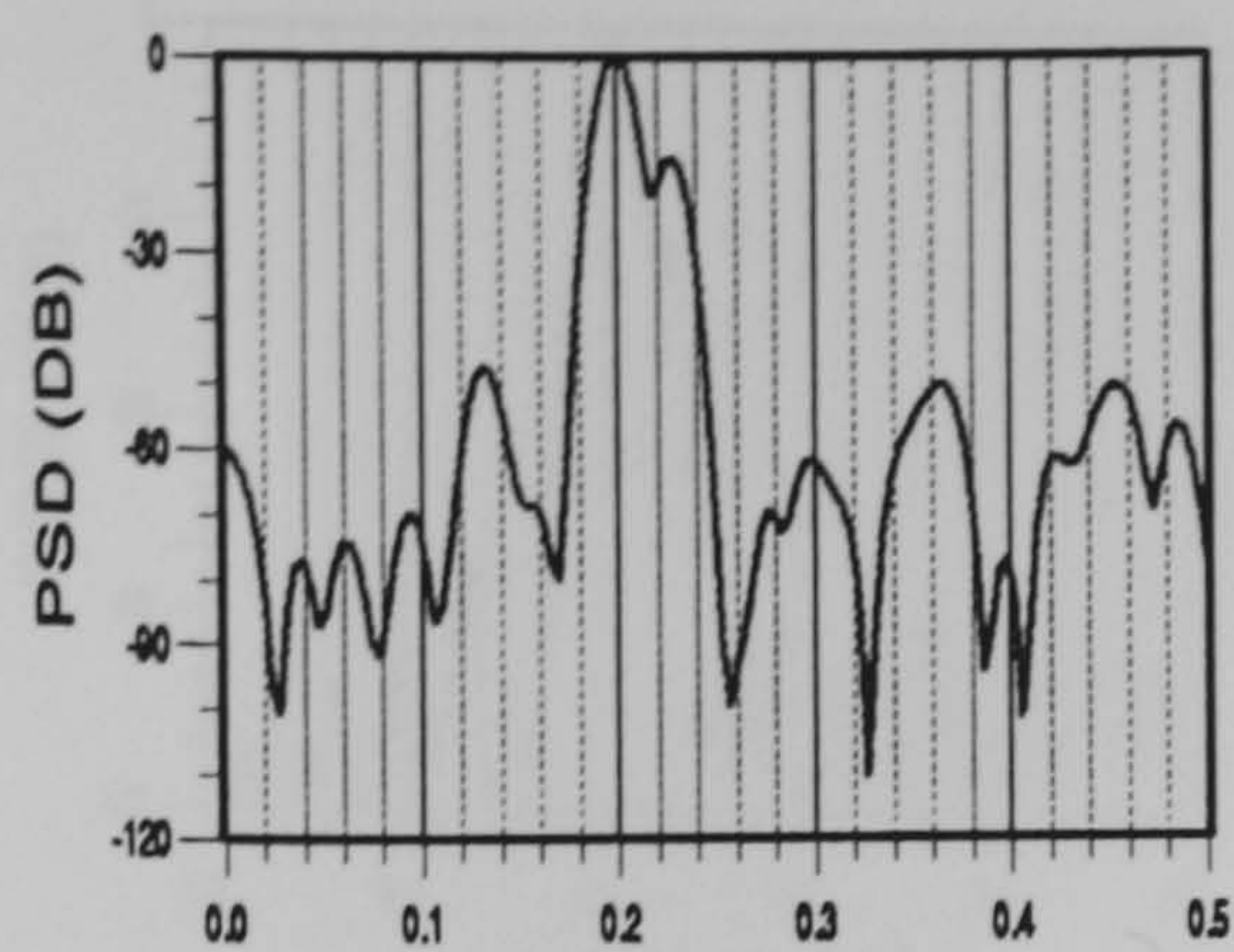
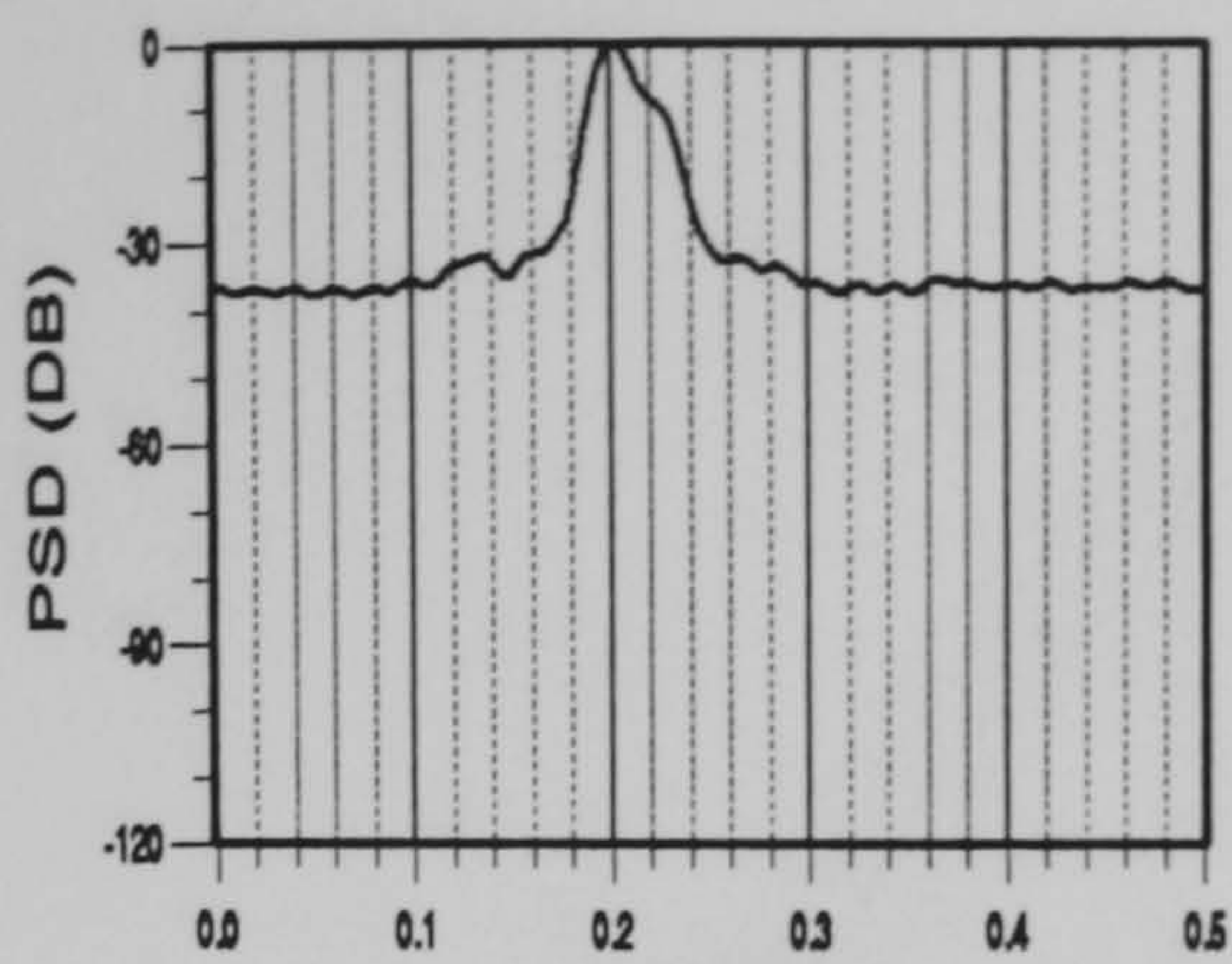


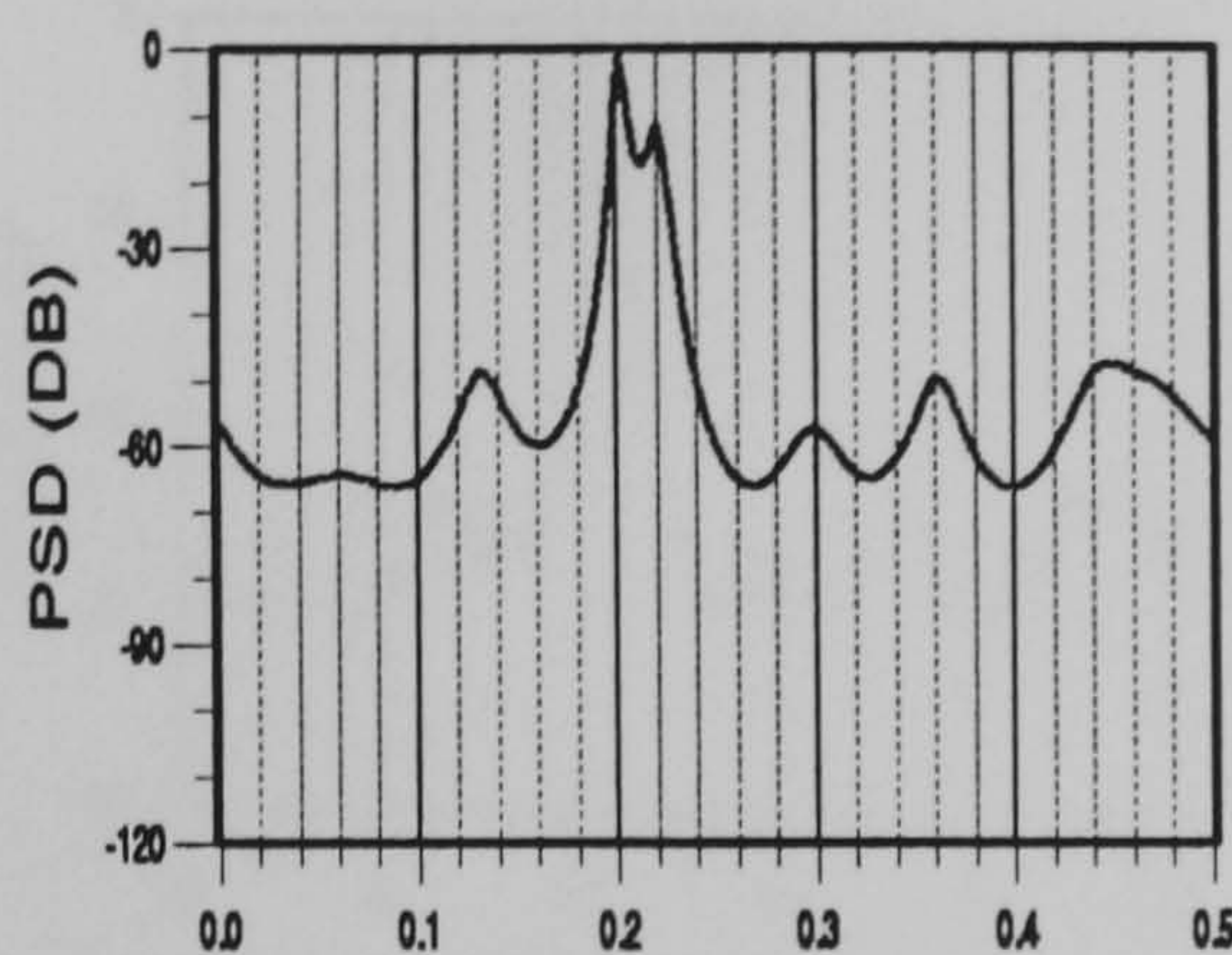
Figure 4.11: Phase difference of 180 degree



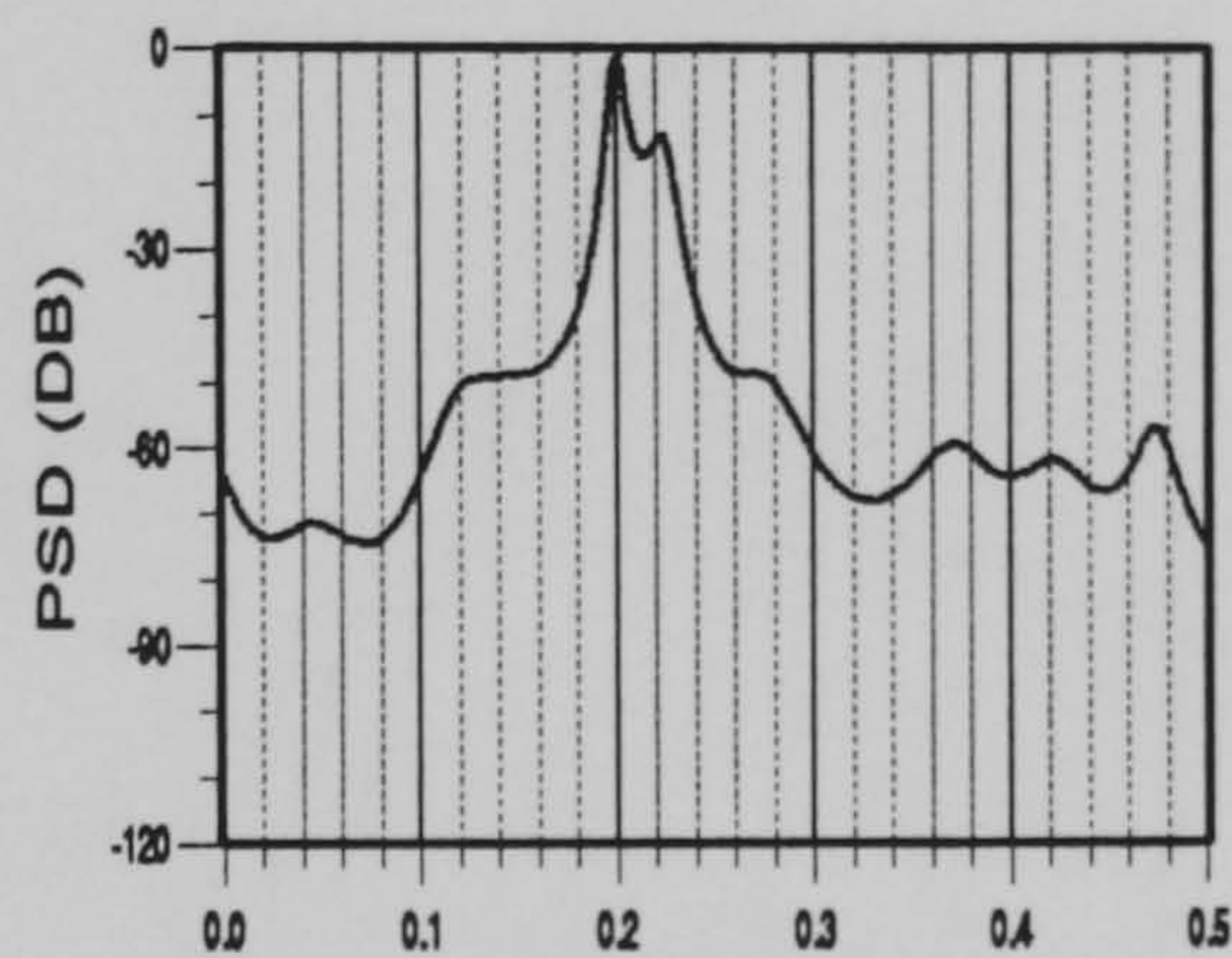
Normalized Frequency
(a) Periodogram



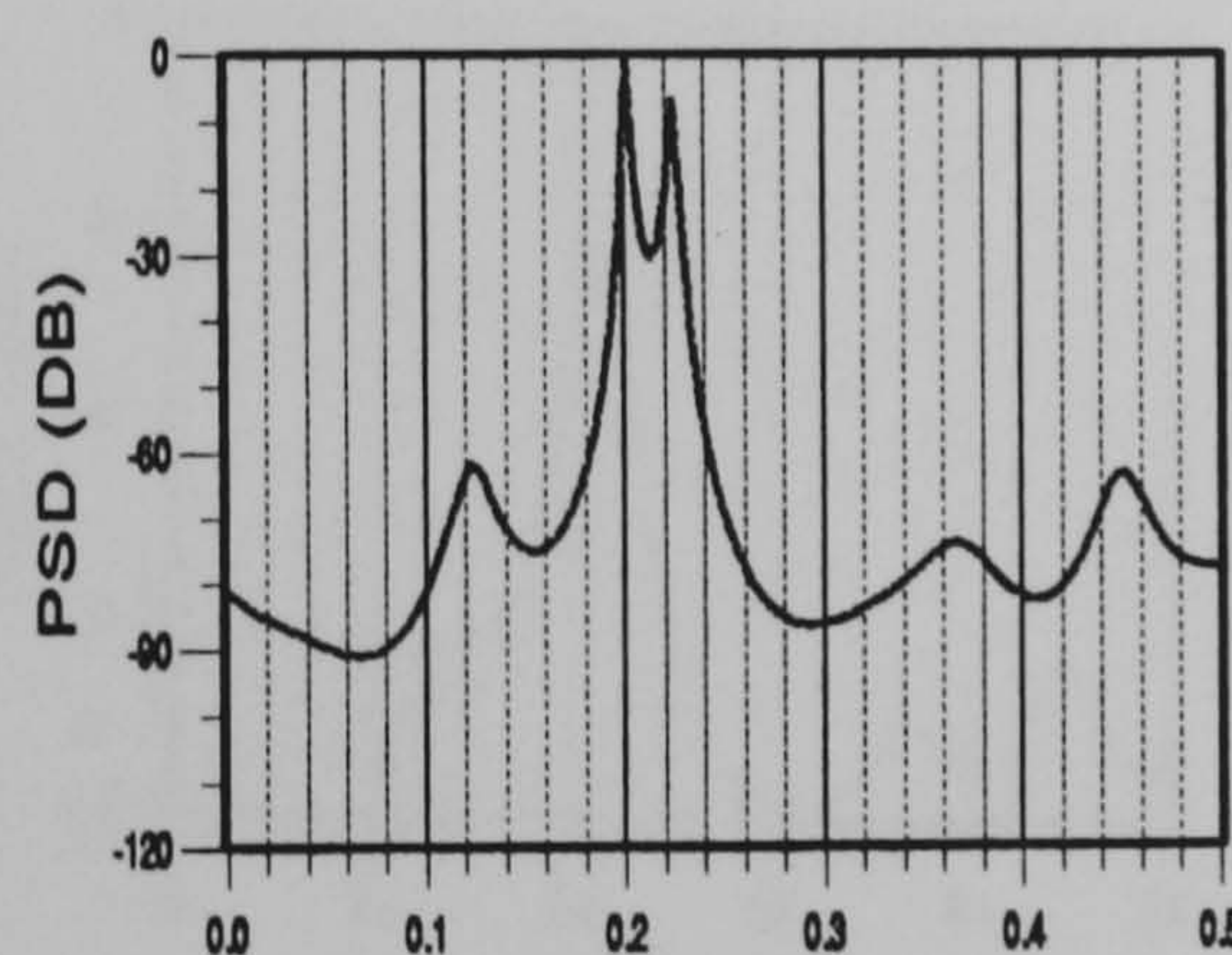
Normalized Frequency
(b) Blackman-Tukey



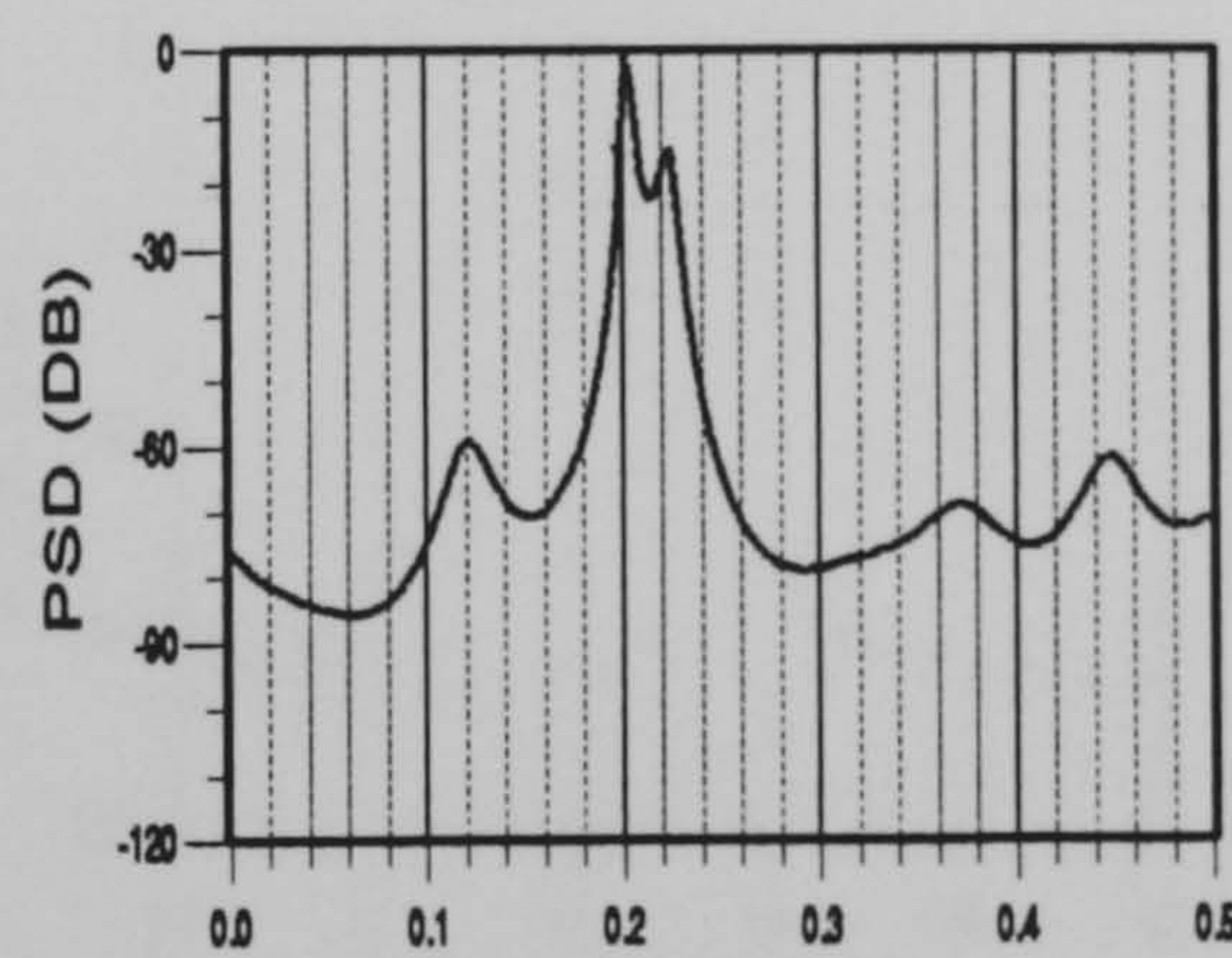
Normalized Frequency
(c) Minimum Variance



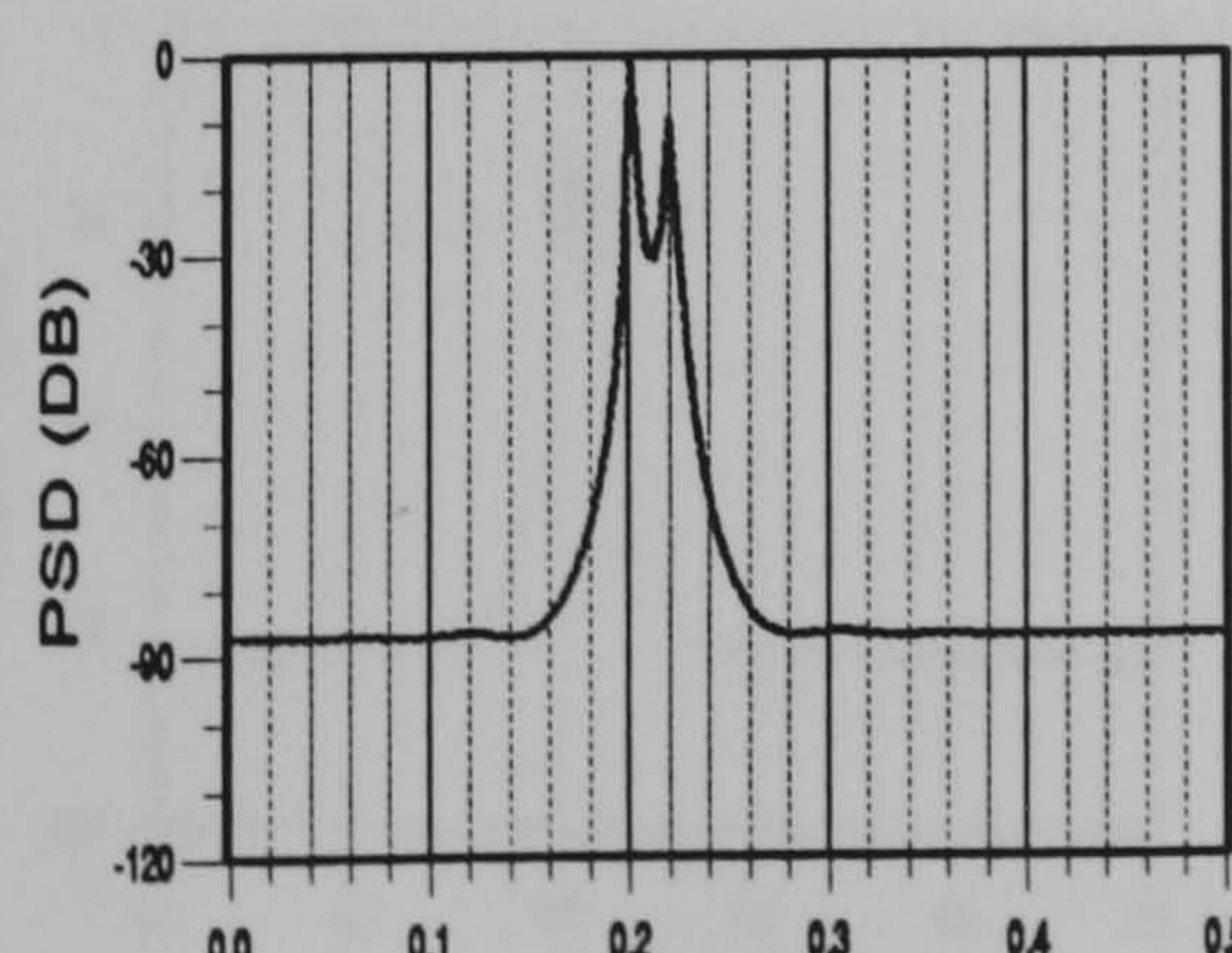
Normalized Frequency
(d) Autocorrelation



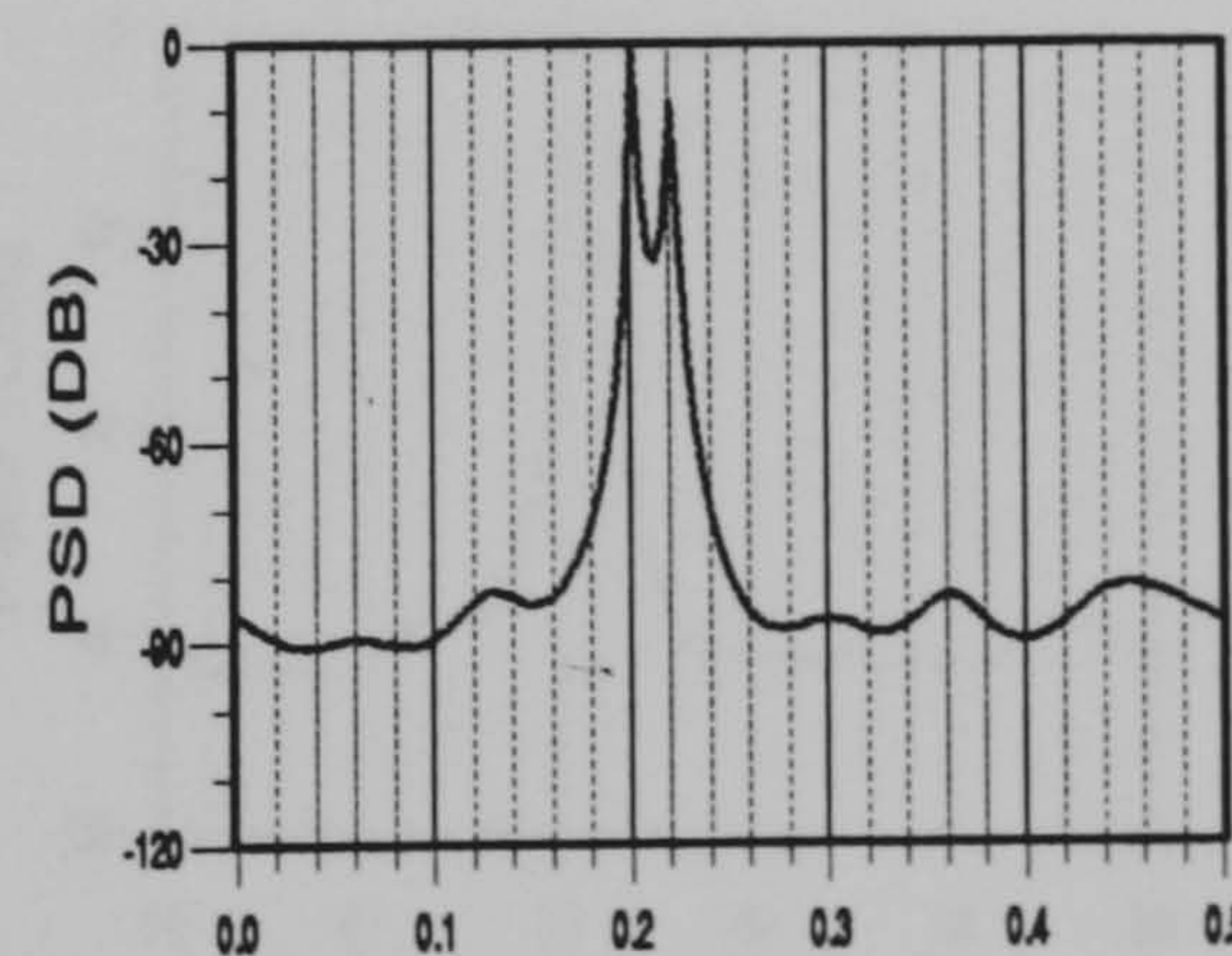
Normalized Frequency
(e) Modified Covariance



Normalized Frequency
(f) Burg



Normalized Frequency
(g) MUSIC



Normalized Frequency
(h) EigenVector

Length = 64

F1 = 0.2

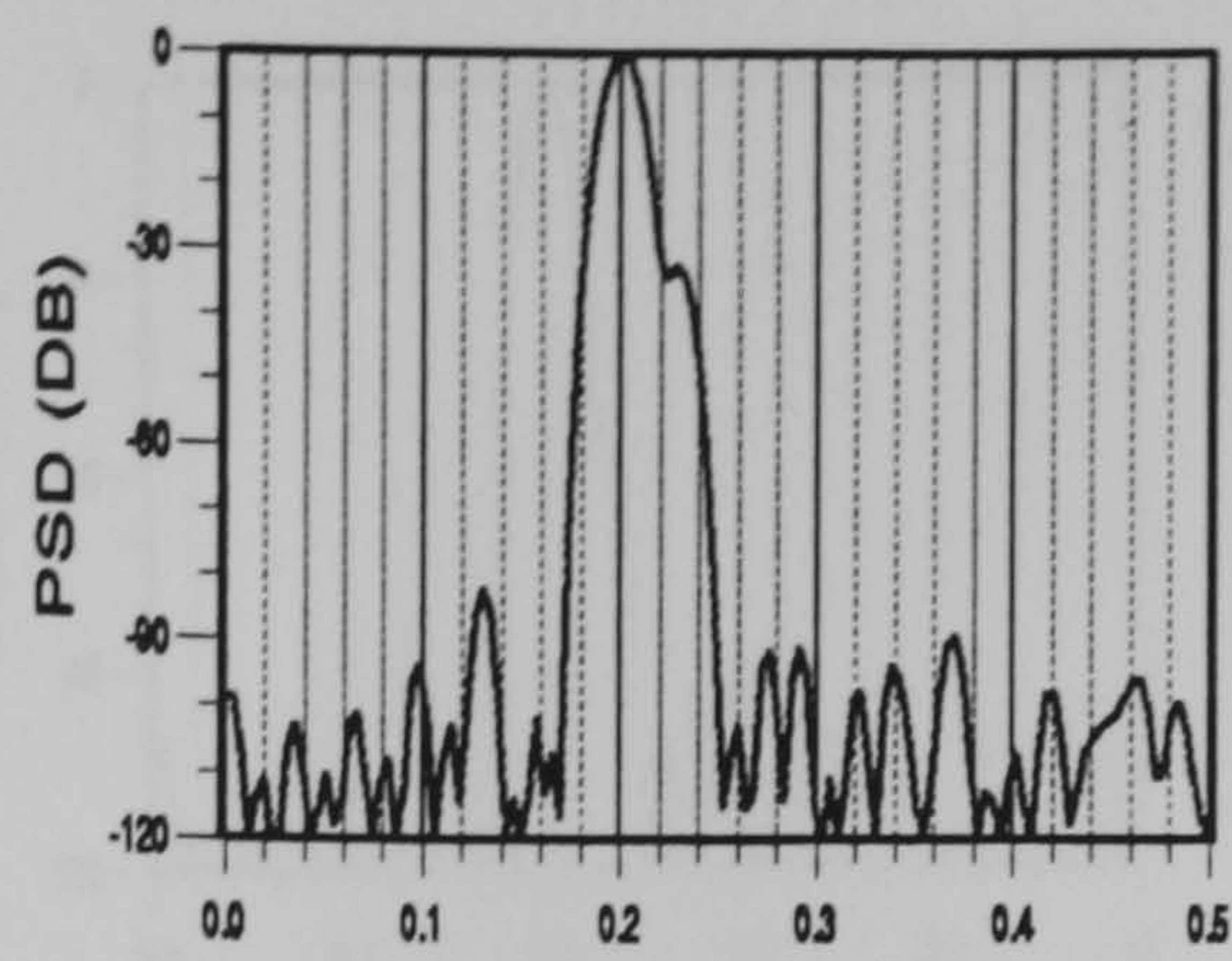
Amplitude 1 = 1.

F2 = 0.22

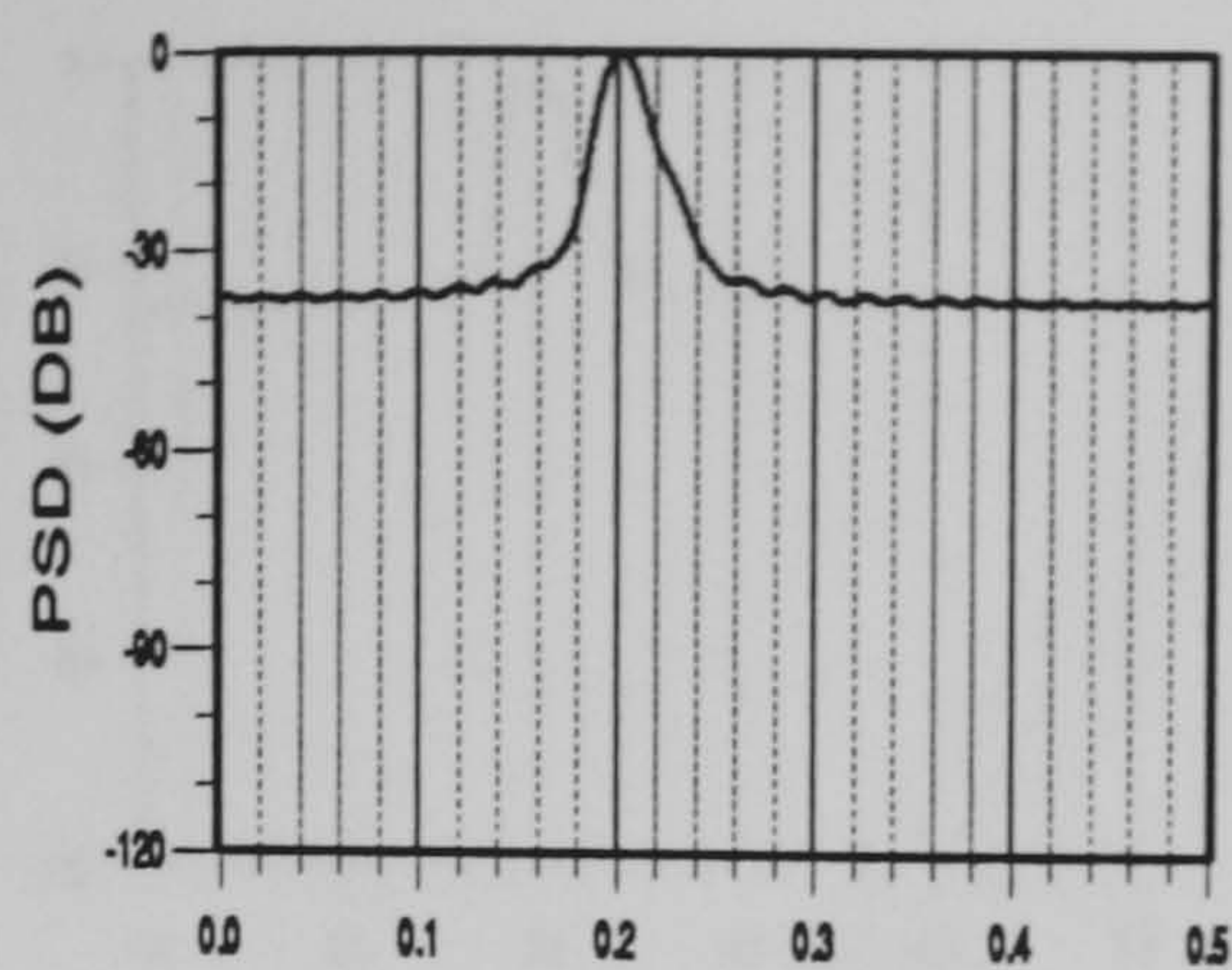
Amplitude 2 = 0.5

SNR = 10 DB

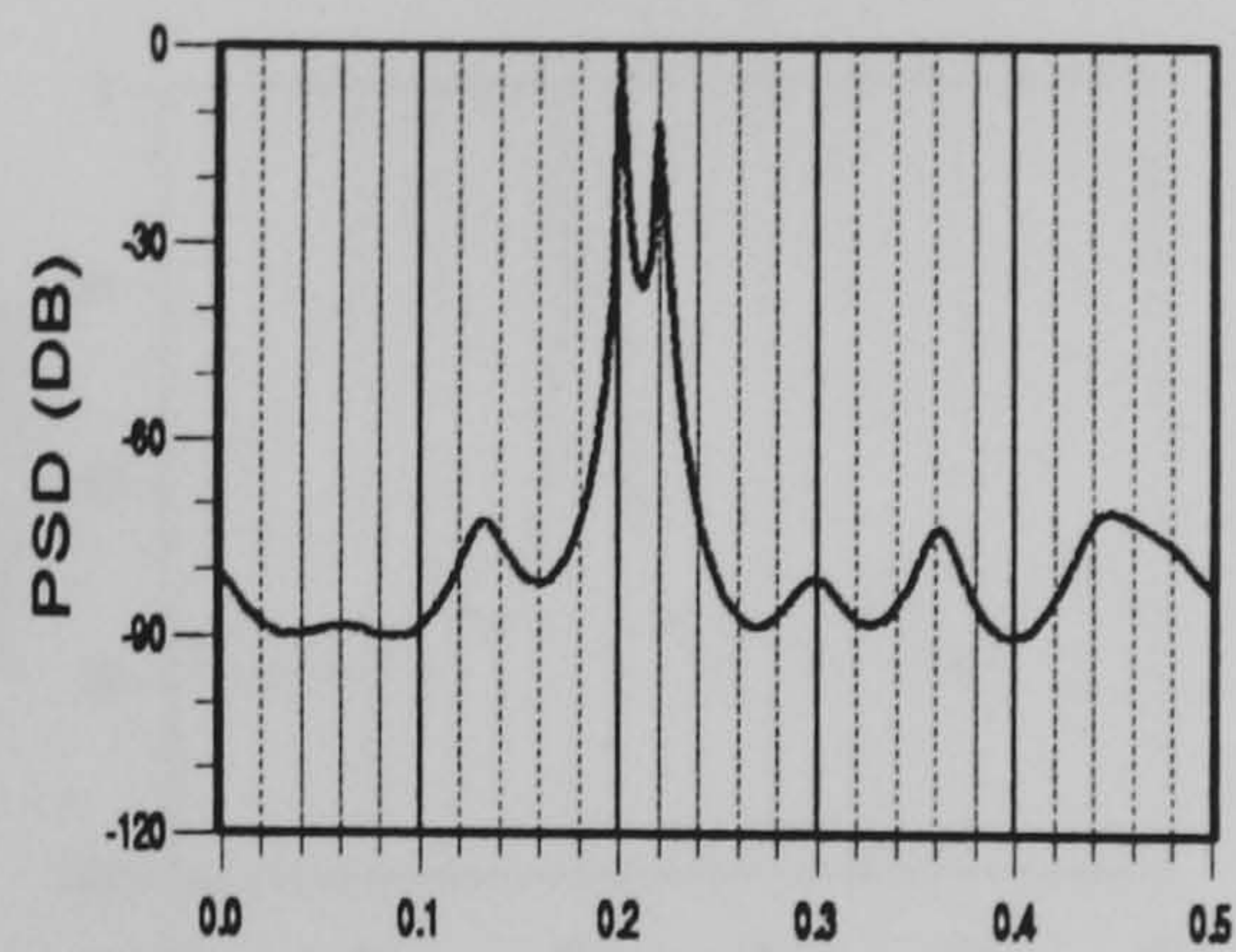
Figure 4.12: Relative amplitude of 0.5



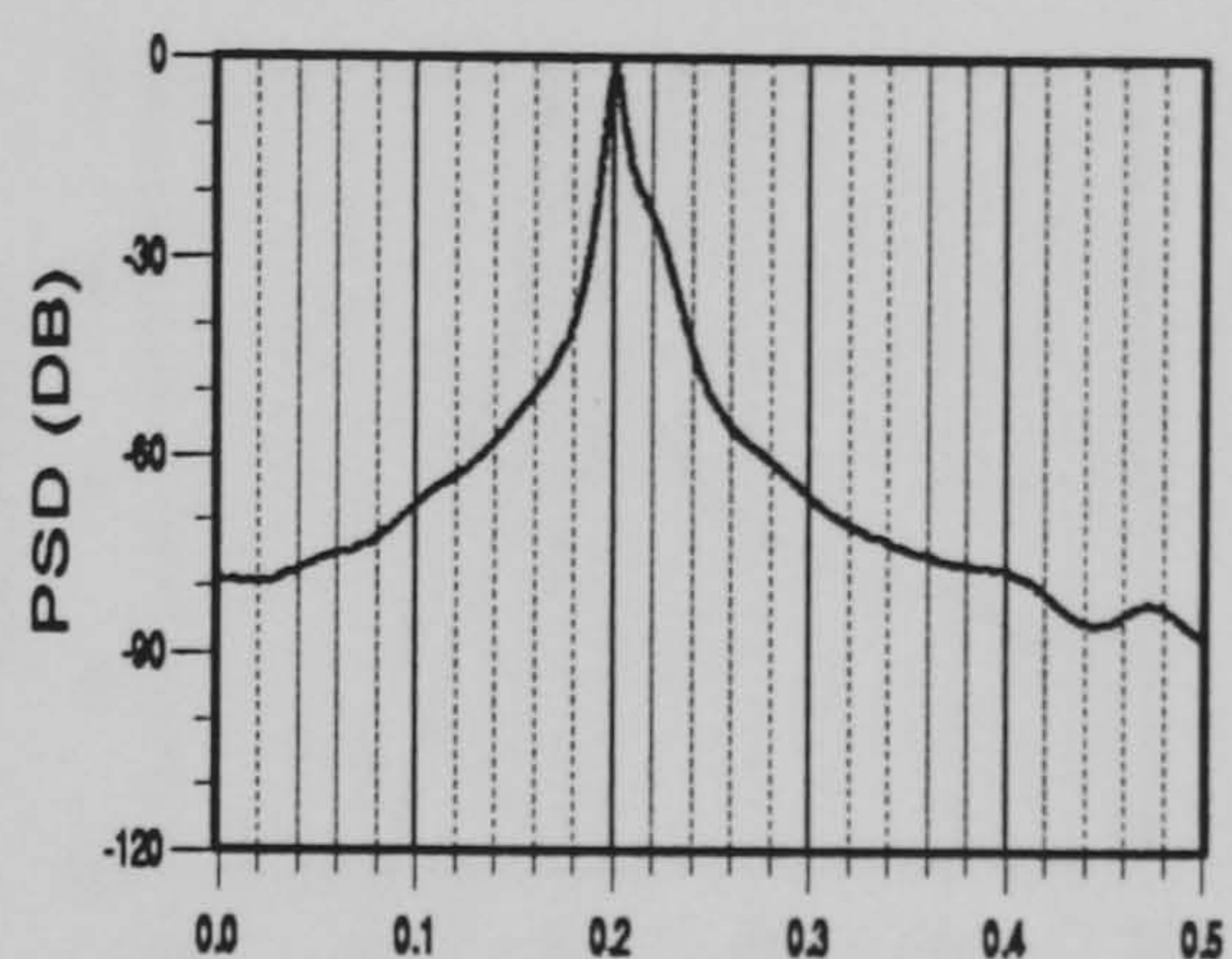
Normalized Frequency
(a) Periodogram



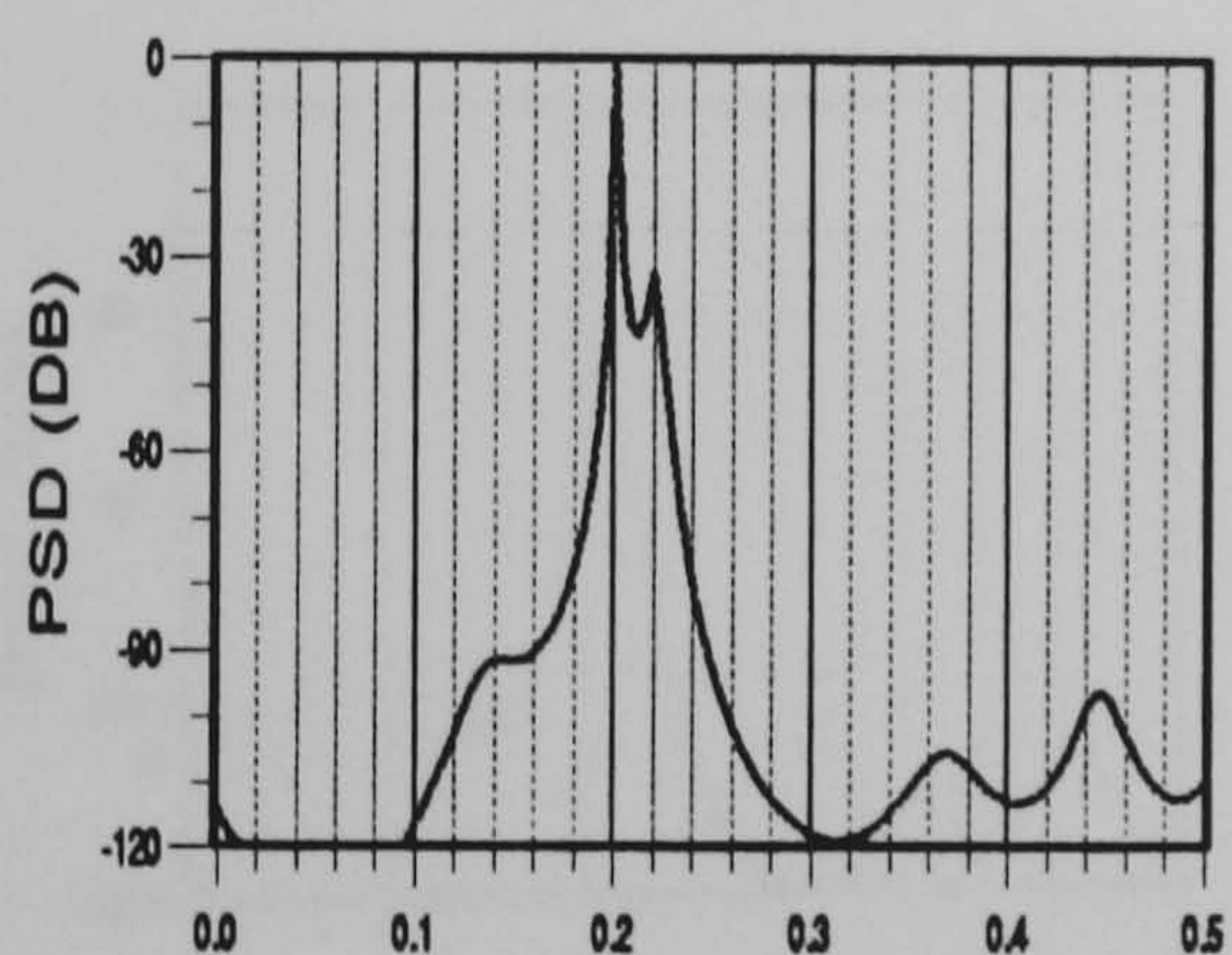
Normalized Frequency
(b) Blackman-Tukzy



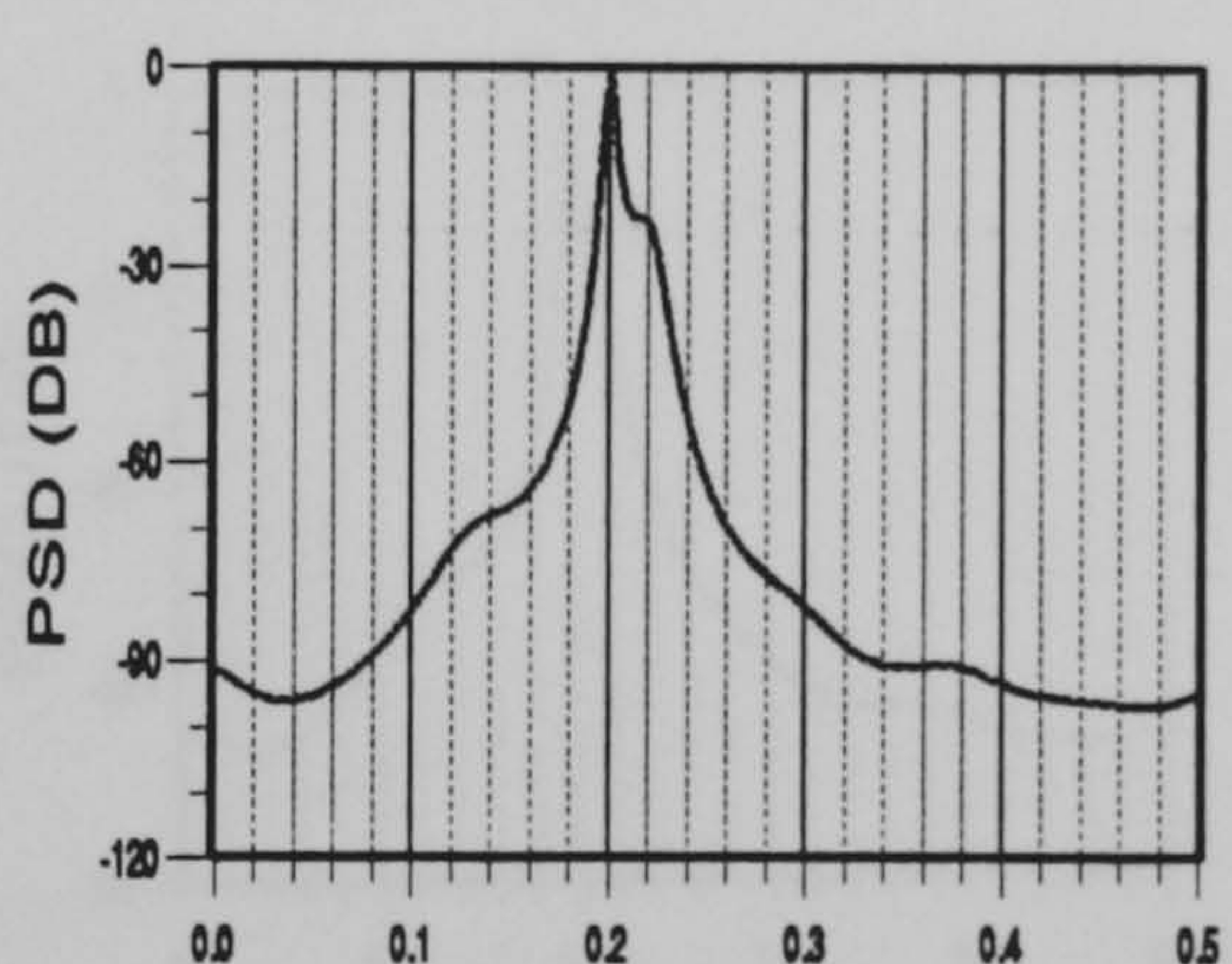
Normalized Frequency
(c) Minimum Variance



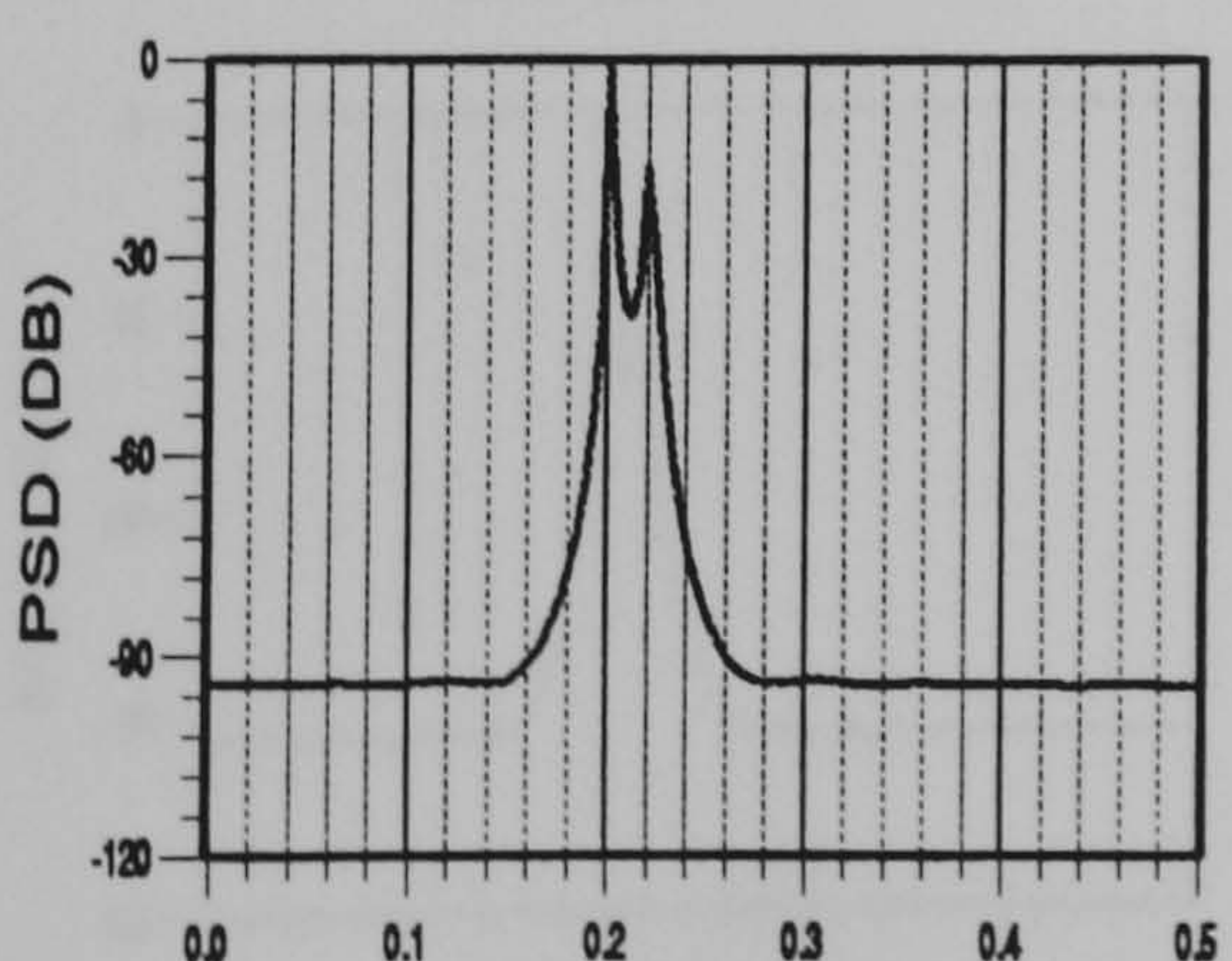
Normalized Frequency
(d) Autocorrelation



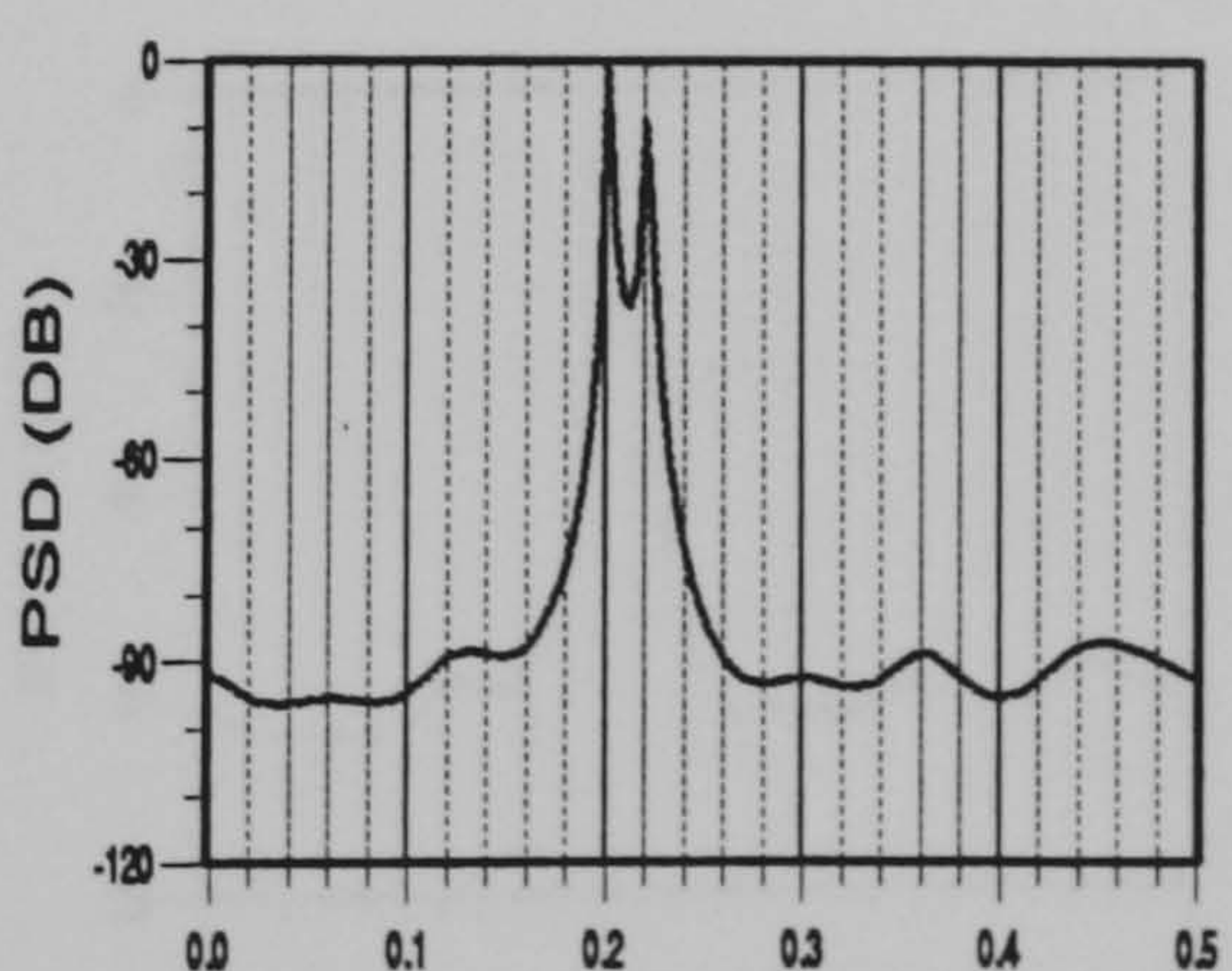
Normalized Frequency
(e) Modified Covariance



Normalized Frequency
(f) Burg



Normalized Frequency
(g) MUSIC



Normalized Frequency
(h) EigenVector

Length = 64

$F1 = 0.2$

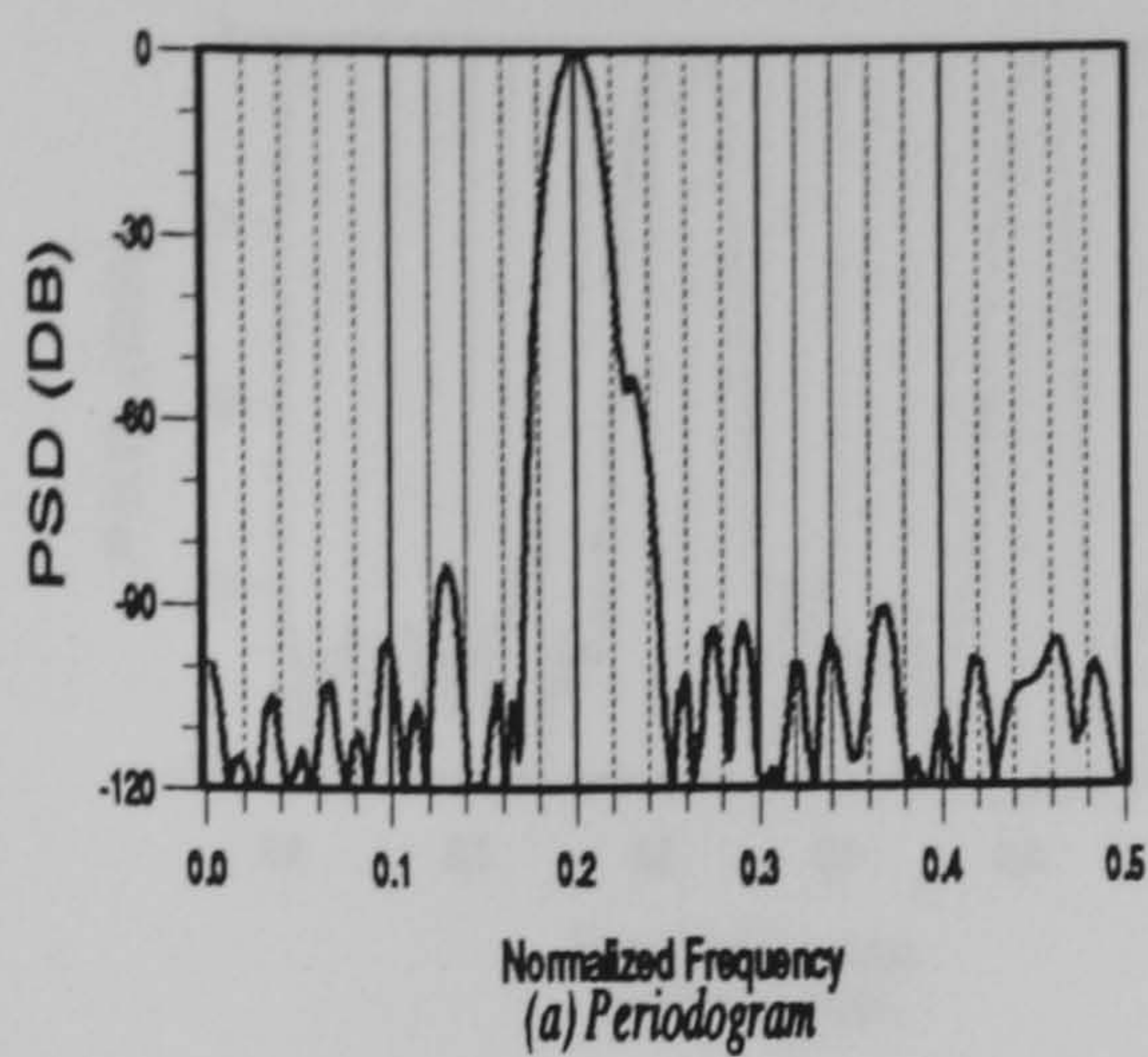
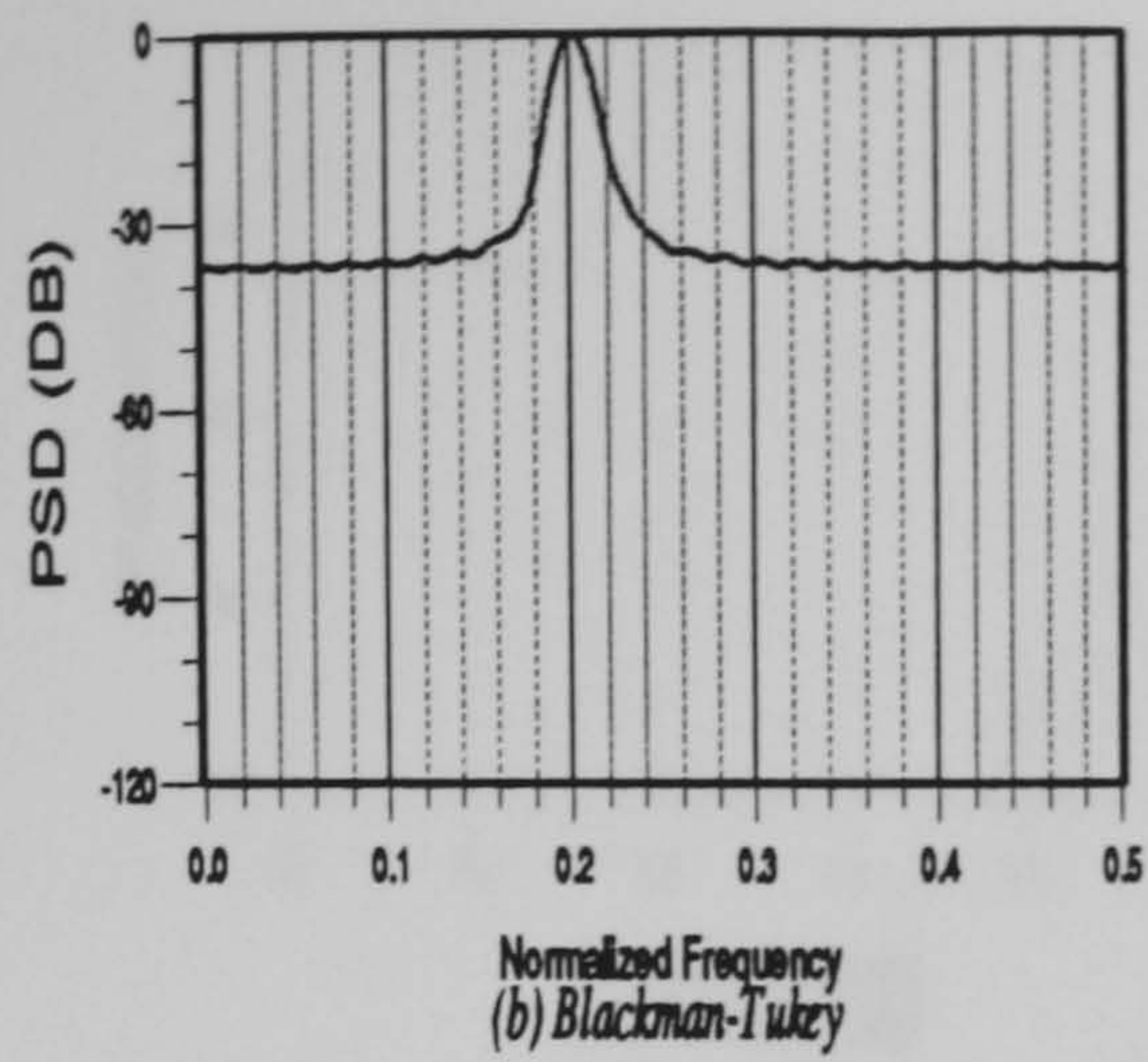
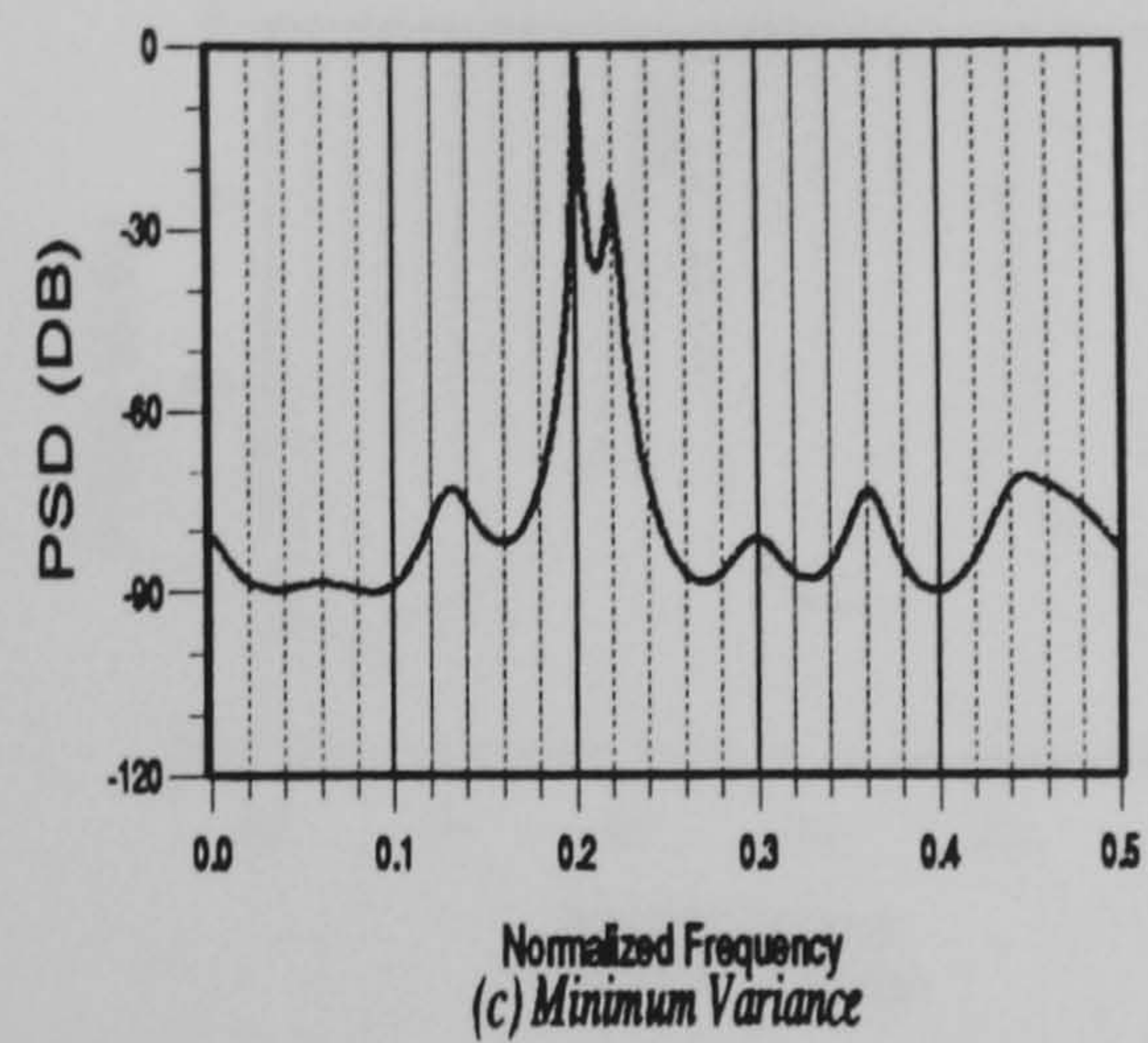
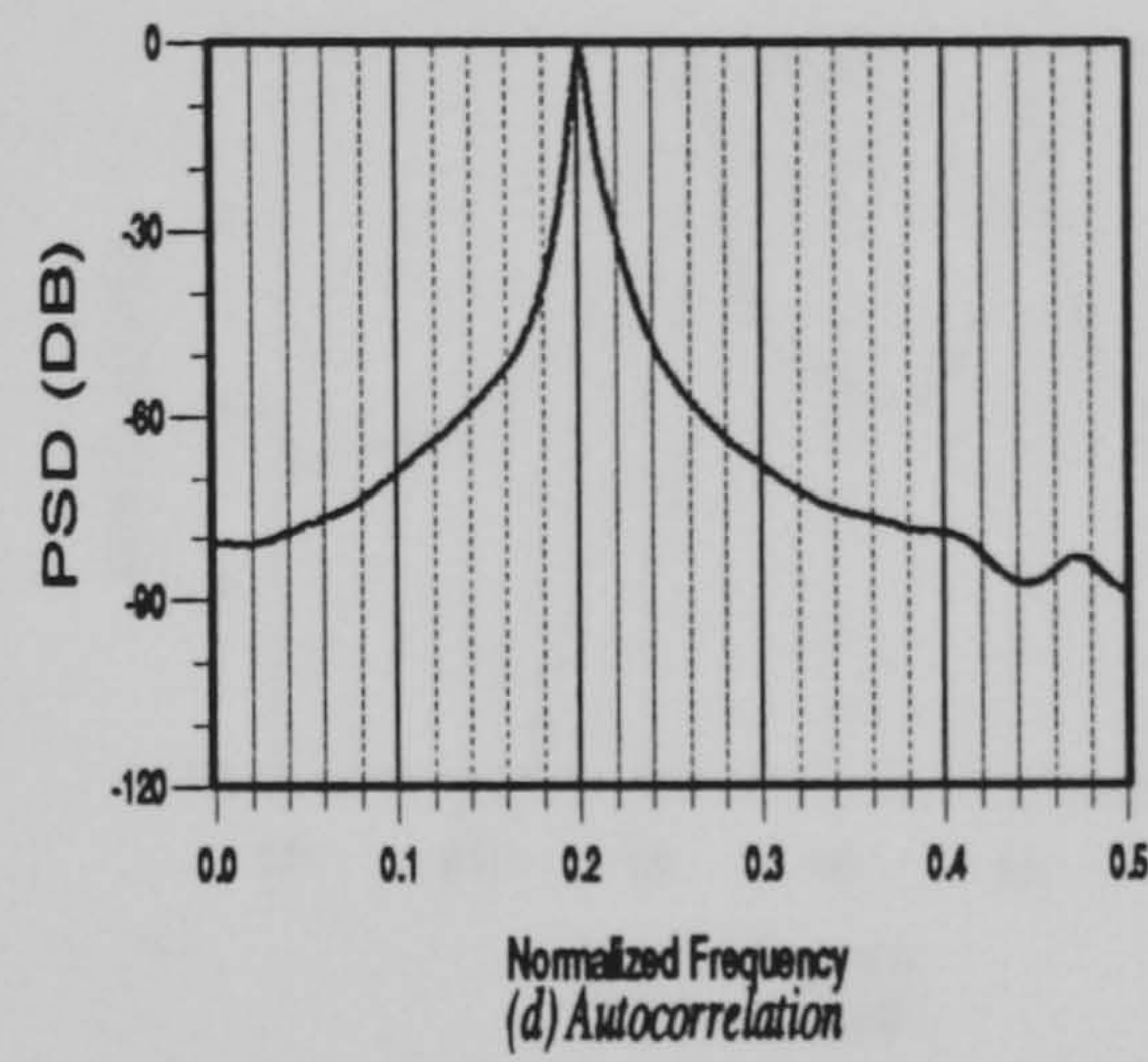
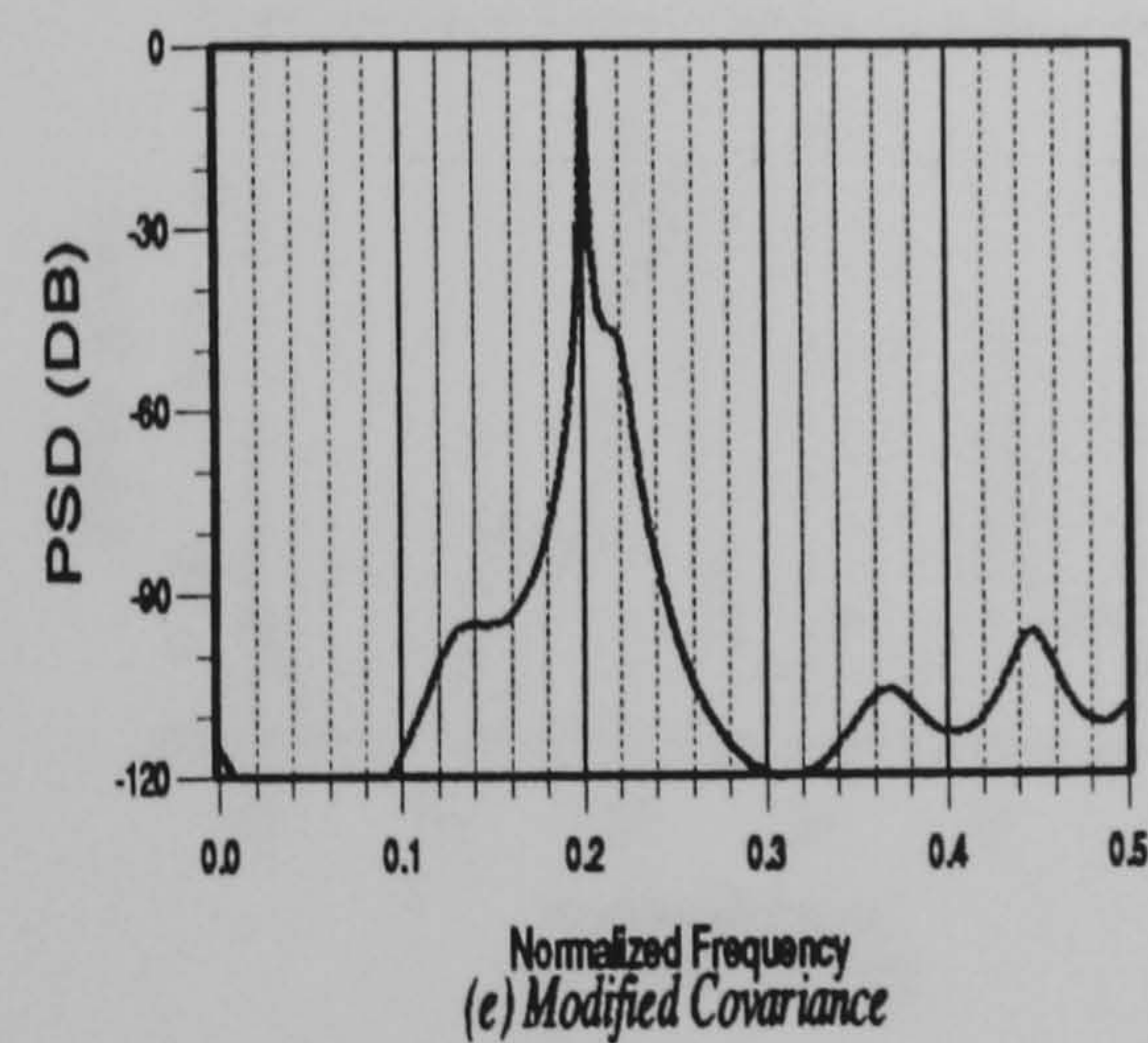
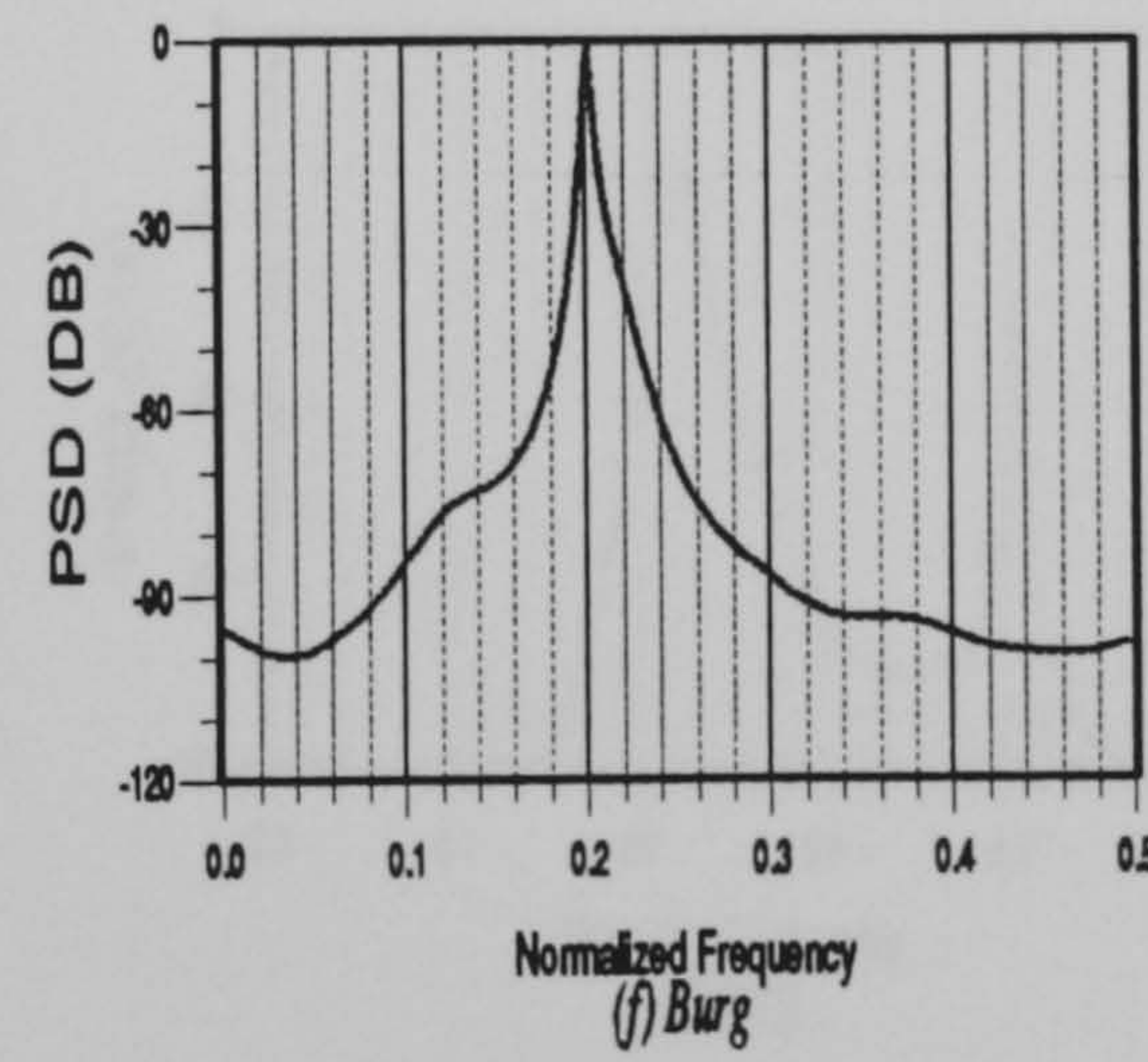
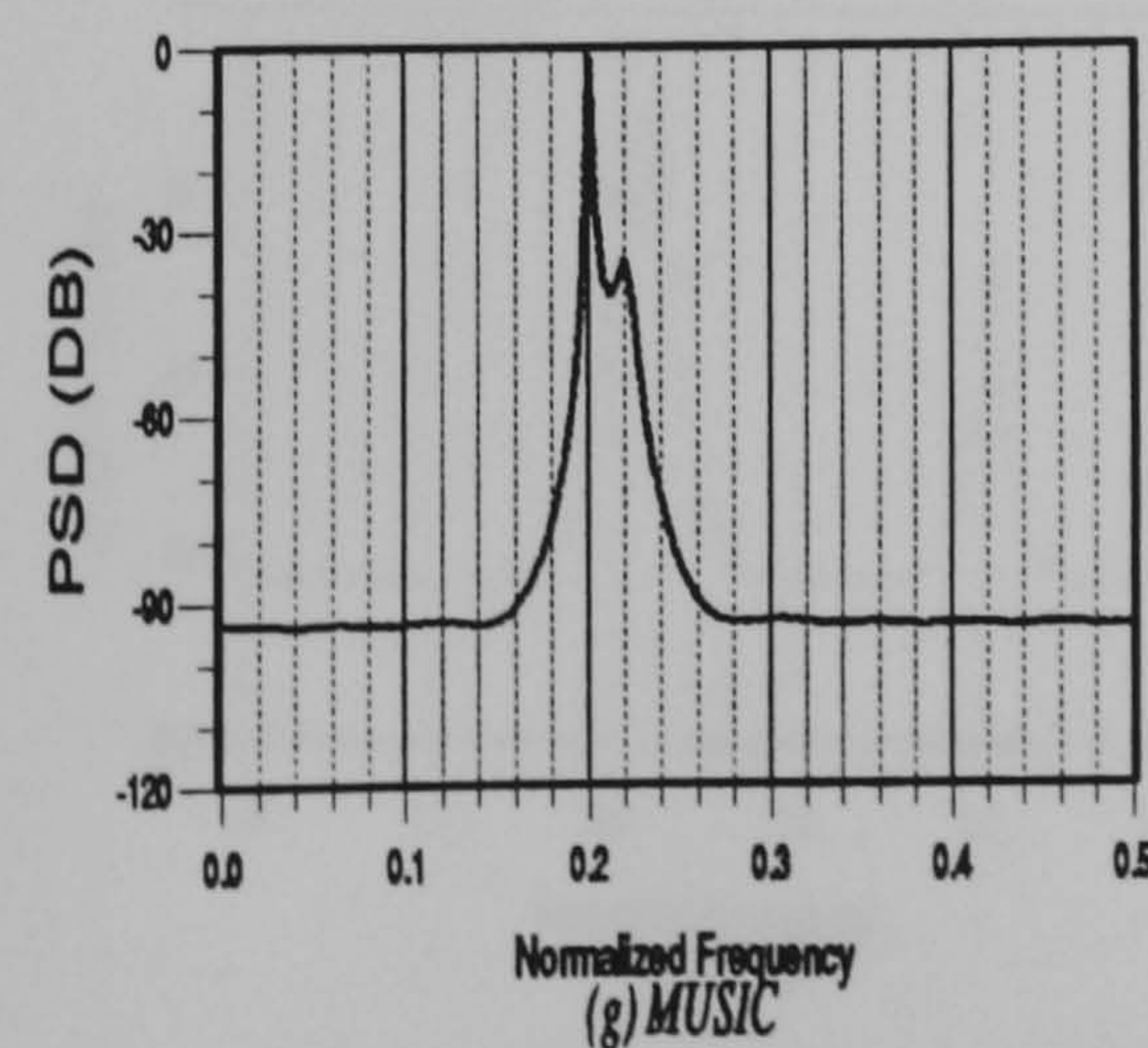
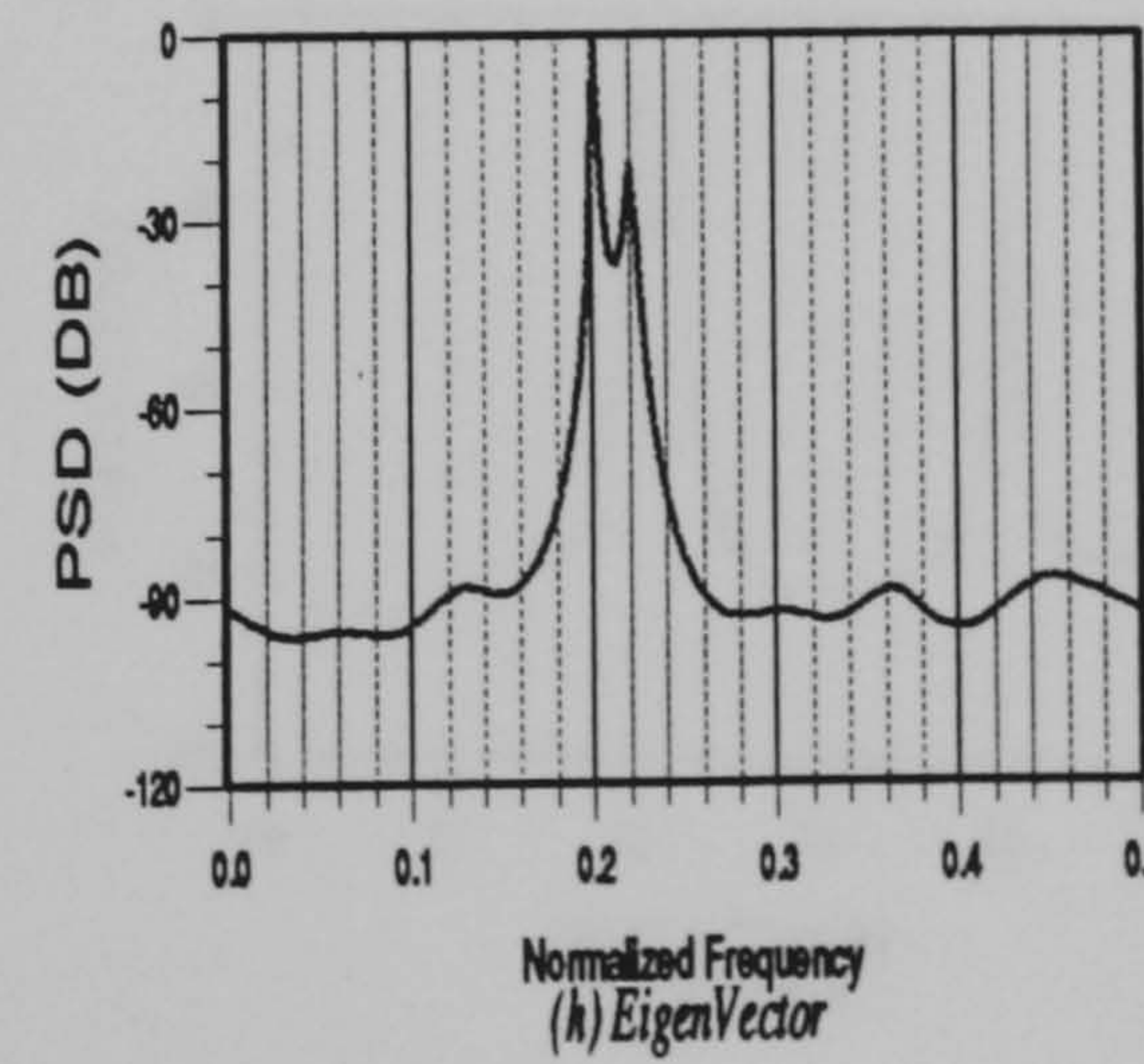
Amplitude 1 = 1.

$F2 = 0.22$

Amplitude 2 = 0.25

SNR = 10 DB

Figure 4.13: Relative amplitude of 0.25



Length = 64

$F1 = 0.2$

Amplitude 1 = 1.

$F2 = 0.22$

Amplitude 2 = 0.1

SNR = 10 DB

Figure 4.14: Relative amplitude of 0.1

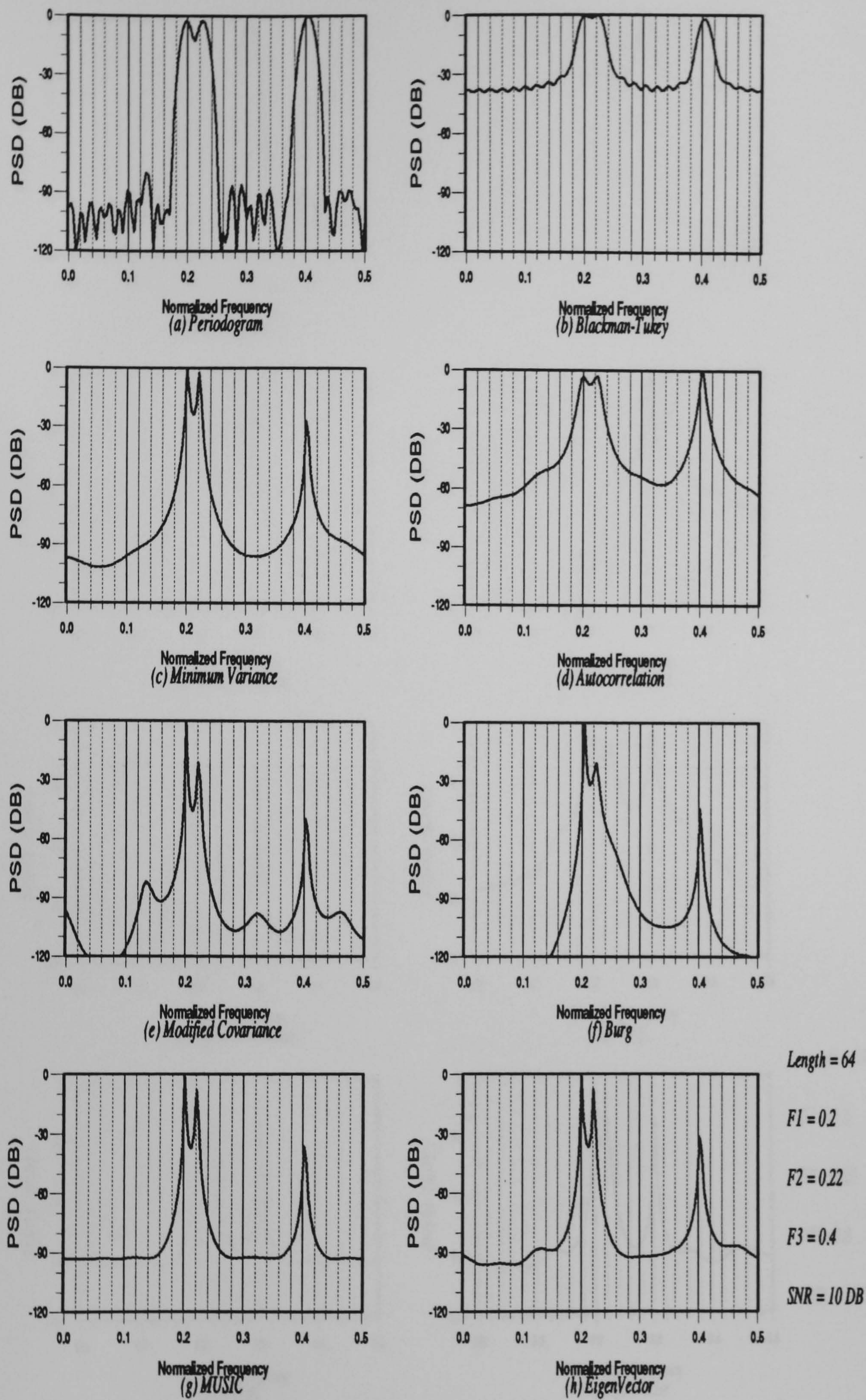


Figure 4.15: The third signal at normalized frequency of 0.4

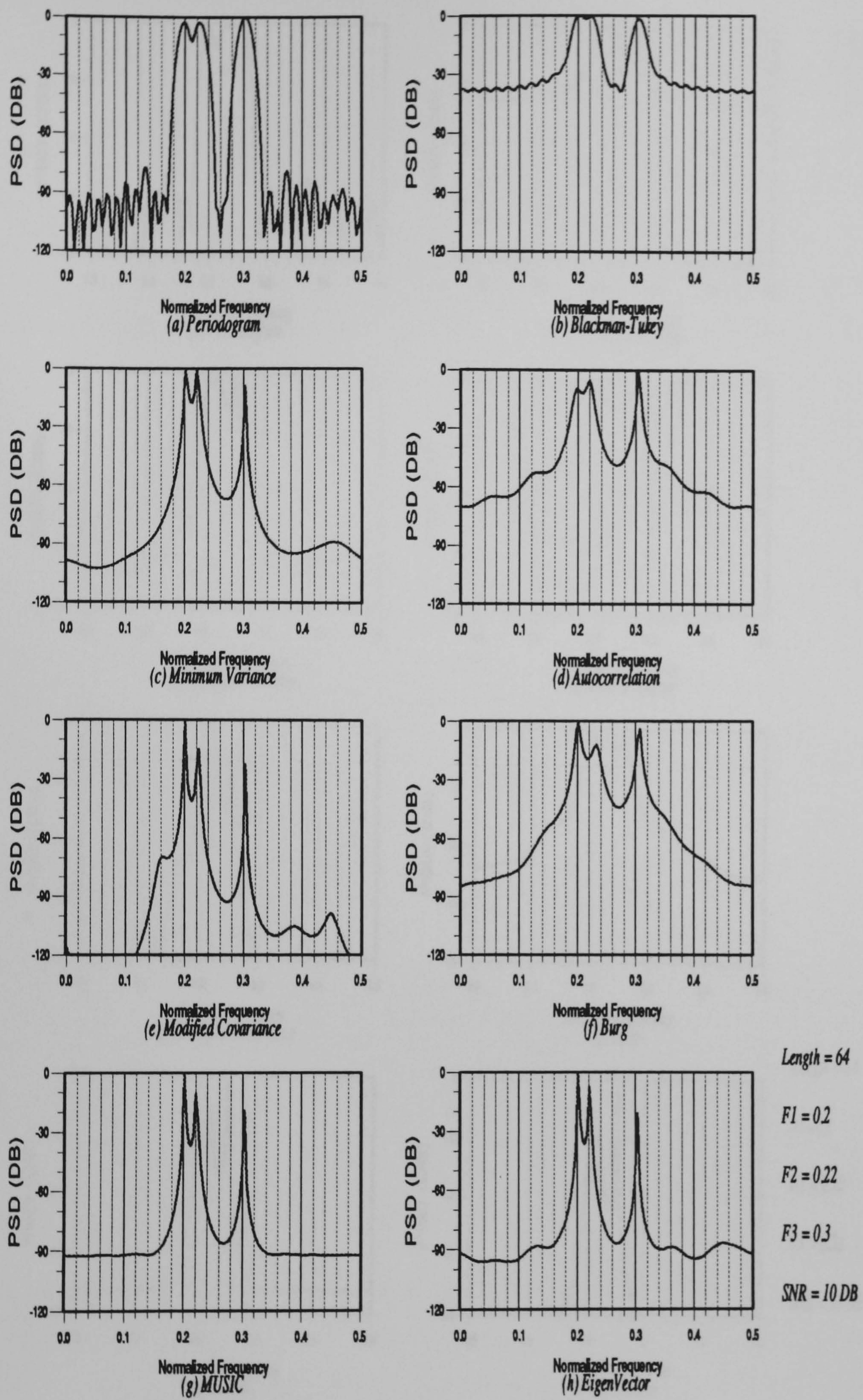


Figure 4.16: The third signal at normalized frequency of 0.3

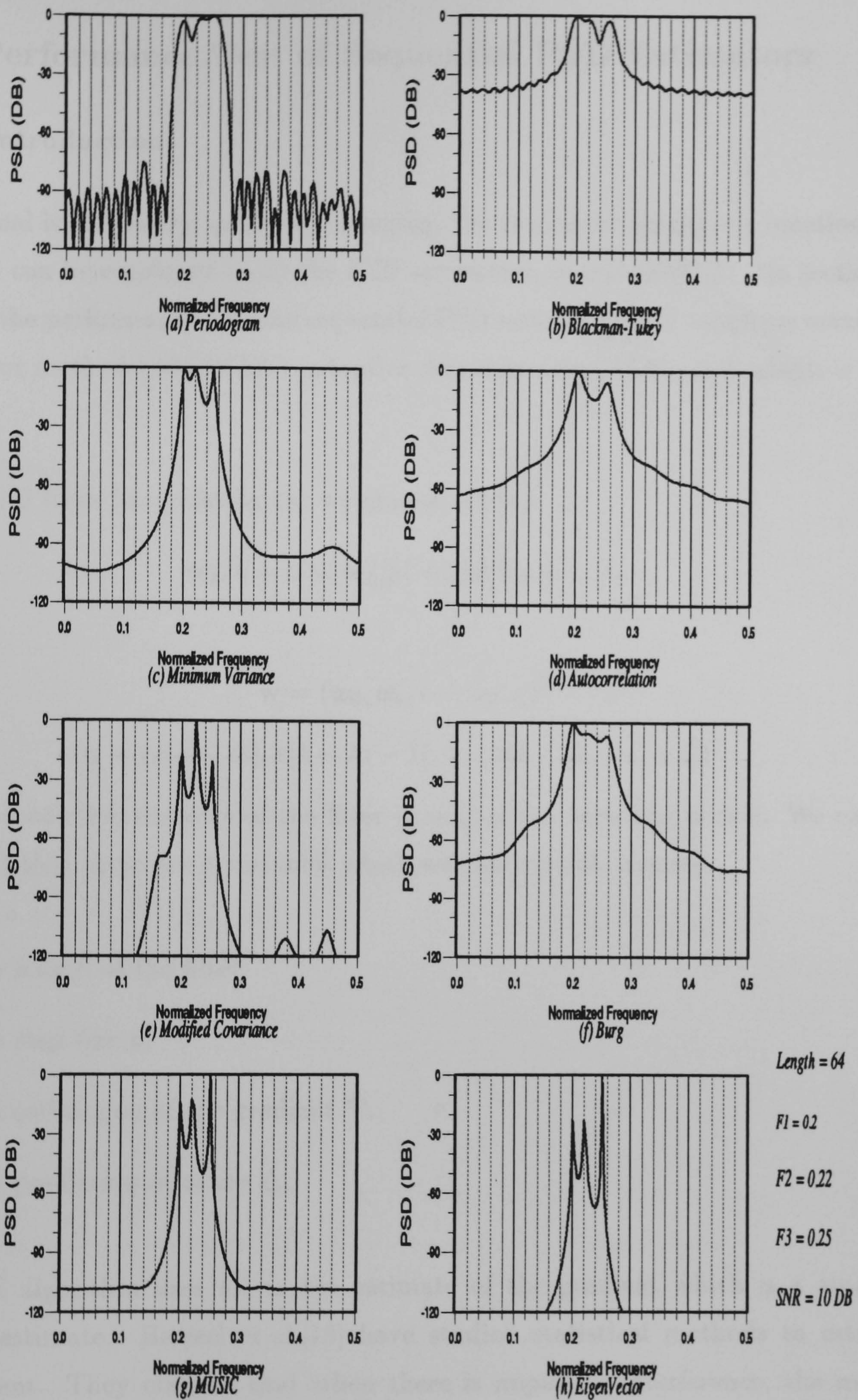


Figure 4.17: The third signal at normalized frequency of 0.25

4.3 Performance Test of Sequential PSD Estimators

4.3.1 Introduction

If the signal is stationary, or slowly changing, the sequential estimators mentioned in chapter 3 can be employed to do the PSD estimation in real-time. In this section we compare the performance of these sequential PSD estimators, i. e. adaptive maximum eigenvector method (AMAXEV), adaptive line enhancer (ALE), and adaptive PHD (APHD).

The general form for updating the weights of ALE is

$$\mathbf{w}_L[k+1] = \mathbf{w}_L[k] + 2\mu e[k]\mathbf{x}[k-\Delta] \quad (4.2)$$

Where

$$\mathbf{w} = (w_0, w_1, \dots, w_{L-1})^T$$

$$\mathbf{x} = (x[k-\Delta], x[k-\Delta-1], \dots, x[k-\Delta-L+1])^T$$

Δ is the prediction distance of the filter in unit of the sampling period. We can see from (4.2) that there are 4 variables which we can control. namely

1. The length of the filter.
2. The step size μ .
3. The estimation of the gradient $\hat{\nabla}$.
4. The prediction distance Δ .

The LMS algorithm uses $\mathbf{ex}$ as the estimate of the gradient which is a stochastic gradient estimate. Haweel et.al.[19] have studied statistical methods to estimate the gradient. They claimed that when there is impulsive interference, the median estimation of the gradient has good performance. However, if the signal is stationary or the characteristics of the signal are changing slowly, the simple estimation $\mathbf{ex}$ is appropriate and is widely used. The prediction distance Δ has been shown [54] to have influence on the noise suppression capability of the ALE. In this chapter we

use $\mathbf{ex}$ as the estimation of gradient, and $\Delta = 1$ as used in the conventional AR time-series analysis techniques. The filter length L and step size μ are assumed fixed to see the response of the adaptive PSD estimators to different input situations.

4.3.2 The Detectability Test

The test scenario is a single sinusoid embedded in white noise. The normalized frequency of the sinusoid is 0.2, the order and the step size μ of the adaptive filter are 20 and 0.001 respectively. The adaptive process is stopped at $N = 512$ and the resulting PSD is then investigated. When the SNR is 5 dB, it can be seen from figure 4.18 ~ figure 4.20 that all four methods can detect the presence of the sinusoid, except that the bandwidth of the AMAXEV is quite wide. However, its peak value is at $f = 0.2$. When the SNR deteriorates to -5dB, it can be seen that the APHD is less influenced by the noise while ALEW has spurious peaks and more bias.

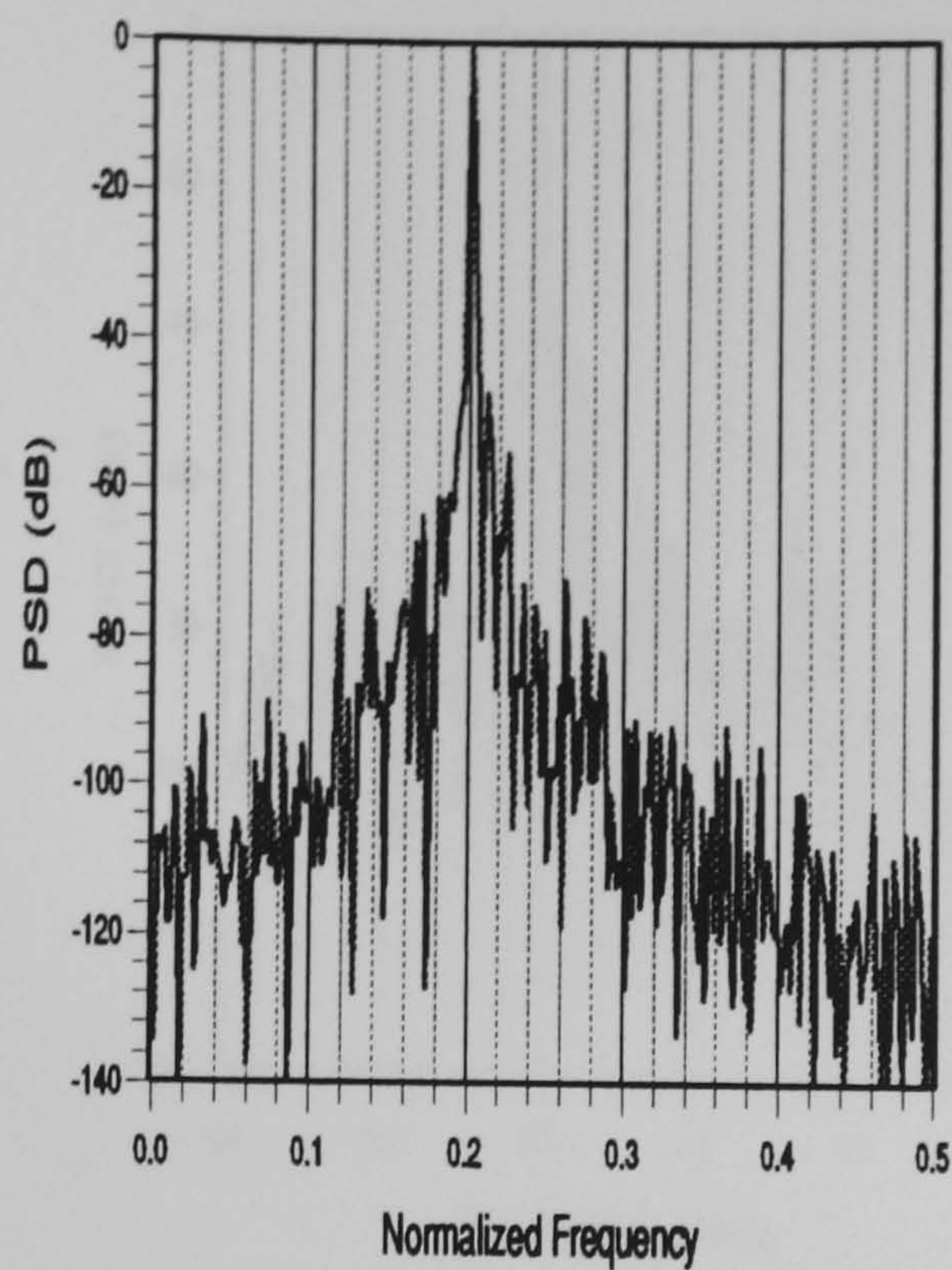
4.3.3 The Frequency Resolution Test

4.3.3.1 The Effect of SNR The test scenario is $f_1 = 0.2$, $f_2 = 0.22$, $SNR = 10dB$. The other parameters are the same as before. It can be seen that the AMAXEV is unable to distinguish these two sinusoids. However, it indicates the location of them. When the SNR is high ALEO, ALEW, and APHD all can resolve those two signals. However, when the SNR is decreased and with μ fixed, the variance in the ALE weight vector noise increases, and the resulting spectral estimation will be degraded. This can be seen from figure 4.21 ~ figure 4.23.

4.3.3.2 The Effect of Relative Amplitude The test scenario is the same as before, except that the amplitude of f_2 is changed. The amplitude of f_1 is fixed at 1. While the amplitude of f_2 is changed to 0.5, 0.25, 0.1 respectively. It can be seen from figure 4.24 ~ figure 4.26 that the ALEO can reflect the power changing while ALEW and APHD should be considered as an frequency indicator rather than PSD estimator. Because their amplitude do not indicate the true power.

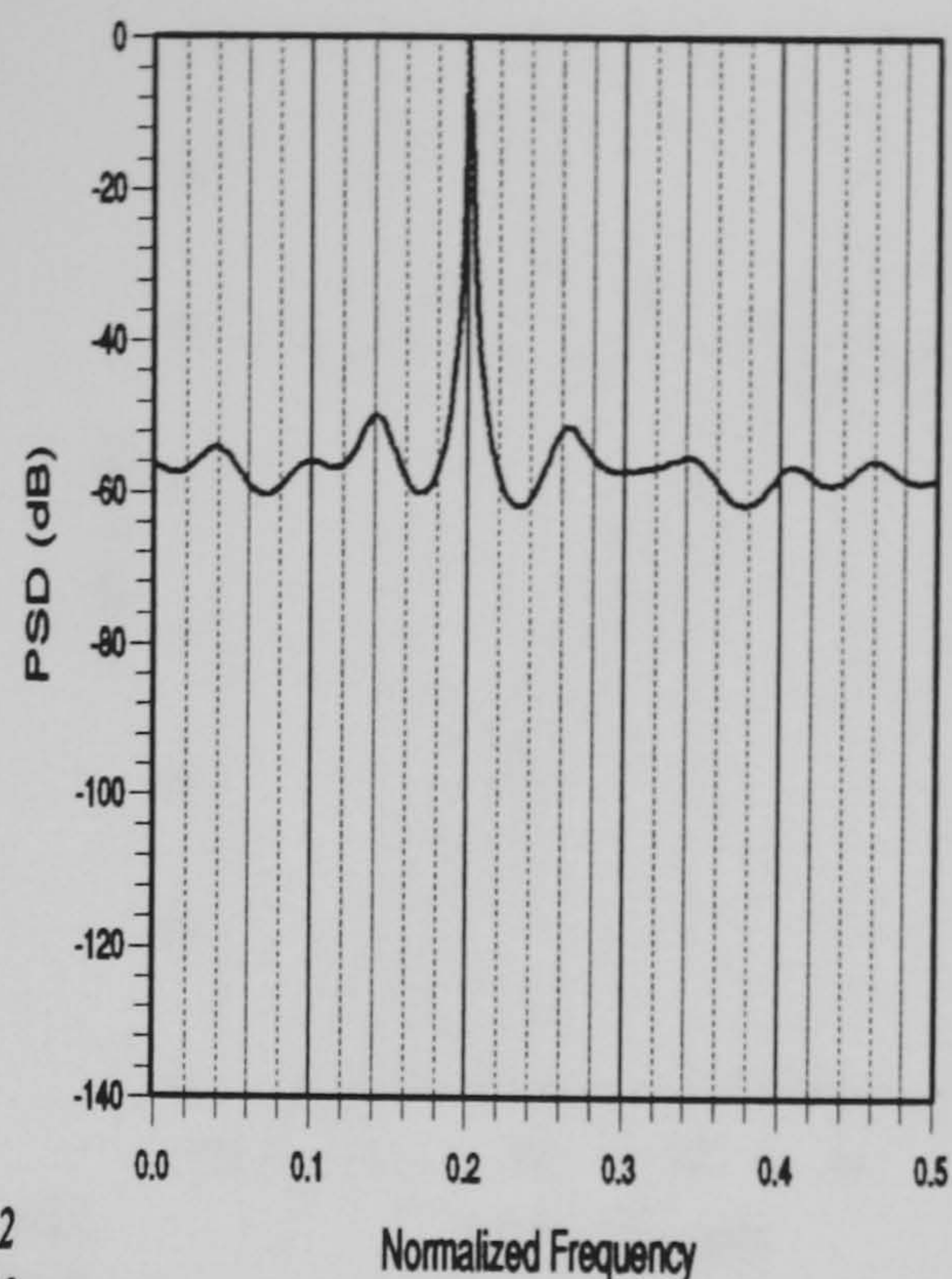
4.3.3.3 The Effect of Relative Phase In this test the phase of f_1 is fixed while the phase of f_2 is changed from 0 degree to 45, 90, 135, and 180 degree respectively.

It can be seen from figure 4.27 ~ figure 4.31 that the phase difference does not affect the performance of the adaptive PSD estimators.

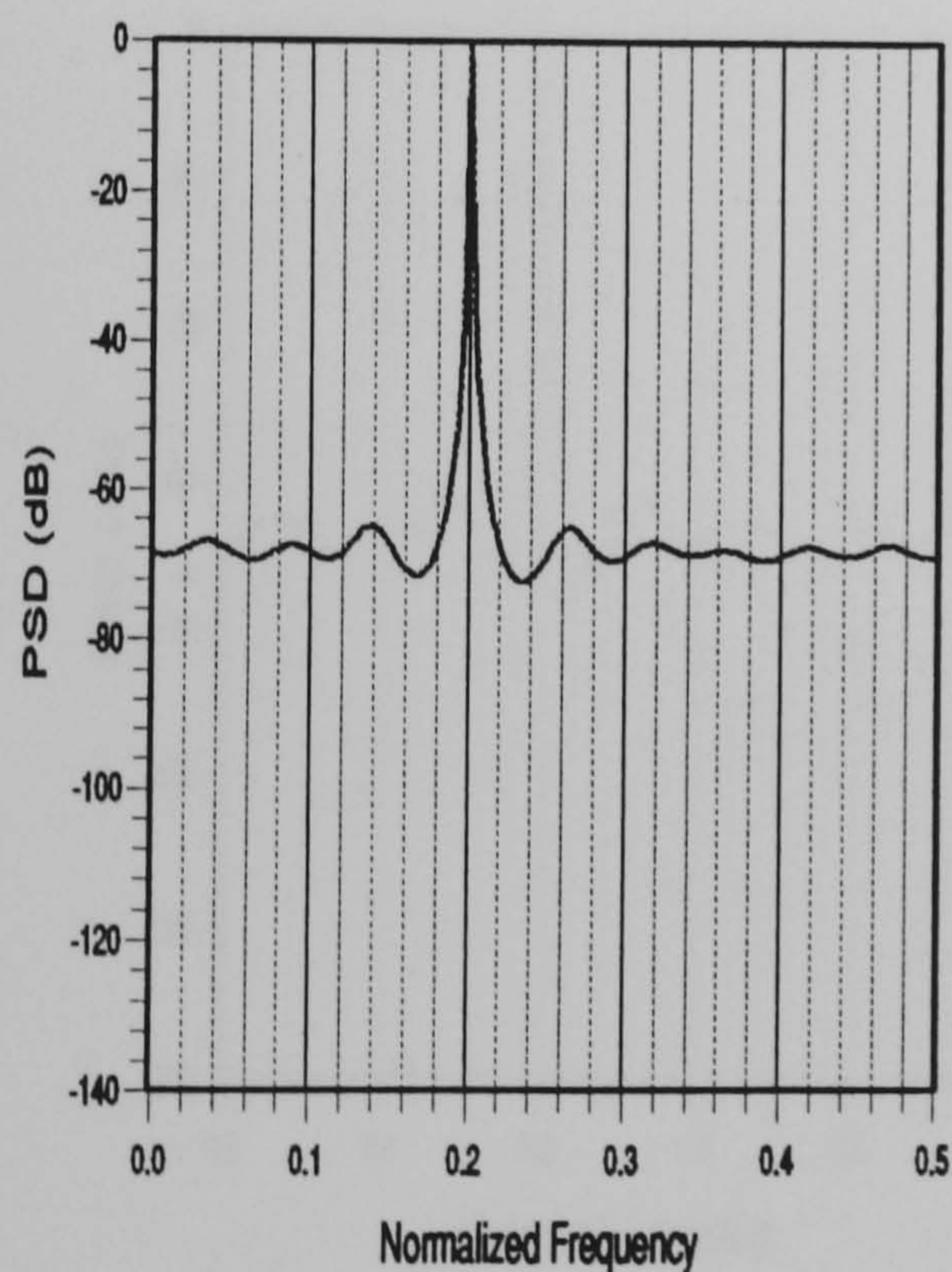


(a) *PSD FROM ALEO*

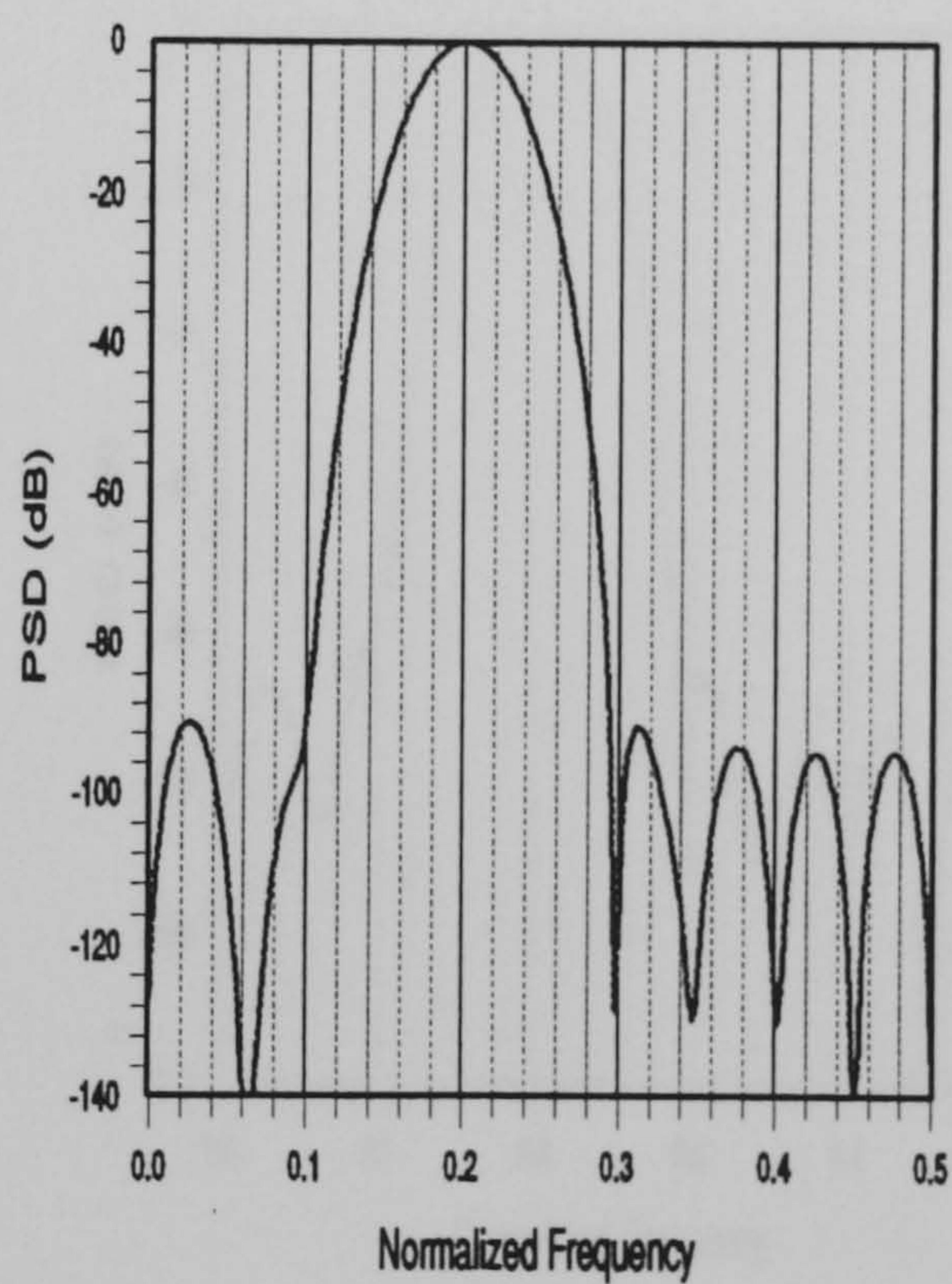
$F = 0.2$
 $IP = 20$
 $\mu = 0.001$
 $SNR = 5 \text{ DB}$



(b) *PSD FROM ALEW*

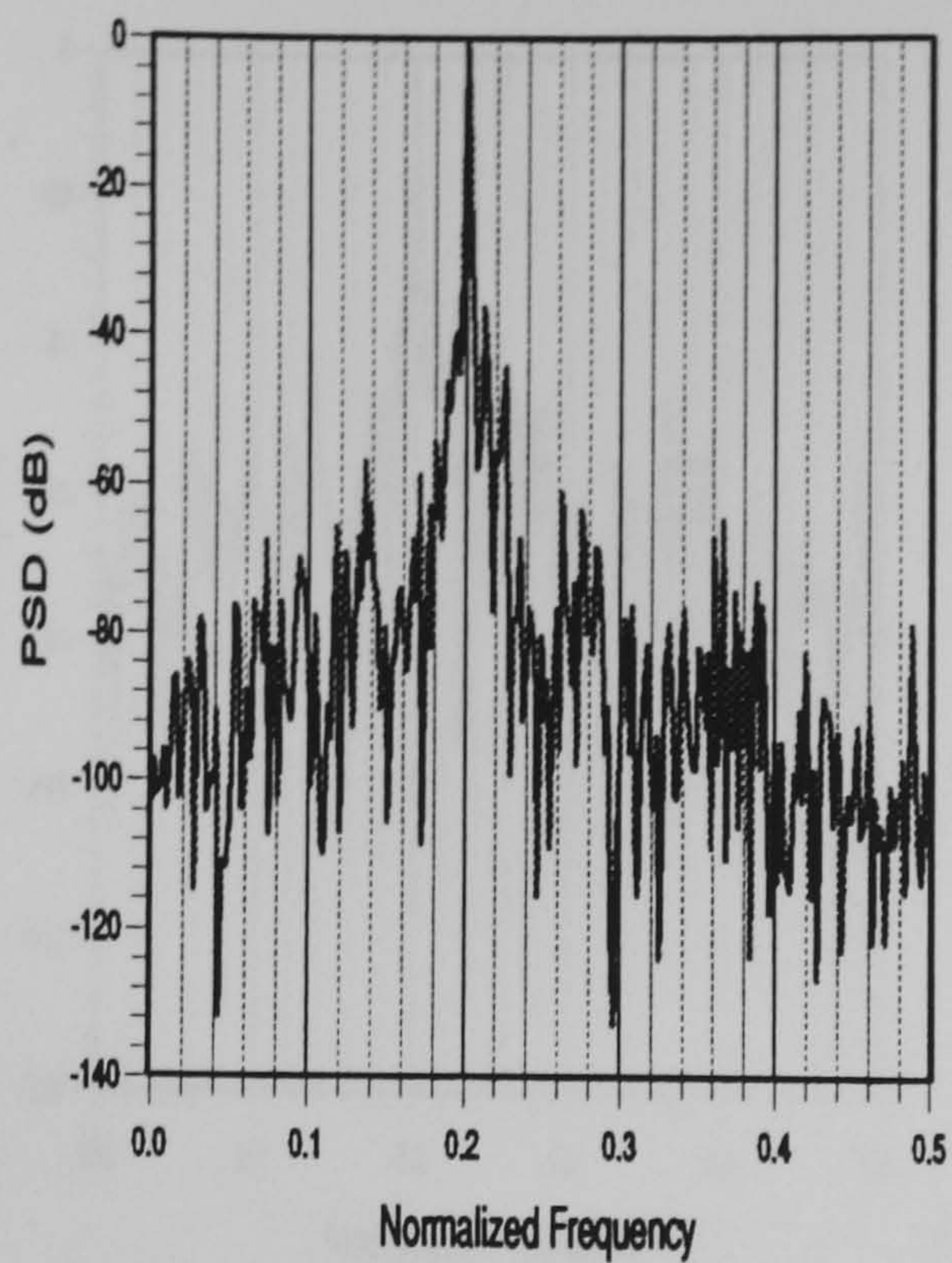


(c) *PSD FROM APHD*

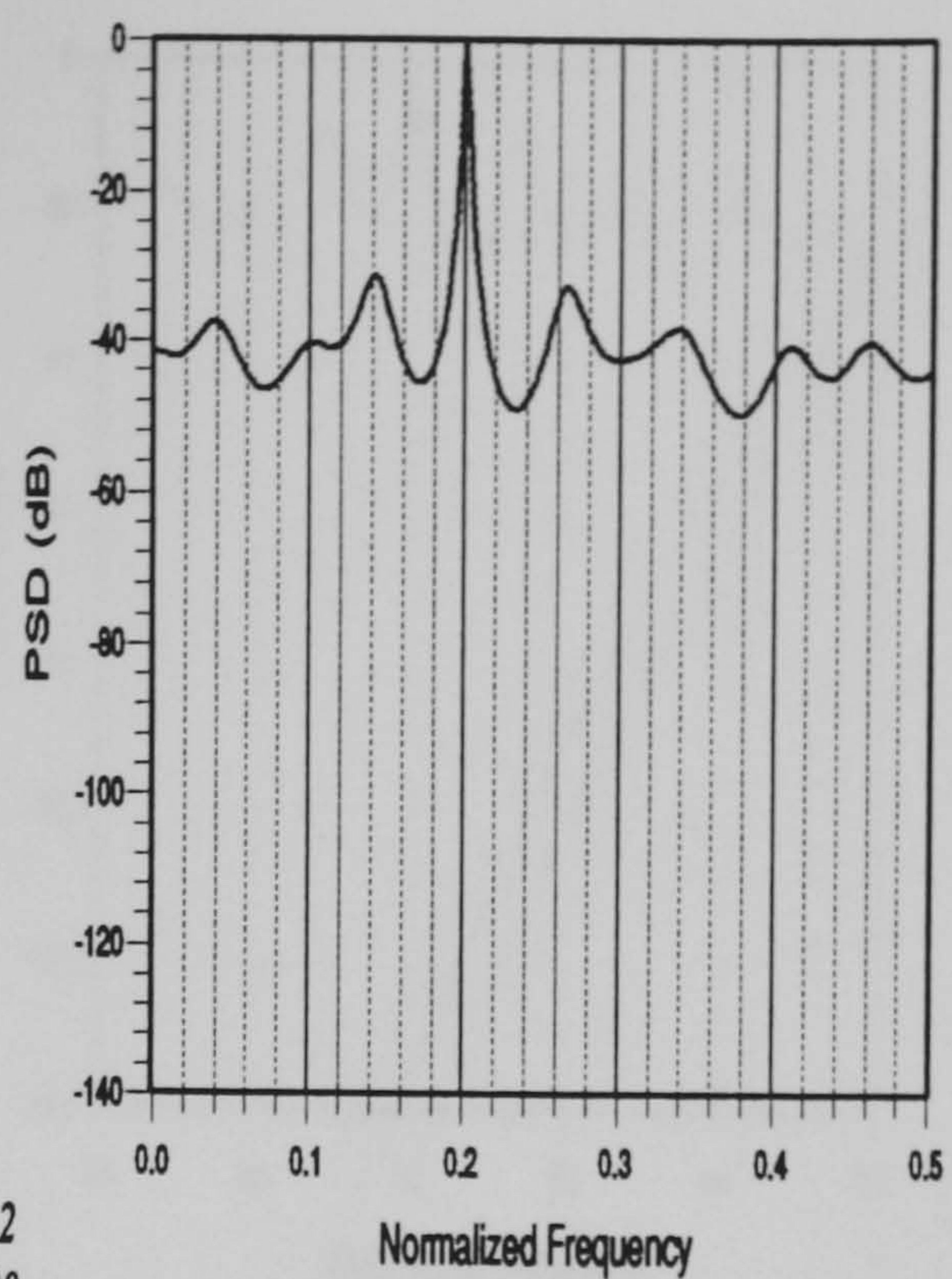


(d) *PSD FROM AMAXEV*

Figure 4.18: The detectability test: $SNR = 5\text{dB}$

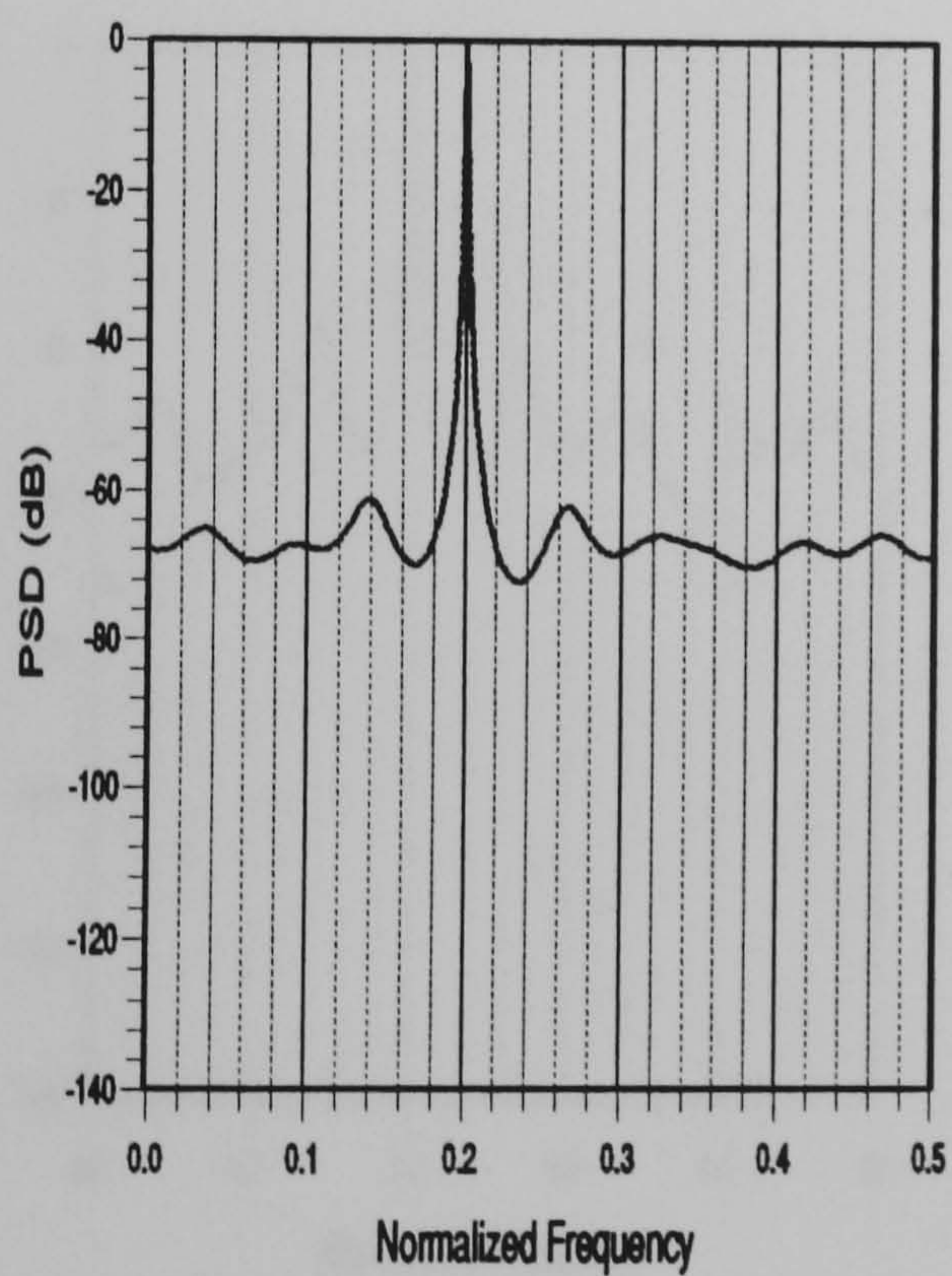


(a) PSD FROM ALEO

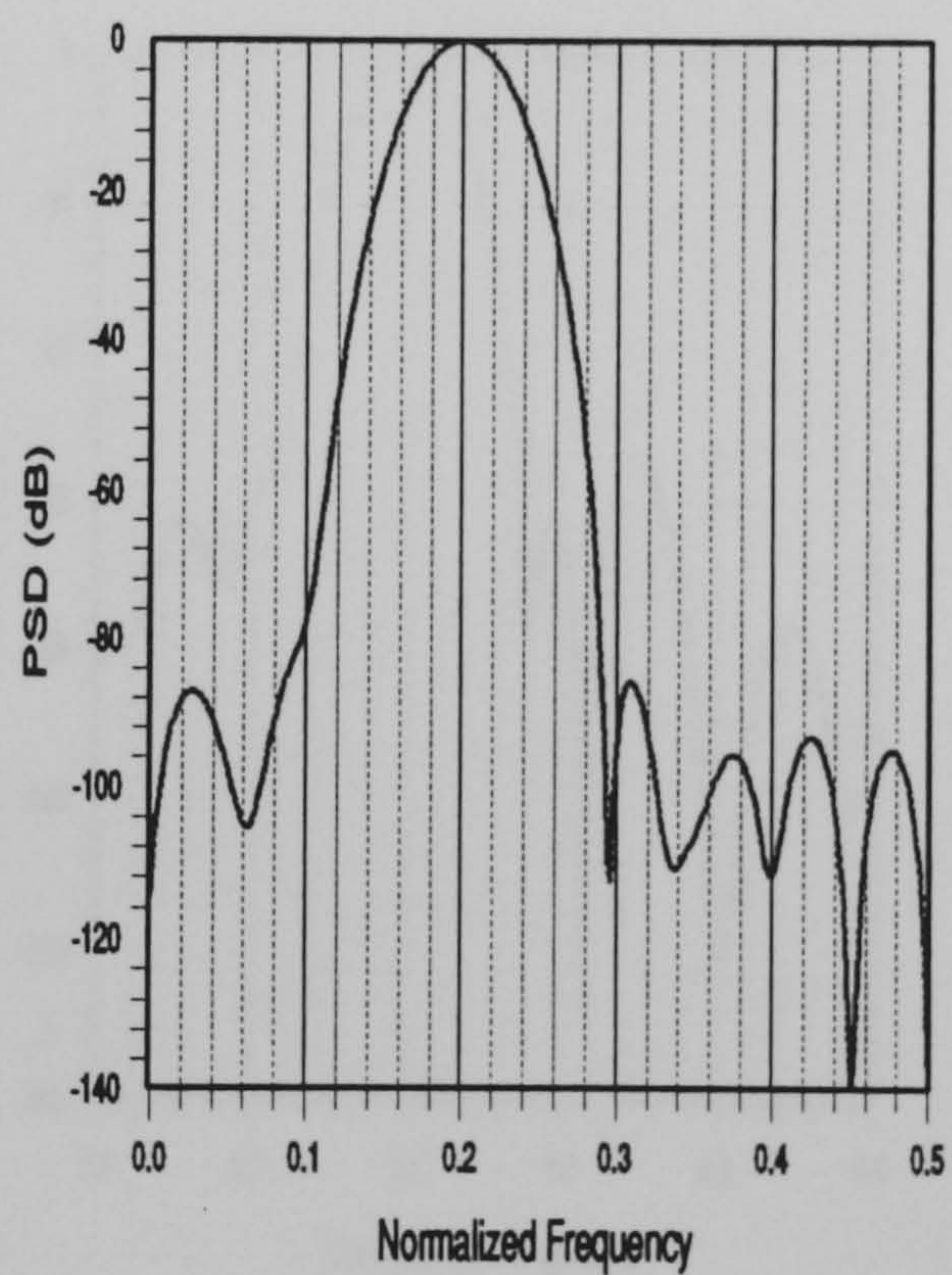


(b) PSD FROM ALEW

$F = 0.2$
 $IP = 20$
 $\mu = 0.001$
 $SNR = 0 \text{ dB}$

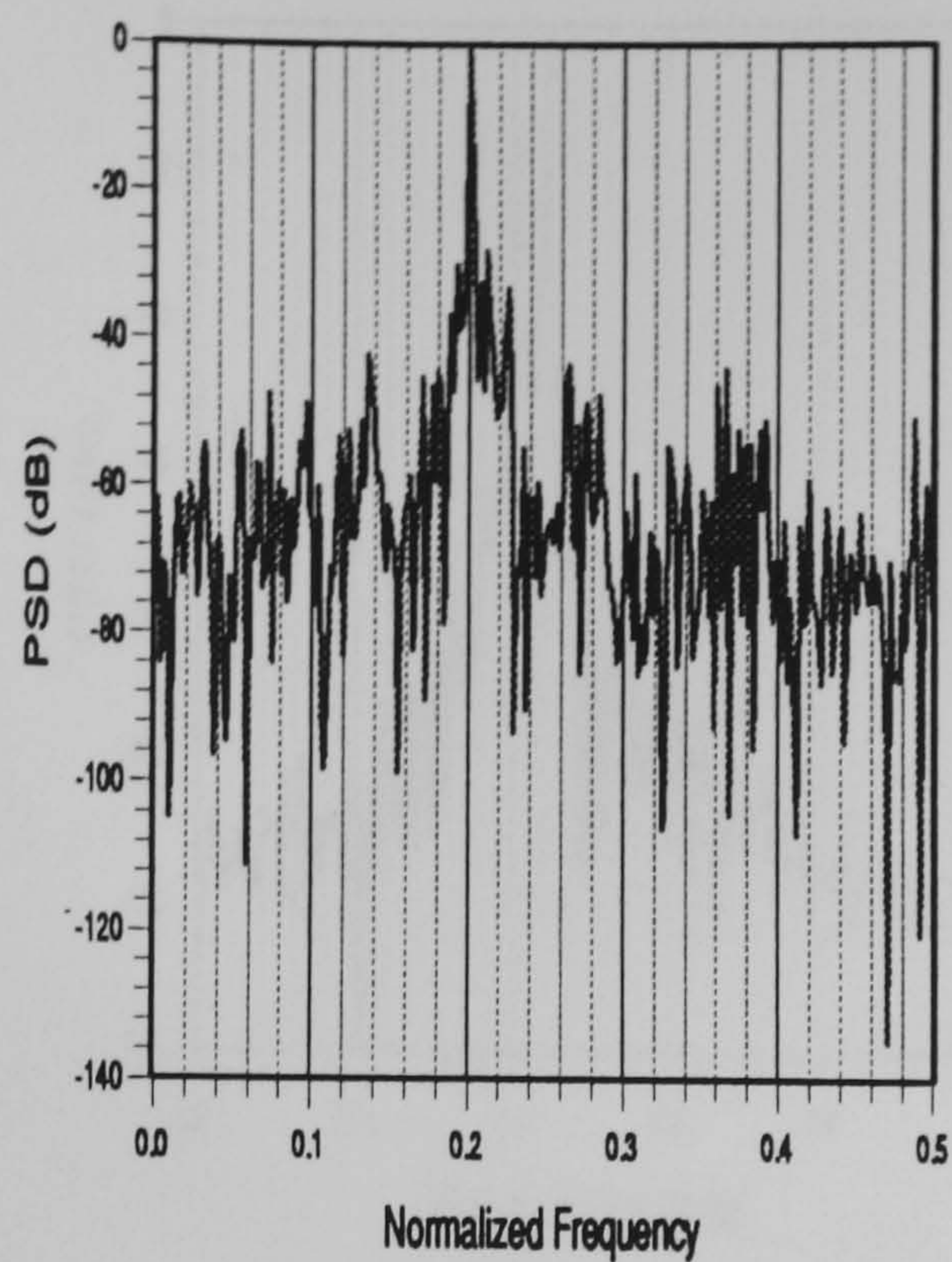


(c) PSD FROM APHD



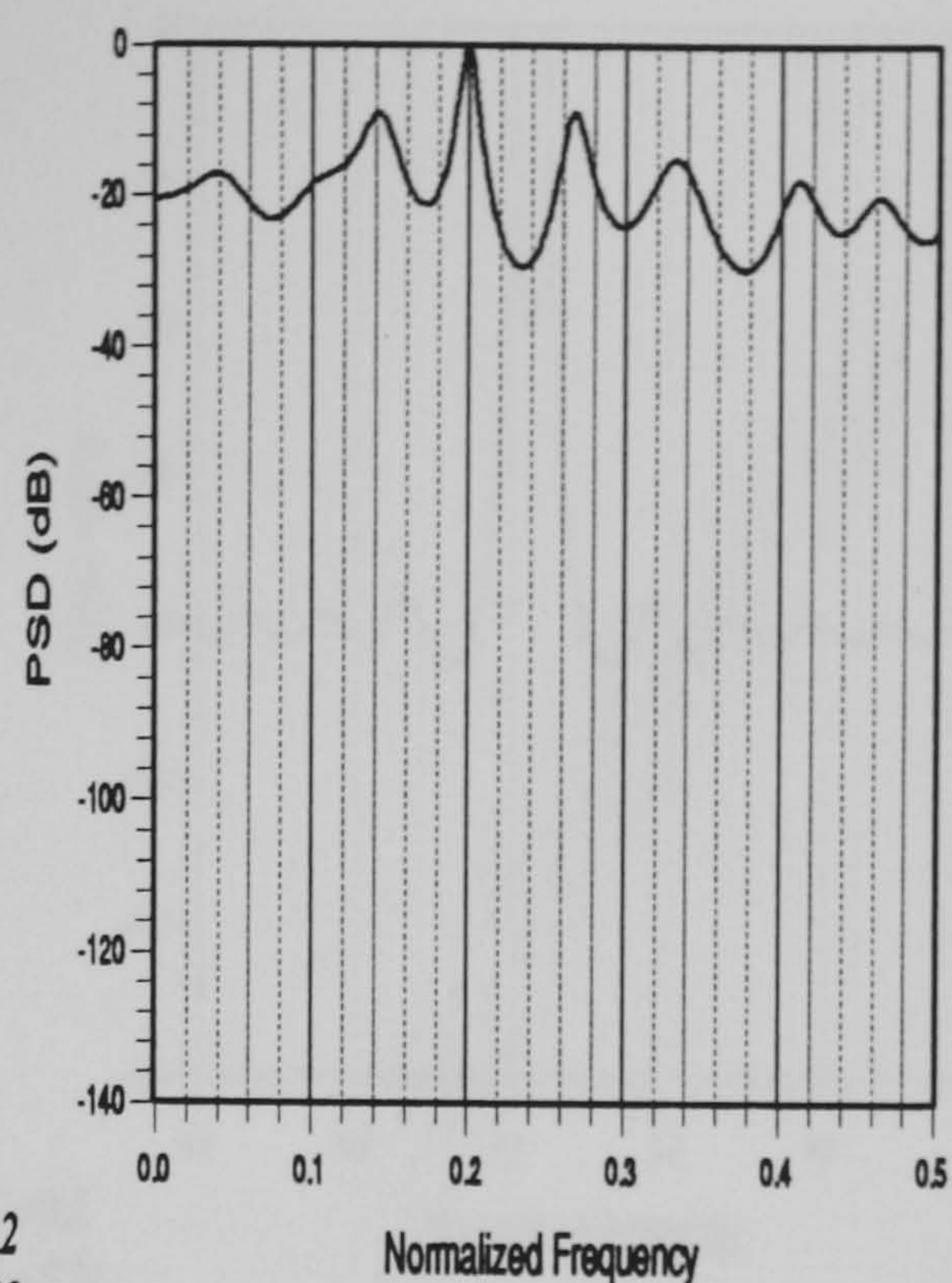
(d) PSD FROM AMAXEV

Figure 4.19: The detectability test: $SNR = 0 \text{ dB}$

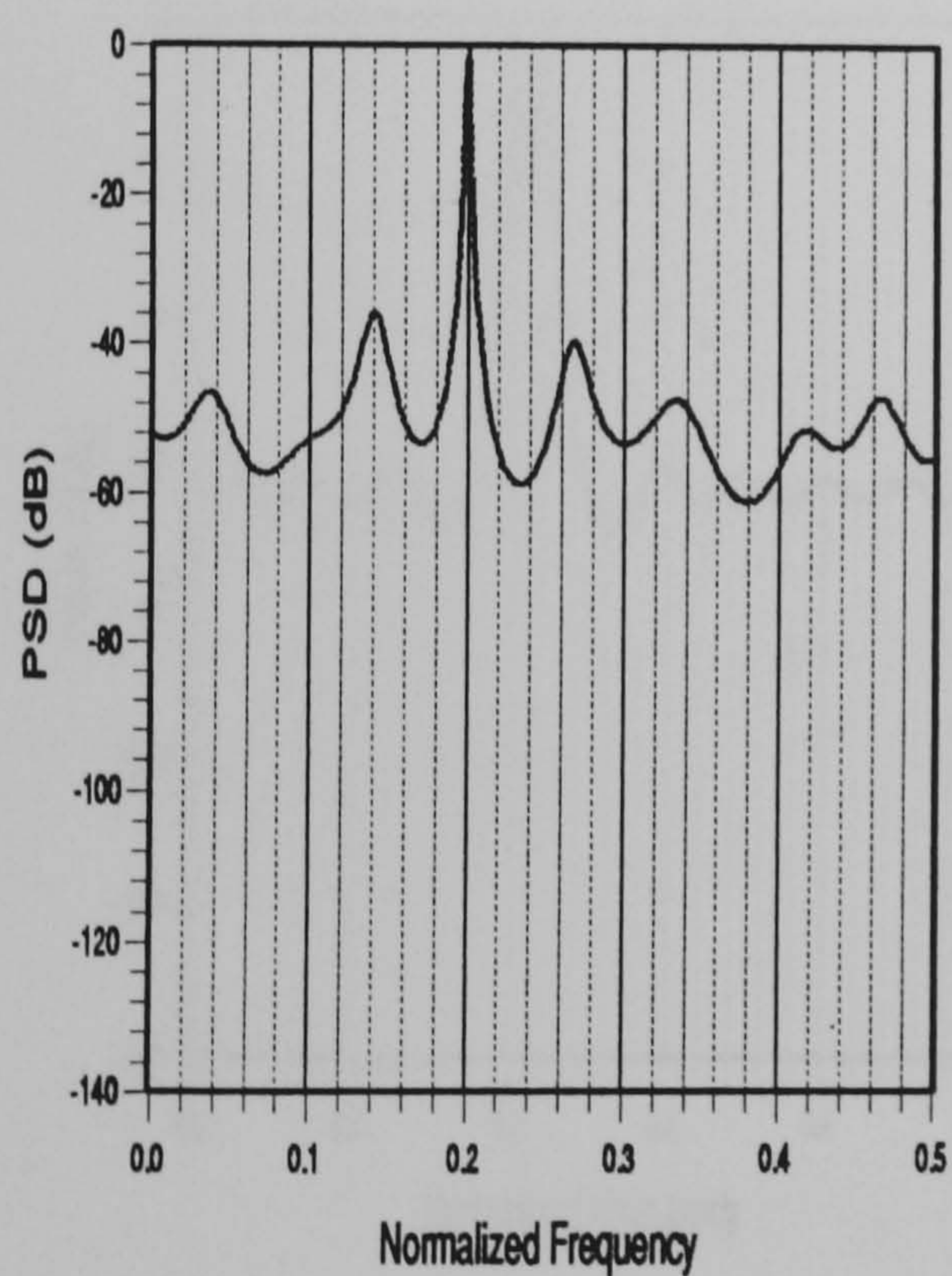


(a) PSD FROM ALEO

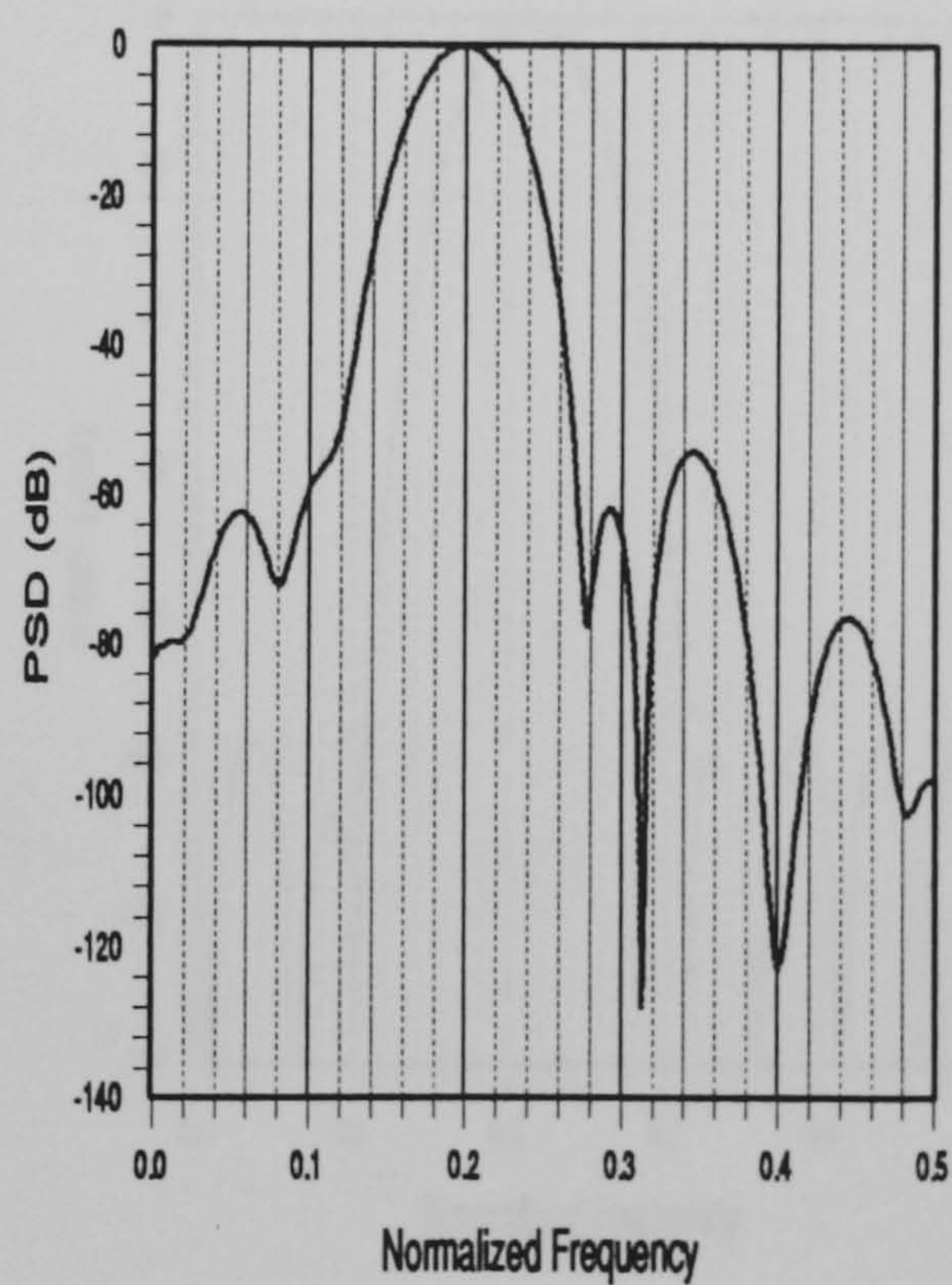
$F = 0.2$
 $IP = 20$
 $\mu = 0.001$
 $SNR = -5 \text{ dB}$



(b) PSD FROM ALEW

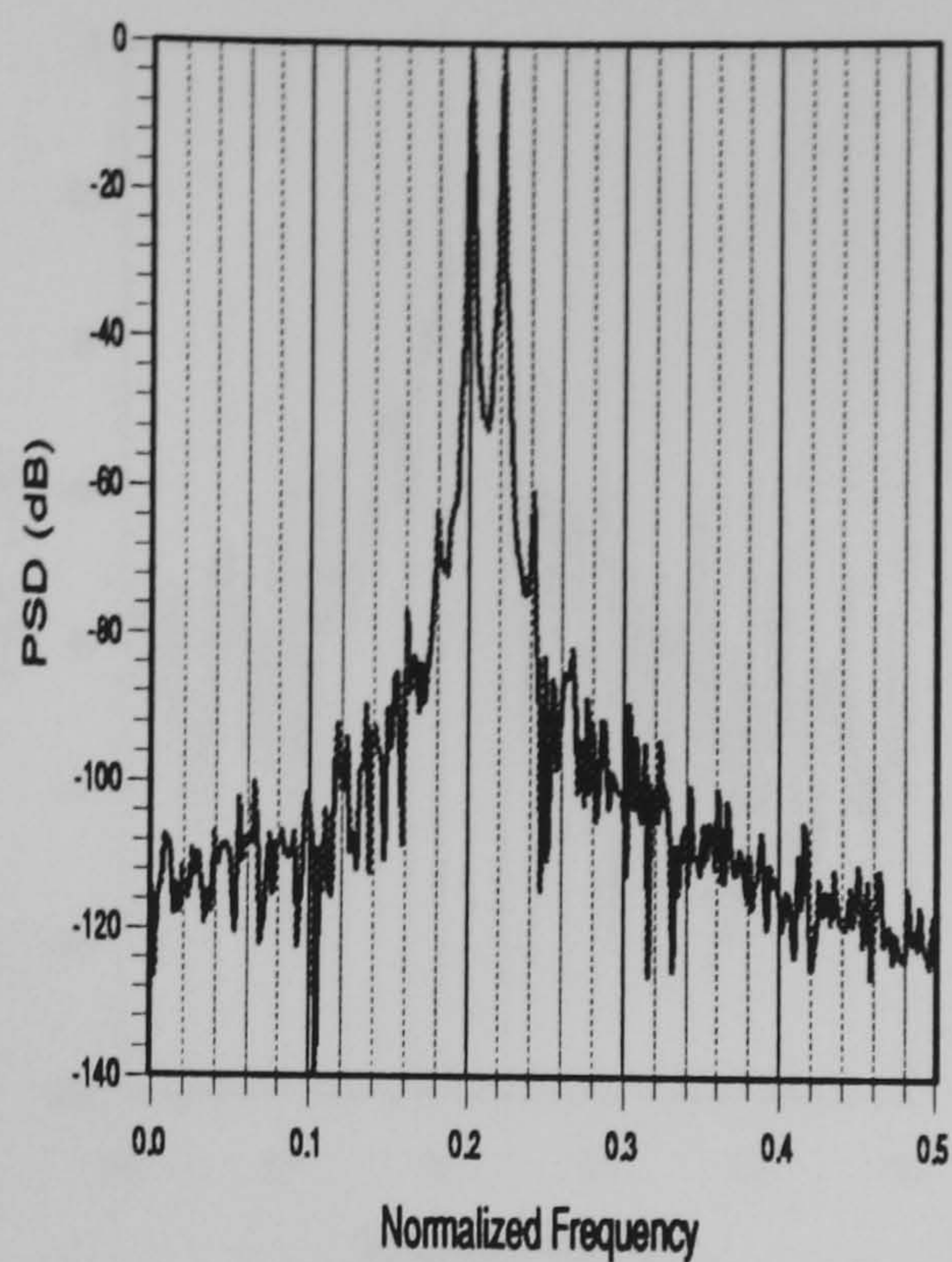


(c) PSD FROM APHD

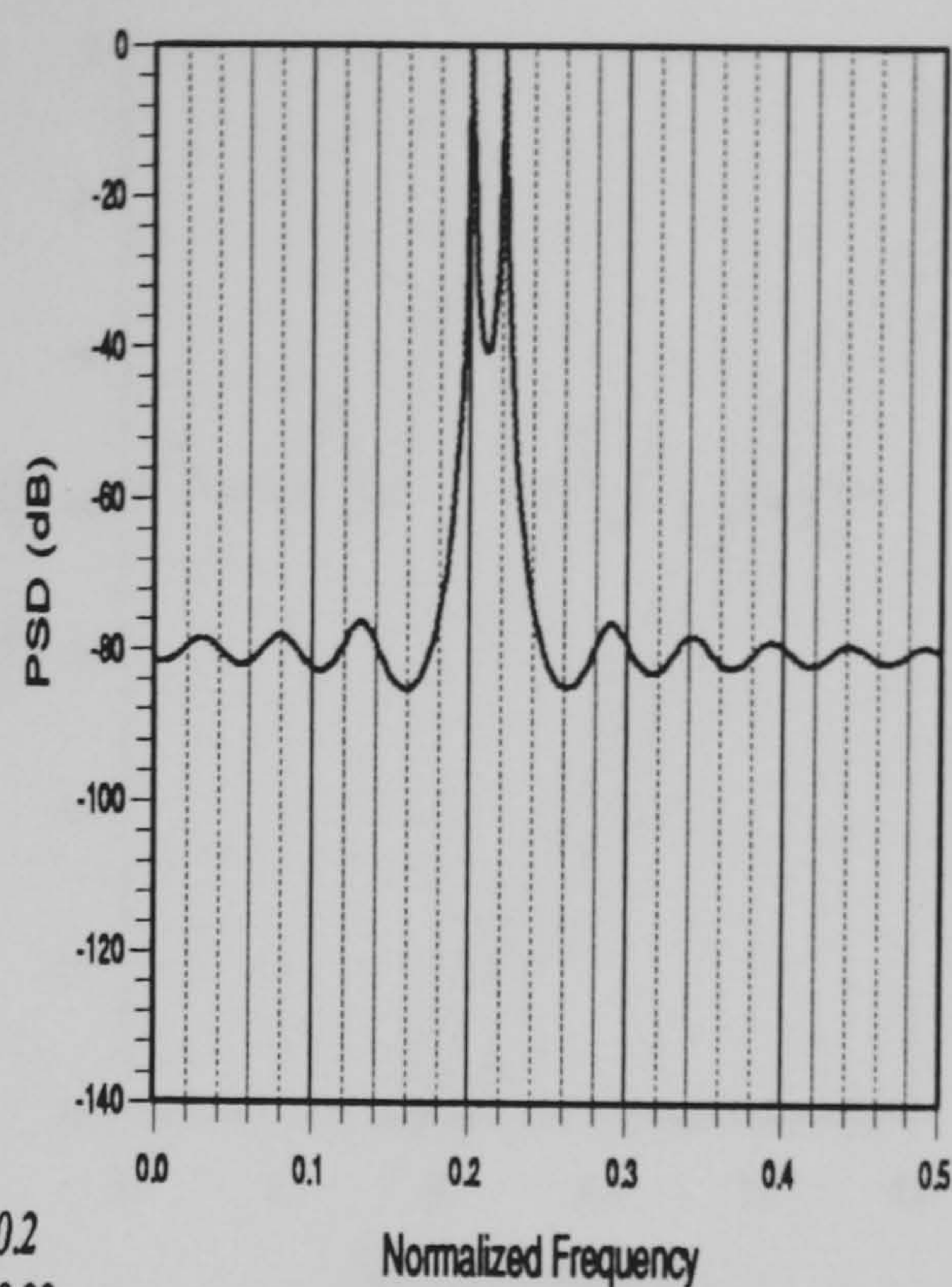


(d) PSD FROM AMAXEV

Figure 4.20: The detectability test: $SNR = -5 \text{ dB}$

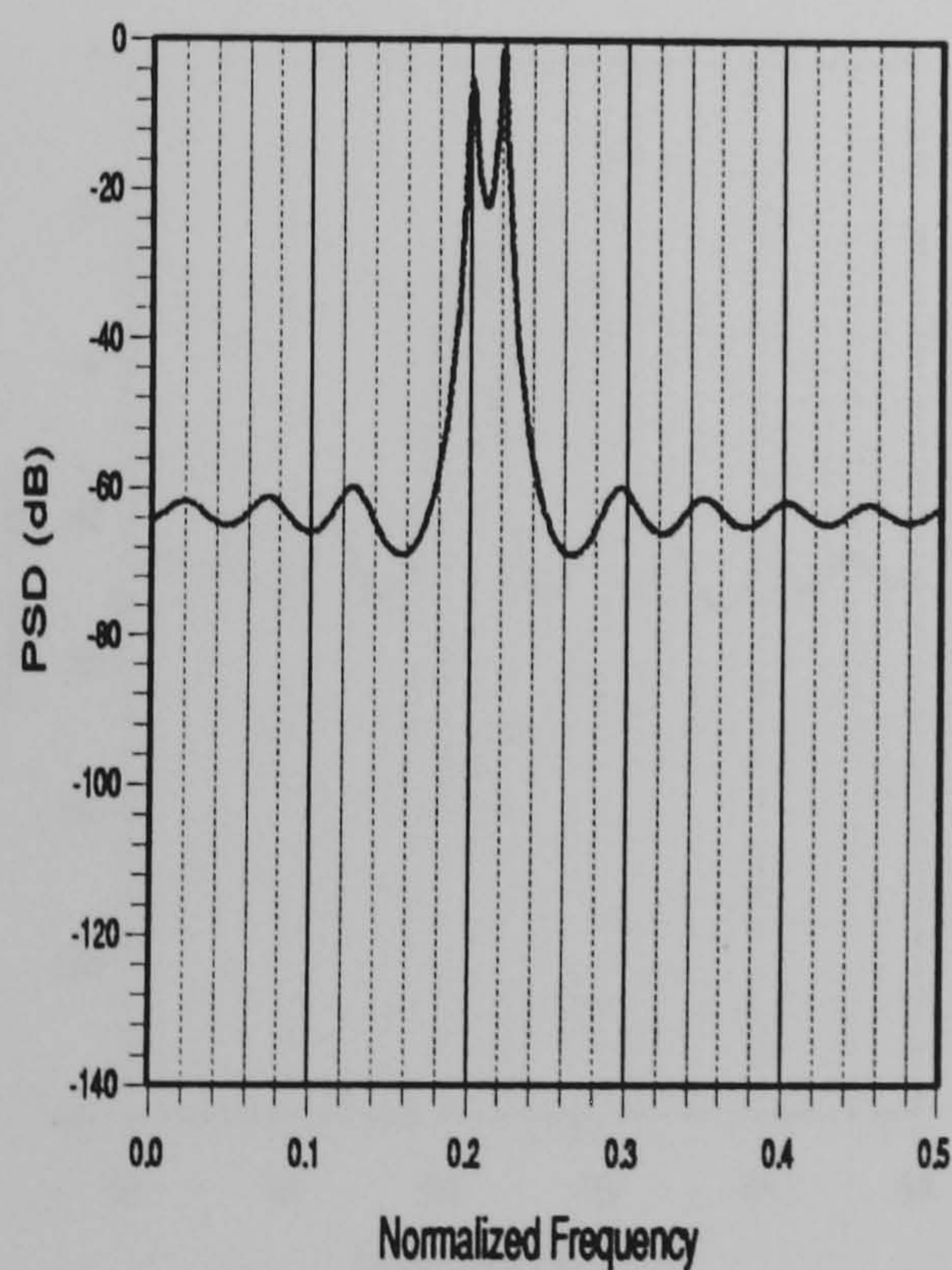


(a) PSD FROM ALEO

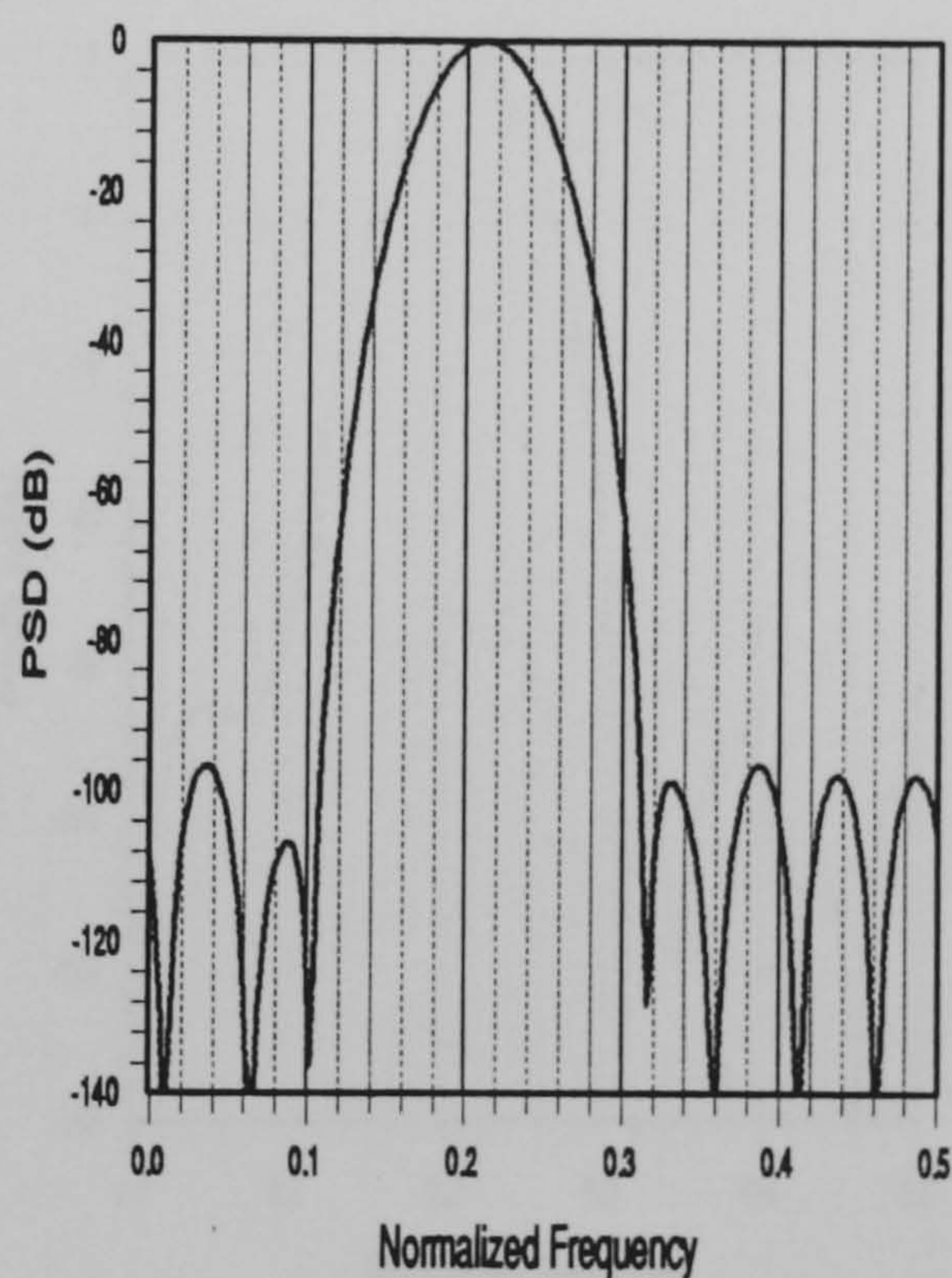


(b) PSD FROM ALEW

$F1 = 0.2$
 $F2 = 0.22$
 $IP = 20$
 $\mu = 0.001$
 $SNR = 20 \text{ DB}$

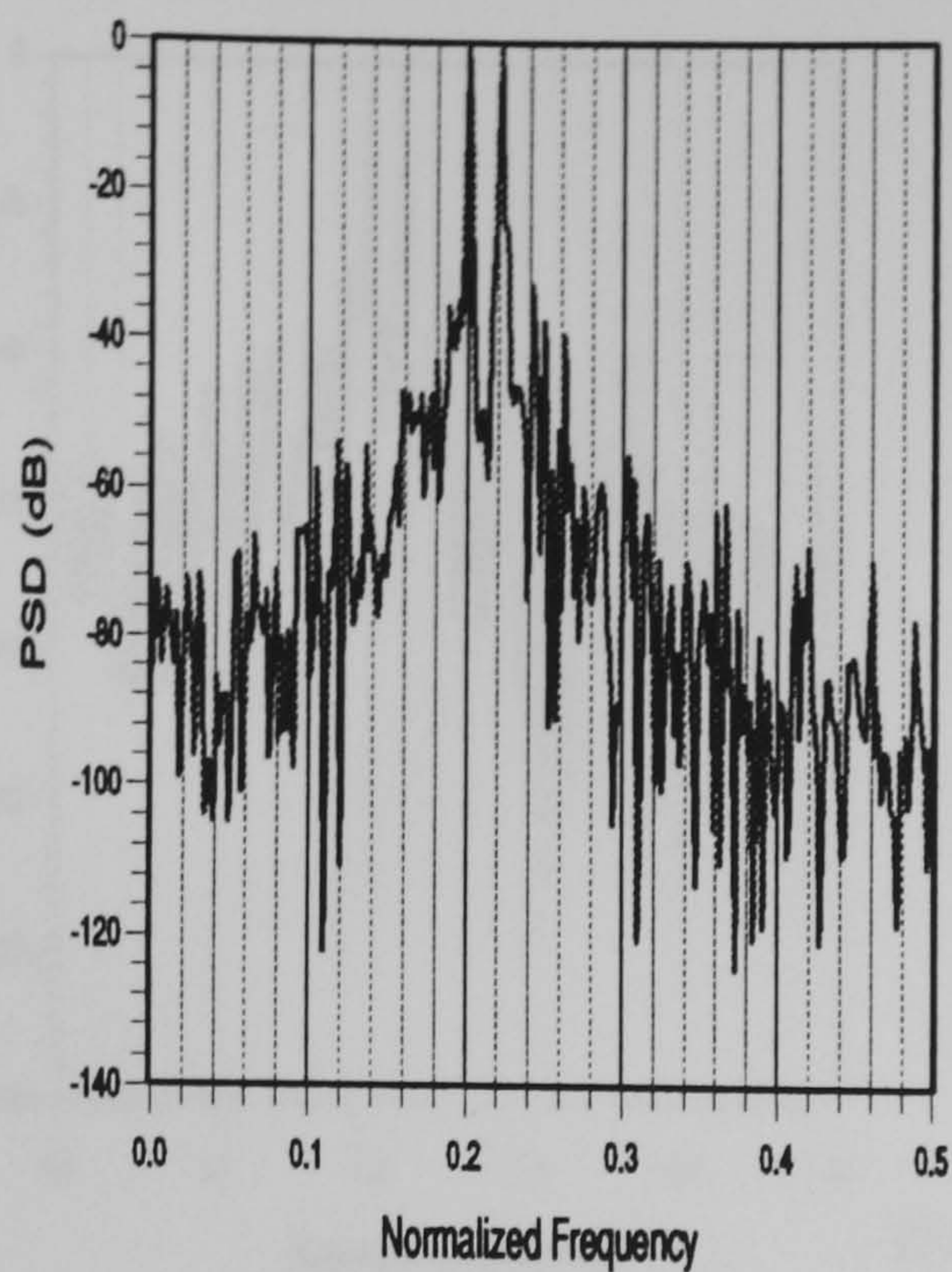


(c) PSD FROM APHD

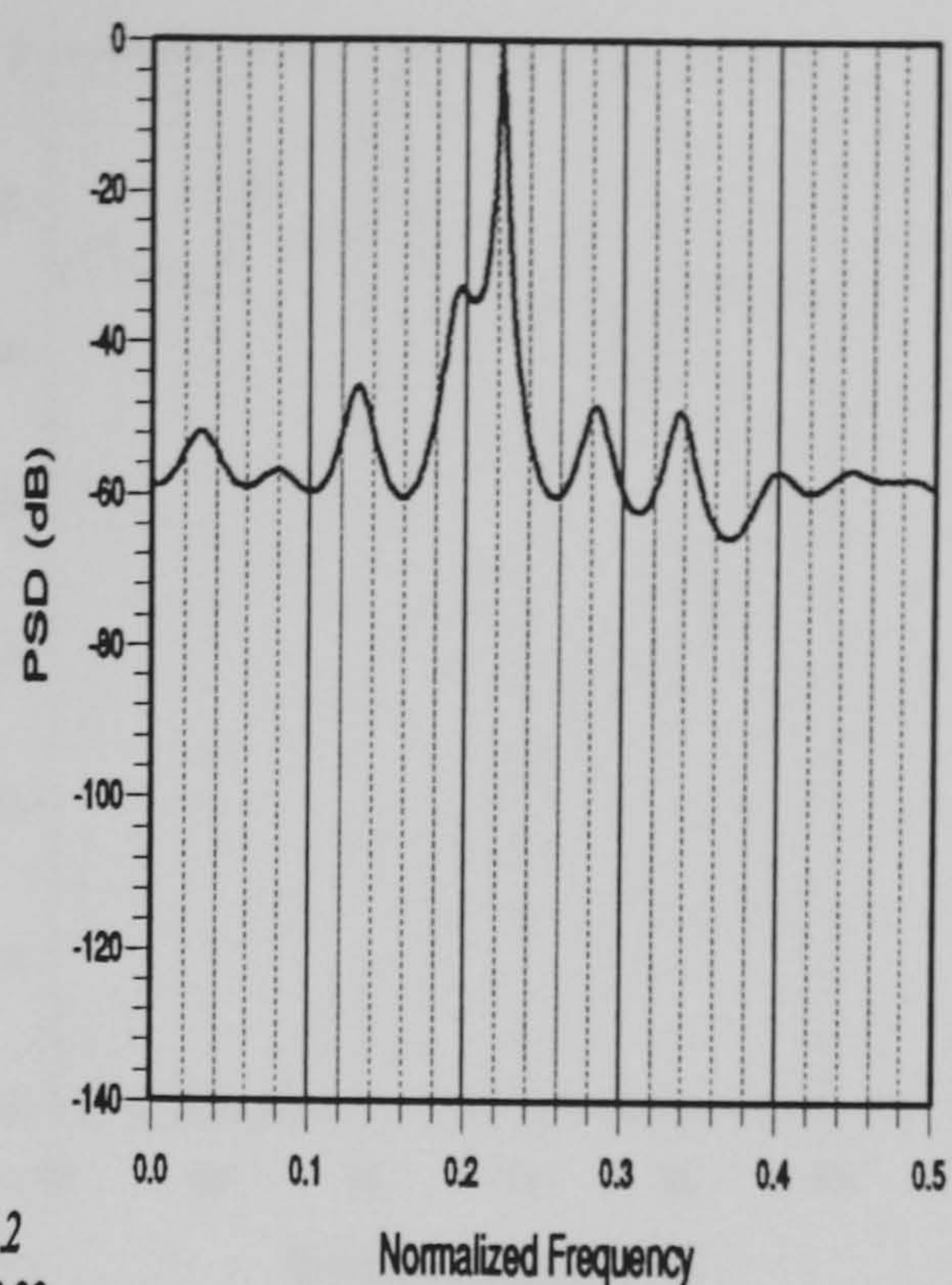


(d) PSD FROM AMAXEV

Figure 4.21: The frequency resolution test: $SNR = 20 \text{ dB}$

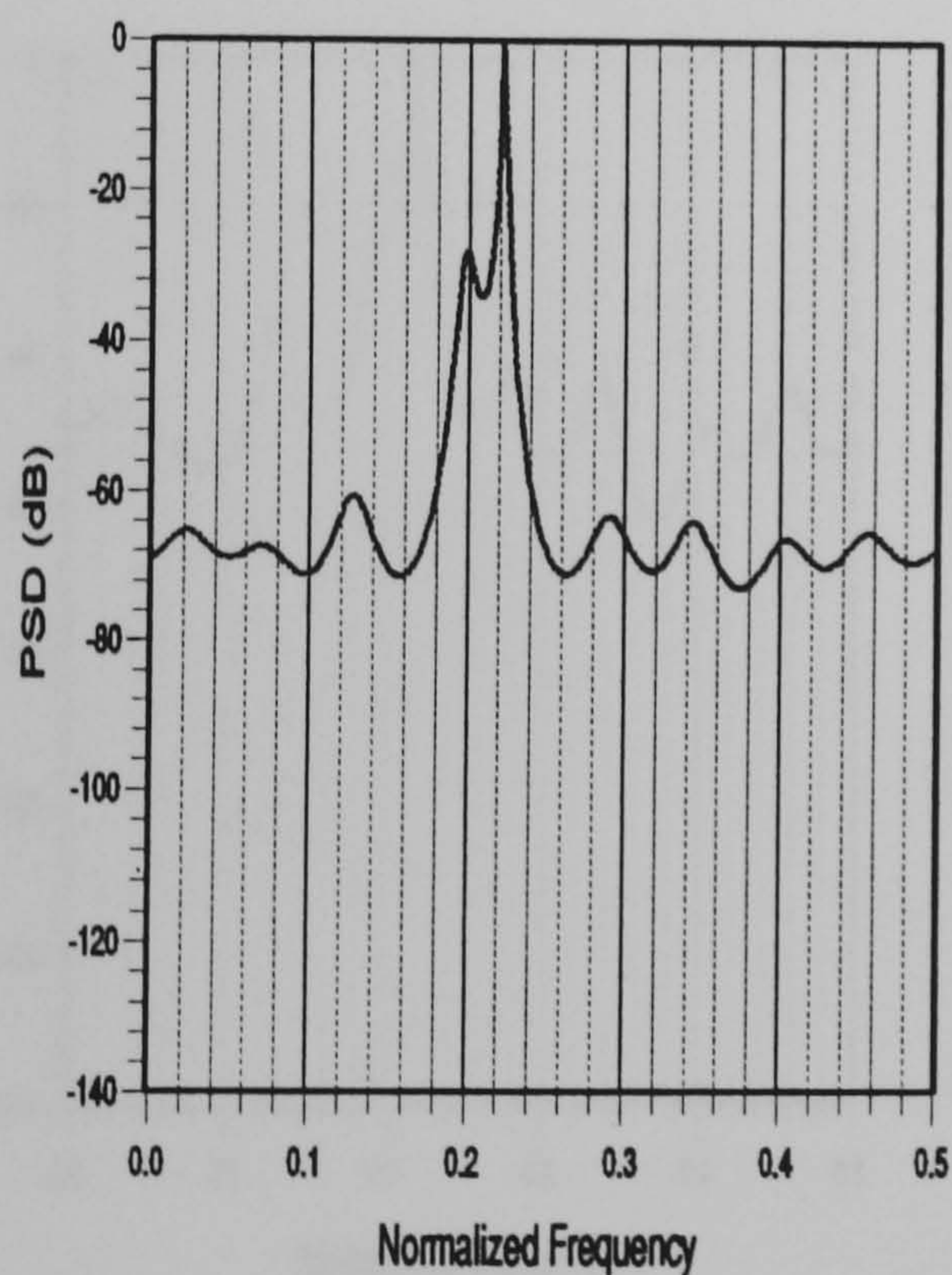


(a) PSD FROM ALEO

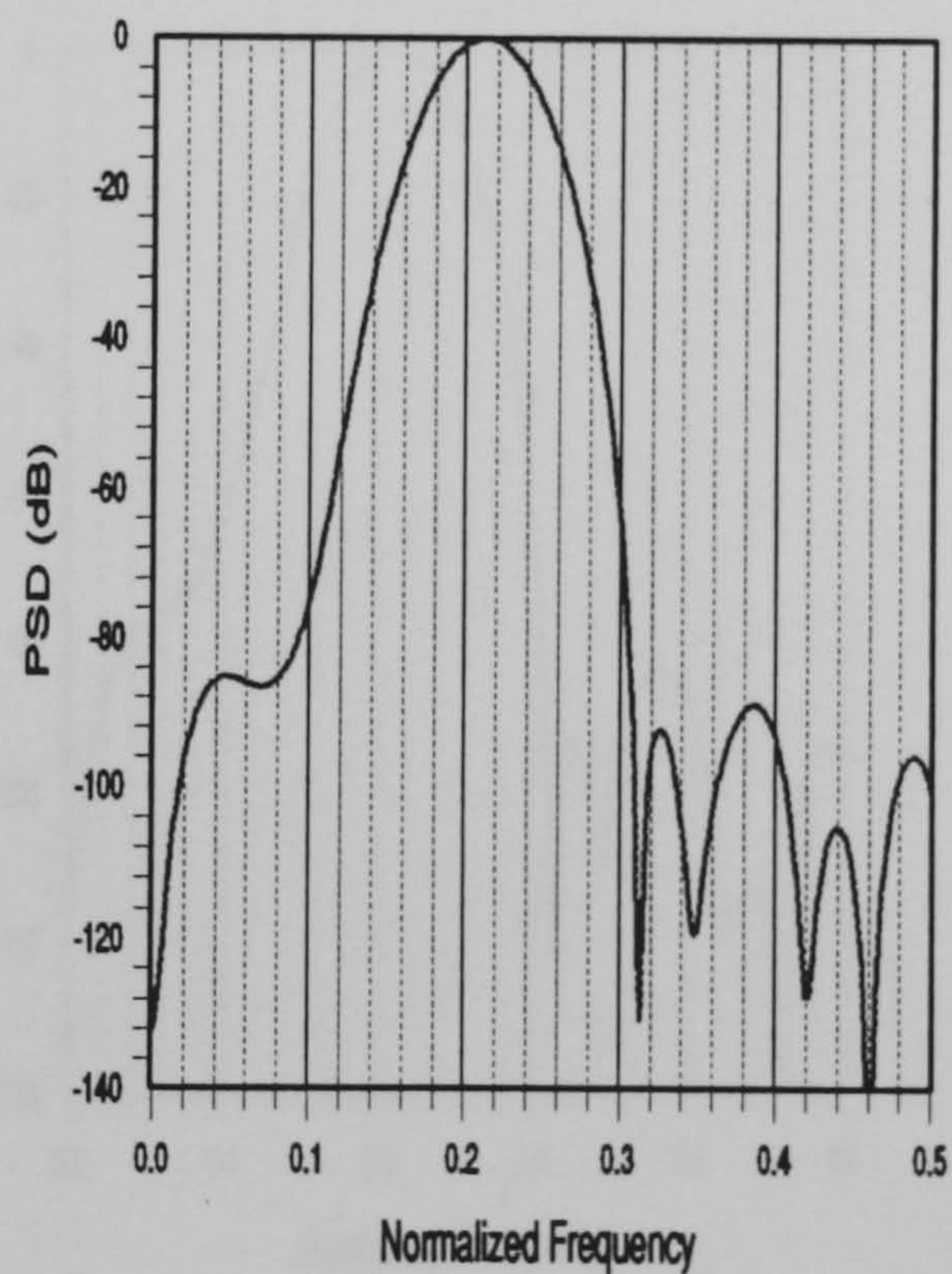


(b) PSD FROM ALEW

$F1 = 0.2$
 $F2 = 0.22$
 $IP = 20$
 $\mu = 0.001$
 $SNR = 0 \text{ dB}$

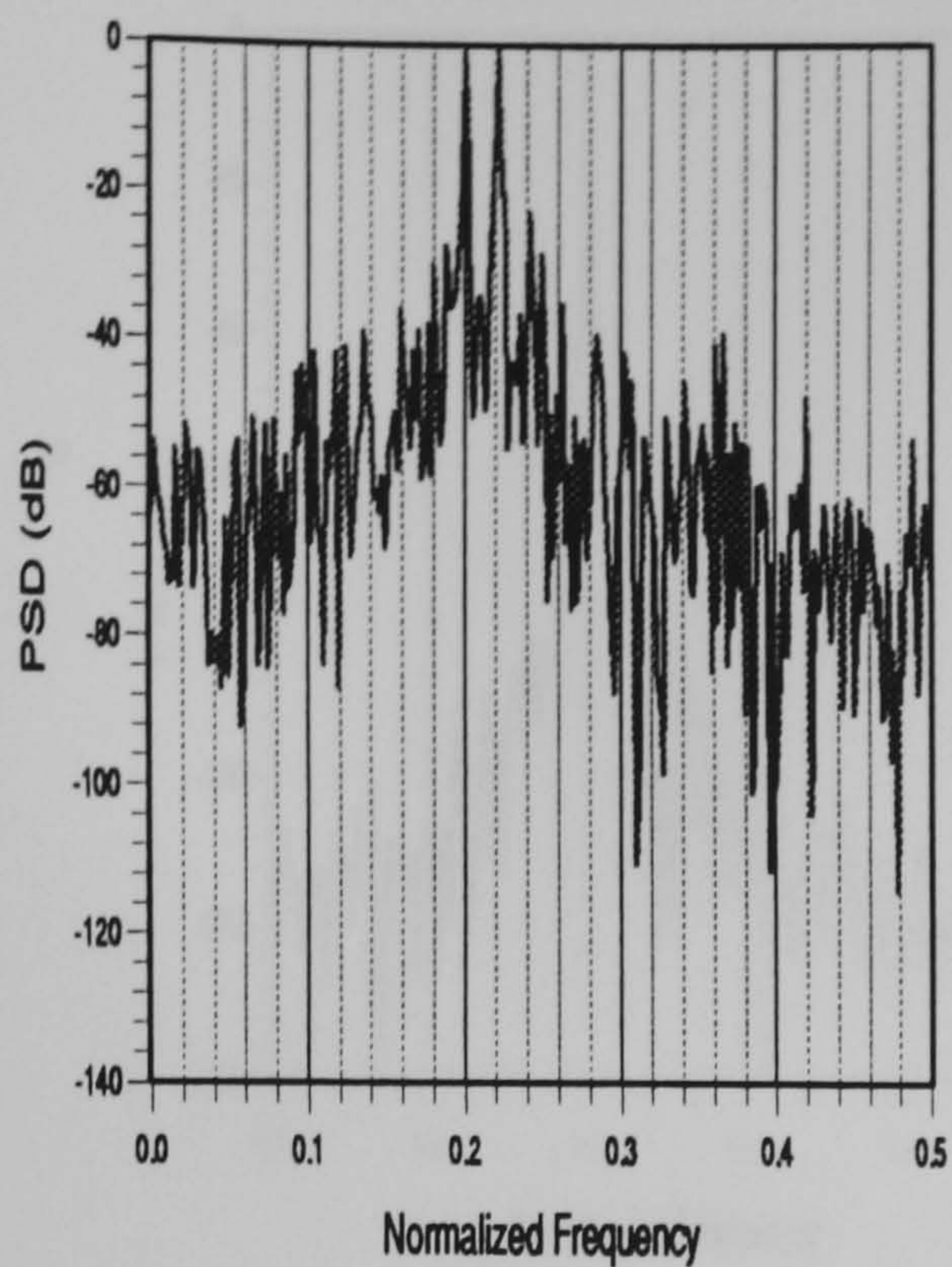


(c) PSD FROM APHD



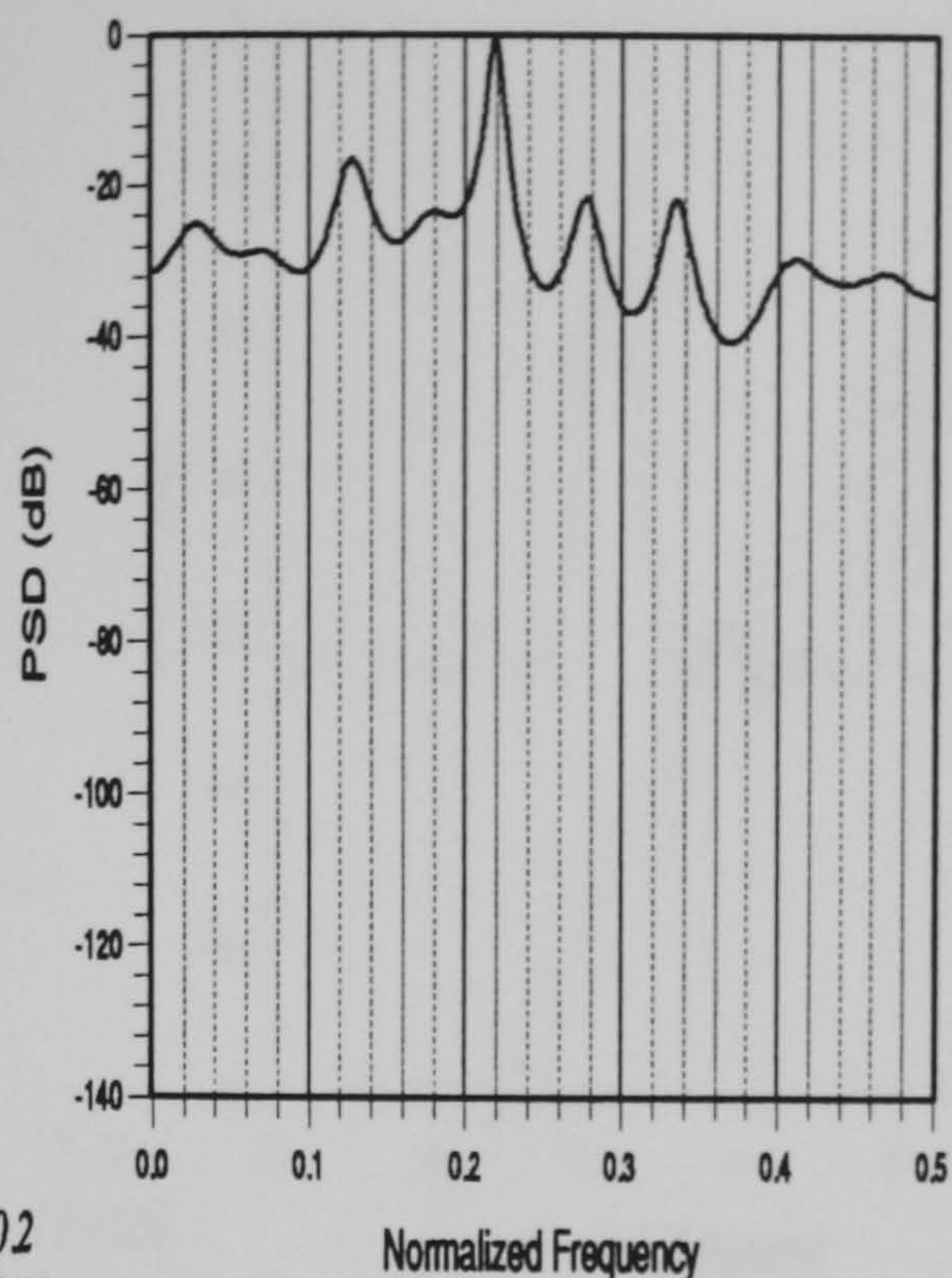
(d) PSD FROM AMAXEV

Figure 4.22: The frequency resolution test: $SNR = 0 \text{ dB}$

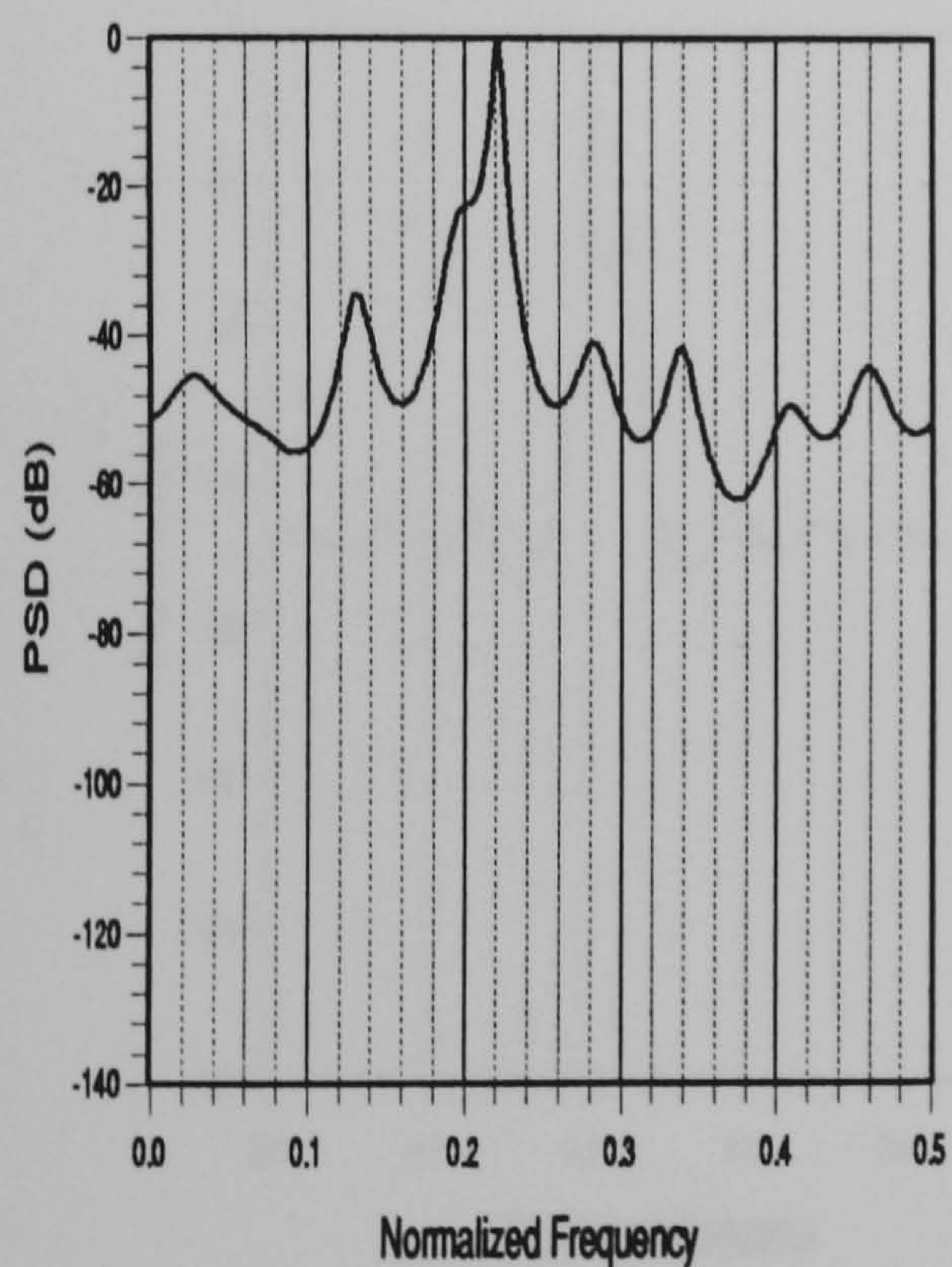


(a) PSD FROM ALEO

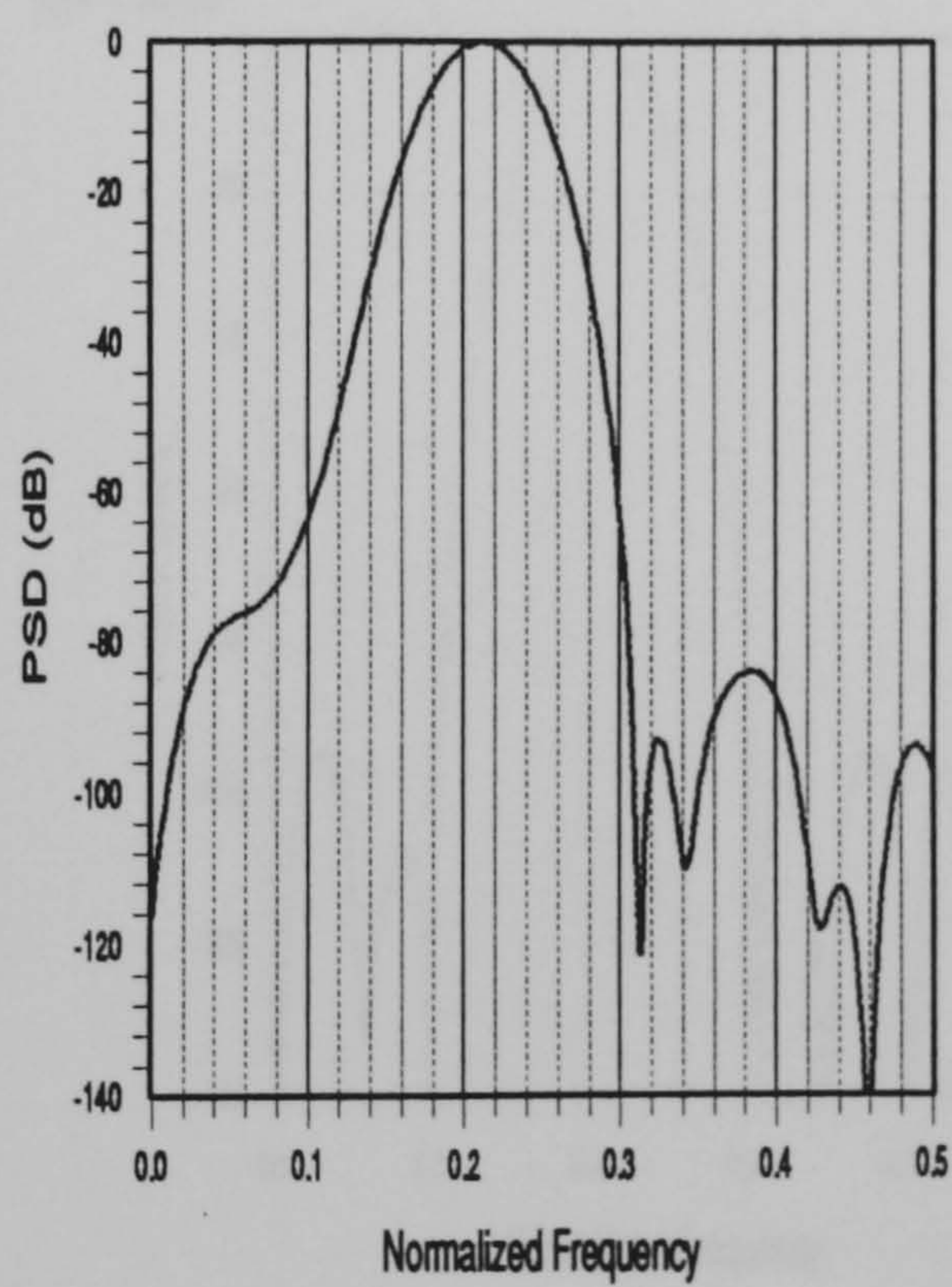
$F1 = 0.2$
 $F2 = 0.22$
 $IP = 20$
 $\mu = 0.001$
 $SNR = -5 \text{ dB}$



(b) PSD FROM ALEW

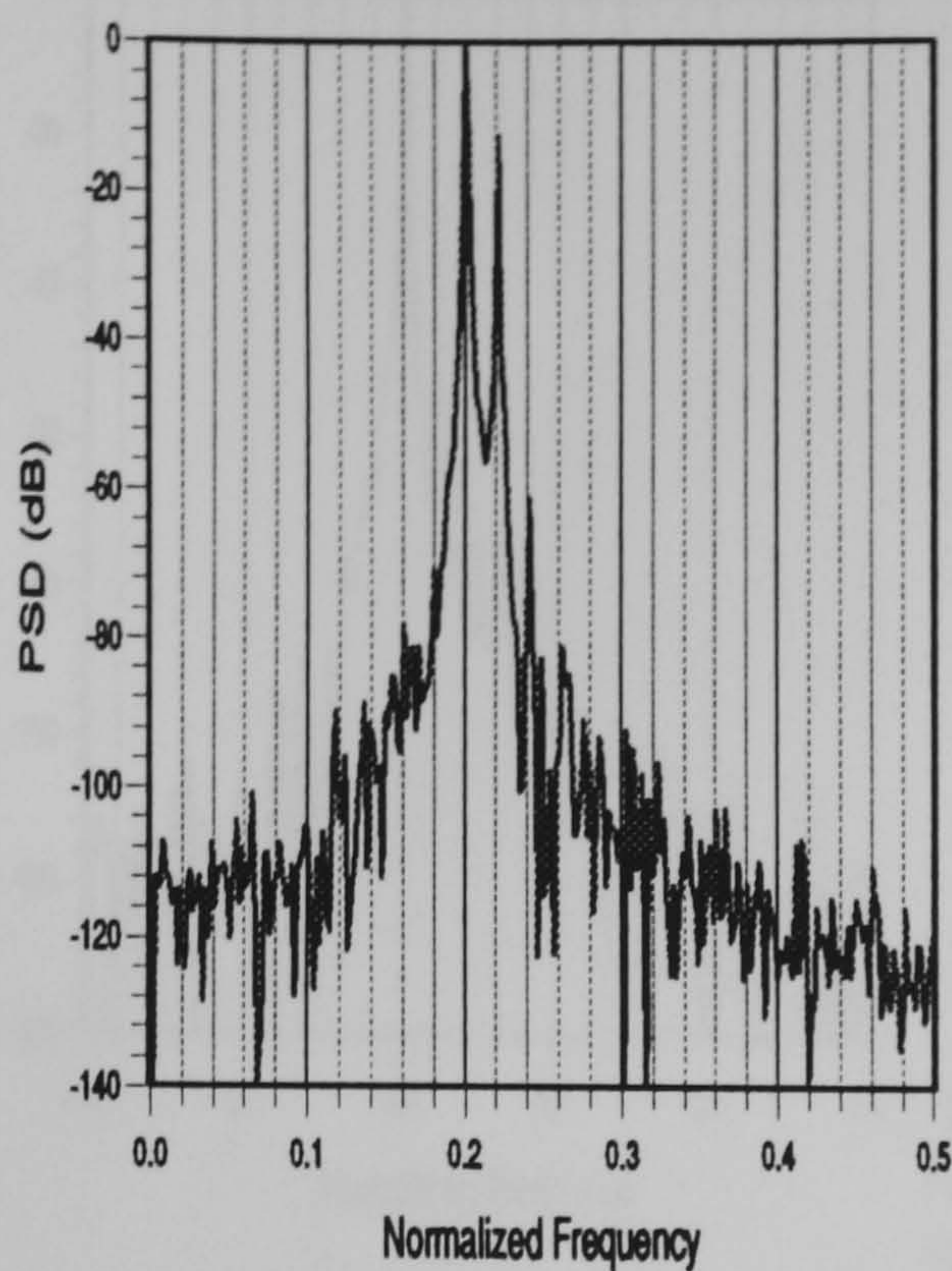


(c) PSD FROM APHD

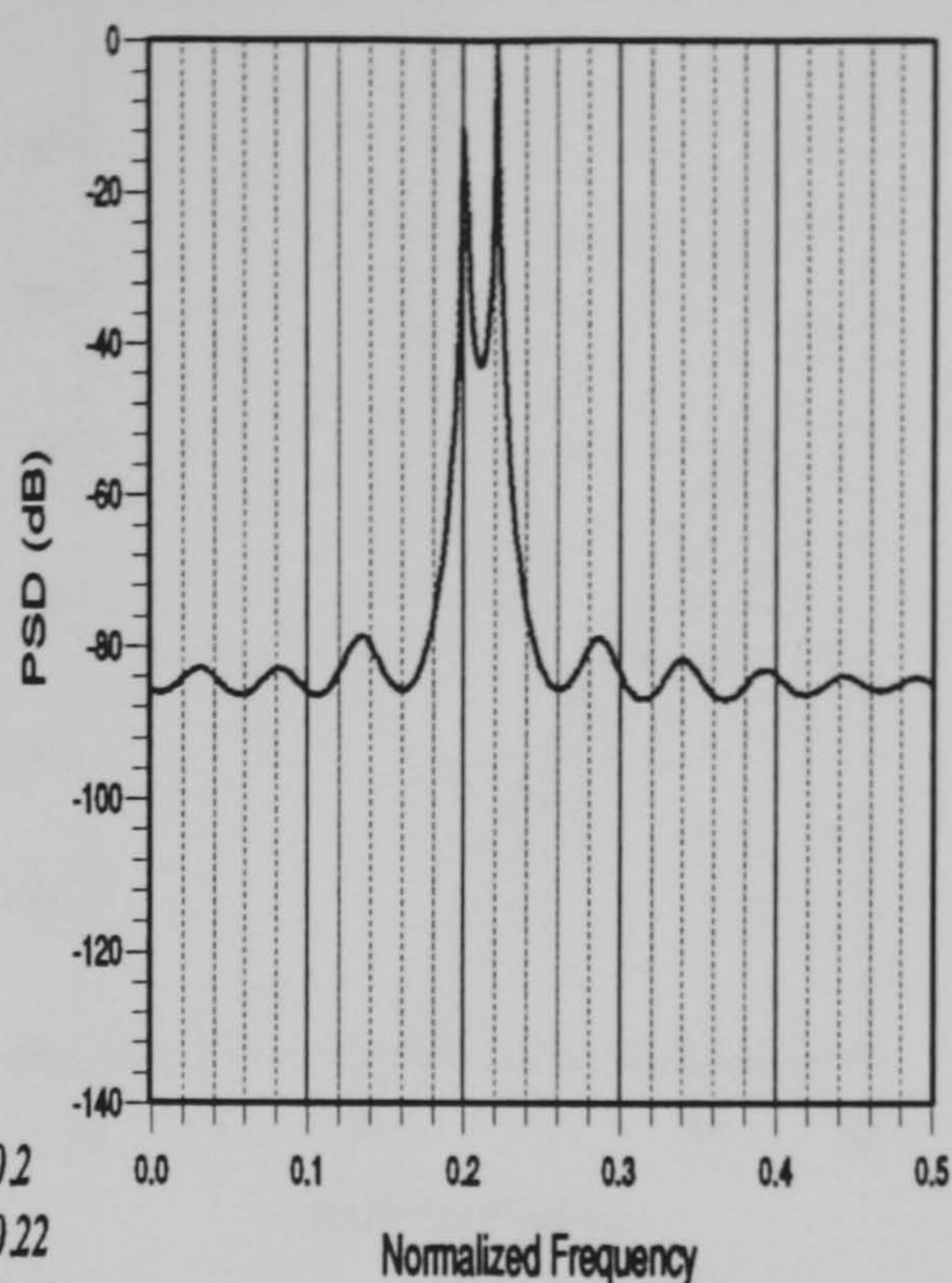


(d) PSD FROM AMAXEV

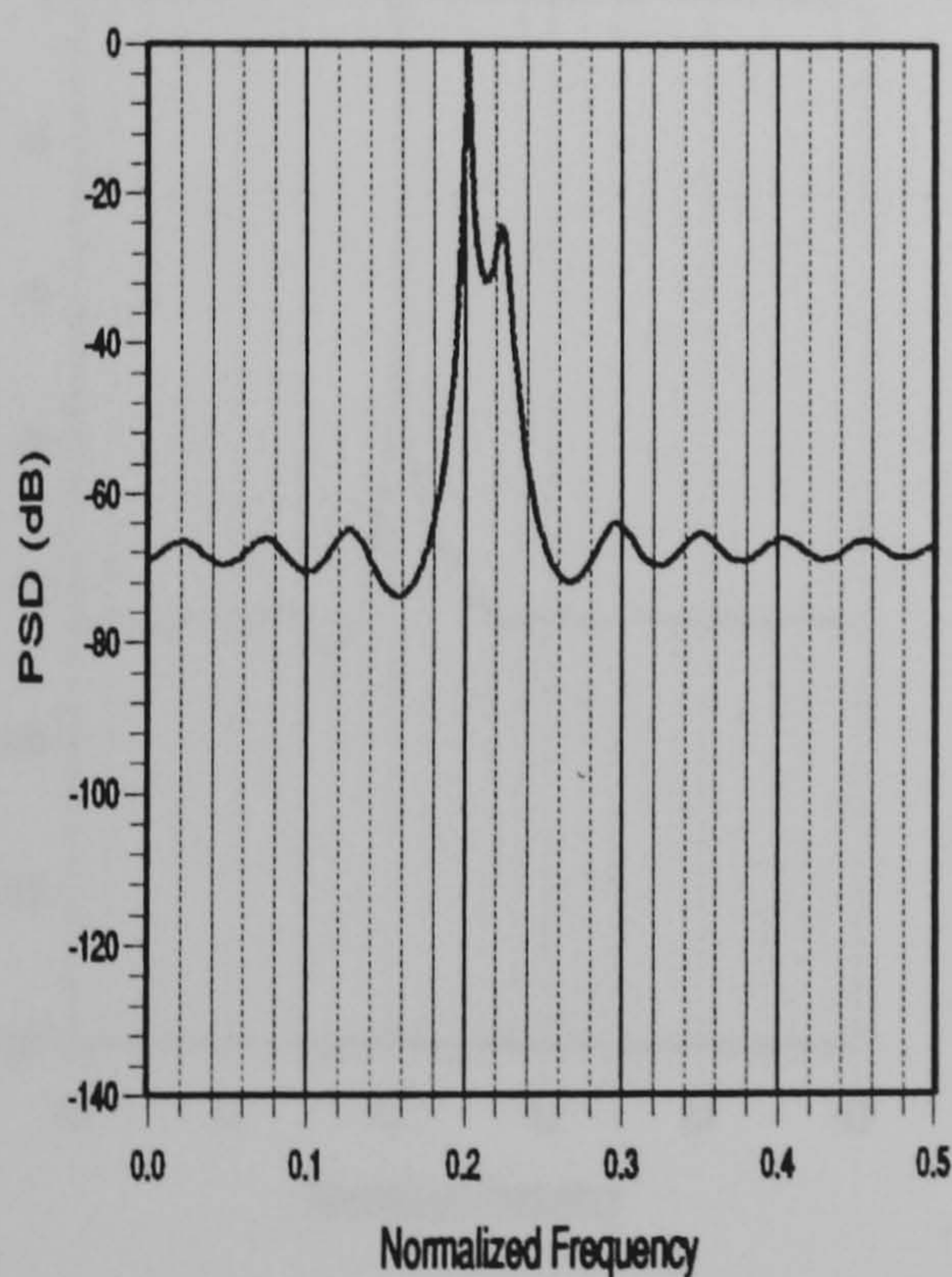
Figure 4.23: The frequency resolution test: $SNR = -5 \text{ dB}$



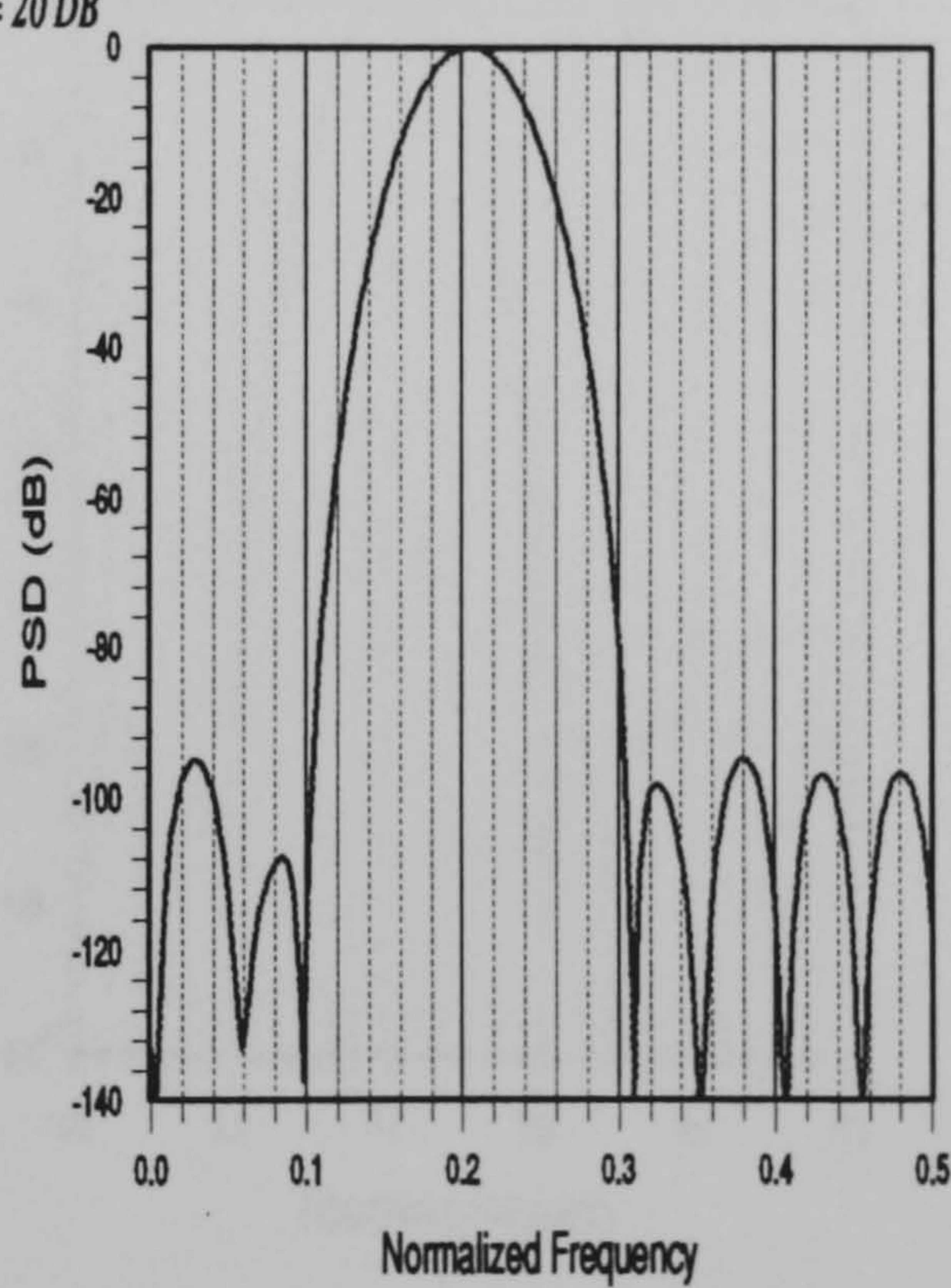
(a) PSD FROM ALEO



(b) PSD FROM ALEW



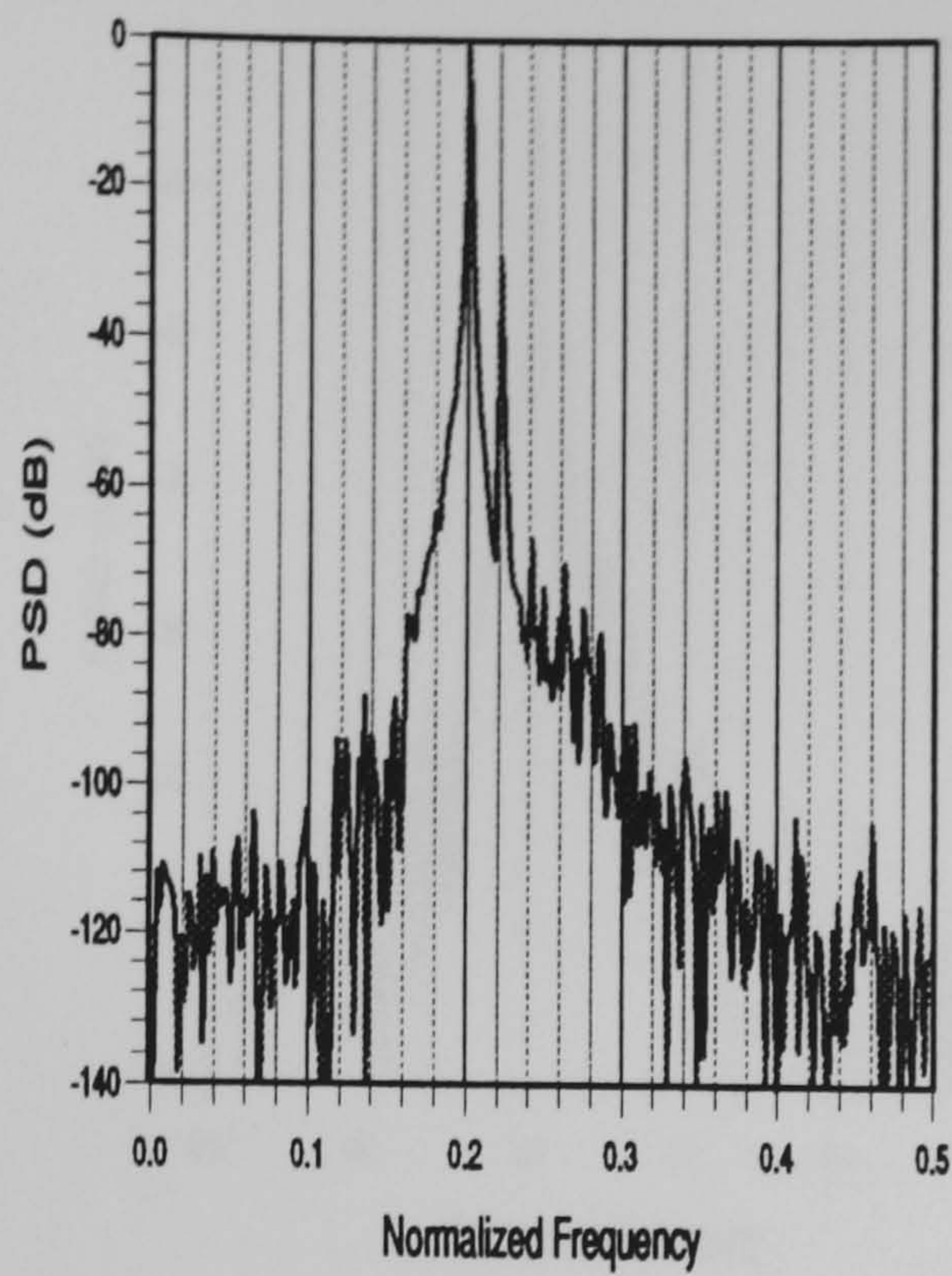
(c) PSD FROM APHD



(d) PSD FROM AMAXEV

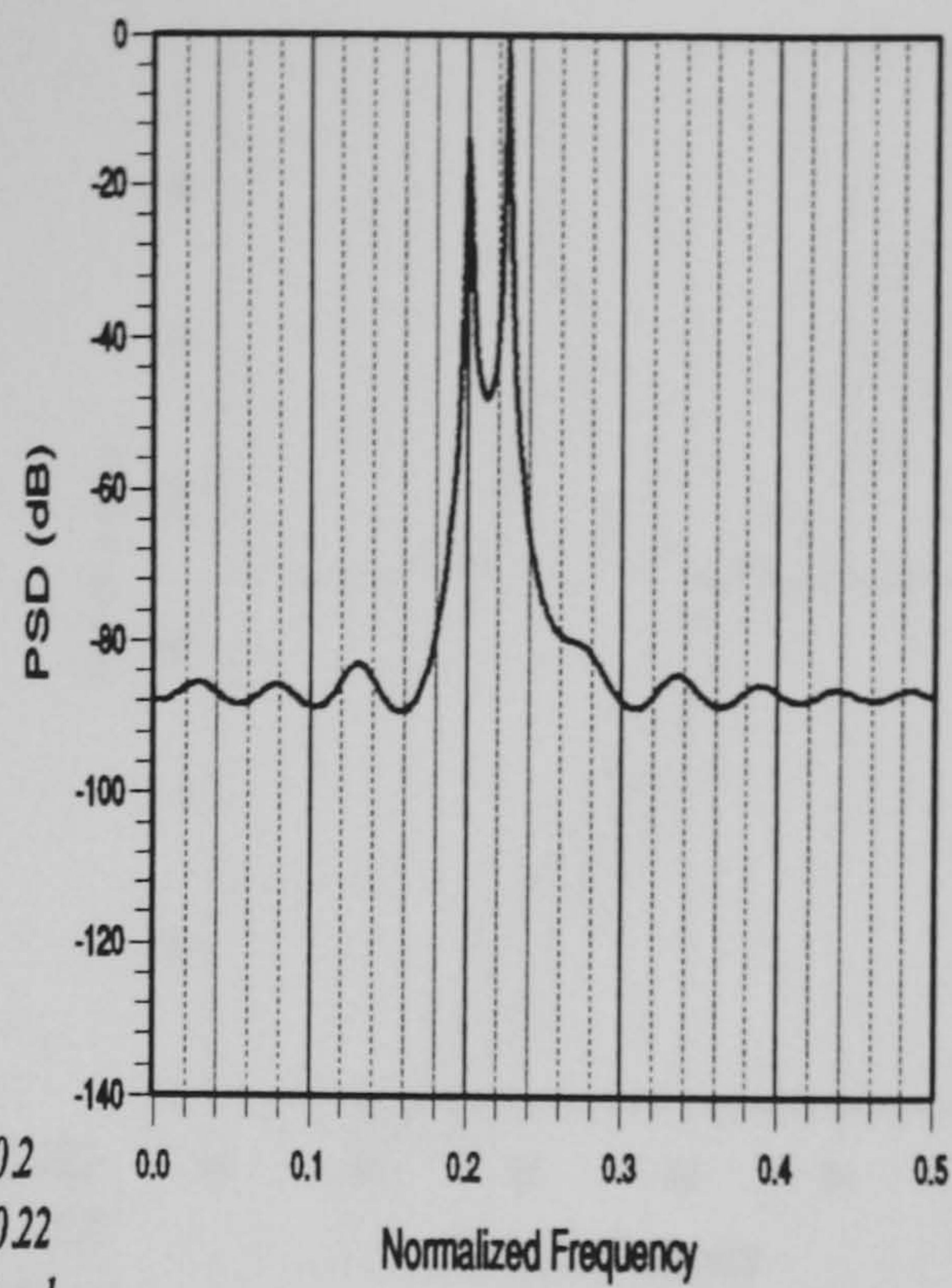
$F1 = 0.2$
 $F2 = 0.22$
 $Amp1 = 1.$
 $Amp2 = 0.5$
 $IP = 20$
 $\mu = 0.002$
 $SNR = 20\text{ DB}$

Figure 4.24: The frequency resolution test: relative amplitude = 0.5

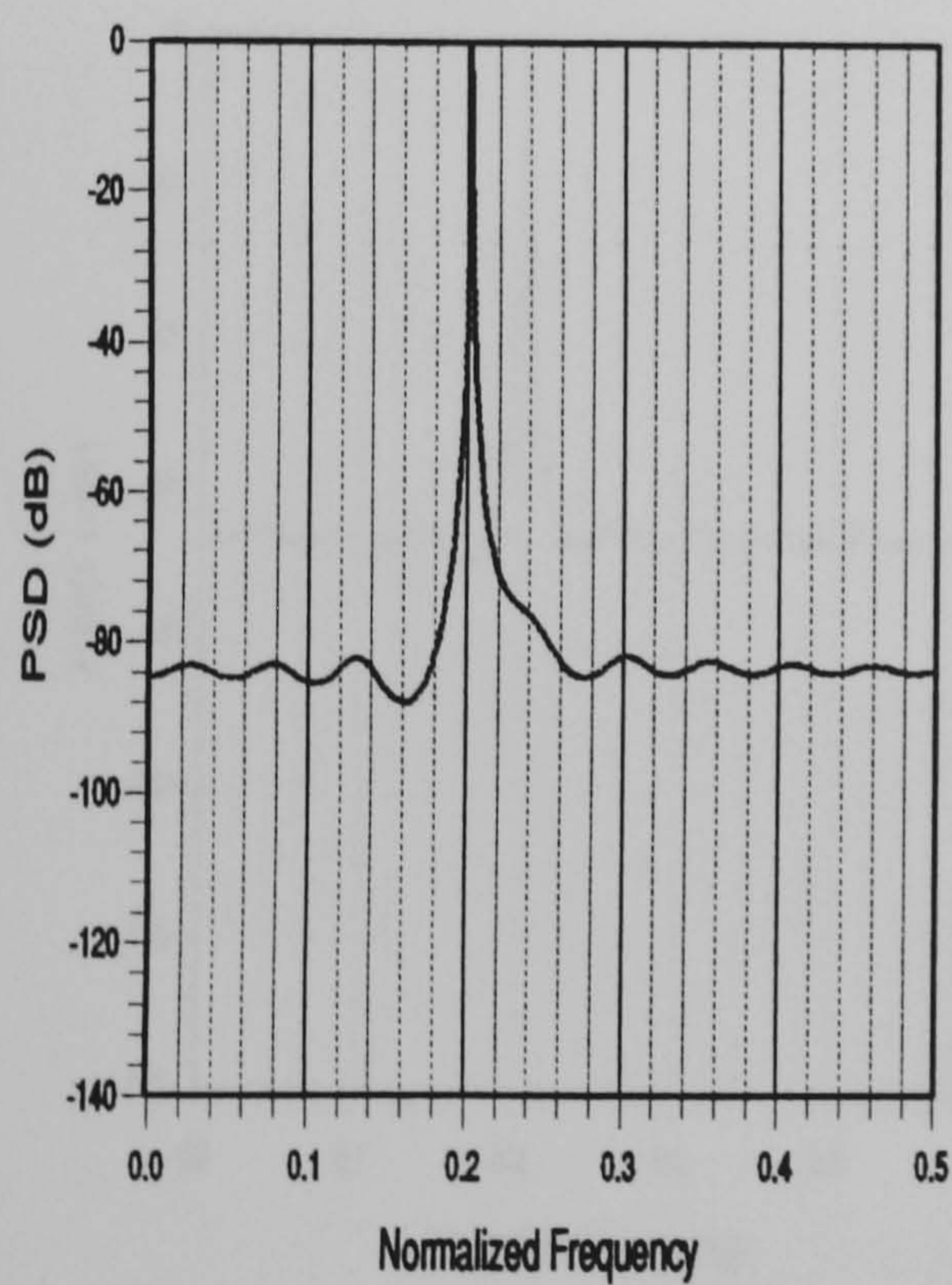


(a) PSD FROM ALEO

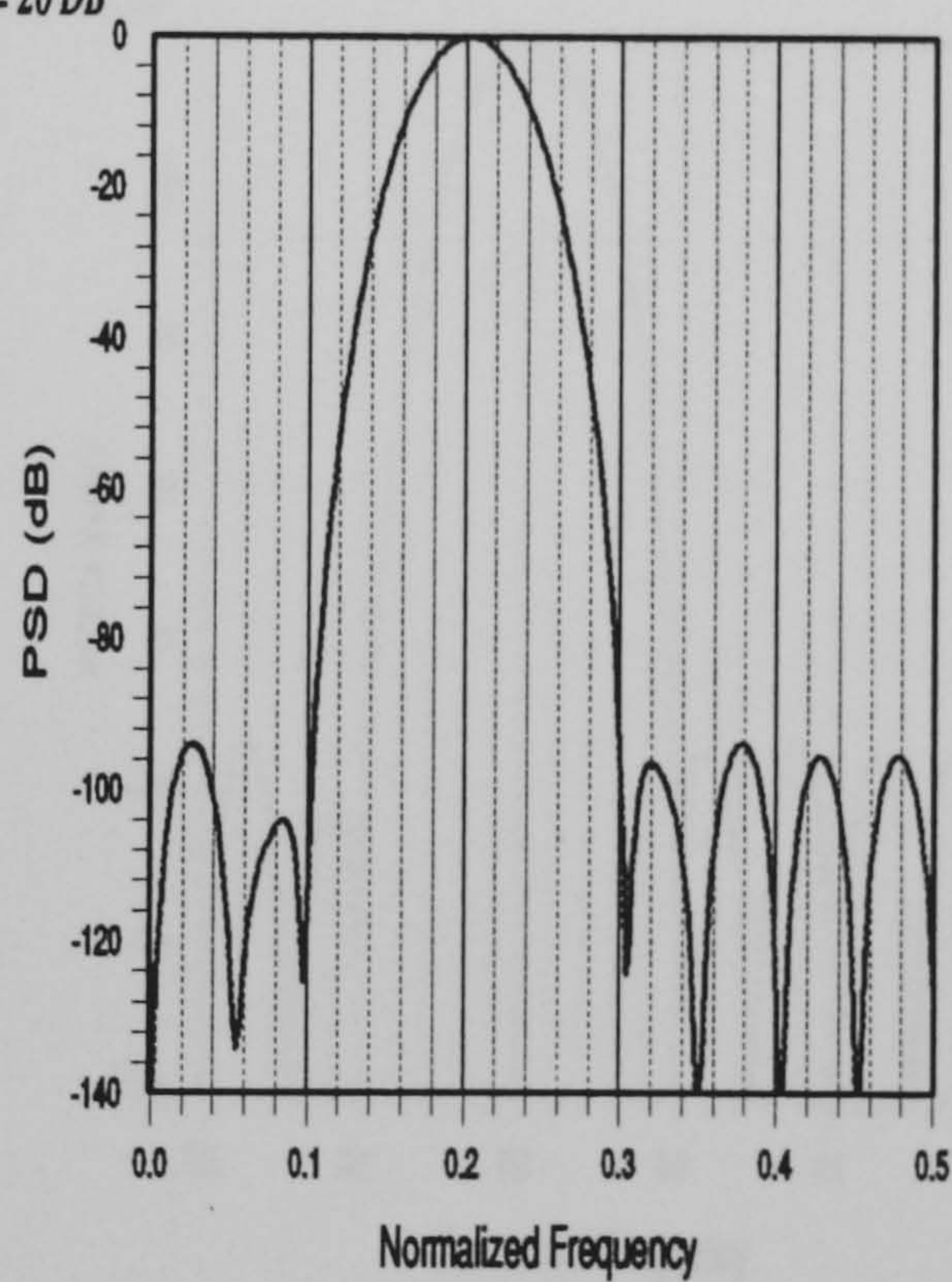
$F1 = 0.2$
 $F2 = 0.22$
 $Amp1 = 1.$
 $Amp2 = 0.25$
 $IP = 20$
 $\mu = 0.002$
 $SNR = 20\text{ DB}$



(b) PSD FROM ALEW

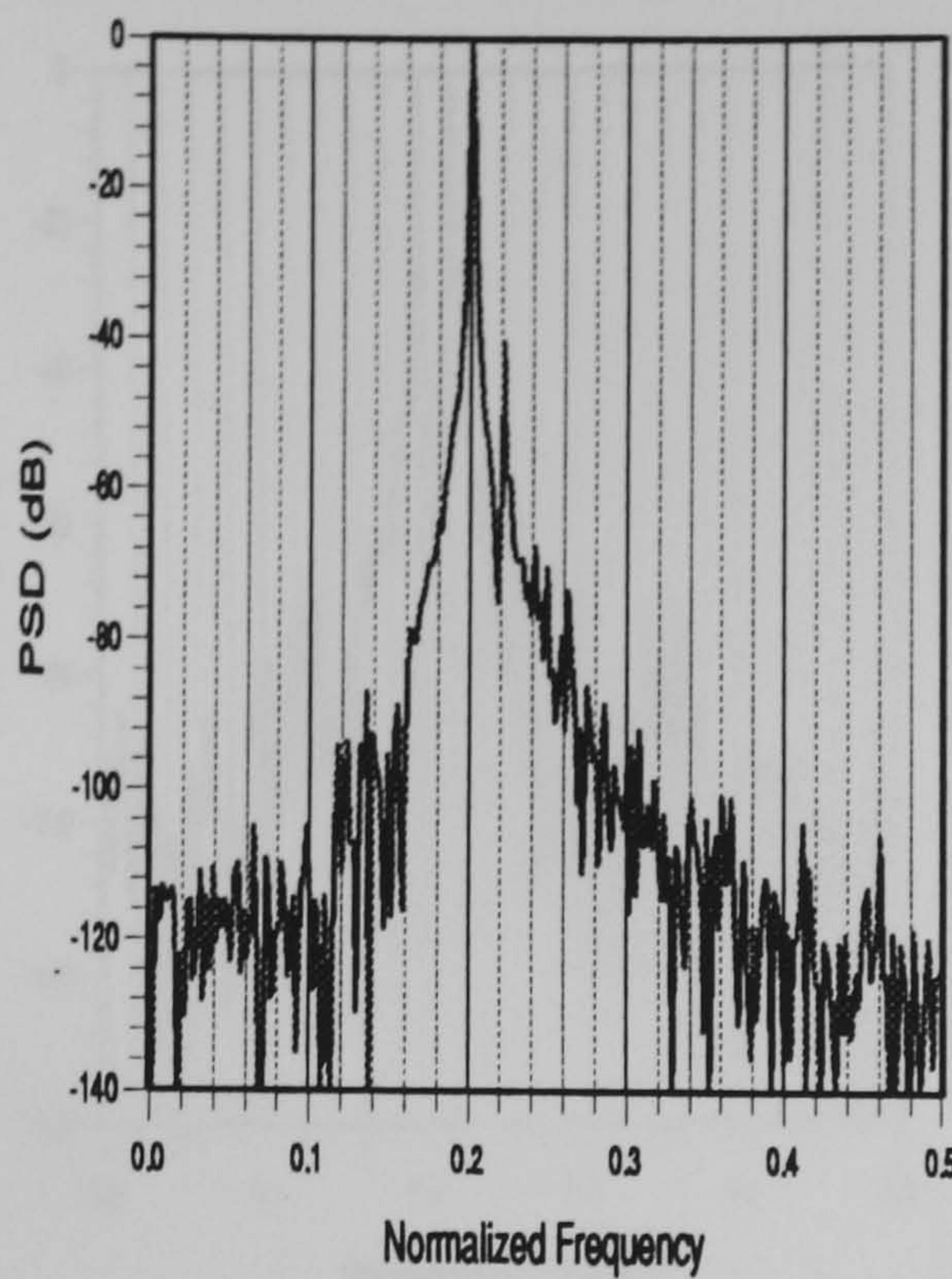


(c) PSD FROM APHD

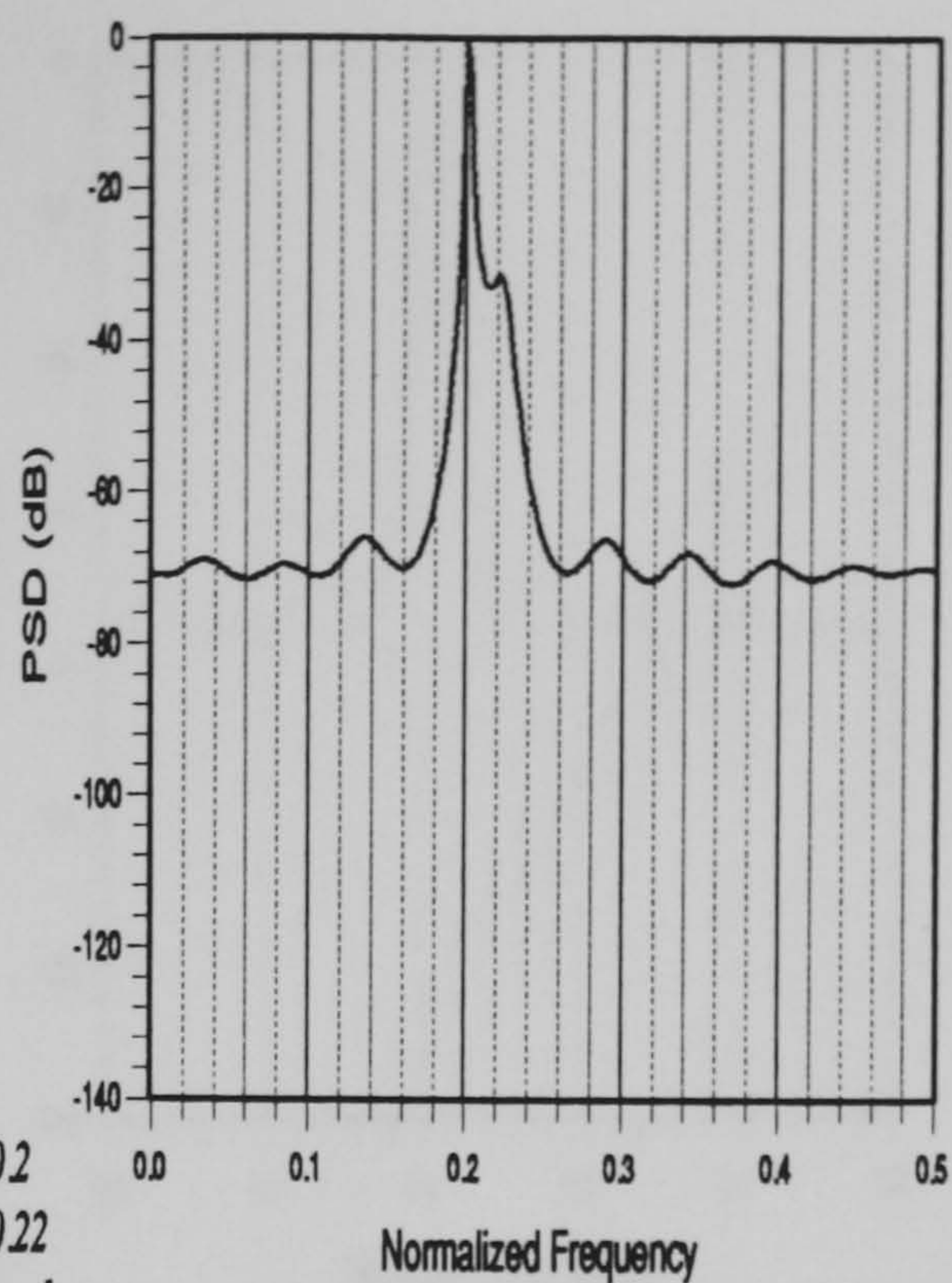


(d) PSD FROM AMAXEV

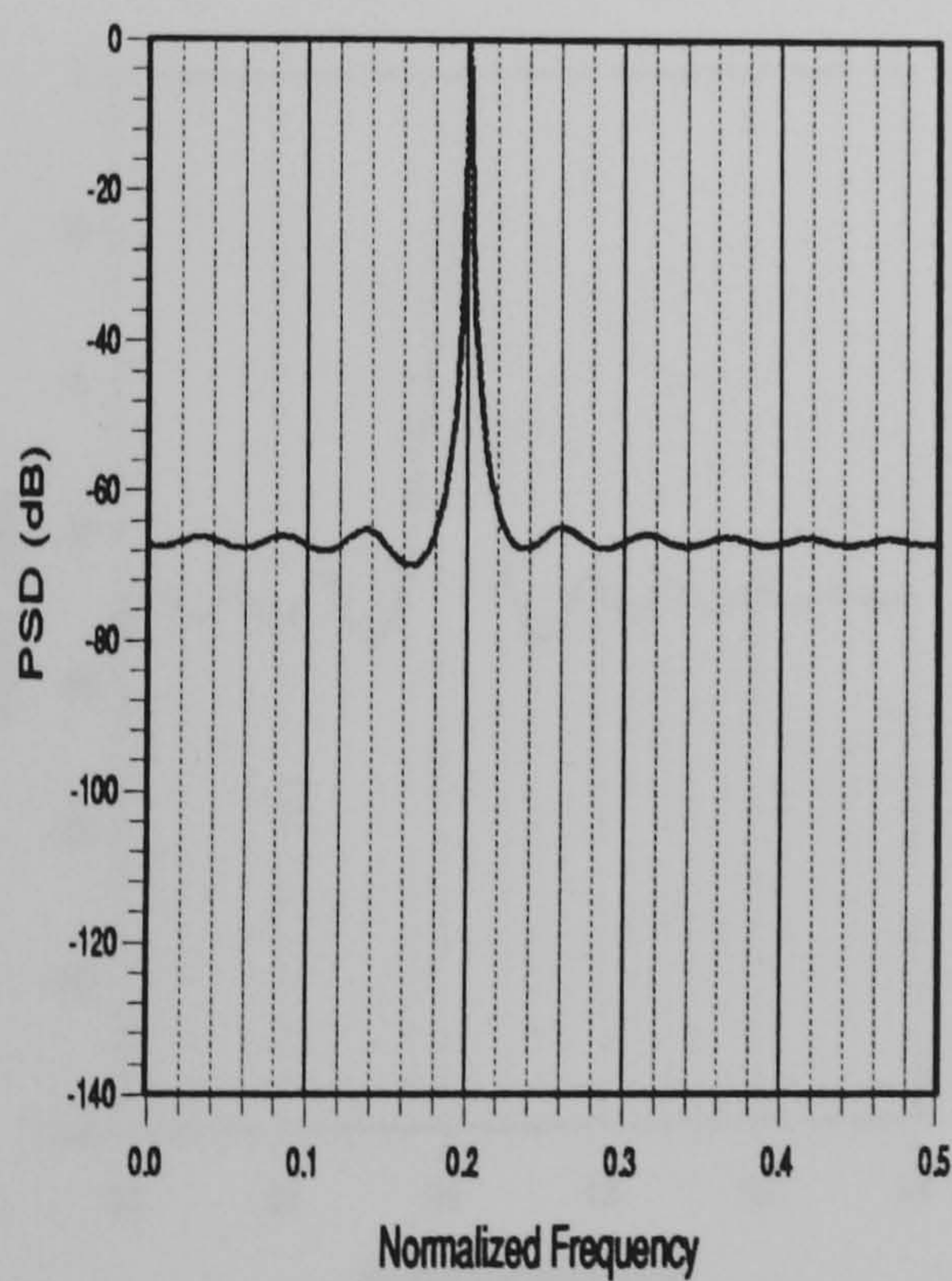
Figure 4.25: The frequency resolution test: relative amplitude = 0.25



(a) PSD FROM ALEO

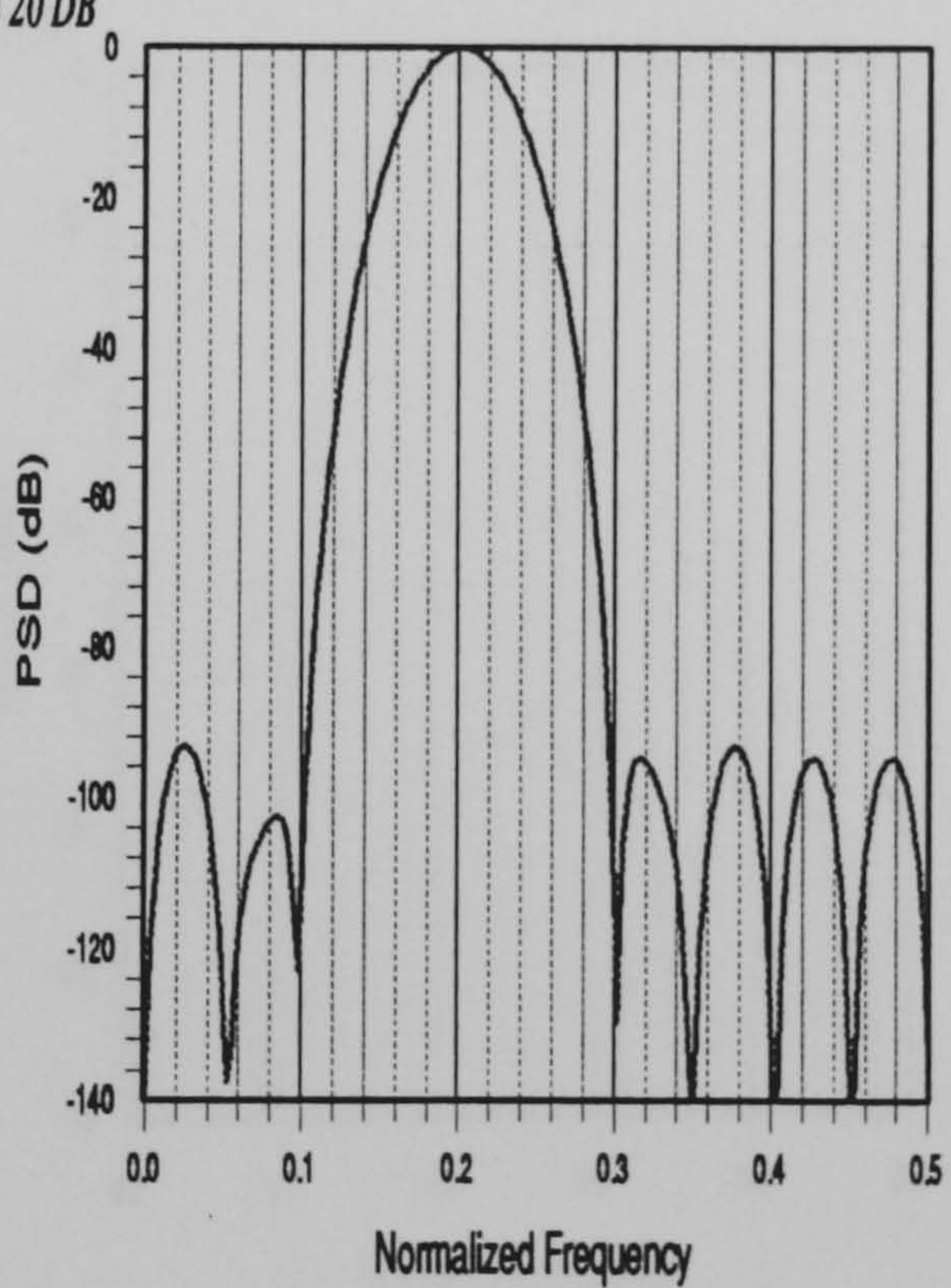


(b) PSD FROM ALEW



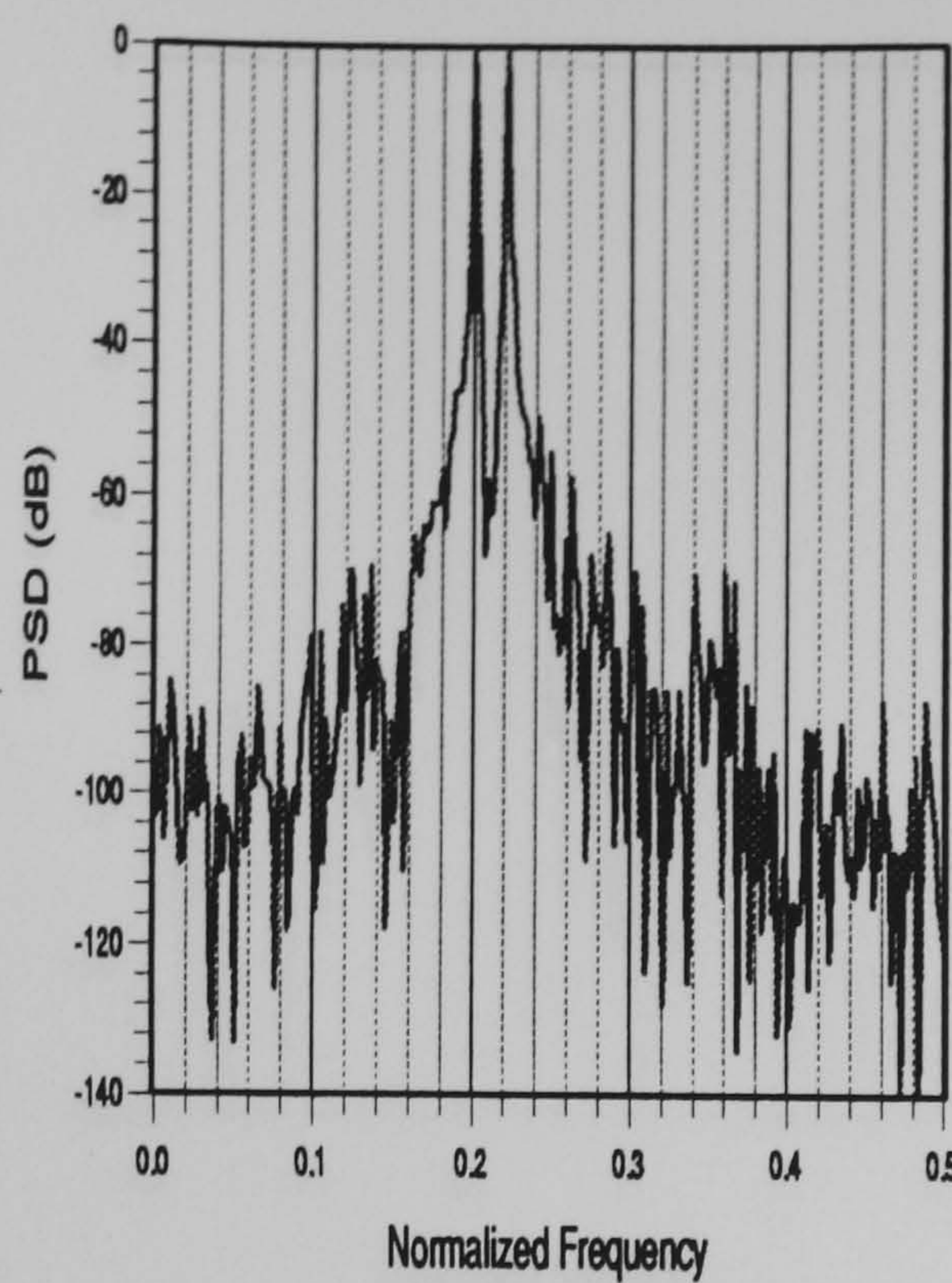
(c) PSD FROM APHD

$F1 = 0.2$
 $F2 = 0.22$
 $Amp\ 1 = 1.$
 $Amp\ 2 = 0.1$
 $IP = 20$
 $\mu = 0.002$
 $SNR = 20\ DB$



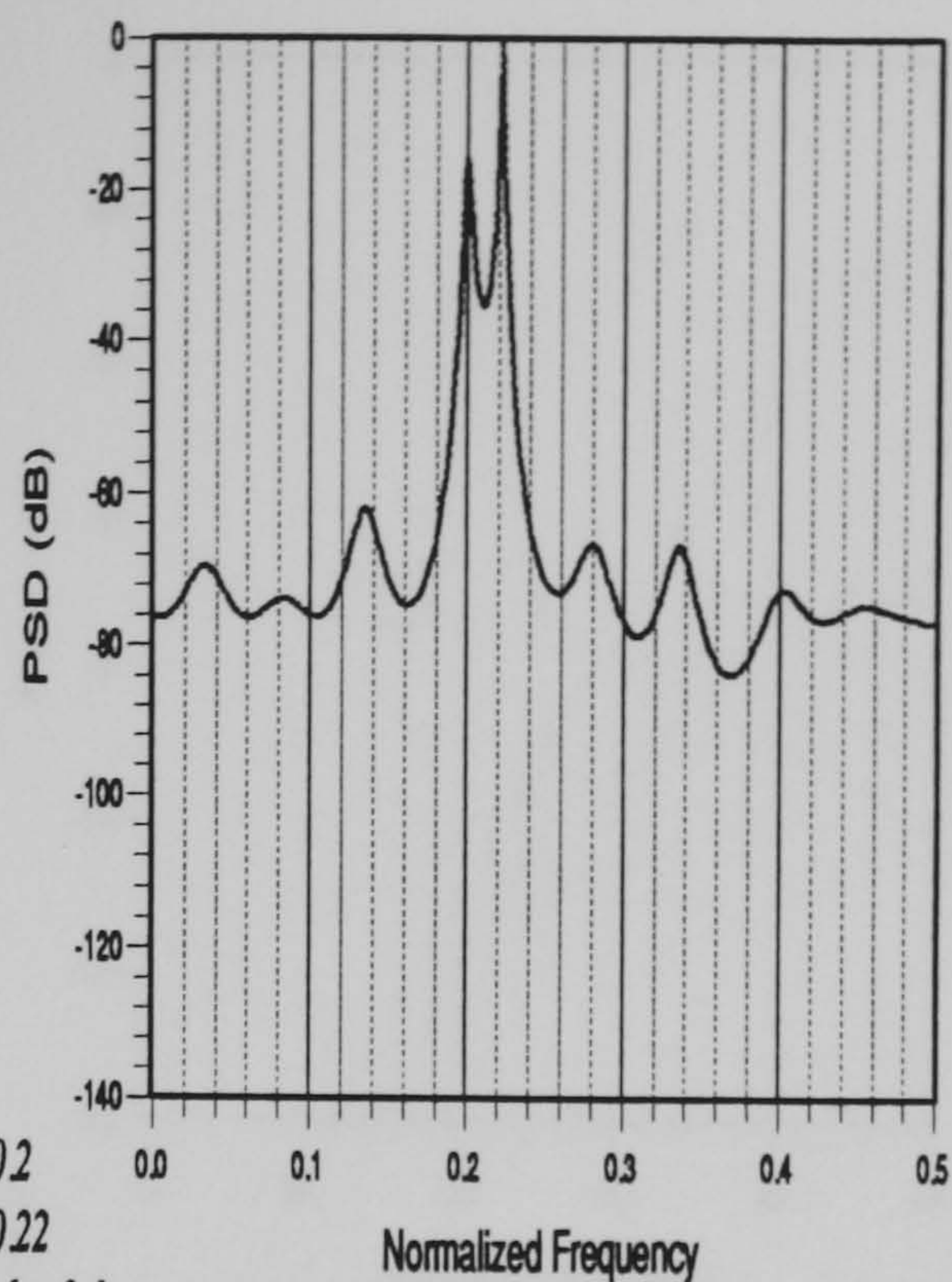
(d) PSD FROM AMAXEV

Figure 4.26: The frequency resolution test: relative amplitude = 0.1

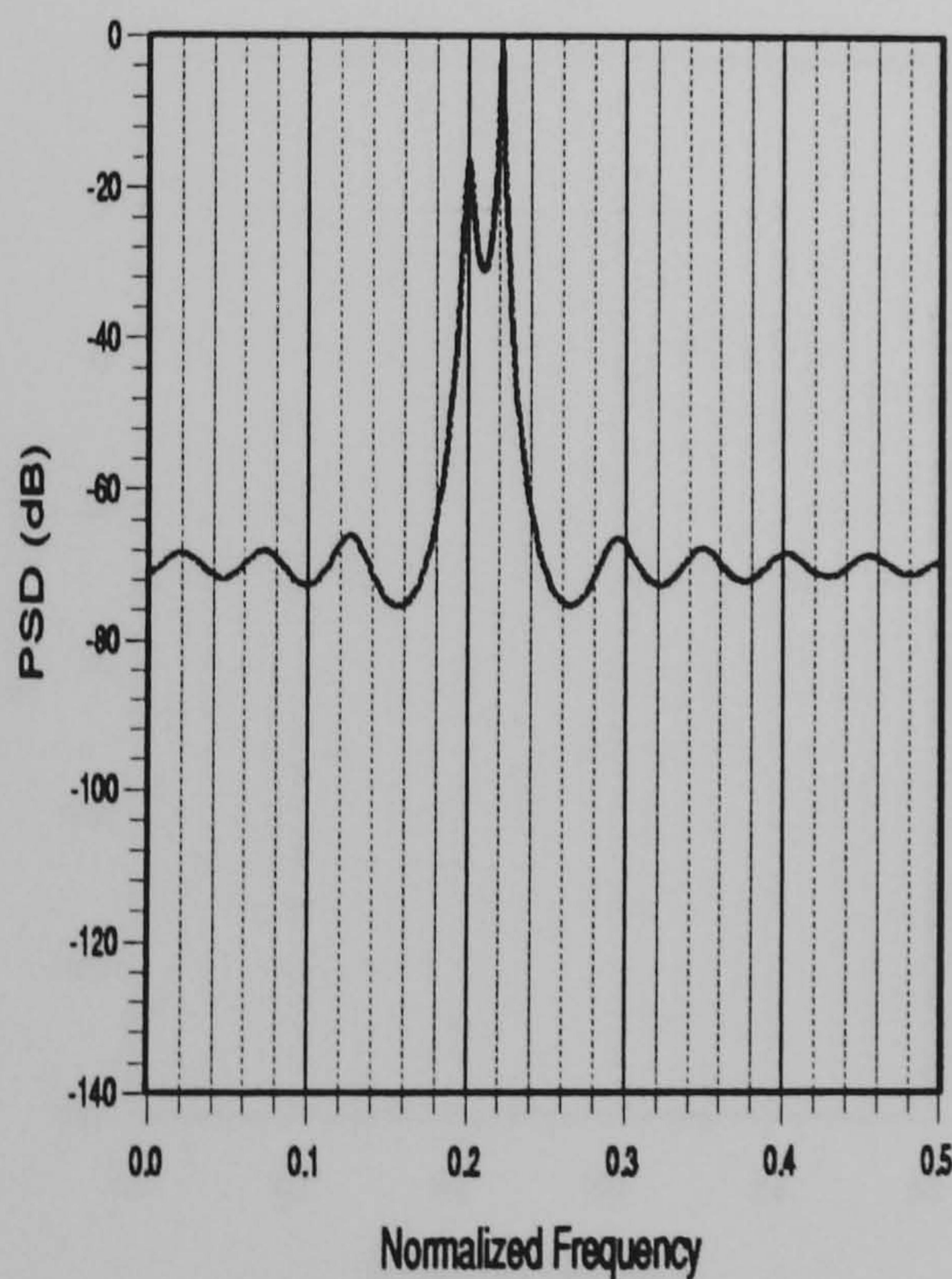


(a) PSD FROM ALEO

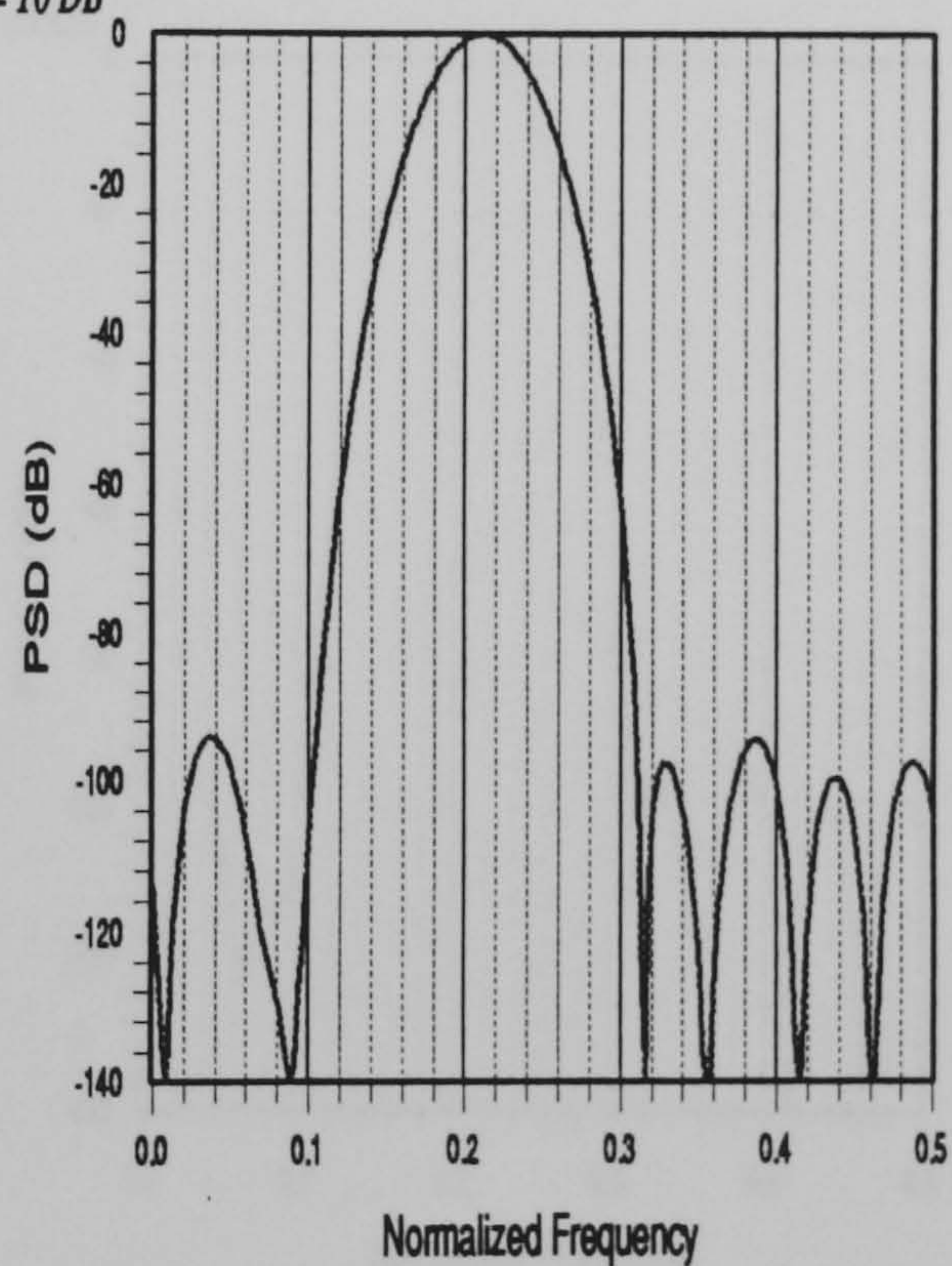
$F1 = 0.2$
 $F2 = 0.22$
 $Phase\ 1 = 0\ deg$
 $Phase\ 2 = 0\ deg$
 $IP = 20$
 $\mu = 0.001$
 $SNR = 10\ DB$



(b) PSD FROM ALEW

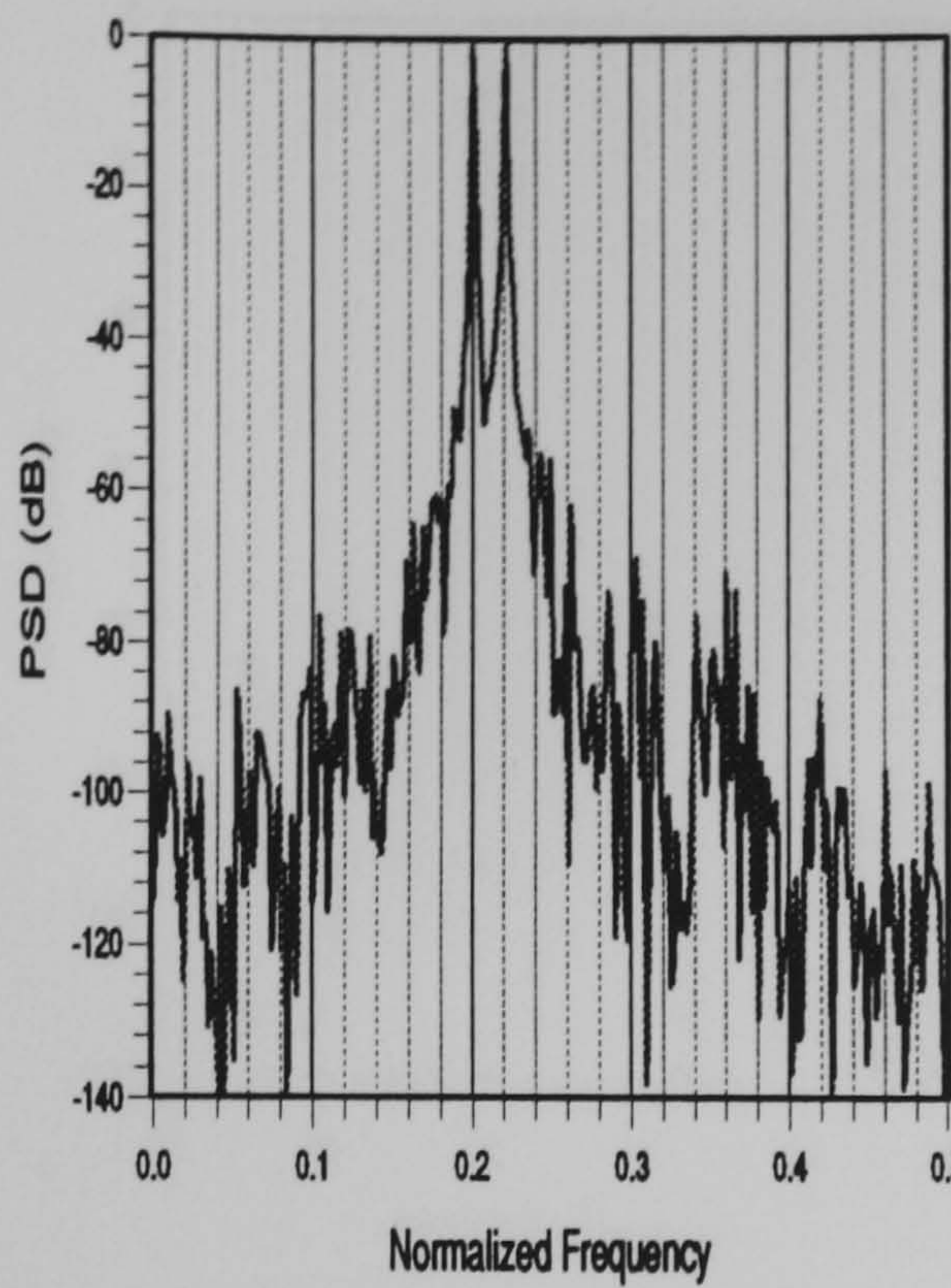


(c) PSD FROM APHD

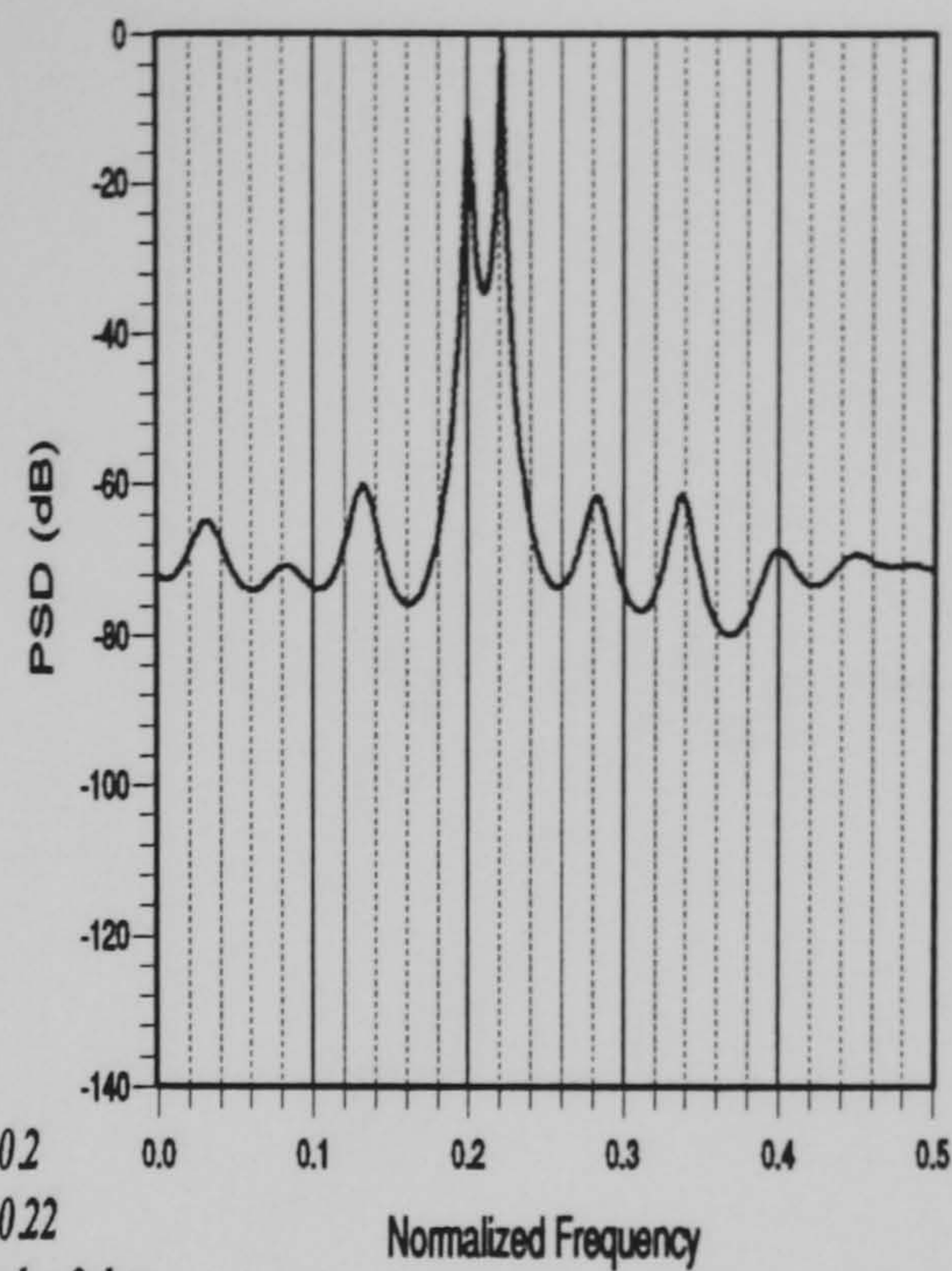


(d) PSD FROM AMAXEV

Figure 4.27: The frequency resolution test: relative phase = 0 degree

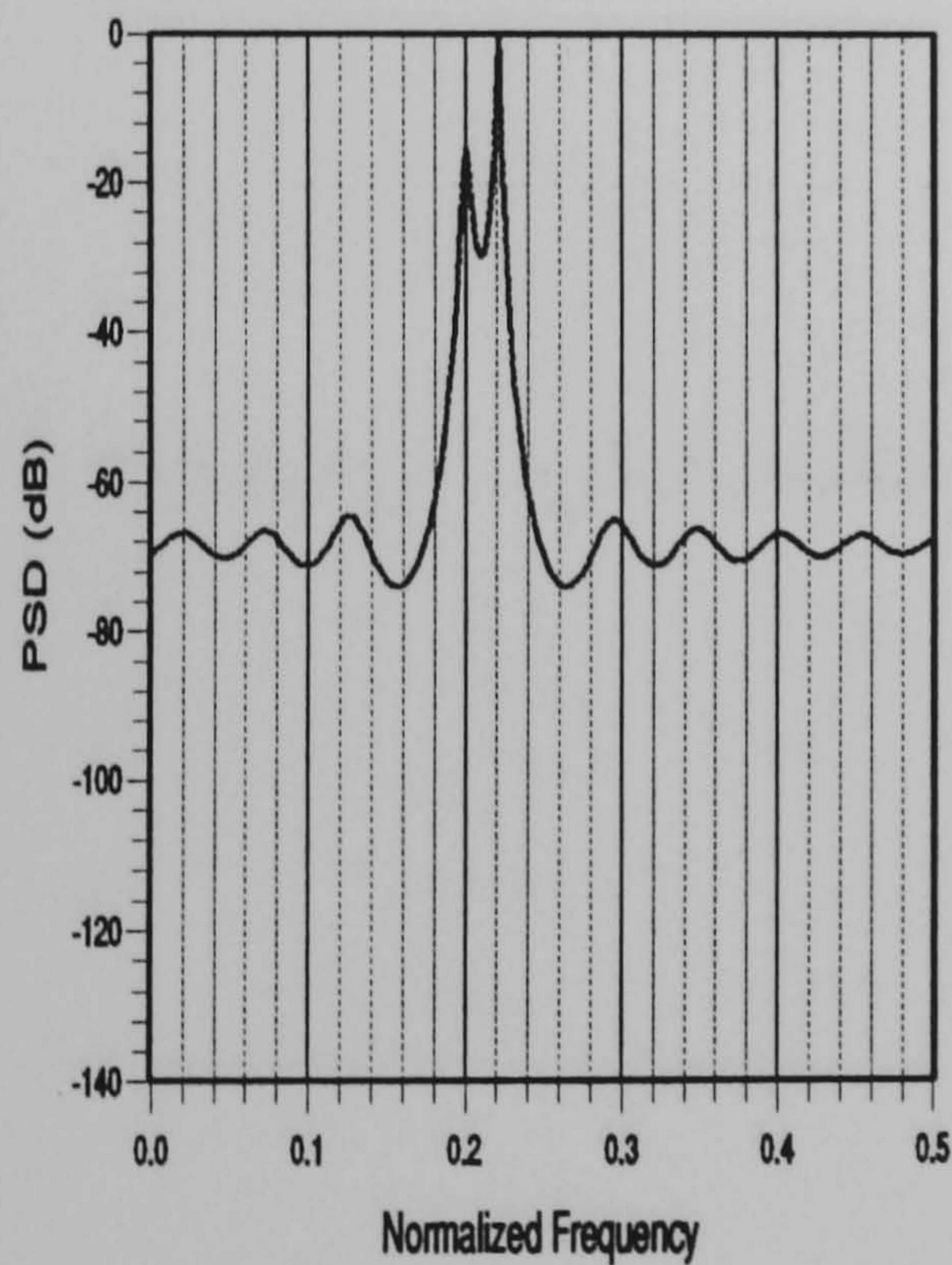


(a) PSD FROM ALEO

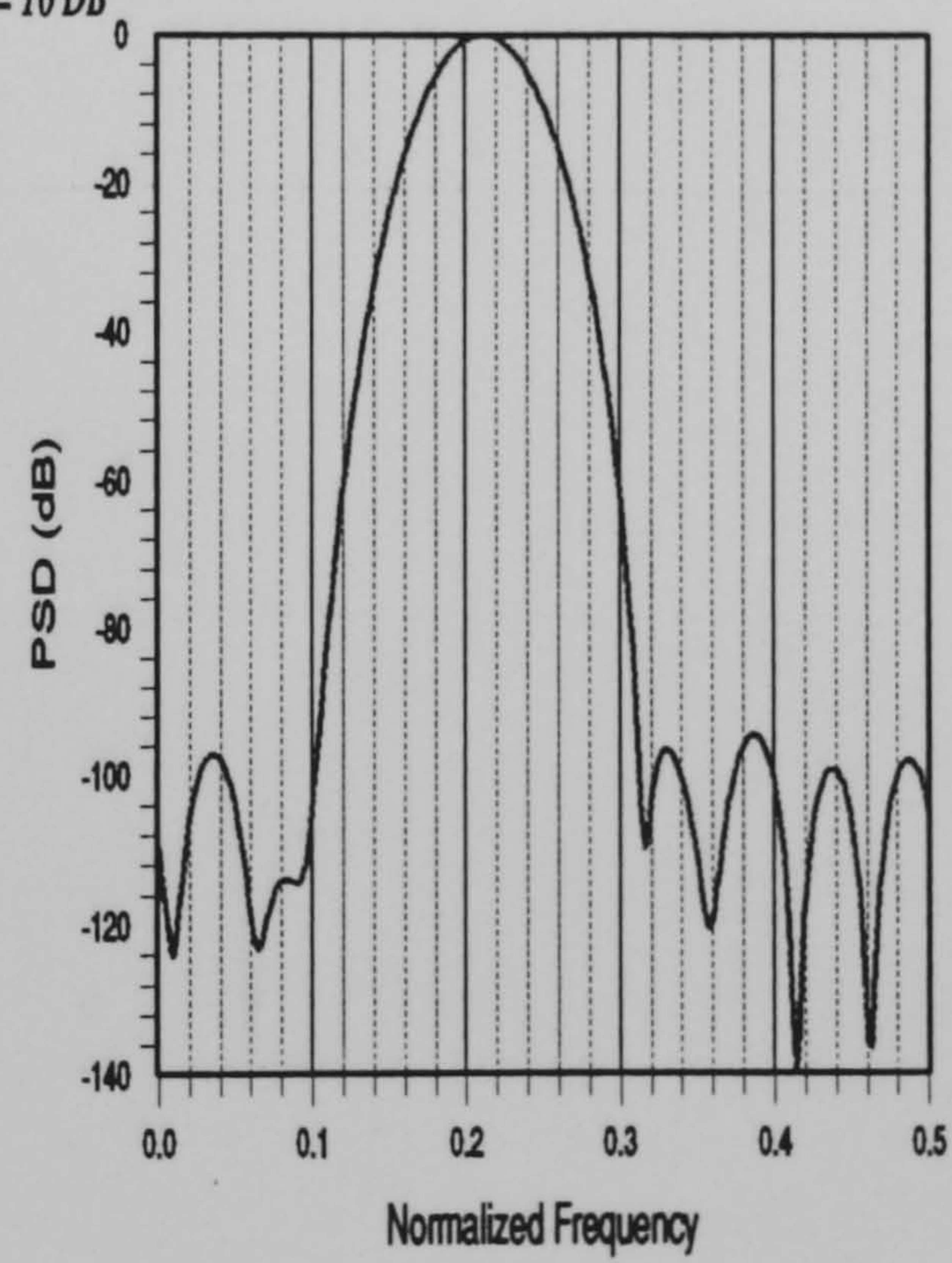


(b) PSD FROM ALEW

$F1 = 0.2$
 $F2 = 0.22$
 $Phase\ 1 = 0\ deg$
 $Phase\ 2 = 45\ deg$
 $IP = 20$
 $\mu = 0.001$
 $SNR = 10\ DB$

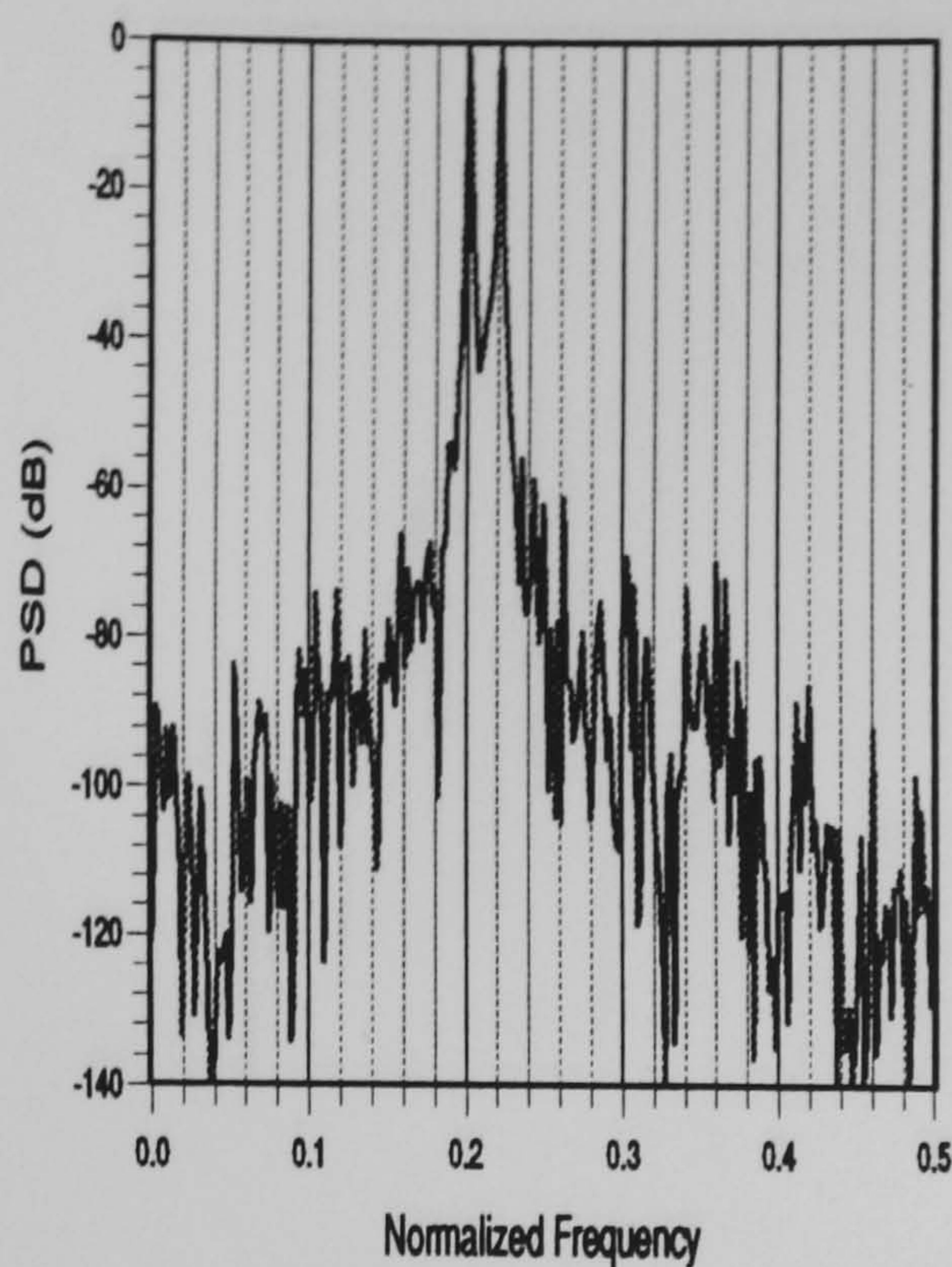


(c) PSD FROM APHD

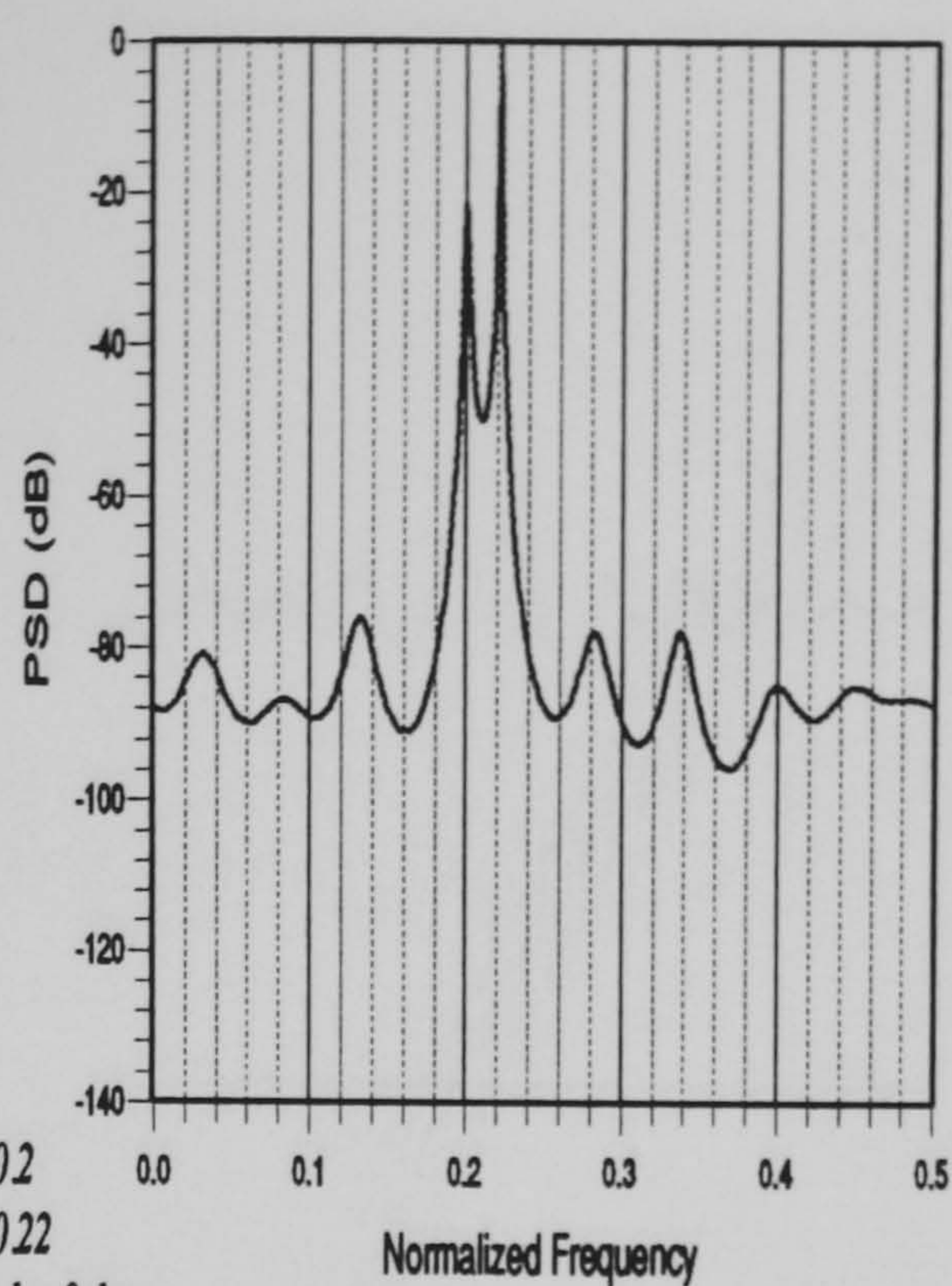


(d) PSD FROM AMAXEV

Figure 4.28: The frequency resolution test: relative phase = 45 degree

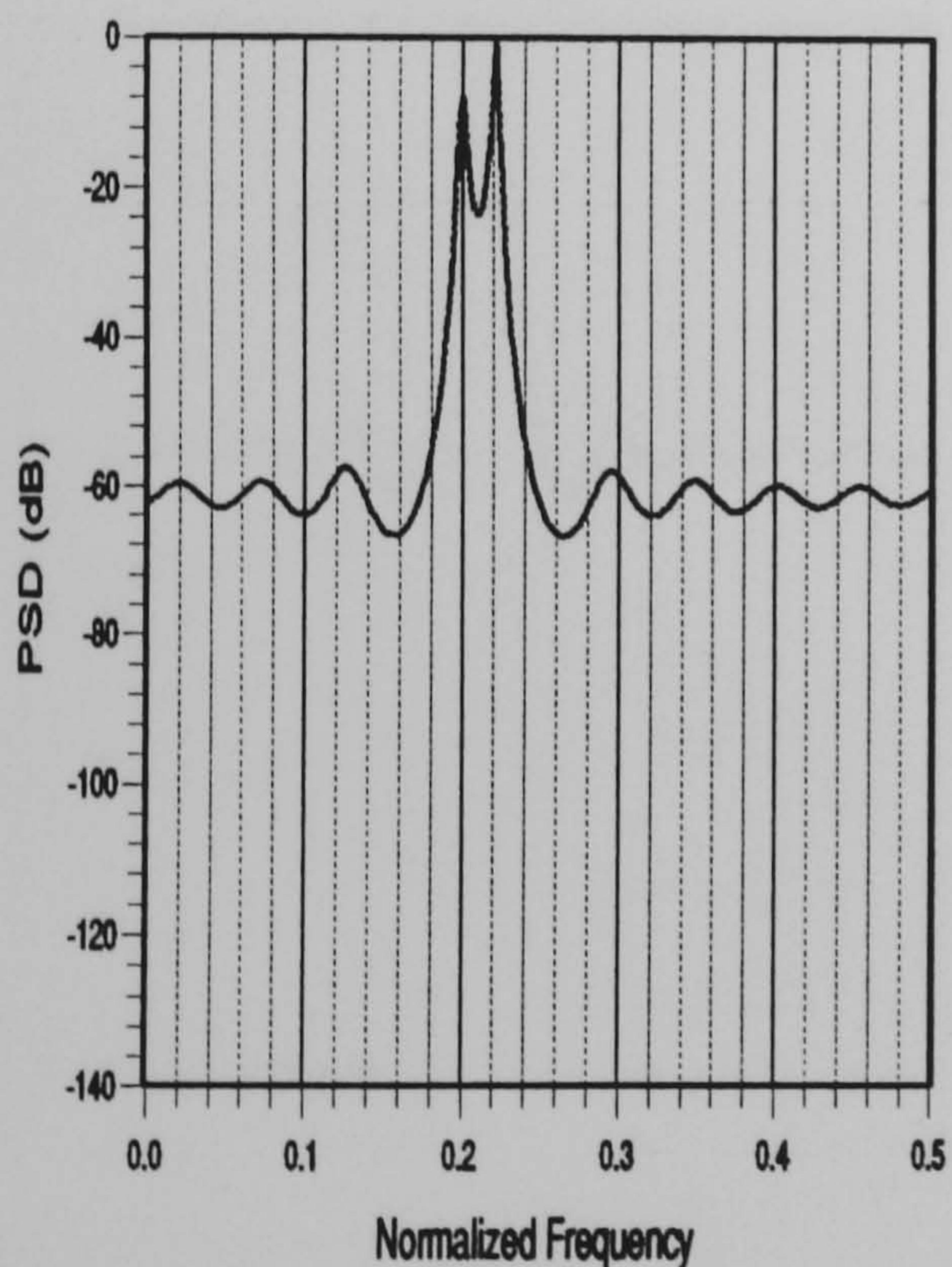


(a) PSD FROM ALEO

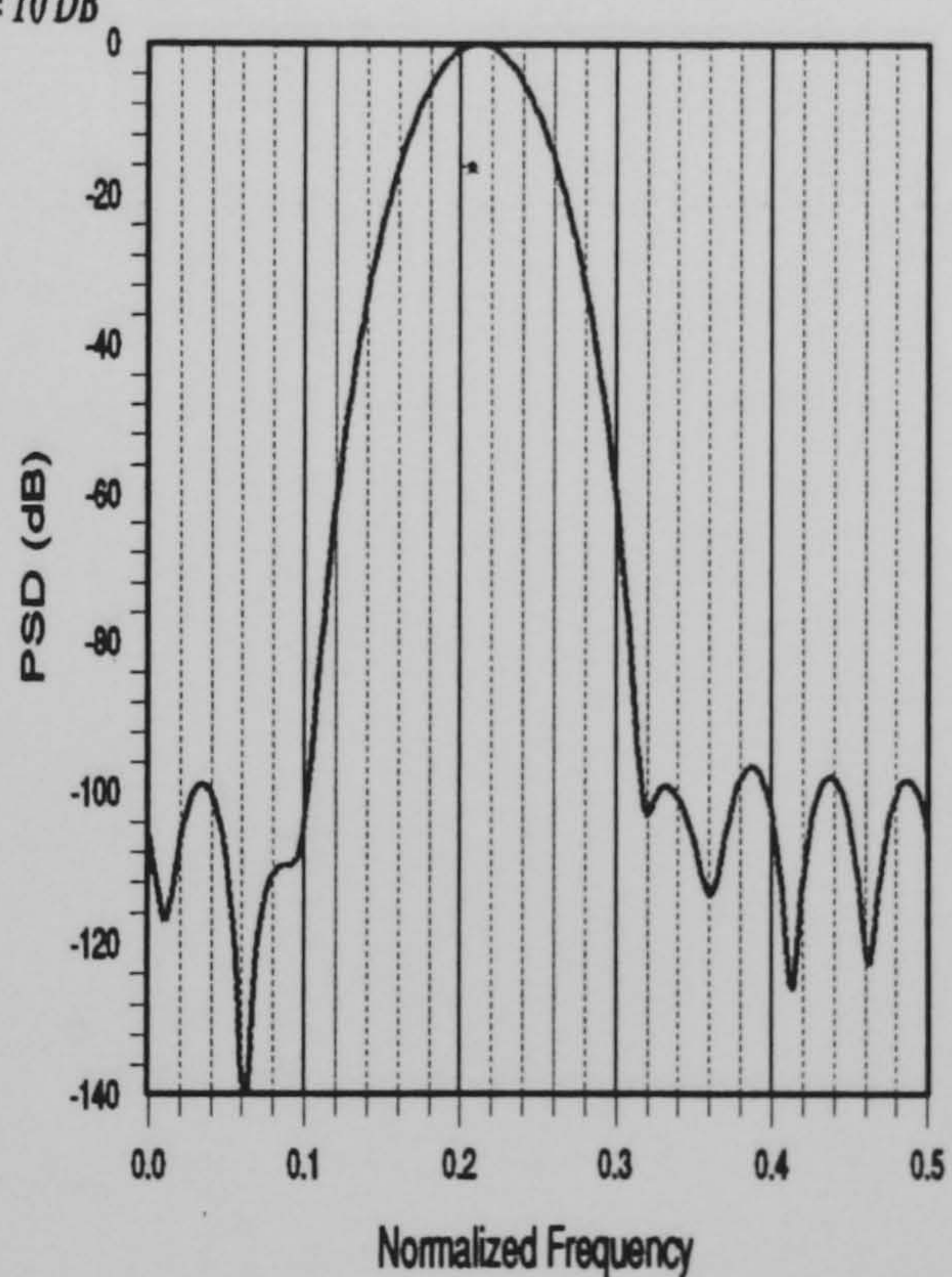


(b) PSD FROM ALEW

$F1 = 0.2$
 $F2 = 0.22$
 $\text{Phase } 1 = 0 \text{ deg}$
 $\text{Phase } 2 = 90 \text{ deg}$
 $IP = 20$
 $\mu = 0.001$
 $SNR = 10 \text{ DB}$

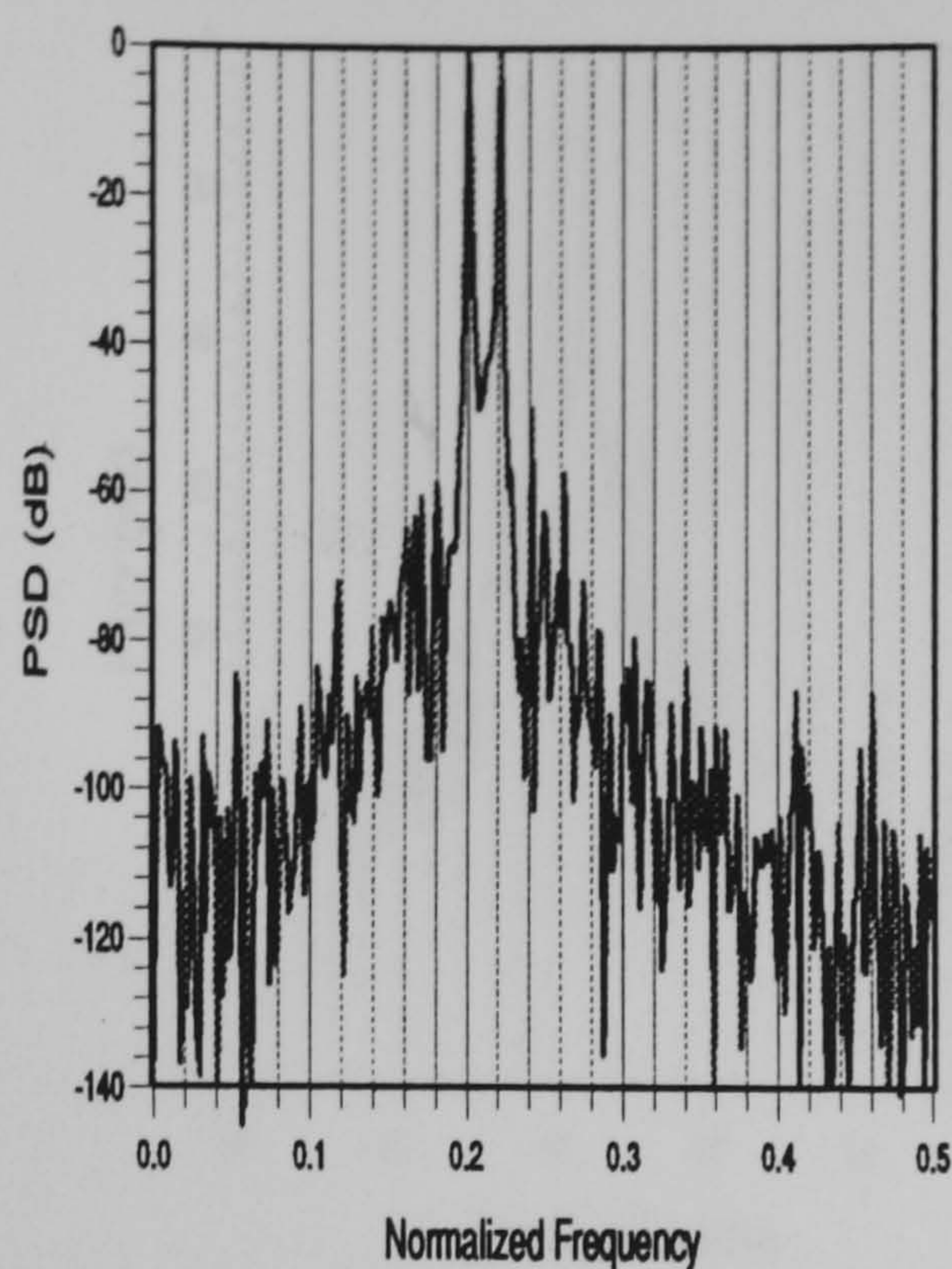


(c) PSD FROM APHD



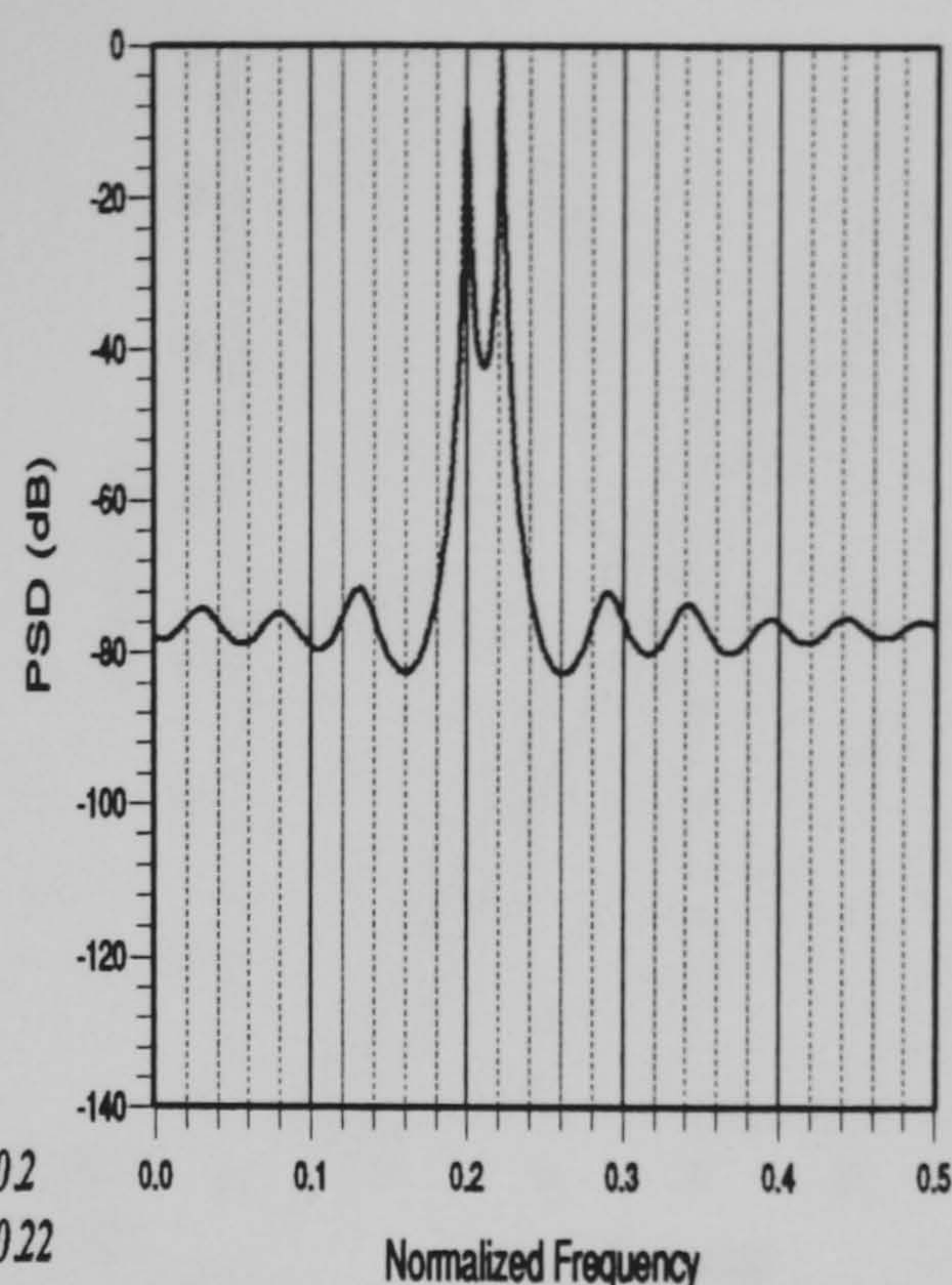
(d) PSD FROM AMAXEV

Figure 4.29: The frequency resolution test: relative phase = 90 degree

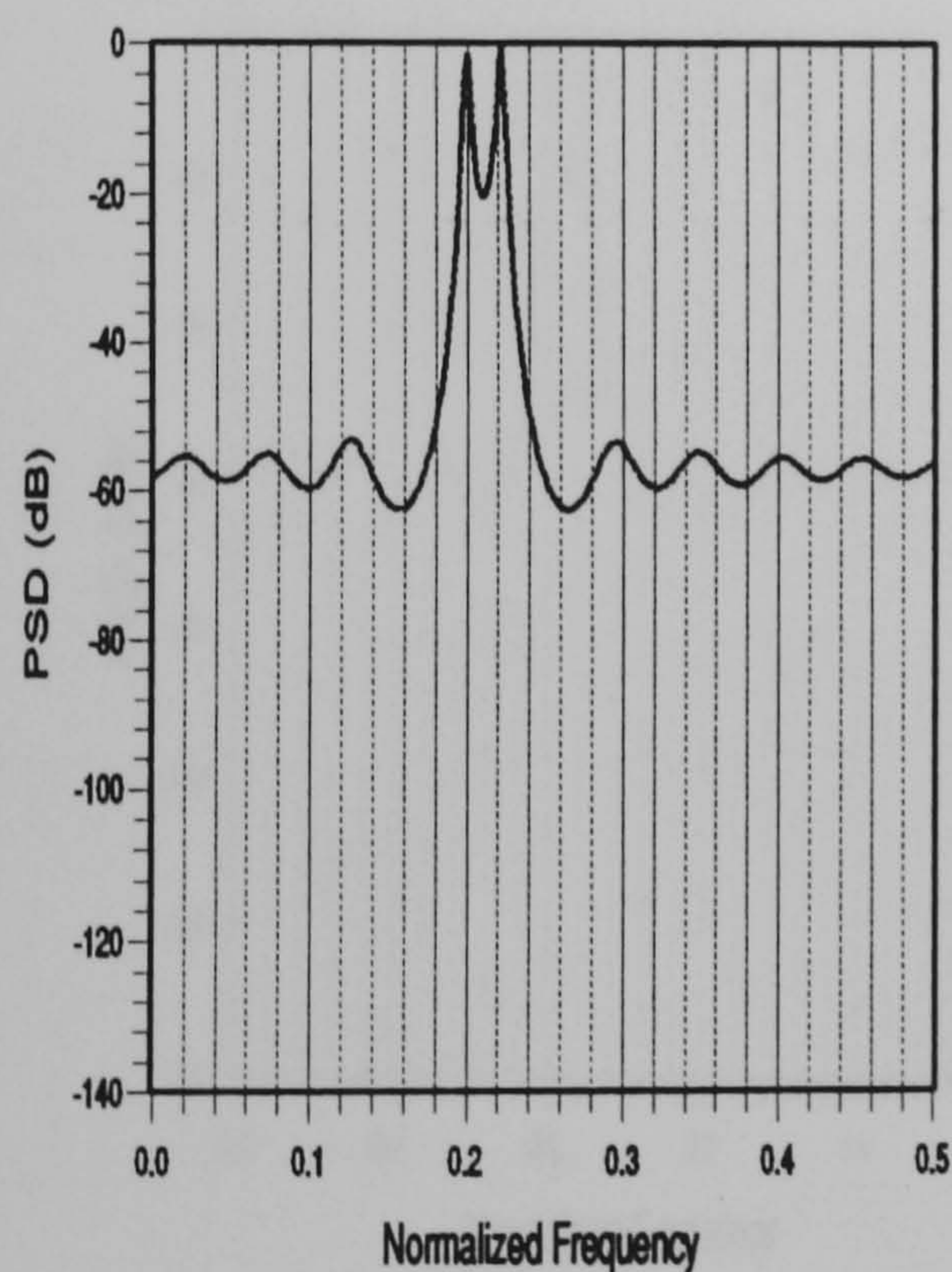


(a) PSD FROM ALEO

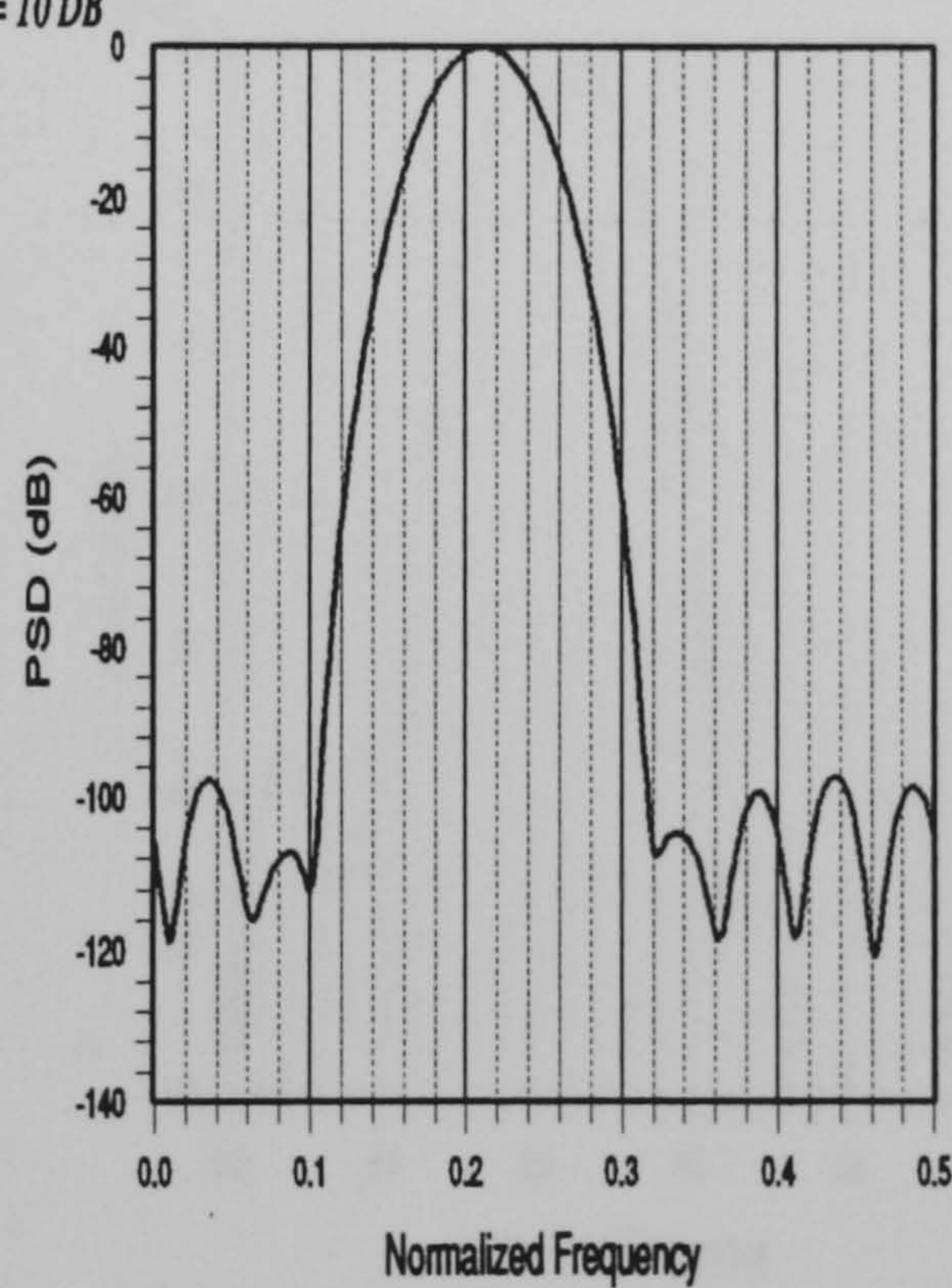
$F1 = 0.2$
 $F2 = 0.22$
 $\text{Phase } 1 = 0 \text{ deg}$
 $\text{Phase } 2 = 135 \text{ deg}$
 $IP = 20$
 $\mu = 0.001$
 $SNR = 10 \text{ DB}$



(b) PSD FROM ALEW

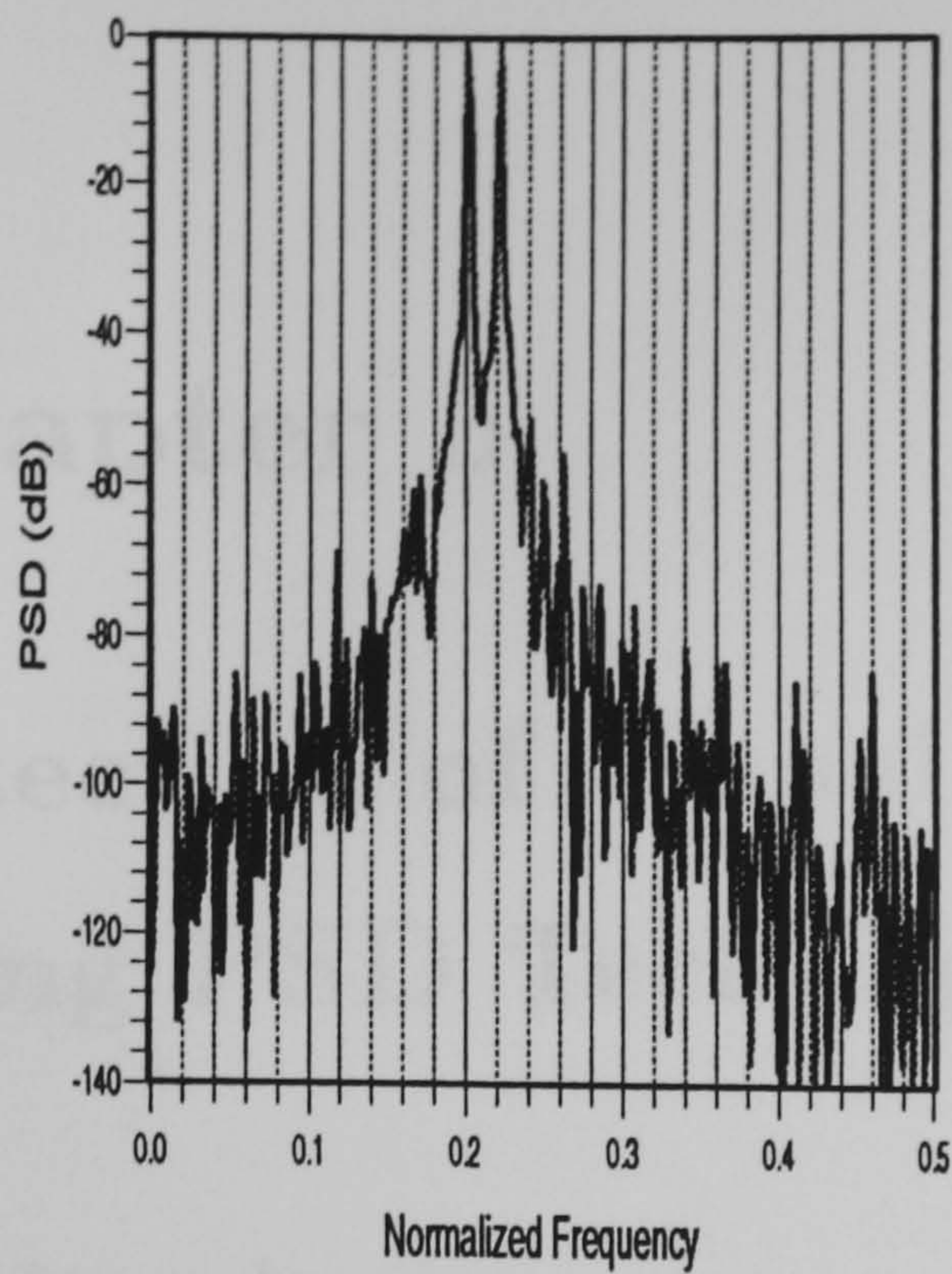


(c) PSD FROM APHD

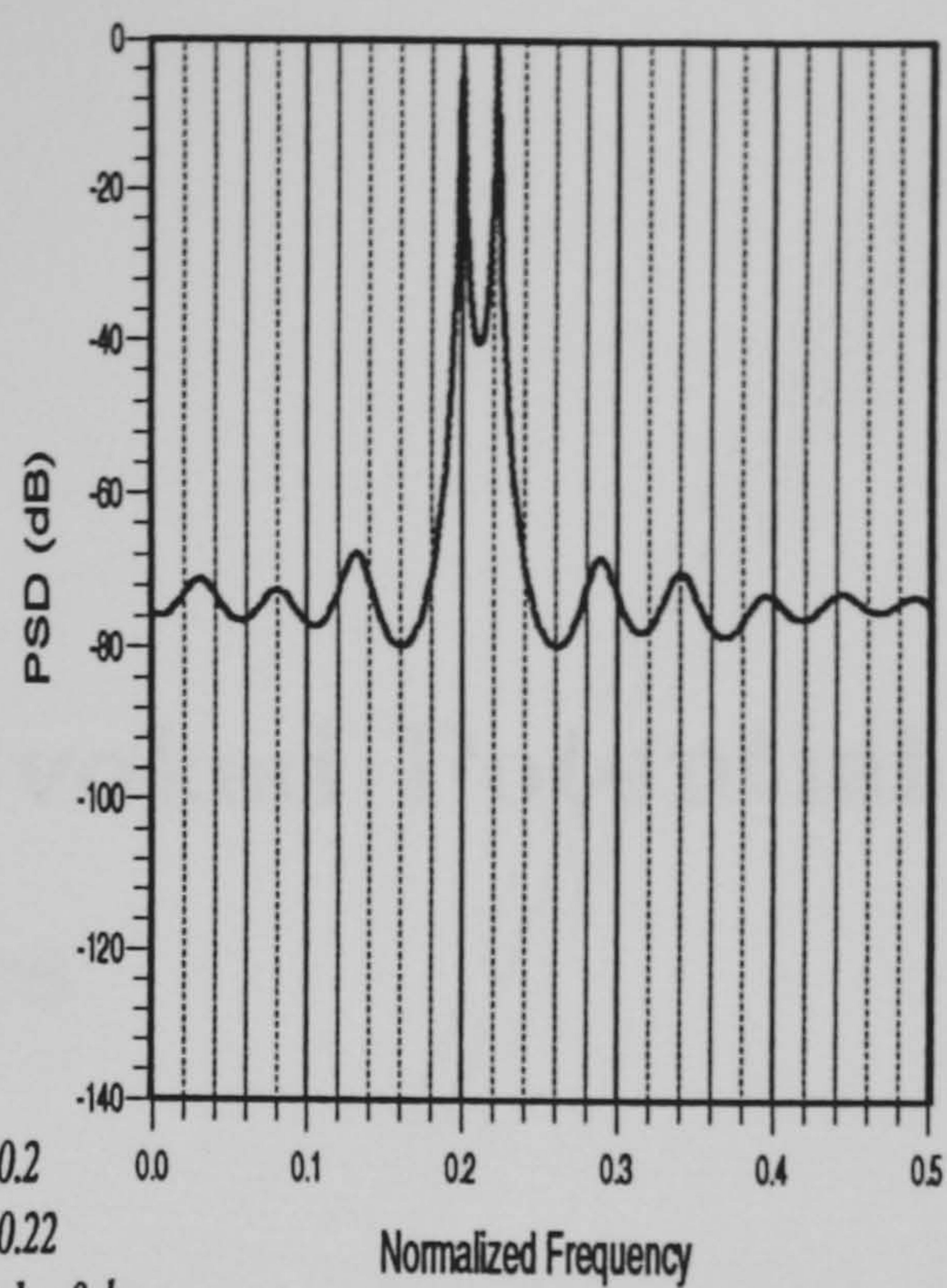


(d) PSD FROM AMAXEV

Figure 4.30: The frequency resolution test: relative phase = 135 degree

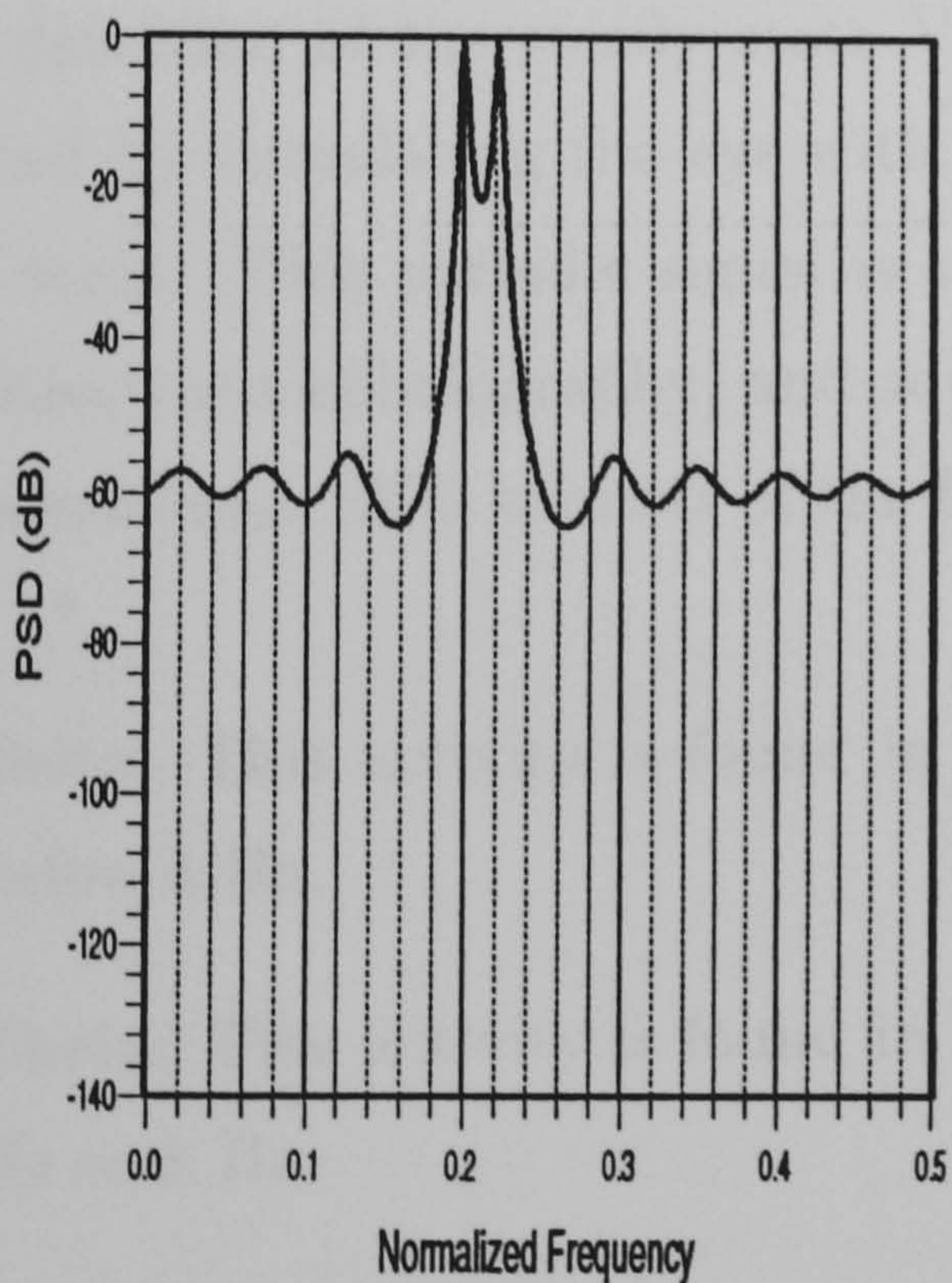


(a) PSD FROM ALEO

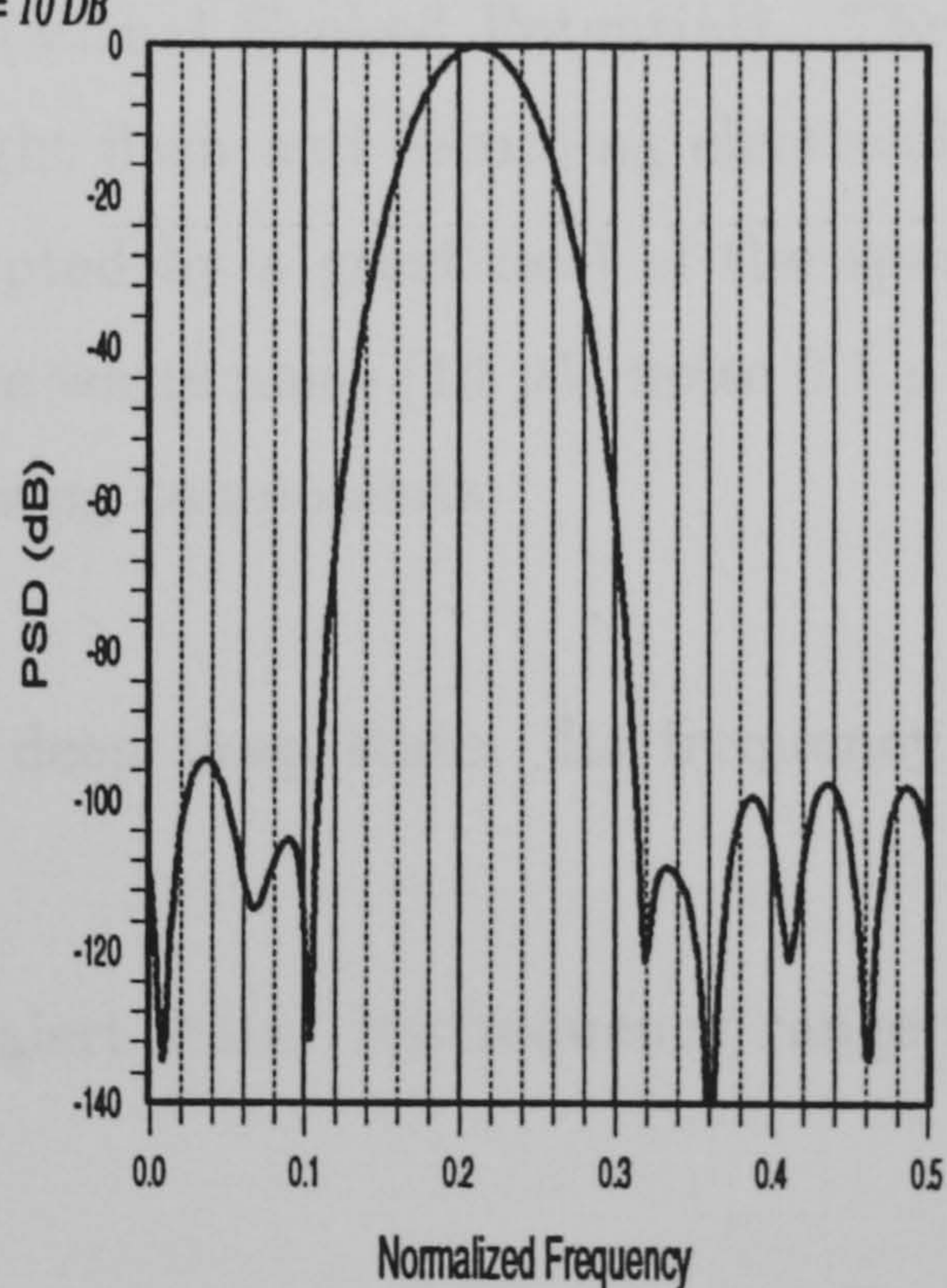


(b) PSD FROM ALEW

$F1 = 0.2$
 $F2 = 0.22$
 $\text{Phase 1} = 0 \text{ deg}$
 $\text{Phase 2} = 180 \text{ deg}$
 $IP = 20$
 $\mu = 0.001$
 $SNR = 10 \text{ DB}$



(c) PSD FROM APHD



(d) PSD FROM AMAXEV

Figure 4.31: The frequency resolution test: relative phase = 180 degree

Chapter 5

Detection of Visual Evoked Potential Using PSD Techniques

5.1 Introduction

Some of the previously investigated PSD estimation techniques have been applied to the detection of the steady state VEP (Visual Evoked Potential). The VEP is generated by stimulating the eye with a light flash and recording electrical activity on the scalp. This periodic signal is corrupted by a great deal of the spontaneous EEG (Electroencephalography) and additive white noise ($10 \mu V$ noise $0.1 \mu V$ VEP). The spontaneous EEG consists of the following components:

1. Delta: This activity is found in the deep sleep state. Its frequency range is below 4 Hz.
2. Theta: This activity is found in the alert state. Its frequency range is from 4 Hz to 8 Hz.
3. Alpha: This is an almost sinusoidal signal with frequency from 8 Hz to 13 Hz. In a stable and relaxed condition with eye closed the spontaneous EEG activity is usually dominated by alpha rhythm. This activity is usually abolished by opening the eyes.
4. Beta: This is a fast activity and is detectable in most normal EEG signals in the frequency range greater than 12 Hz.

When the eye is repeatedly stimulated by a flash, the brain will generate a signal with corresponding frequency. The magnitude of that voltage is very small. It can be about $\frac{1}{10}$ of the order of the spontaneous EEG activity which is in the range of $10 \sim 100\mu\text{V}$. Besides spontaneous EEG there are also additive white noise so the VEP is usually embedded in the unwanted signal and noise. This VEP signal can be used to monitor human performance, for example, pilot alertness. The VEP occupies the same frequency band as the spontaneous EEG but it is periodic. If we apply the Fourier transform to the measured data, the effect of the FT is to concentrate the energy of the periodic signal into a narrowband filter while spreading the energy of the additive white noise over a wide band of frequencies. It is much easier to detect the presence of the VEP in the frequency domain than in the time domain. Therefore the PSD technique is used for this purpose.

In this real-time PSD application the main issues are :

1. The detectability: Since the signal can be smaller than the noise the PSD estimator used must be able to cope with this difficult situation.
2. The speed of response: Since this is a real-time application, we would like to get the response as soon as possible. Thus the PSD estimator should use as few samples as possible and calculate the PSD as fast as possible.

The periodogram is used to estimate the PSD because it is robust and fast. The sampling rate is 200 Hz and 1024 points FFT is used to do the DFT. So it takes $\frac{1024}{200} = 5.12$ sec. to get one set of the PSD estimation. When the SNR is 0 dB, the spectrum will be $10 \log 1024 \approx 30$ dB above the noise floor. This is the advantage of doing the detection in the frequency domain. The resolution in this case is $\frac{200}{1024} \approx 0.2$ Hz which is adequate. The price paid for the detectability and frequency resolution by using periodogram method is the reaction time. The sequential periodogram using a cyclic input buffer as described in chapter 3 is used to give a smooth real-time PSD estimation and display. When the VEP is changing the PSD is gradually built-up, i. e. the sequential periodogram can track the variation of the VEP. However, it still needs 5.12 sec. to reach the steady state VEP representation. In order to reduce the

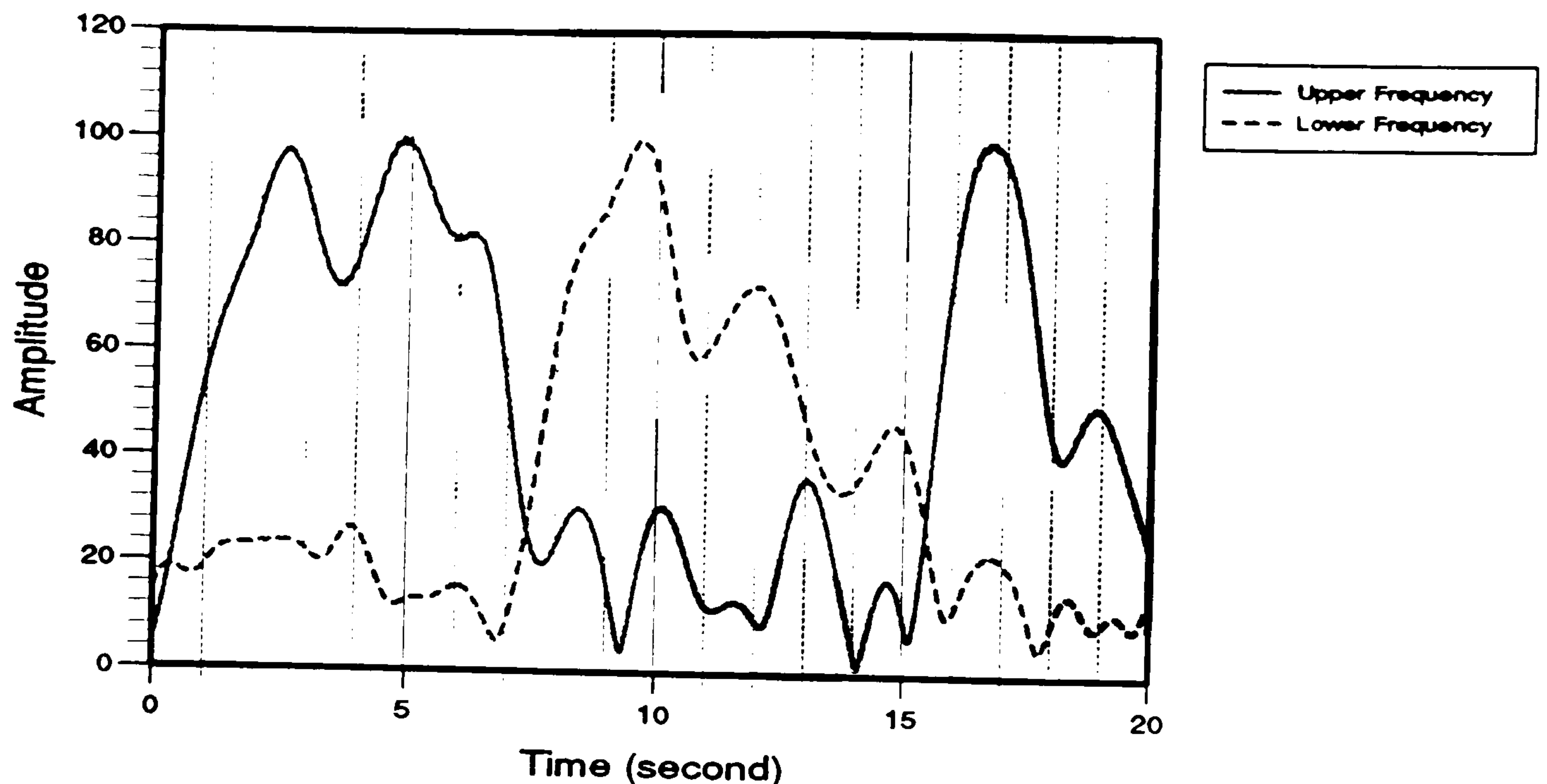


Figure 5.1: The VEP response of a pilot

number of samples needed for detecting the VEP. The MUSIC method is employed. If we preprocess the data to get rid of the spontaneous EEG then we have the *a priori* information that the number of signals is one. This prior information is essential for the MUSIC method.

In this chapter we compare the performance of the MUSIC and periodogram PSD estimation method when they are applied to the detection of VEP. We chose the MUSIC algorithm in particular for comparison because we are primarily interested in the frequency estimation of the periodic signal, and MUSIC is basically a frequency estimator. It will give a clean frequency domain picture. The Periodogram on the other hand has a noisy appearance.

5.2 The Description of Experimental Data

The EEG signals contain a great deal of information including some related to the processing, by the brain, of visual data. These signals are extremely small (a few microvolts) and are notoriously difficult to detect even in a benign and controlled environment. However, a new noninvasive technique has been developed in this research to enable the acquisition of these biopotential signals by the non-specialist, with a minimum of discomfort to the subject. Sophisticated signal processing algorithms, e. g. filtering, PSD estimation $\cdots$ etc. , have been developed which will extract the

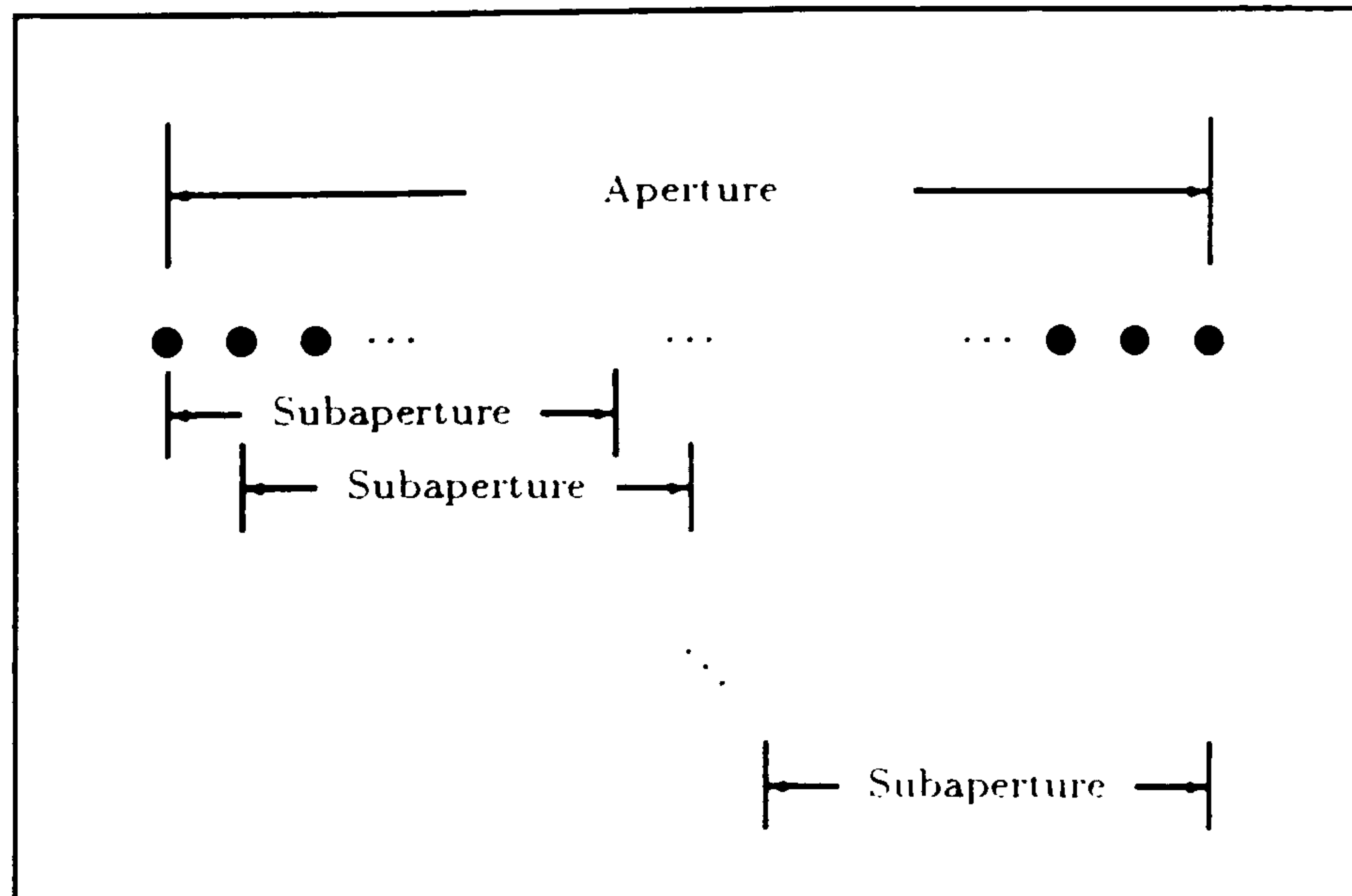


Figure 5.2: The partition of data to form matrix

required visual information from other neural activity, even in the presence of noise from environment. Figure 5.1 illustrates the results of applying the algorithm to a pilot/instrument interaction. In this case the pilot attends the instrument 1 (i.e. the upper frequency in the figure), then his attention moves to the instrument 2 (i.e. the lower frequency in the figure), then returns to instrument 1. The amplitude of the signal is an indication of the degree of attention. The sampling rate is 200 Hz. The VEP corresponding to instrument 1 and instrument 2 have frequency of 27 Hz (normalized frequency = $\frac{27}{200} = 0.135$) and 23 Hz (normalized frequency = $\frac{23}{200} = 0.115$) respectively.

5.3 The Snap Shot Implementation of MUSIC Algorithm

The number of samples collected in this experiment is 4096. In order to process the data in continuous fashion, the total record is broken into segments which are referred to as snap shots. The size of each segment is referred to as the aperture size. The data matrix is formed by further partitioning the aperture into subapertures as shown in figure 5.2. Let N be the aperture size, i. e. the number of samples per snap shot, and C be the subaperture size, i. e. the column numbers of the data matrix. In matrix form the data matrix is formed as (5.1). A program called VEP was written for the evaluation of the VEP signals. The flow chart is shown in figure 5.3 and the program is in appendix 1.

$$\mathbf{X} = \begin{bmatrix} x_C & x_{C-1} & \cdots & x_1 \\ x_{C+1} & x_C & \cdots & x_2 \\ \vdots & \vdots & \vdots & \vdots \\ x_2^* & x_3^* & \cdots & x_{C+1}^* \\ x_3^* & x_4^* & \cdots & x_{C+2}^* \\ \vdots & \vdots & \vdots & \vdots \\ x_{N-C+1}^* & x_{N-C+2}^* & \cdots & x_N^* \end{bmatrix} \quad (5.1)$$

5.4 Comparison of Periodogram and MUSIC Frequency Tracking Performance

Because the VEP was deeply embedded in the noise, the periodogram needs a long record to detect its presence. While the MUSIC method being blessed by the priori knowledge of number of signals needs less data. The 32 point periodograms of the VEP signal are shown in figure 5.4. It can be seen that it can not resolve the two VEP frequencies. The 3D spectrogram, figure 5.5, of MUSIC clearly tracks the variation of the frequencies of the VEPs. It starts at upper frequency (0.135) and then shifts to lower frequency VEP (0.115) and back to upper frequency (0.135) again.

5.5 Some Comments

The MUSIC algorithm needs the information of a number of signals. In this application it is one. So it will avoid the spurious peaks if we preprocess the data by passing the raw VEP through a band-pass filter. Occasionally there will be harmonics of the VEP signals. Since the eigenvalues corresponding to these frequencies are smaller than the one associated with the fundamental frequency, MUSIC will peak out the fundamental frequency which is what we want. The problem to implement the MUSIC algorithm in real-time is that it is computationally intensive. However, faster DSP chips have been designed, as will be mentioned in chapter 7, and parallel algorithms can be employed to speed up the computation.

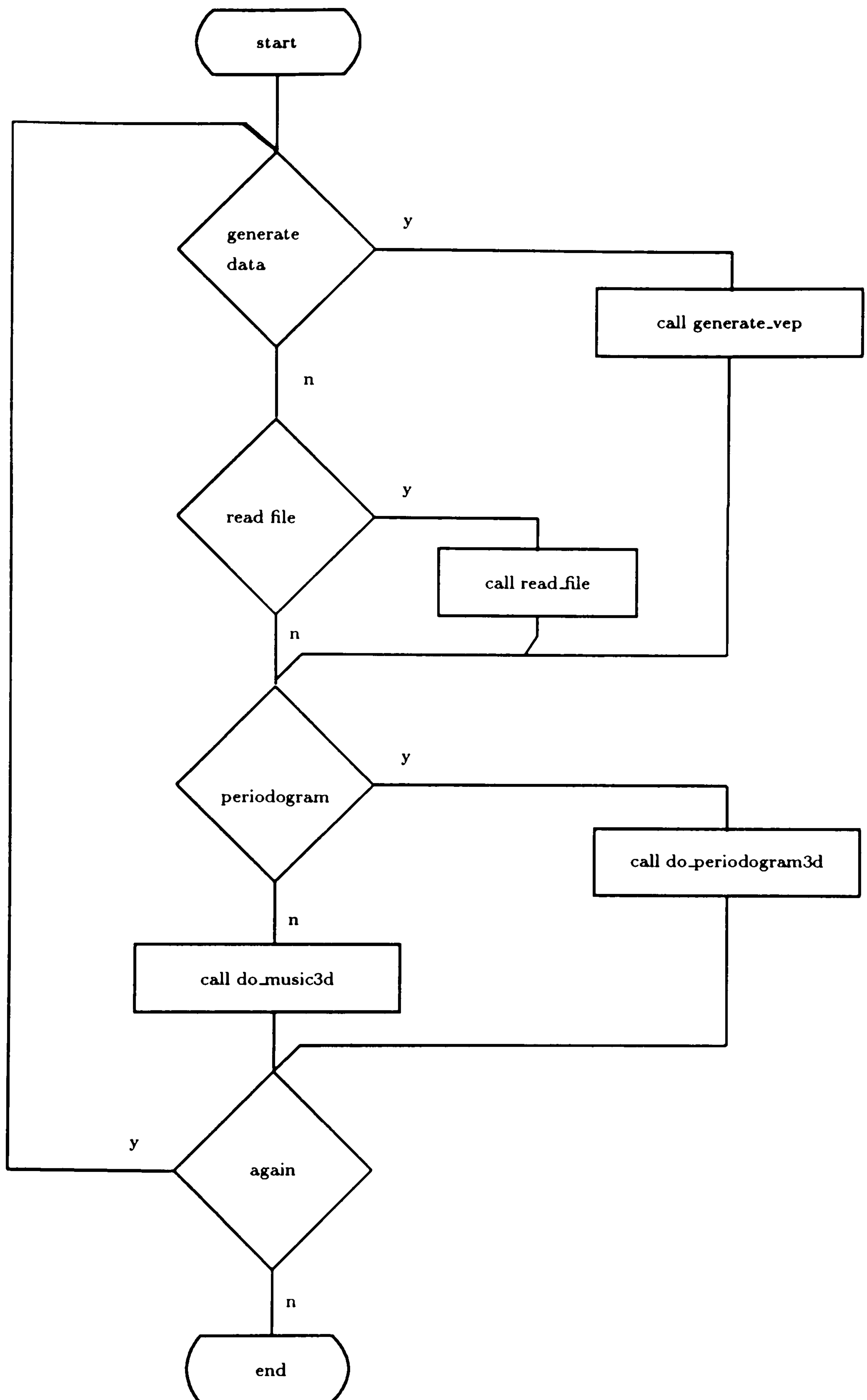
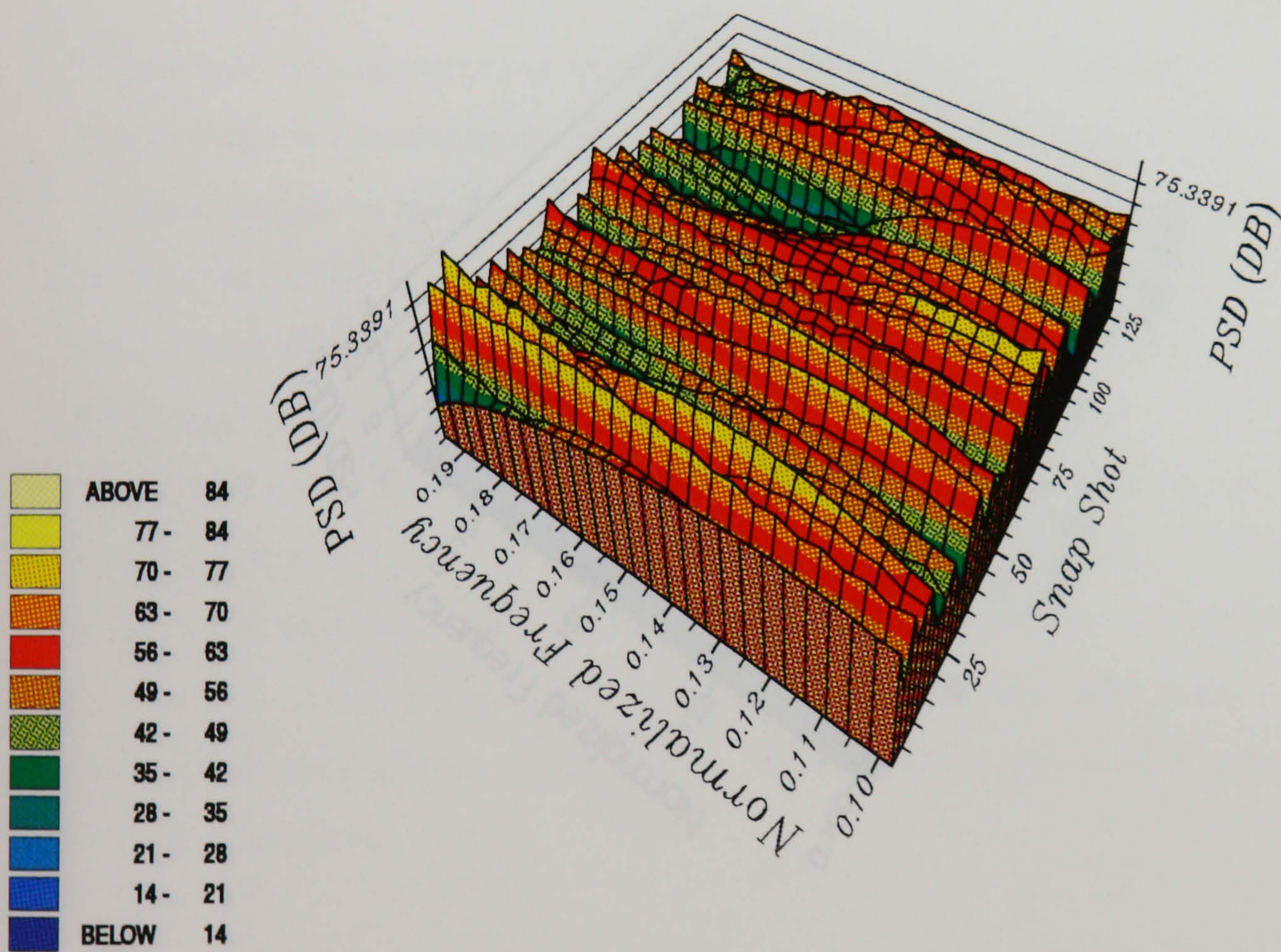


Figure 5.3: The flow chart of program VEP

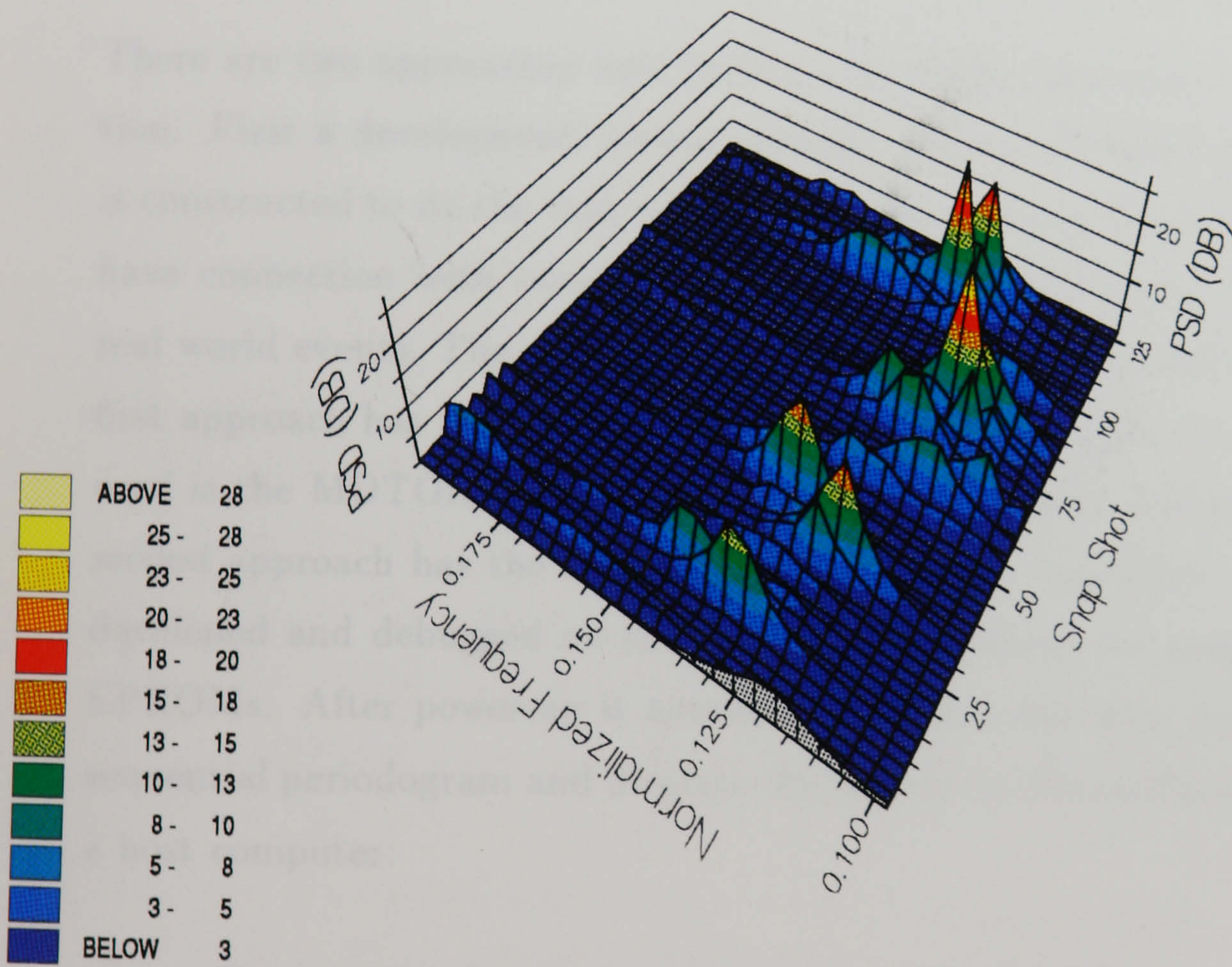
Figure 5.4: The 3D Spectrogram Display of Periodogram Method



Chapter 6

The Real-Time MUSIC Method Estimators

Figure 5.5: The 3D Spectrogram Display of MUSIC Method



Chapter 6

The Real-Time Implementation of PSD Estimators

6.1 Introduction

There are two approaches used in this research to implement real time PSD estimation. First a development system is used. Second a digital signal processing board is constructed to do the real-time PSDE. They are real-time processing because they have connection with outside world via A/D and D/A and it can keep pace with real world events. The digital signal processor used is the MOTOROLA 56001. The first approach has the advantage of fast development time. The development system used is the MOTOROLA DSP56000 ADS (Application Development System). The second approach has the advantage of stand-alone capability. When the program is developed and debugged on the development system, the program is transferred to EPROMs. After power-up it automatically takes the data from A/D, executes the sequential periodogram and displays the output on the oscilloscope. It does not need a host computer.

6.2 The Development System Approach

The DSP56000 ADS consists of an Application Development Module (ADM), a Host-Bus Interface Board, and a software program. The ADM contains a DSP56001 processor and control circuitry. The software program interacts with the user and controls the ADM. The DSP56001 ADS has $2K \times 24$ bits of monitor EPROM and $8K \times 24$ bits of static RAM on the DSP56001 bus. These RAMs are partitioned using a full

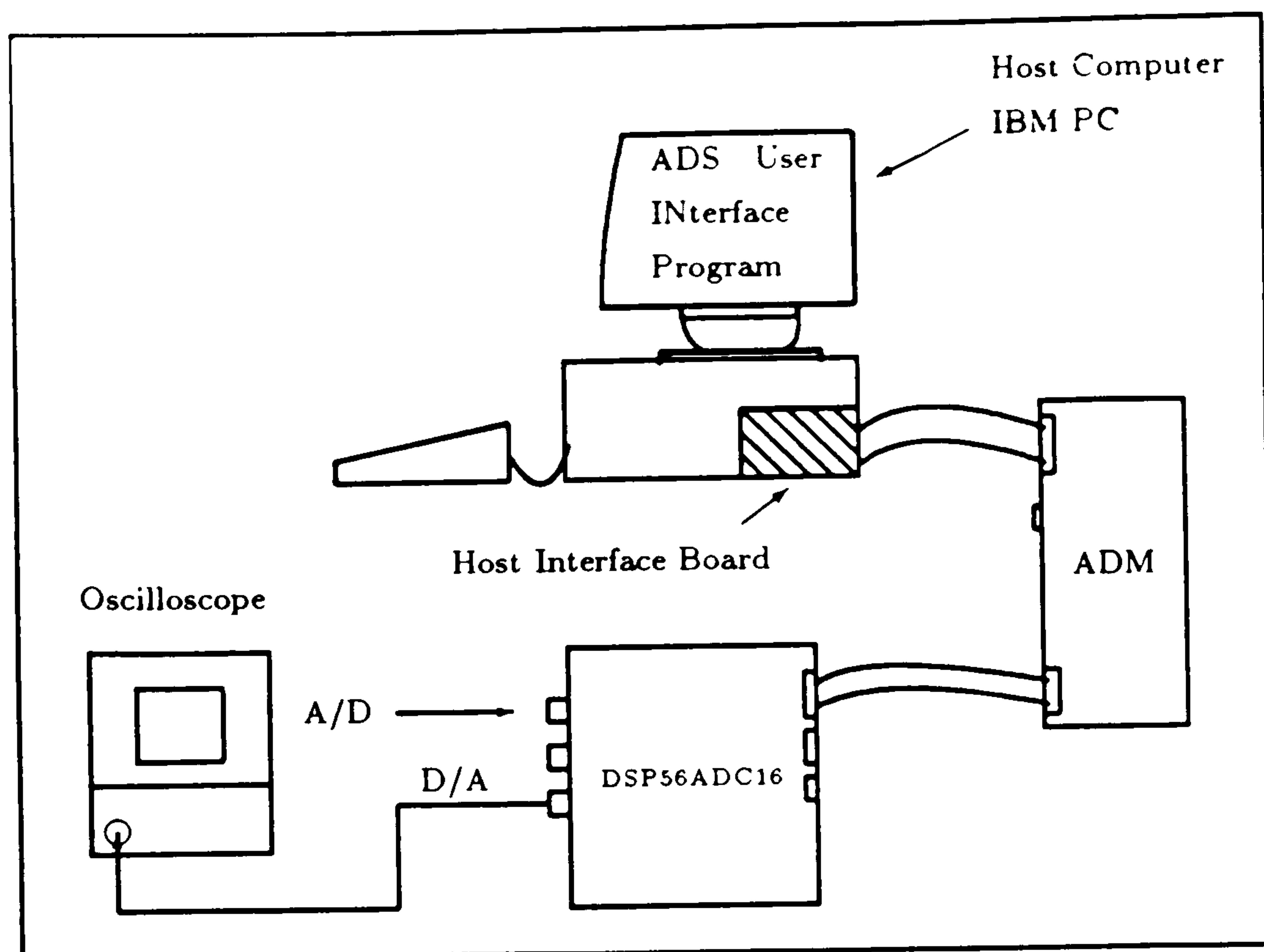


Figure 6.1: The DSP56000 ADS

decoding scheme implemented by jumpers and PAL (Programmable Array Logic) to allow the user to configure the RAM in all three of the memory maps. There are 24 commands available in four basic categories: memory, register modification, file I/O, ADM execution control and miscellaneous tasks.

The DSP56ADC16 Evaluation Board (EVB) is a complete A/D and D/A conversion system for either stand-alone operation or use in conjunction with the DSP56000 ADS. In this case the DSP56ADC16 A/D part is interfaced directly to the SSI (Synchronous Serial Interface) receiver port of the DSP56001 chip on the DSP56000 ADS. The D/A part of the DSP56ADC16 is connected to the DSP56001 SSI transmitter port. This allows analog signals to be converted to digital form, received and processed by the DSP56001 using DSP algorithms, and then sent back to the D/A converter. This completes a programmable data acquisition and processing system. The output of the D/A is connected to the oscilloscope for displaying in real-time as shown in figure 6.1. Additionally the data can be logged and analyzed off-line.

6.2.1 The Real-Time Sequential Periodogram Implemented on The Development System

A sequential periodogram is implemented on the development system to see the VEP response signal in real-time. The flow chart is shown in figure 6.2. There are two

input data buffers. The main buffer is 512 points long. This is the input buffer that the FFT is operating on. The temporary buffer is 8 samples long . When 8 samples are accumulated they are copied to the main buffer (which is a cyclic buffer) and the FFT is triggered to process the data. The program is in appendix B.

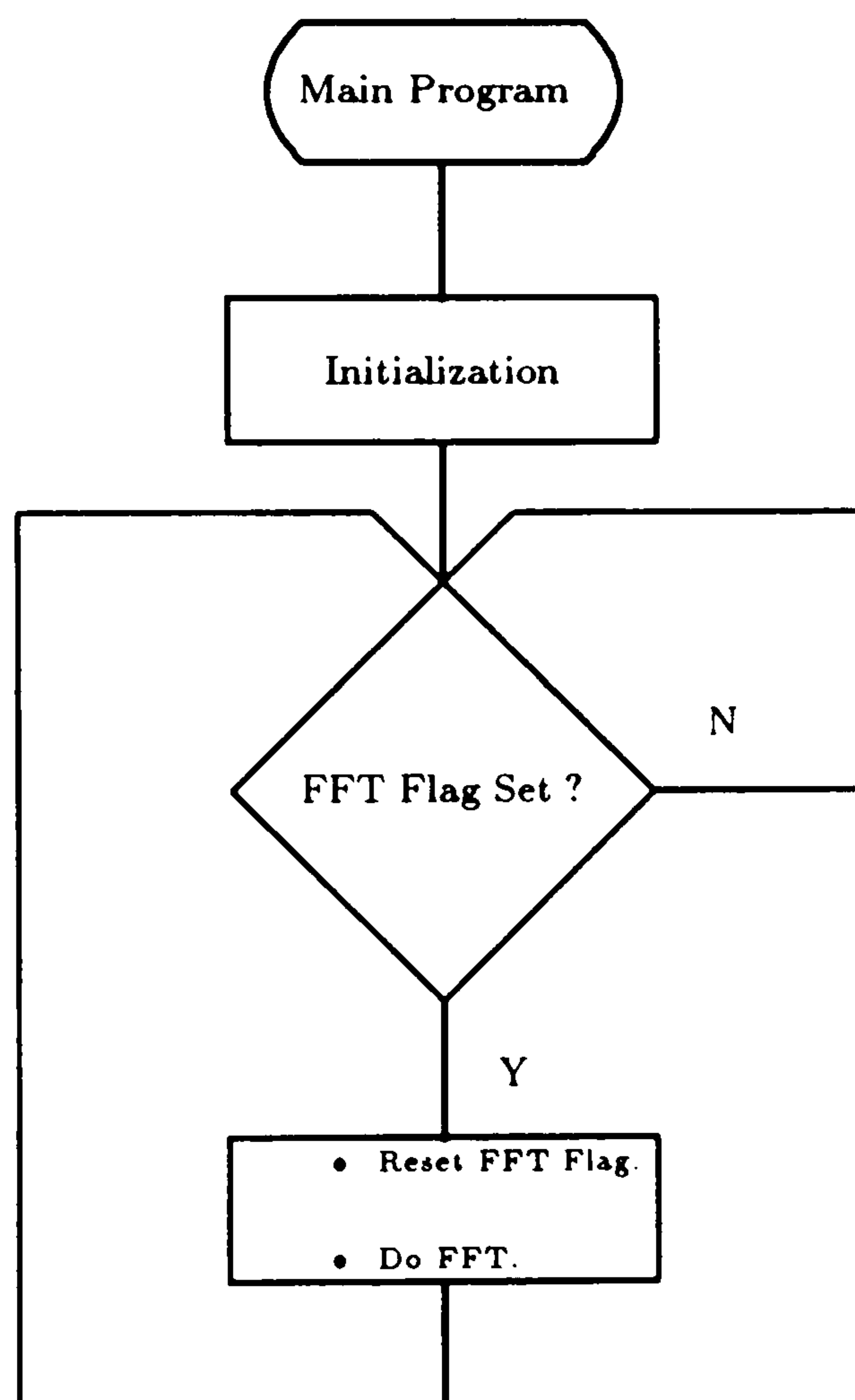


Figure 6.2: Flow chart of the main program of the sequential periodogram implemented on DSP56000ADS

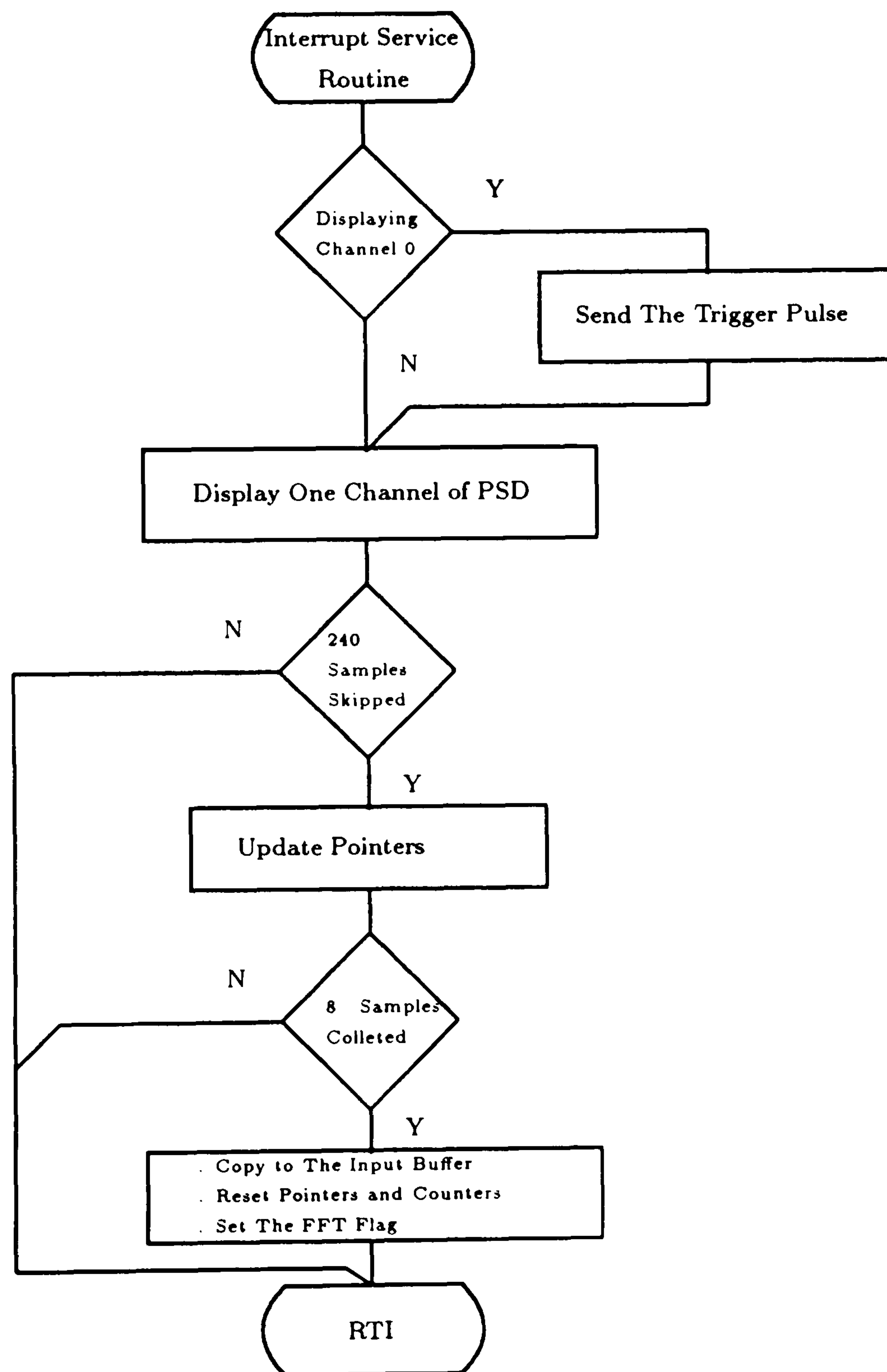


Figure 6.2: Flow chart of the interrupt service subroutine of the sequential periodogram Implemented on DSP56000ADS (cont.)

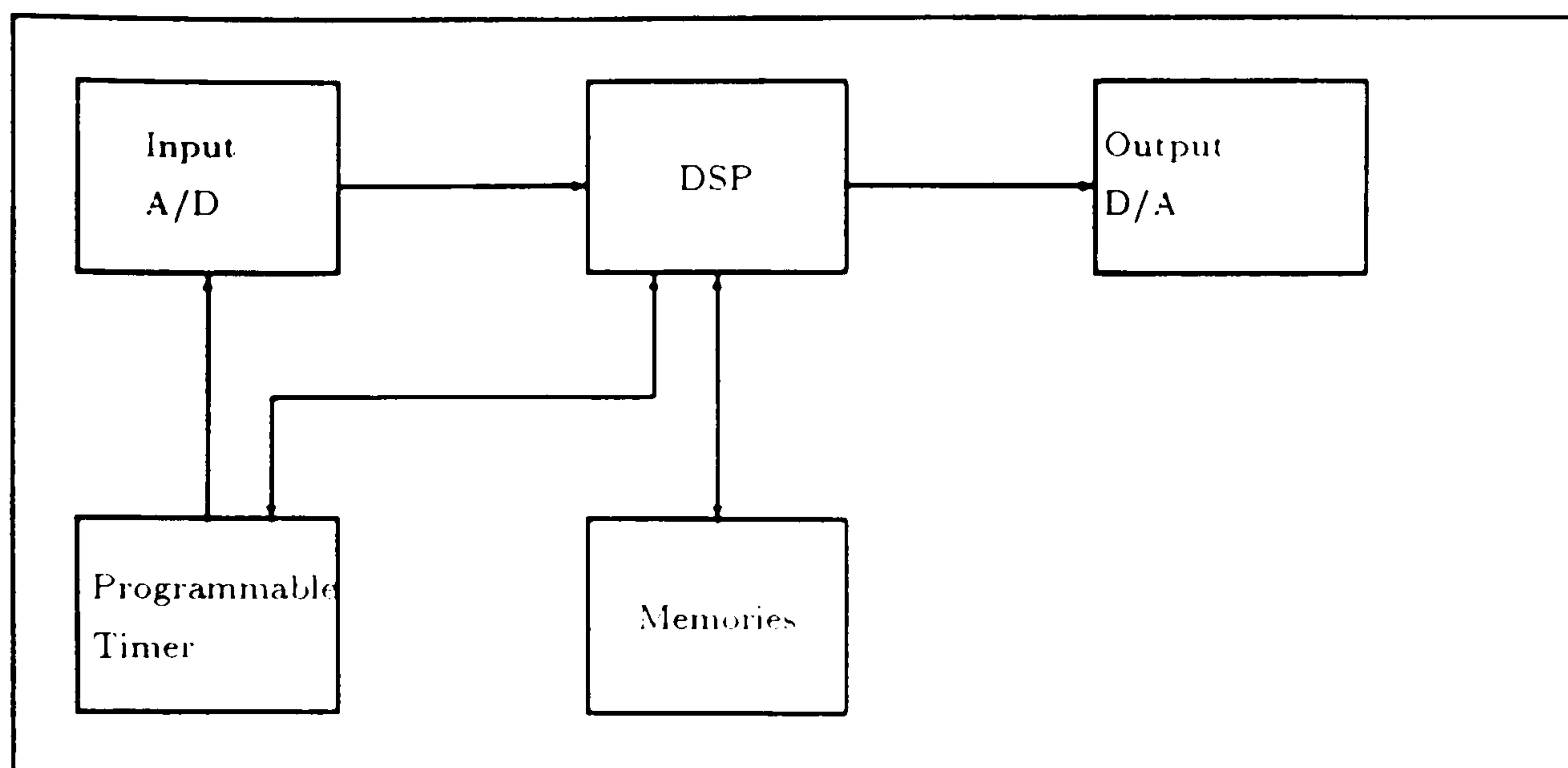


Figure 6.3: Block diagram of the Digital Signal Processing Board

6.3 The Design of A Digital Signal Processing Board

The block diagram is as shown in figure 6.3. The complete circuit diagram is shown in the appendixes. In this project it is used in stand-alone mode. The periodogram PSD estimator is burnt into EPROMs, when powered up it will execute the program automatically. The program is written in MOTOROLA DSP56000 assembly language. The MOTOROLA DSP56000 Application Development System is used for program development. The algorithm is the same as that implemented on the development system except the memory map for the peripherals are different.

6.3.1 The Analog to Digital Converter

The ADC574 is used for the analog to digital conversion. It has 12 bits digital output which gives 72 dB dynamic range. The connection is shown in figure 6.4. When pin 6 is high and pin 5 is has a transition from high to low it starts conversion. The maximum conversion time is 25 μ s, so the maximum sampling frequency is 40 KHz which in turn set the maximum unambiguous input frequency to 20 KHz according to Shannon's sampling theorem. The unit gain amplifier is to lower the impedance of the signal source. This is necessary because the changes in output current of the internal D/A converter as it tests the various bits will cause transient in A/D input current. The output voltage of the driving source must remain constant while furnishing these

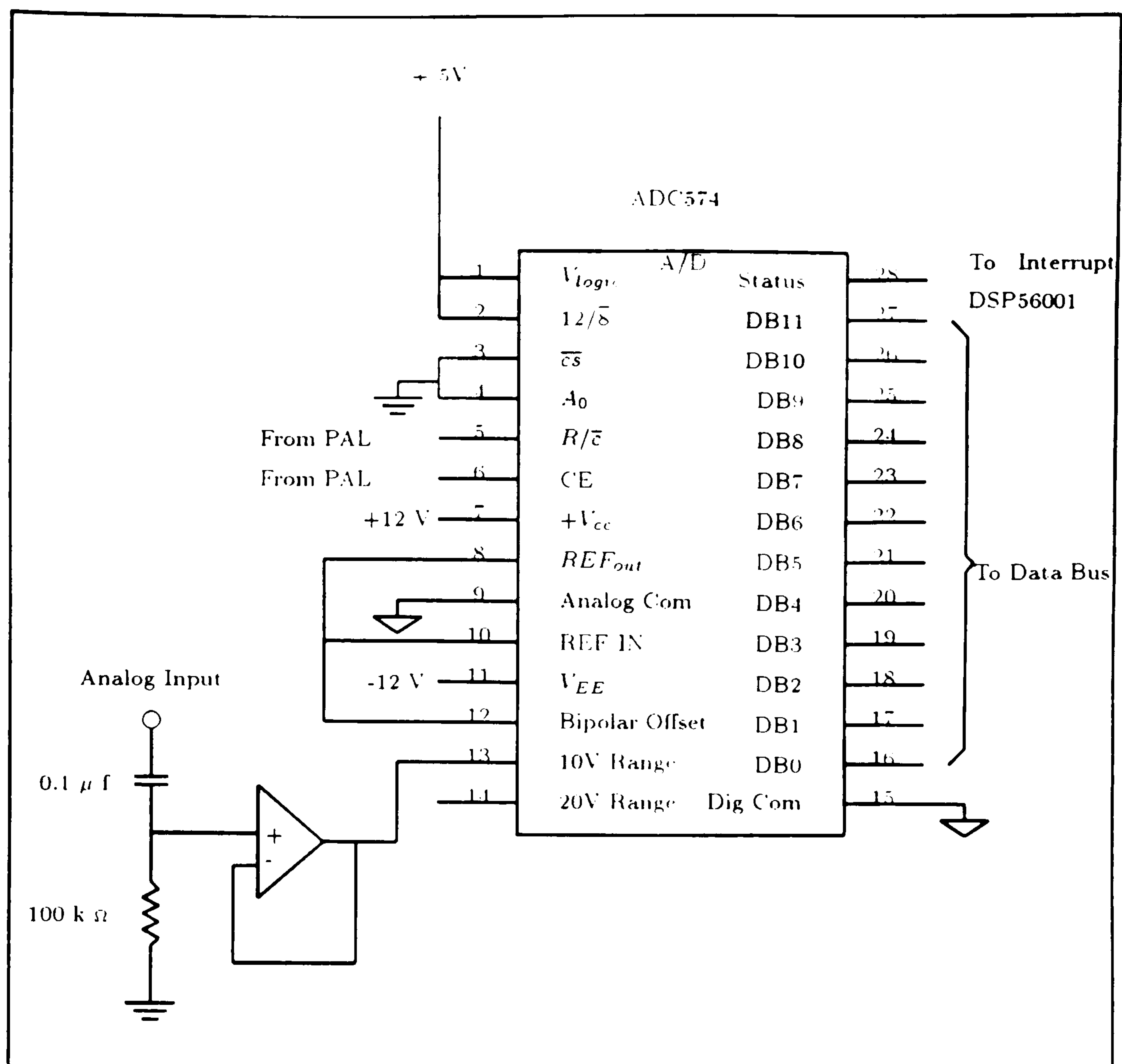


Figure 6.4: The analog to digital converter

fast current changes. The input range of The ADC574 is set as unipolar from $0V \sim 10V$ which in turn gives the resolution of $1 \text{ LSB} = \frac{10V}{2^{12}-1} = 2.44mV$. When the conversion is ready, STATUS will become high and interrupt DSP56001 to read the data. Since the data is already available, no wait state is necessary for reading data.

6.3.2 The Digital Signal Processor DSP56001

Figure 6.5 is a block diagram of the DSP56001 showing all the major blocks with their connection buses [30]. It has a dual Harvard architecture optimized for MAC (multiply and accumulate) operations. The DSP56001 has 512 words on chip program RAM, two preprogrammed data ROMs, and special on-chip bootstrap hardware to facilitate downloading of user program into the program RAM from the host computer. The heart of the processor consists of three execution units operating in parallel: the data Arithmetic Logic Unit (ALU), the Address Generation Unit (AGU), and the

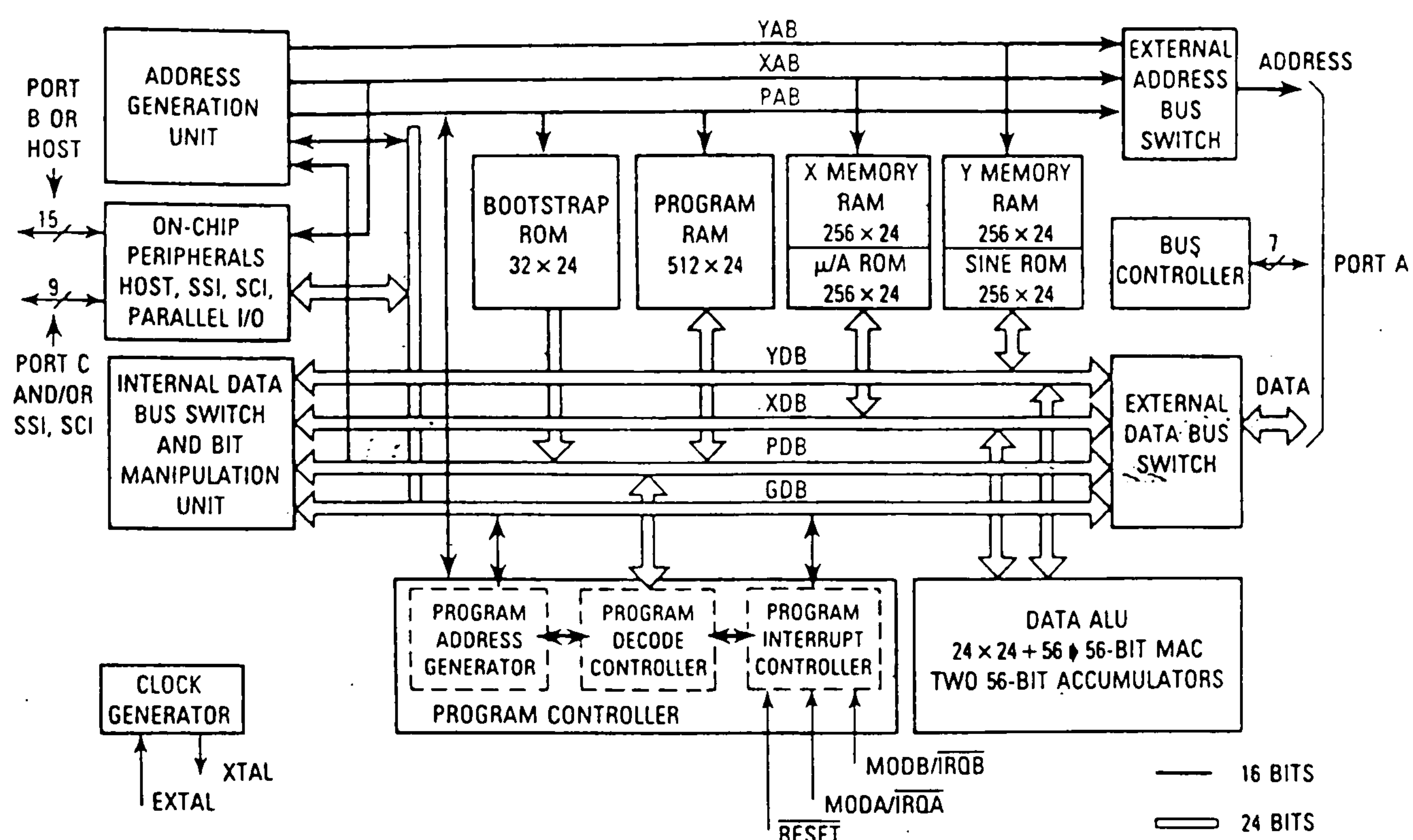


Figure 6.5: Block diagram of DSP56001

Program Counter (PC). The main features of DSP56001 are as follows:

- The maximum clock rate is 20.5 MHz which gives the minimum instruction cycle time of 97.5 ns.
- The data buses are 24 bits wide, Providing 144 dB of dynamic range. The 56 bits accumulator which can range over 336 dB gives good precision and reduces overflow problems.
- The three-stage instruction pipeline is essentially transparent to the programmer allowing straightforward program development in either assembly language or high level language, such as C.

6.3.3 EPROMS

Three 2764 EPROMs (Erasable Programmable Read Only Memory) are used to store the program. These EPROMs are $8K \times 8$ bits nonvolatile devices. This gives the

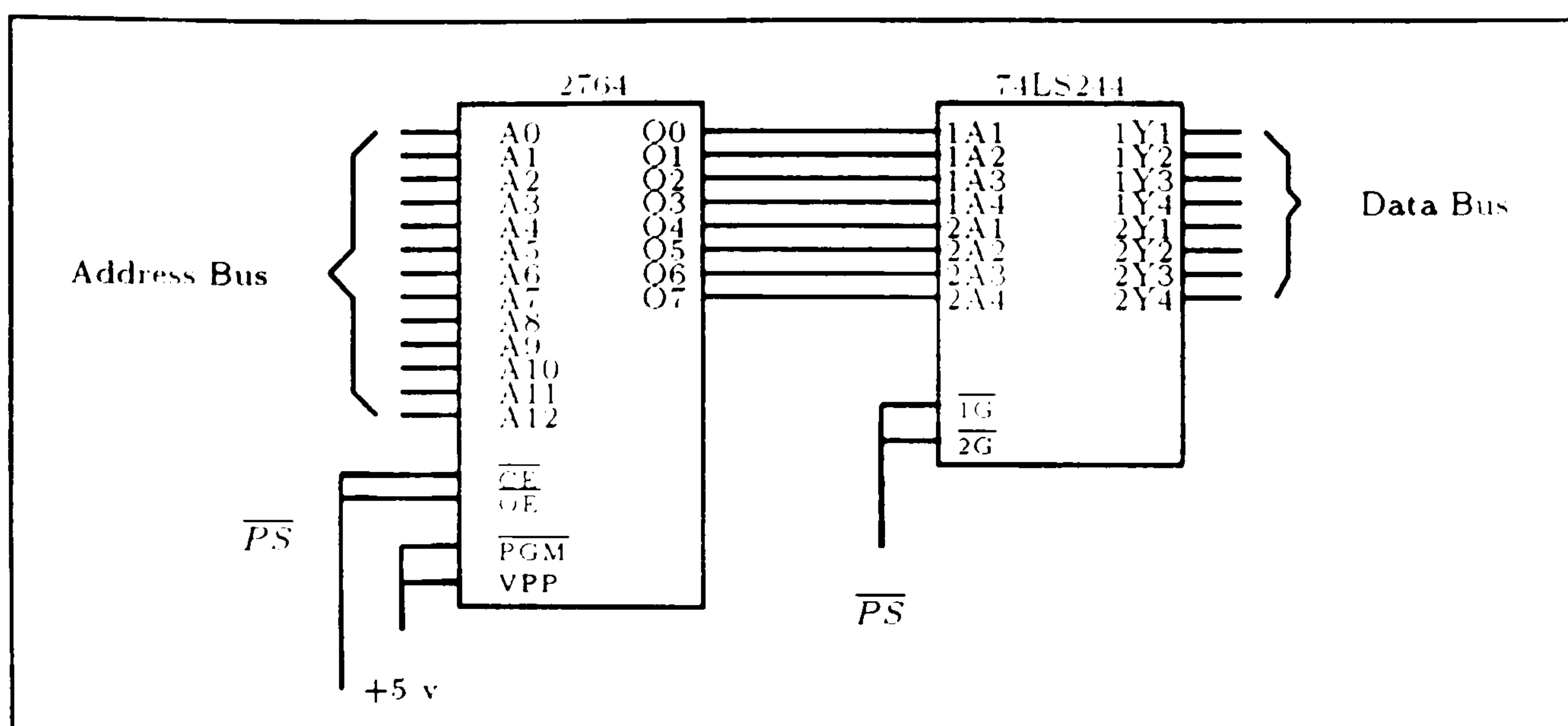


Figure 6.6: The EPROM connections

DSPB (Digital Signal Processing Board) stand-alone capability. The connection between EPROMs and DSP56001 is straightforward. When DSP56001 wants to read the program from EPROMs, it puts the address on the address bus and generates $\overline{PS}$ to enable the EPROMs to enable the program at the output. The snag is that when the $\overline{OE}$ is set high to float the output of 2764 a delay of 50 ns can be experienced, but the $\overline{RD}$ deassertion to $\overline{WR}$ assertion of DSP56001 is 40 ns minimum. So there is a chance of bus contention problem, i. e. DSP56001 and EPROMs both try to put data on the data bus while DSP56001 wants to write data on RAMs. To solve the bus contention problem three 74LS244 are cascaded between the EPROMs and the DSP56001 as shown in figure 6.6. The maximum delay time between disabling the 74LS244 and the output becoming float is 25 ns. This avoids the bus contention problem.

6.3.4 RAMs

Three $8K \times 8$ bits 6264 RAM (Random Access Memory) are used for X memory and Y memory respectively. There is no consecutive read or write problem. Since the $\overline{OE}$ high to output float of 6264 is only 35 ns which is smaller than the minimum consecutive read or write time (40 ns) of DSP56001. The minimum write pulse width required by the 6264 is 60 ns minimum. DSP56001 adds one-half instruction cycle to each memory access for each wait state (WS), i. e. 50 ns for the 20 MHz

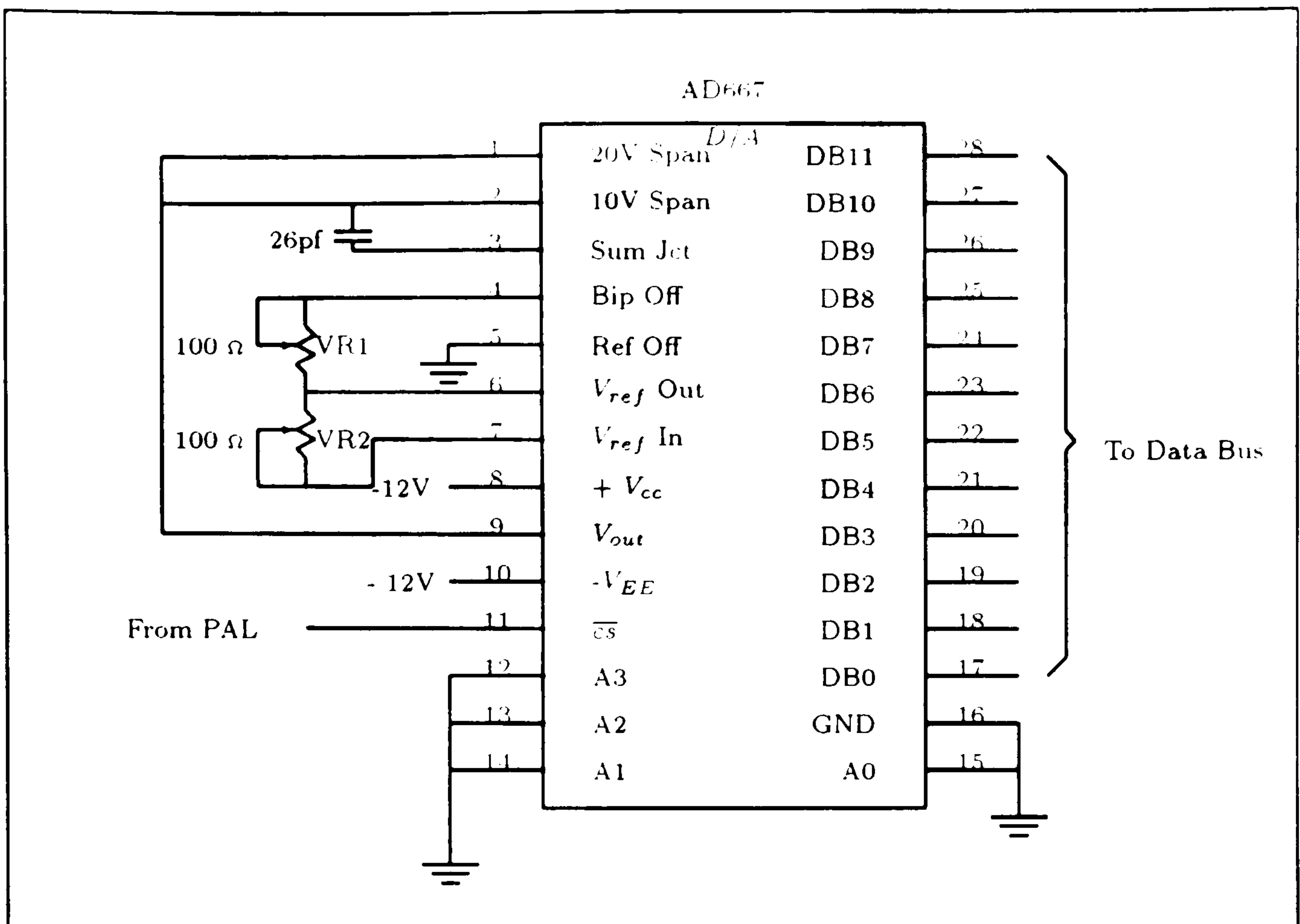


Figure 6.7: The digital to analog converter

clock in this design. The write assertion width is 41 ns ($WS = 0$), 66 ns ($WS = 1$), 116 ns ($WS = 2$). Two wait states are used to ensure reliable writing.

6.3.5 The Digital to Analog Conversion

The AD667 is used for digital to analog conversion, see figure 6.7. It has 12 bits of digital input and ± 5 V analog output. The $\overline{CS}$ starts the conversion. VR1 is used to adjust the offset. VR2 is used to adjust the gain. The minimum pulse width of $\overline{CS}$ is 100 ns. A wait state of two is needed. The maximum conversion speed is $4\mu s \pm 0.01$

6.3.6 The Timer

The 8254-2 programmable interval timer is used for generating a trigger signal to the A/D, see figure 6.8. OUT is originally high and goes low N clock pulses after the the trigger, where N is the number loaded into the down counter. There are three 16 bits presetable down counters. For a clock frequency of 10 MHz the maximum counting

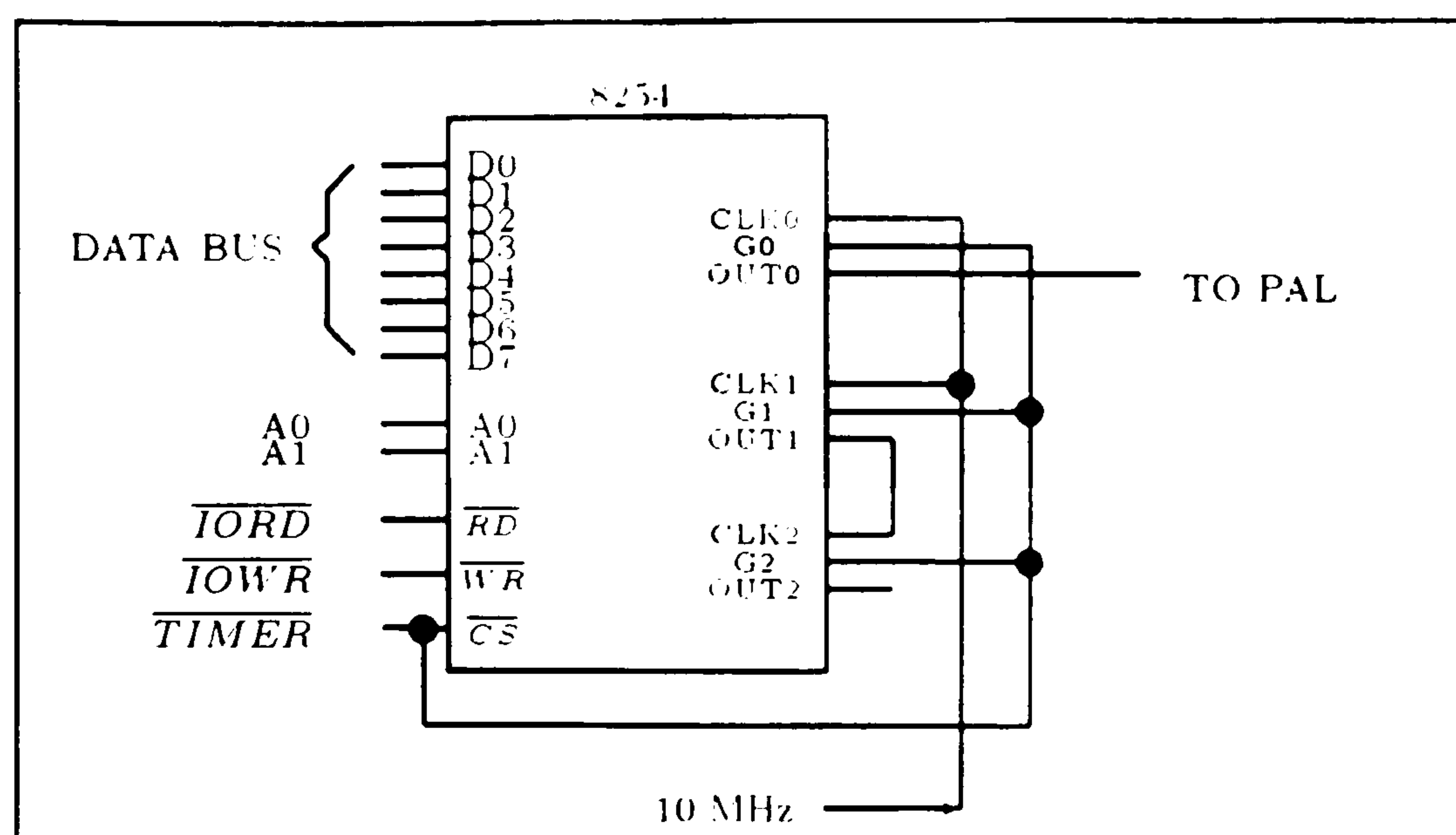


Figure 6.8: The timer

interval for a single counter is

$$\frac{2^{16} - 1}{10^7} = 6.5535ms.$$

Counter 1 and counter 2 are cascaded, therefore the maximum time interval is

$$6.5535ms \times (2^{16} - 1) = 429sec.$$

To write the data to the timer the $\overline{CS}$ signal has to arrive no later than $\overline{WR}$. The address decoding delay caused by the address decoder PAL 16L8 is 20 ns maximum. Whereas the address valid to $\overline{WR}$ assertion of DSP56001 is 16 ns maximum. Therefore there are possibilities that $\overline{CS}$ will arrive later than $\overline{WR}$. To avoid this problem $\overline{WR}$ is gated by $\overline{CS}$ in the PAL and becomes $\overline{IOWR}$. This ensures that $\overline{IOWR}$ is always derived later than $\overline{CS}$. The same arrangement is applied to the signal $\overline{RD}$.

6.3.7 The Address Decoder

The 16L8 PAL is used for the address decoding purpose, see figure 6.9 The address assignments for the peripheral are as table 6.1

The PAL is programmed by using CUPL (Universal Compiler for Programmable Logic). CUPL is a computer aided design tool. It is a high level compiler. It can do logic minimization and DeMorgan expansion. The programming consists of two

Device	Address	A15	A3	A2
Counter 0 of 8254	FFC0	1	0	0
Counter 1 of 8254	FFC1	1	0	0
Counter 2 of 8254	FFC2	1	0	0
CTRL Word of 8254	FFC3	1	0	0
D/A	FFC4	1	0	1
A/D	FFC8	1	1	0

Table 6.1: The Memory Map for Peripherals

parts: one is the logic declaration, the other is the polarity declaration. The logic output equations for PAL written in terms of its inputs are:

$$AD = A15 \& A3 \& Y$$

$$RC = OUT0$$

$$DA = A15 \& A2 \& Y$$

$$IORD = RD \& Y \& A15 \& !A2 \& !A3$$

$$IOWR = WR \& Y \& A15 \& !A2 \& !A3$$

$$TIMER = Y \& A15 \& !A2 \& !A3$$

$$MODINT = RESET \# INT$$

$$YMEM = Y \& DS \& !A15$$

Where $\&$ means AND, $\#$ means OR, $!$ means NOT.

The polarity declaration of pins are: A2, A3, A15, AD, !RC, !DA, !IORD, !IOWR, !TIMER, YMEM, !Y, !WR, !RD, A15, !DS, A3, A2, !RESET, INT. Where $!$ means the variable is true when its voltage is low. YMEM is to avoid the bus contention problem when DSP56001 wants to communicate with peripherals. Because the peripherals are memory-mapped in the Y-memory, it is disabled when DSP56001 is communicating with peripherals.

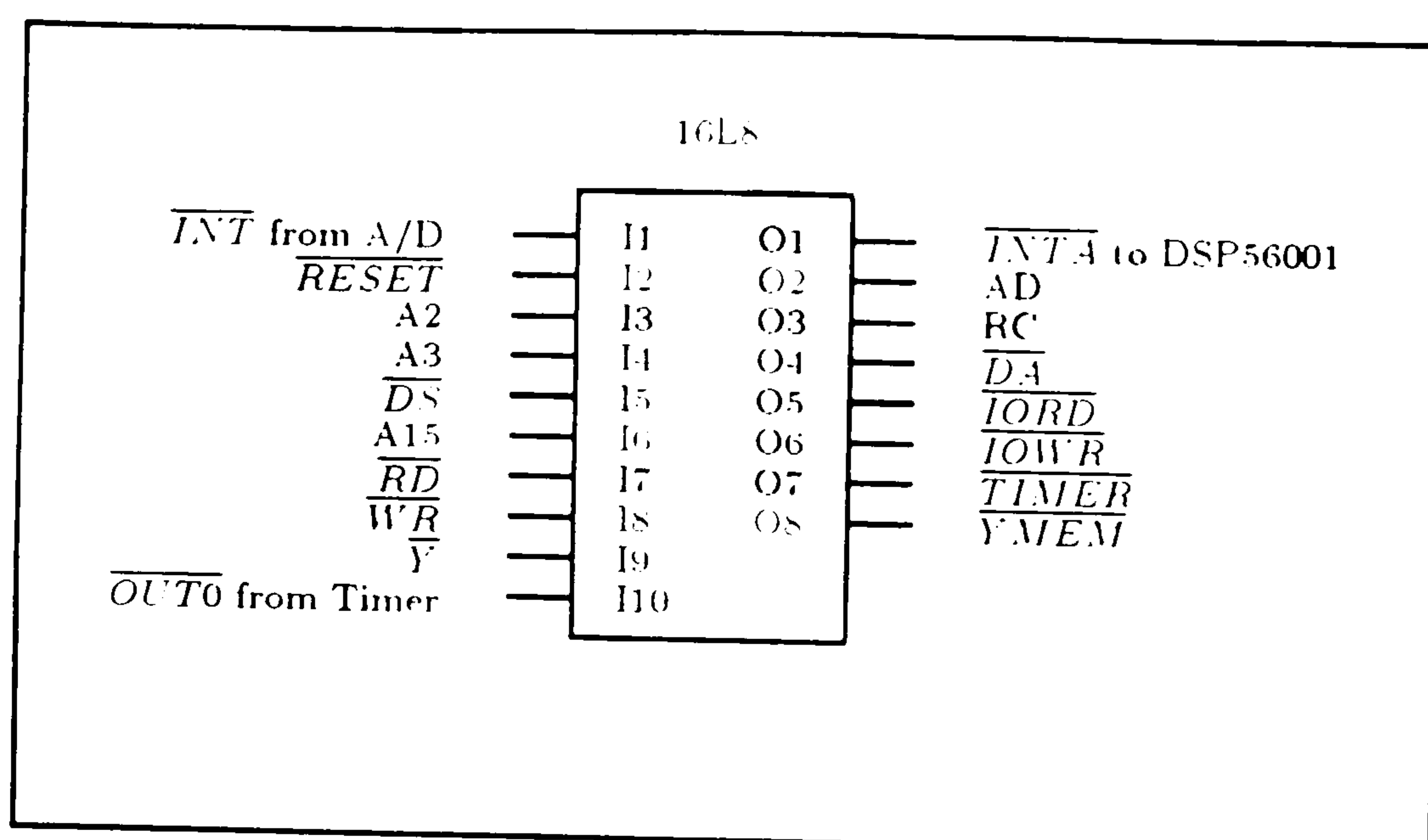


Figure 6.9: The PAL as address decoder

6.4 The Proposed Multiprocessor Architecture for PSD Estimation Application

The singular value decomposition method (or eigendecomposition method) mentioned before needs a great amount of computation. The high speed VLSI has been designed to facilitate real-time implementation. However, the speed of the VLSI techniques will eventually be limited by the physical laws. System performance improvements must then come from the application of parallelism. A parallel processor system achieves optimum system performance by distributing tasks between two or more processors. The processors are working independently. However, each processor needs to be able to pass the results of its work to another processor or vice versa. Processor to processor communication is critical in multiprocessor system design. The TMS320C4X generation DSP is the latest DSP that is designed particularly for the parallel processing. It has 6 high speed bidirectional communication ports. Each port has independent 8 data lines and 4 control lines. The direct (glueless) processor to processor communication can be done in high speed (160 megabits per second or 5 megawords per second at 40 ns cycle time). Those communication ports can be configured in a wide variety of multiprocessor architectures, including rings, trees, hypercubes, bidirectional pipelines, three dimensional grid, ... etc. Another important feature that the TMS320C4X provides to facilitate parallel processing is a DMA

(Direct Memory Access) facility. It has 6 independent DMA channels. The primary benefit of the DMA coprocessor is to maximize CPU performance by completely relieving the CPU of burdensome data movement duties. The DMA coprocessor can perform transfer to and from anywhere in the processor memory map including the six on-chip communication ports. This allows the data (or results) to be shared among processors.

6.4.1 The PSD Estimation Using The TMS320C4X Parallel Processing Features

The algorithm proposed in chapter 2 and 3 needs to calculate the spectrum of a common data set by using two different PSD estimators. The multiprocessor parallel processing is suitable for this application. The task for each processor is easily partitioned: it is just different PSD estimators. The configuration is shown in figure 6.10. The ping-pong memory configuration described in chapter 3 is used here. The size of each input data buffer is to ensure each processor finishes its jobs before switching to the other block of data. Each processor has its own EPROM and RAM connected to its local bus. There are two sets of control signal $\overline{LSTRB0}$ and $\overline{LSTRB1}$. The local memory control registers define which set of registers is active. This allows glueless connection between memories and processor. The A/D and D/A are memory mapped into the local memory. The selection is via an address decoder. The processor to processor connection is via the communication ports. The flow chart of the processors are shown in figure 6.11 and figure 6.12.

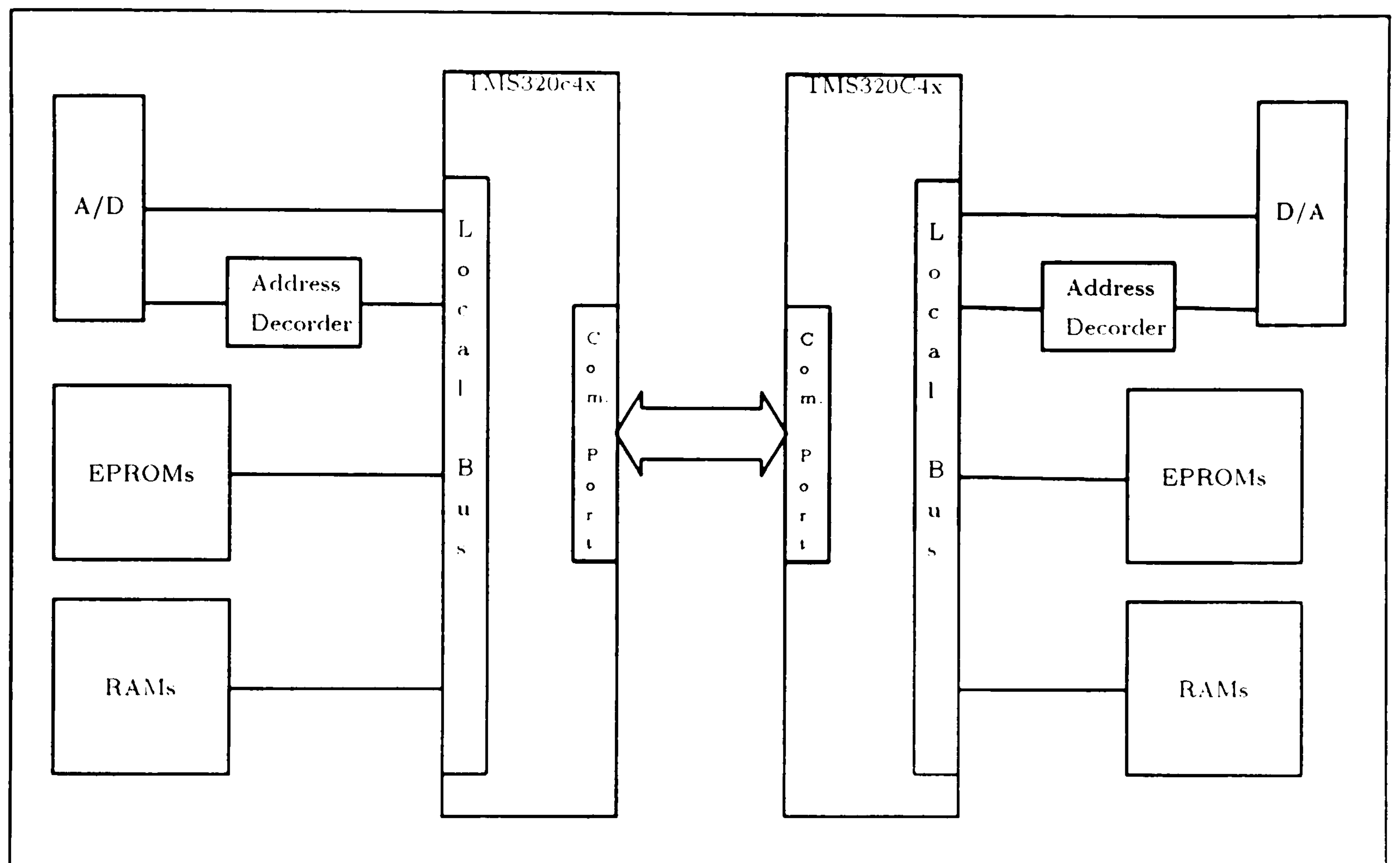


Figure 6.10: The multiprocessor configuration using TMS320C4X

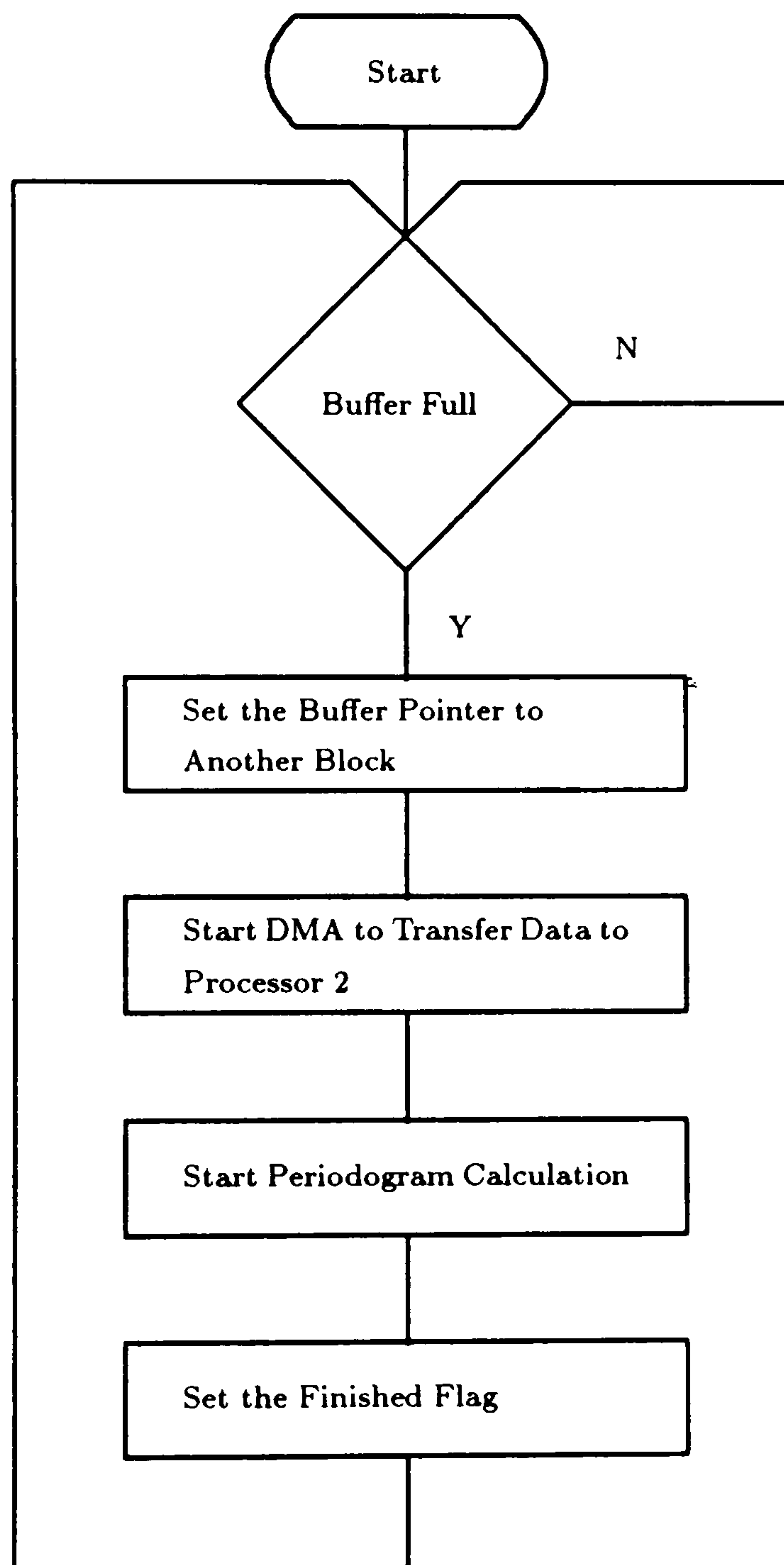


Figure 6.11: The program flow chart of processor 1

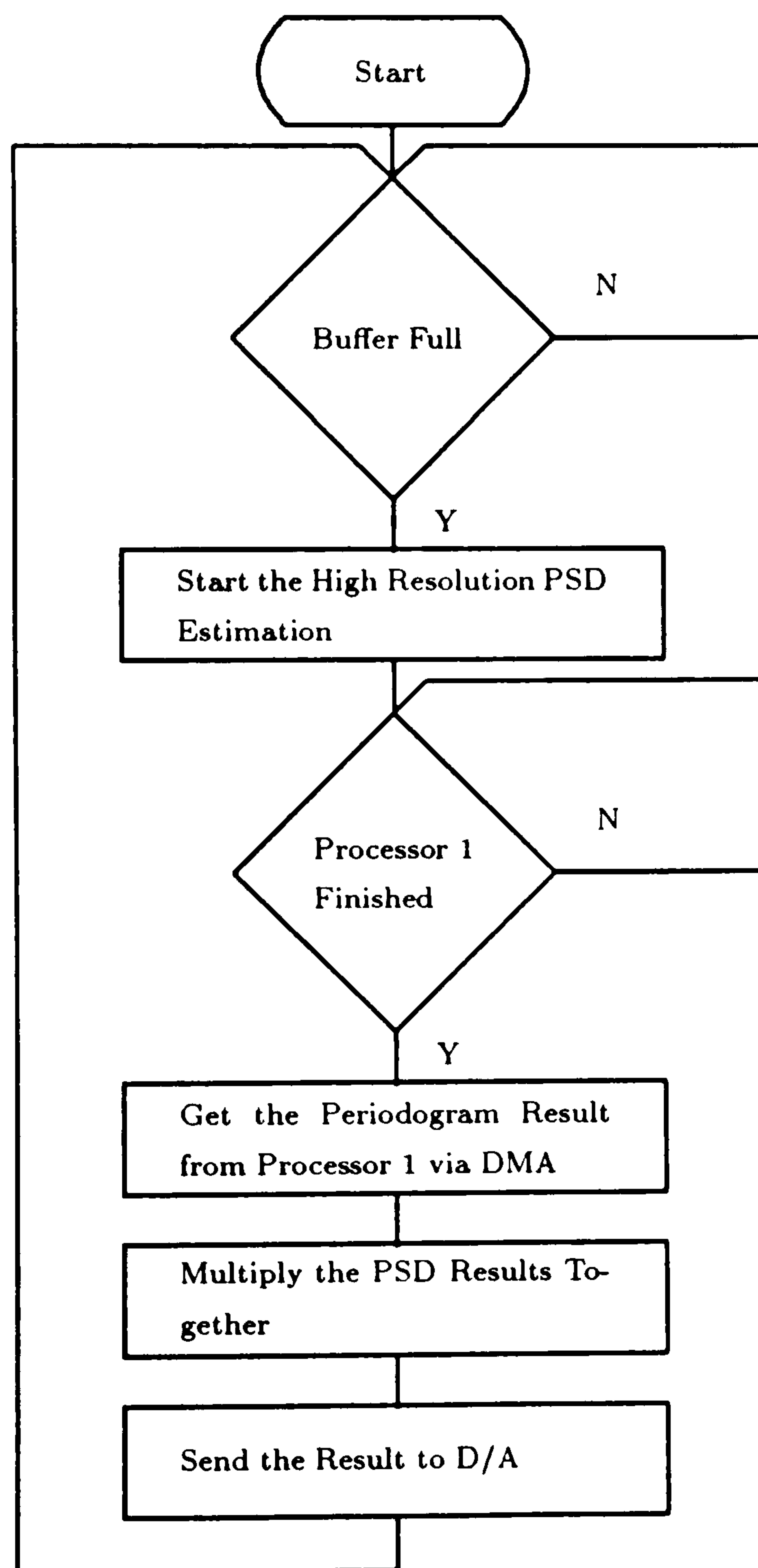


Figure 6.12: The program flow chart of processor 2

Chapter 7

Conclusion and Suggestion for Further Work

7.1 Conclusion

From the above discussion it is clear that there is no universal PSD estimator. Depending on the particular application a suitable PSD estimator should be chosen. According to the definition of 1.1 in chapter 1 the true spectrum of a random process needs infinite samples which is impossible, therefore there is no true PSD function as a reference to assess the PSD estimation results. i. e. we may face the dilemma that we are not sure a sharp peak is due to the true signal or an artifact. The general guide lines are:

1. When enough data are available and the $\frac{1}{NT}$ resolution is sufficient then the classical methods are to be preferred, because they are robust.
2. If possible, different algorithms should be used to confirm the result.
3. It is not reliable to do the estimation based on just one evaluation. As many evaluations as possible should be used to obtain the statistical result.

The behaviour of classical methods are well understood. Especially the periodogram which is widely used in practice, e. g. radar, sonar and biomedical. But there are some situations for which the classical methods are not suitable, e. g. short available data length, higher resolution requirement. As a result, more and more AR parametric PSD estimators have been implemented in real-time [37]. The main problem of the

AR parametric methods is the determination of the order. Various criteria have been proposed, e. g. AIC (Akaike Information Criterion), FPE (Final Prediction Error), CAT (Criterion Autoregressive Transfer Function), and MDL (Minimum Description Length). [29]. These methods generally work well. However, in difficult situations, e. g. the SNR is low and there is coloured noise, the AR methods tend to generate spurious peaks. These spurious peaks will compete with the true peaks of the signal. We have proposed a method in this situation to identify the true peaks. We use the periodogram (the correlogram may be used as well) which is not affected by the order selection to serve as an indication of the area in which the signals are located. We find the intersection of the spectrum generated by periodogram and AR PSD estimator by multiplying them together. This will suppress spurious peaks generated by the AR parametric methods while retaining the high-resolution capability of them (or this may be thought as improving the resolution capability of the periodogram).

The eigenanalysis based parametric methods can estimate the frequency accurately by using just a few data samples. This property is very suitable for array signal processing, e. g. DOA (Direction Of Arrival) estimation, because the elements of the array are usually limited. One intensive research area about this method is to use the parallel algorithm to find the eigenvalues/eigenvectors. Recursive algorithms have also been developed to avoid the necessity of finding the eigenvalues/eigenvectors from scratch every time but to update the old eigenvalues/eigenvectors based on newly arrived data. This will speed up the eigendecomposition process. Table (1.1) provides a summary of the key PSD estimators that were presented in this research. The main contributions of this work are described in the introduction 1.2.

7.2 Suggestion for Further Work

The advance of VLSI technology has made many digital signal processing algorithms viable for real-time implementation. Digital spectral estimation as one of the main digital signal processing tools will continue to be under intensive research. Following areas are of interest:

Category	Model	Method	Decision Factors for The Resolution Performance	Resolution Capability
Nonparametric	None	periodogram	Window	Low
	None	Correlogram	Window	Low
	None	MVSE	Dimension $\mathbf{R}_{xx}$	Medium
Parametric	AR	Yule-Walker	AR order	High
	AR	Modified Cov	AR order	High
	AR	Burg	AR order	High
	MA	High-order AR	MA order	Low
	ARMA	Modified YW	AR,MA order	High
	Sinusoids + Noise	MUSIC	no. of signal	High
	Sinusoids + Noise	EV	no. of signal	High

Table 7.1: Summary of Key PSD Estimators

1. The ARMA model: The ARMA model has the potential for estimation of both broadband and narrowband or high dynamic range spectra. However, this potential can only be realized if accurate parameter estimation is made. So fast and accurate algorithms for finding these parameters are needed.
2. Fast algorithms for eigendecomposition: The eigenanalysis based methods are of special interest for spatial data analysis due to their ability to perform high-resolution beamforming. But to implement these algorithms in real-time, fast and reliable algorithms for calculating the eigenvalues/eigenvectors are needed.
3. Multidimensional spectral analysis: There are several applications that use spatial data array, e. g. radar, sonar, seismology etc. More research is needed to extend the one-dimensional algorithm to a practical algorithm for those multidimensional applications.
4. The development of an expert system: There exist many PSD estimators. The choice of one particular spectral estimator usually depends on some *a priori* information about the input data, e. g. narrowband or broadband, the SNR, the number of signals and their approximate frequency separation etc. An expert system may be employed to choose a suitable algorithm. Ideally it can choose the best suit estimator that matches the input data situation, i. e. "data adaptive".

Appendix A

FORTRAN Program for VEP PSD Estimation

```
      PROGRAM VEP
C
C      *****
C      *
C      * FUNCTION: THIS IS THE MAIN PROGRAM TO DISPLAY *
C      *          THE VEP SIGNAL. THE INPUT DATA CAN *
C      *          BE GENERATED BY SOFTWARE SIMULATION *
C      *          OR READ FROM FILE. THE PSD          *
C      *          ESTIMATION ALGORITHM CAN BE         *
C      *          PERIODOGRAM OR MUSIC                *
C      *
C      *****
C
      COMPLEX AS(512,512)
      REAL S(512),PSD(512)
      CHARACTER *1 METHOD,OPTION
C
C      SELECT THE TERMINAL CODE FOR UNIRAS
C
      CALL GROUTE('S MREGIS; E') ! Terminal code for VT240
C      call groute('s mx11; e') ! Terminal code for DEC workstation
C      call groute('s lt4109; e') ! Terminal code for Pericom terminal
      CALL GOPEN
C
C——— PROMPT THE USER TO SELECT THE METHOD
C
1      WRITE(6,*)'INPUT THE METHOD TO USE'
      WRITE(6,*)' 1. PERIODOGRAM'
      WRITE(6,*)' 2. MUSIC '
      READ(5,10) METHOD
10     FORMAT(A1)
C
C——— PROMPT THE USER TO SELECT THE WAY DATA IS GENERATED
C
```



```

        WRITE(6,*)'CHOOSE DATA OPTION'
        WRITE(6,*)' 1. GENERATE DATA'
        WRITE(6,*)' 2. READ DATA FROM FILE'
        READ(5,20) OPTION
20      FORMAT(A1)
        NAS = 512 ! ARRAY SIZE FOR DYNAMIC MEMORY ALLOCATION
C
C—— CALL CORRESPONDING SUBROUTINES TO GENERATE DATA
C
        IF(OPTION.EQ.'1') CALL GENERATE_VEP(AS,NS,IC,N,NAS)
        IF(OPTION.EQ.'2') CALL READ_VEP(AS,NS,IC,N,NAS)
C
C—— CALL CORRESPONDING SUBROUTINES TO CALCULATE THE PSD
C
        IF(METHOD.EQ.'1') CALL DO_PERIODVEP(AS,NS,N,NAS)
        IF(METHOD.EQ.'2') CALL DO_MUSICVEP(AS,NS,IC,N,NAS)
C
C—— PROMPT THE USER TO DO IT AGAIN OR NOT ?
C
        CALL DISPLAY_OPTIONS(IR)
        IF(IR.EQ.1) GOTO 1
        CALL GCLOSE
        STOP
        END

```

SUBROUTINE GENERATE_VEP(AS,NS,IC,N,NAS)

```

C
C      *****
C      *
C      * FUNCTION: GENERATES SIMULATED TWO VEP
C      *          SIGNALS
C      * PARAMETERS:
C      * INPUT: NONE
C      * OUTPUT:
C      *      AS : ARRAY SIGNAL
C      *      NS : NUMBER OF SNAP SHOT
C      *      IC : NUMBER OF COLUMN OF DATA MATRIX
C      *      N : NUMBER OF DATA POINTS
C      *      NAS: DYNAMIC ARRAY SIZE DECLARATION
C      *
C      *****
C
C      COMPLEX AS(NAS,NAS),V(512)
C      REAL Z(512)
C
C      WRITE(*,*)'INPUT APERTURE SIZE: (N)'
C      READ(*,*) N

```



```

WRITE(*,*)'INPUT SUBAPERTURE SIZE: (IC)'
READ(*,*) IC
WRITE(*,*)'INPUT STARTING FREQUENCY OF VEP 1'
READ(*,*) F1
WRITE(*,*)'INPUT FREQUENCY CHANGE OF VEP 1 PER SNAP SHOT'
READ(*,*) DF1
WRITE(*,*)'INPUT STARTING FREQUENCY OF VEP 2'
READ(*,*) F2
WRITE(*,*)'INPUT FREQUENCY CHANGE OF VEP 2 PER SNAP SHOT'
READ(*,*) DF2
WRITE(*,*)'INPUT NUMBER OF SNAP SHOT: (NS) '
READ(*,*) NS
WRITE(*,*)'INPUT S/N: (DB) '
READ(*,*) DB
PI = 4. * ATAN(1.)
ISEED = 69069
VAR = 1./(2*10**(0.1*DB))
NR = 2 * N
CALL WGN(NR,VAR,ISEED,Z)
DO I = 1,N
    V(I) = CMPLX(Z(2*I-1),Z(2*I))
ENDDO
DO I = 1,NS
    F11 = F1 + (I-1) * DF1
    F22 = F2 + (I-1) * DF2
DO J = 1,N ! Do columnwise to minimize page fault
    AS(J,I) = CEXP(CMPLX(0.,2.*PI*F11*(J-1))) + CEXP(CMPLX(0.,2.*PI*F22*(J-1))) + V(J)
ENDDO
ENDDO
RETURN
END

```

```

SUBROUTINE READ_VEP(AS,NS,IC,N,NAS)

```

```

C
C *****
C *
C * FUNCTION: READ INPUT DATA FROM FILE FOR ANALYSIS *
C * PARAMETERS:
C * INPUT: NONE
C * OUTPUT:
C *      AS : ARRAY SIGNAL
C *      NS : NUMBER OF SNAP SHOT
C *      IC : NUMBER OF COLUMN OF DATA MATRIX
C *      N : NUMBER OF DATA POINTS
C *      NAS: DYNAMIC ARRAY SIZE DECLARATION
C *
C *****

```



```

C
COMPLEX AS(NAS,NAS),X(4096)
C
CALL READ_FILE(X,NF)
WRITE(*,*)'INPUT APERTURE SIZE: (N)'
READ(*,*)N
WRITE(*,*)'INPUT SUBAPERTURE SIZE: (IC)'
READ(*,*)IC
WRITE(*,*)'INPUT NUMBER OF SHIFT BETWEEN SHIFT'
READ(*,*)NSHIFT
NS = (NF - N)/NSHIFT + 1
DO I = 1,NS
DO J = 1,N ! Do columnwise to minimize page fault
    INDEX = J + (I-1) * NSHIFT
    AS(J,I) = X(INDEX)
ENDDO
ENDDO
RETURN
END

SUBROUTINE DO_PERIODVEP(AS,NS,N,NAS)
C
C *****
C *
C * FUNCTION: MAIN SUBROUTINE FOR THE PERIODOGRAM *
C * PSD ESTIMATION *
C * PARAMETERS: *
C * INPUT: *
C * AS : ARRAY SIGNAL *
C * NS : NUMBER OF SNAP SHOT *
C * N : NUMBER OF DATA POINTS *
C * NAS: DYNAMIC ARRAY SIZE DECLARATION *
C * OUTPUT: NONE *
C *
C *****
C
COMPLEX AS(NAS,NAS),X(512)
REAL PSD3D(512,512),W(512),PSD(512)
C
CALL HAMMING(W,N)
CALL ZPVEP(AS,NS,N,MPOINTS,NEXP,NAS)
NPSD = 2**NEXP
DO I = 1,NS
DO J = 1,N
    X(J) = AS(J,I)
ENDDO
CALL PERIODOGRAM(X,N,W,NEXP,PSD)

```



```

DO J=1,NPSD
    PSD3D(J,I)=PSD(J)
ENDDO
ENDDO
CALL DISPLAY_3D(PSD3D,NPSD,NS,NAS)
RETURN
END

SUBROUTINE DISPLAY_OPTIONS(IR)
C
C *****
C *
C * FUNCTION: PROVIDES OPTION FOR REPEATING A
C *          PROCESS, e.g. USING DIFFERENT INPUT
C *          DATA OR DIFFERENT PARAMETERS.
C *
C *****
C
C CHARACTER*1 ANSWER
C
C *****PROMPT THE USER TO DO IT AGAIN OR NOT ? *****
C
C WRITE(6,1)
1  FORMAT(1x,'DISPLAY ANOTHER ;N; ?'$)
C READ(5,10)ANSWER
10 FORMAT(A1)
C
C READ THE ANSWER: DEFAULT NO (RETURN) , IR = 0
C          OPTION YES (INPUT'y' OR 'Y') IR = 1
C
C IF((ANSWER.EQ.'y').OR.(ANSWER.EQ.'Y')) THEN
C     IR=1
C ELSE
C     IR=0
C ENDIF
C RETURN
C END

SUBROUTINE DO_MUSICVEP(AS,NS,IC,N,NAS)
C
C *****
C *
C * FUNCTION: MAIN SUBROUTINE FOR MUSIC PSD
C *          ESTIMATION
C *
C * PARAMETERS:

```



```

C      * INPUT:
C      *      AS : ARRAY SIGNAL
C      *      NS : NUMBER OF SNAP SHOT
C      *      IC : NUMBER OF COLUMN OF DATA MATRIX
C      *      N : NUMBER OF DATA POINTS
C      *      NAS: DYNAMIC ARRAY SIZE DECLARATION
C      * OUTPUT: NONE
C      *
C      *****
C
C      COMPLEX AS(NAS,NAS)
C      REAL PSD3D(512,512)
C      CHARACTER *1 ANSWER
C
C      WRITE(*,*)'INPUT NUMBER OF SIGNAL: (NSIG)'
C      READ(*,*)NSIG
C      WRITE(*,*)'ZERO PADDING ? (N)'
C      READ(5,5)ANSWER
5      FORMAT(A1)
C      IF((ANSWER.EQ.'y').OR.(ANSWER.EQ.'Y')) THEN
C          WRITE(*,*)'HOW MANY POINTS'
C          READ(*,*)MPOINTS
C          CALL DO_NEXP(MPOINTS,NEXP)
C      ELSE
C          CALL DO_NEXP(N,NEXP)
C      ENDIF
C      NPSD = 2**NEXP
C      CALL EIGENFREQ3D(1,NS,NEXP,N,IC,NSIG,AS,NAS,PSD3D,ISTAT)
C      CALL DISPLAY_3D(PSD3D,NPSD,NS,NAS)
C      RETURN
C      END
C
C      SUBROUTINE READ_FILE(X,NF)
C
C      *****
C      *
C      * FUNCTION: READS A SIGNAL FROM A NAMED DATA FILE
C      *      (FREE FORMATED).
C      * INPUT: NONE
C      * OUTPUT: X - DATA
C      *      NF - SIZE OF SIGNAL
C      * NOTE: MAXIMUM SIZE OF NF IS 4096
C      *
C      *****
C
C      COMPLEX X(4096)
C      REAL S(4096)

```



```

      CHARACTER *1 ANSWER
C
C      CALL SUBROUTINE OPEN_READ_FILE WHICH ASSIGNS UNIT NUMBER 1
C      TO DATA FILE AND PROMPT TO USER FOR SIZE AND NAME OF FILE
C
1      CONTINUE
      CALL OPEN_READ_FILE(NF)
C
C      SEND MESSAGE TO USER 'READING DATA...'
C
      WRITE(6,5)
5      FORMAT(1X,'READING DATA ....')
C
C      READ REAL DATA FROM FILE AND STORE IN S AND THEN CONVER TO COMPLEX DATA X
C
      DO I = 1,NF
          READ(1,*,END=10,ERR=15)S(I)
          X(I) = CMPLX(S(I),0.)
      ENDDO
      GOTO 25
C
C      IF READ GOES BEYOND END OF FILE, OUTPUT APPROPRIATE MESSAGE
C      WITH OPTIONS ALLOWING USER TO EITHER INPUT ANOTHER FILE OR
C      STOP PROCESS.
C
10     WRITE(6,*)'ATTEMPT TO READ BEYOND END OF FILE !'
      WRITE(6,*) !SKIP A LINE
      WRITE(*,*)'CONTINUE ? (N)'
      READ(5,11) ANSWER
11     FORMAT(A1)
      IF((ANSWER.EQ.'y').OR.(ANSWER.EQ.'Y')) GOTO 1
      IF((ANSWER.NE.'y').OR.(ANSWER.NE.'Y')) GOTO 25
C
C      IF ERROR OCCURS WHILE READING FILE, OUTPUT APPROPRIATE MESSAGE
C      WITH OPTION OF ALLOWING USER TO INPUT ANOTHER FILE
C      OR TO STOP PROCESS.
C
15     WRITE(*,*)'ERROR WHEN READING DATA FROM FILE !'
      WRITE(*,*) !SKIP A LINE
      WRITE(*,*)'CONTINUE ? (N)'
      READ(5,20) ANSWER
20     FORMAT(A1)
      IF((ANSWER.EQ.'y').OR.(ANSWER.EQ.'Y')) GOTO 1
C
C      CLOSE FILE
C
25     CLOSE(1)
C
      SKIP A LINE TO INDICATE THAT PROCESS HAS FINISHED

```



```

C
    WRITE(*,*)
    RETURN
    END

C
C
C
    SUBROUTINE OPEN_READ_FILE(NF)

C
    CHARACTER *40 FILENAME

C
    FUNCTION: ASKS THE USER TO 'ENTER SIZE OF SIGNAL...'
    AND 'ENTER NAME OF INPUT DATA FILE...'.

C
    NMAX = 4096

C
    PROMPT USER TO INPUT SIZE OF SIGNAL TO READ.
C
    1    WRITE(*,*)'ENTER SIZE OF SIGNAL (MAX 4096)'
        READ(5,*,ERR=99)NF

C
    CHECK THAT N IS  $\leq$  NMAX AND  $\geq$  0
C
    IF((NF.GT.NMAX).OR.(N.LT.0)) THEN
        WRITE(*,*)'WRONG SIZE , TRY AGAIN ? (Y) '
        READ(5,2)ANSWER
    2    FORMAT(A1)
        IF((ANSWER.EQ.'y').OR.(ANSWER.EQ.'Y')) GOTO 1
    ENDIF
    GOTO 5

C
    5    CONTINUE

C
    PROMPT USER TO INPUT NAME OF DATA FILE
C
    WRITE(*,*)'ENTER NAME OF DATA FILE (INPUT) :'
    READ(5,15)FILENAME
    15    FORMAT(A40)

C
    OPEN FILE - UNIT FILE NUMBER=1
C
    OPEN(UNIT=1,FILE=FILENAME,STATUS='OLD',ERR=100)
    REWIND 1
    GOTO 1000

C
    IF ERROR OCCURS IN INPUT TYPE SEND MESSAGE TO '... TRY AGAIN'
C
    99    WRITE(*,*)'WRONG TYPE, TRY AGAIN !'
        GOTO 1

```



```

C
C      IF FILE DOES NOT EXIST, INFORM USER AND LOOP
C      BACK PROMPTING USER TO INPUT NAME OF DATA FILE.
C
100    WRITE(*,*)'FILE DOES NOT EXIST !'
      GOTO 5
1000   CONTINUE
C
      RETURN
      END

      SUBROUTINE PERIODOGRAM(X,N,W,NEXP,PPER)
C
C      *****
C      *
C      * FUNCTION: THIS PROGRAM COMPUTES A DATA
C      *              WINDOWED PERIODOGRAM.
C      * INPUT PARAMETERS:
C      * X - COMPLEX ARRAY OF DIMENSION 2**NEXP × 1 WITH
C      *      FIRST N LOCATION CONTAINS THE DATA POINTS
C      * N - NUMBER OF DATA POINTS.
C      * W - REAL ARRAY OF DIMENSION N × 1 OF DATA
C      *      WEIGHTS. THE WEIGHTS SHOULD BE SYMMETRICAL
C      *      ABOUT THE POINT (N+1)/2.
C      * NEXP - THE NUMBER OF FREQUENCY IS 2**NEXP
C      * OUTPUT PARAMETERS:
C      * PPER - REAL ARRAY OF DIMENSION L=2**NEXP×1 OF
C      *      SAMPLES OF THE PERIODOGRAM, WHERE
C      *      PPER(I) CORRESPONDS TO THE SPECTRAL
C      *      ESTIMATE AT THE FREQUENCY  $F = -1/2 + (I-1)/L$ 
C      *
C      * REFERENCE: S.M. KAY " MODERN SPECTRUM ESTIMATION"
C      *
C      *****
C
      COMPLEX X(512)
      DIMENSION W(512),PPER(512)
C
C      WINDOW THE DATA
C
      DO 10 I=1,N
10    X(I)=X(I)*W(I)
C      COMPUTE FFT OF WINDOWED DATA
      L=2**NEXP
      IF(L.EQ.N)GO TO 25
      DO 20 I=N+1,L
20    X(I)=(0.,0.)

```



```

25      INVRS=1
      CALL FFT(X,NEXP,INVRS)
C C    COMPUTE SAMPLES OF PERIODOGRAM AND TRANSPOSE HALVES OF
C      FFT OUTPUT SO THAT FIRST PSD SAMPLE IS AT A FREQUENCY OF -1/2
C      DO 30 I=1,L/2
      PPER(I+L/2)=(CABS(X(I))**2)/N
30      PPER(I)=(CABS(X(I+L/2))**2)/N
      RETURN
      END

```

```

      SUBROUTINE HAMMING(W,N)
      REAL W(512)

C
C      *****
C      *
C      * FUNCTION: GENERATE THE HAMMING WINDOW.
C      * PARAMETERS:
C      * INPUT: N - ARRAY SIZE
C      * OUTPUT: W - HAMMING WINDOW.
C      *
C      *****
C

      PI=4.*ATAN(1.) ! DEFINE PI
      DO J=1,N ! GENERATE HAMMING WINDOW
          W(J)=0.54-0.46*COS(2.*PI*FLOAT(J)/FLOAT(N))
      ENDDO
      RETURN
      END

```

```

      SUBROUTINE ZPVEP(AS,NS,N,MPOINTS,NEXP,NAS)

C
C      *****
C      *
C      * FUNCTION: ZERO PADDING A ARRAY FROM N+1 TO
C      *              2**NEXP
C      * PARAMETERS:
C      * INPUT:
C      *      AS : ARRAY SIGNAL
C      *      NS : NUMBER OF SNAP SHOT
C      *      N : NUMBER OF DATA POINTS
C      *      NAS: DYNAMIC ARRAY SIZE DECLARATION
C      * OUTPUT:
C      *      MPOINTS: THE EXTENDED DATA POINTS
C      *      NEXP: THE SMALLEST INTEGER SUCH THAT
C      *              2**NEXP ≥ MPOINTS.
C

```



```

C      *
C      *****
C
COMPLEX AS(NAS,NAS)
CHARACTER *1 ANSWER
C
WRITE(*,*)'ZERO PADDING DATA ? (N)'
READ(5,5) ANSWER
5  FORMAT(A1)
IF((ANSWER.EQ.'y').OR.(ANSWER.EQ.'Y')) THEN
    WRITE(*,*)'HOW MANY POINTS'
    READ(*,*)MPOINTS
    CALL DO_NEXP(MPOINTS,NEXP)
    NPSD = 2**NEXP
    DO I = 1,NS
    DO J = N+1,NPSD
    AS(J,I) = (0.,0.)
    ENDDO
    ENDDO
ELSE
    CALL DO_NEXP(N,NEXP)
ENDIF
RETURN
END

SUBROUTINE DO_NEXP(N,NEXP)
C
C      *****
C      *
C      * FUNCTION: FIND AN INTEGER EXPONENTIAL SUITABLE *
C      *          FOR FFT *
C      * PARAMETERS *
C      * INPUT: *
C      *      N: NUMBER OF DATA *
C      * OUTPUT: *
C      *      NEXP: THE SMALLEST INTEGER THAT 2**NEXP ≥ N *
C      *
C      *****
C
FN=FLOAT(N)
NEXP=INT(LOG(FN)/LOG(2.))
IF(MOD(LOG(FN),LOG(2.)).NE.0.)NEXP=NEXP+1
RETURN
END

SUBROUTINE DISPLAY_3D(S,NPSD,NS,NAS)

```



```

C
REAL S(NAS,NAS),X(512),Y(512,512),Z(512)
REAL YMAX,YMIN,XMAX,XMIN,ZMAX,ZMIN
REAL SIZFAC,SIZRAT,HEIFAC
DATA SIZFAC,SIZRAT,HEIFAC/0.9,1.6,0.07/
CHARACTER *1 ANSWER

C
C *****
C *
C * FUNCTION: DISPLAY A DIGITAL SIGNAL USING UNIRAS 6.3 *
C * PARAMETERS: *
C * INPUT: S - DIGITAL SIGNAL *
C *      NPSD - NUMBER OF PSD ( leq 512) *
C *      NS - NUMBER OF SNAP SHOT *
C *      NAS - DYNAMIC MEMORY ALLOCATION *
C * *
C * OUTPUT: NONE *
C * *
C *INTERNAL SUBROUTINES: AXIS, PLOT *
C * *
C *****
C
DO I = 1,NS
DO J = NPSD/2+1,NPSD
      Y(J,I) = 10*LOG(S(J,I)+1) ! S unchanged on exit
ENDDO
ENDDO

C
C COMPUTE LARGEST VALUE OF Y: YMAX.
C
YMAX = 0.
DO I = 1,NS
DO J = NPSD/2+1,NPSD
      IF(Y(J,I).GT.YMAX) YMAX = Y(J,I)
ENDDO
ENDDO

C
C COMPUT THE SMALLEST VALUE OF Y: YMIN
C
YMIN = YMAX
DO I = 1,NS
DO J = NPSD/2+1,NPSD
      IF(Y(J,I).LT.YMIN) YMIN = Y(J,I)
ENDDO
ENDDO

C
C DEFINE THE MINIMUM AND MAXIMUM VALUES OF THE X-AIXS.
C
XMIN = 0.

```



```

      XMAX = FLOAT(NS)
C
C      DEFINE THE MINIMUM AND MAXIMUM VALUES OF THE Z-AXIS
C
      FNPSD = FLOAT(NPSD)
      DZ = 1./FNPSD
      ZMIN = 0.
      DO J = 1,NPSD/2
          FJ= FLOAT(J)
          Z(J) = FJ * DZ
      ENDDO
      ZMAX = 0.5
C
C      PLOT AXIS.
C
      CALL AXIS3D(XMIN,XMAX,YMIN,YMAX,ZMIN,ZMAX,SIZEFAC,HEIFAC)
C
C      PLOT SIGNAL.
C
      CALL PLOT3D(X,Y,Z,NPSD,NS,NAS)
      CALL GEMPTY ! Empty Buffer
C
C      KEEP PLOT UNTIL USER 'HITS' THE RETURN KEY.
C
      WRITE(*,*)'HIT RETURN TO CONTINUE'
      READ '(A)'
C
C      PROMPT USER FOR GENERATING A HARDCOPY
C
      WRITE(*,*)'FILE GRAPH ? (N)'
      READ(5,10)ANSWER
10    FORMAT(A1)
      IF((ANSWER.EQ.'y').OR.(ANSWER.EQ.'Y')) THEN
          CALL WRITE3D(Y,NS,NPSD,NAS)
      ENDIF
C
C      CLEAR PLOT IN UNIRAS WINDOW.
C
      CALL GCLEAR
C
      RETURN
      END
C
C
C
      SUBROUTINE AXIS3D(XMIN,XMAX,YMIN,YMAX,ZMIN,ZMAX,SIZEFAC,HEIFAC)
C
C      FUNCTION: PLOTS THE AXES OF THE USER 3D COORDINATE SYSTEM.
C

```



```

REAL XMIN,YMIN,XMAX,YMAX,ZMIN,ZMAX,WID
REAL XLEN,YLEN,ZLEN,XSIZE,YSIZE,SIZE,XSIZMM,YSIZMM
REAL SIZFAC, HEIFAC, HEIGHB
REAL BOX2MM
INTEGER LENGTH(6),PLANE(3)
CHARACTER *20 TEXT(6)
DATA IGNORE/9999/
DATA RIGNOR/999.999/
DATA TEXT/' ','SNAP SHOT',' ','AMPLITUDE',' ','FREQUENCY'/
DATA PLANE/1,1,1/
DATA LENGTH/0.,-2,0,-2,0,-2/
EXTERNAL BOX2MM

C
C STEP 1: DEFINE THE 3D LIMITS OF THE AXES
C
CALL GLIMIT(XMIN,XMAX,YMIN,YMAX,ZMIN,ZMAX)
C
C STEP 2: CALCULATE FOR A SUITABLE SQUARE VIEWPORT SIZE
C
CALL GRPSIZ(XSIZMM,YSIZMM)
SIZE = SIZFAC * MIN(XSIZMM,YSIZMM)
XSIZE = SIZE
YSIZE = SIZE
C
C SET UP THE VIEWPORT
C
XORG = (XSIZMM-XSIZE)/2.
YORG = (YSIZMM-YSIZE)/2.
C
CALL GVPORT(XORG,YORG,XSIZE,YSIZE)
C
C STEP 3: DEFINE THE 3D WORKBOX ASPECT RATIO
C TO BE EQUAL TO THE VIEWPORT'S.
C ASSIGN THE SHORTER SIDE OF THE WORLBOX TO BE 1 AND THE OTHER
C SIDE PROPORTION TO IT.
C
XLEN = XSIZE/MIN(XSIZE,YSIZE)
YLEN = YSIZE/MIN(XSIZE,YSIZE)
ZLEN = MAX(XLEN,YLEN)
CALL GWBOX(XLEN,YLEN,ZLEN)
C
C STEP 4: CALCULATE A SUITABLE AXIS-LABEL HEIGHT BASED ON THE SHORTER
C SIDE OF THE WORKBOX. THE FUNCTION "BOX2MM" CONVERTS BOX UNITS
C TO mm.
C
HEIGHB = HEIFAC * MIN(XSIZE,YSIZE)
C
C STEP 5: PLOT TWO NUMERIC AND USER-LABELED AXES AT THE LIMITS OF
C THE 3D COORDINATE SYSTEM.

```



```

C
CALL GEYE(2.*XLEN,20.*YLEN,2.*ZLEN)
CALL RAXDIS(1,1,IGNORE)
CALL RAXDIS(6,1,IGNORE)
CALL RAXDIS(2,1,IGNORE)
CALL RAXDIS(3,1,IGNORE)
CALL RAXDIS(4,1,IGNORE)
CALL RAXDIS(7,1,IGNORE)
CALL RAXTIL(7,2,RIGNOR,IGNORE)
DO IAXIS = 1,3
CALL RAXIS3(IAXIS,HEIGHB,PLANE,LENGTH,TEXT)
ENDDO
RETURN
END

C
C
C

SUBROUTINE PLOT3D(X,Y,Z,NPSD,NS,NAS)
REAL X(512),Y(NAS,NAS),Z(512),YY(512),XLEN,YLEN,ZLEN

C
C FUNCTION: PLOTS THE SIGNAL
C

CALL GVPROJ(1) ! Y Up Coordinate
CALL GACT(-1) ! This is to enquire the GBOX parameters
CALL GWBOX(XLEN,YLEN,ZLEN)
CALL GEYE(2.*XLEN,20.*YLEN,2.*ZLEN)
IPF = NPSD/2
DO I = 1,NS
DO J = 1,NPSD/2
YY(J) = Y(J+IPF,I)
X(J) = I ! Snap Shot Coordinate
ENDDO
CALL BLIVE3(X,YY,Z,NPSD/2,ITREAT)
ENDDO
RETURN
END

SUBROUTINE EIGENFREQ3D(METHOD,NS,NEXP,N,IP,NSIG,X,NAS,PSD3D,ISTAT)

C
C *****
C
C *
C * FUNCTION: EXECUTES THE EIGENANALYSIS BASED PSD
C * ESTIMATION.
C * INPUT PARAMETERS:
C *
C * METHOD - SET TO 1 FOR MUSIC METHOD, 2 FOR EV METHOD *
C * NEXP - POWER OF 2

```



```

C      * N - NNUMBER OF DATA SAMPLES *
C      * IP - DIMENTION OF DATA MATRIX *
C      * NSIG - ASSUMED NUMBER OF "SIGNAL SPACE" VECTORS *
C      * X - ARRAY OF COMPLEX DATA SAMPLES *
C      * *
C      * OUTPUT PARAMETERS: *
C      * *
C      * PSD3D - ARRAY OF REAL FREQUENCY ESTIMATOR VALUES. *
C      * ISTAT - STATUS INDICATOR : *
C      * 0 FOR NORMAL EXIT *
C      * 1 IF NSIG > IP-1 OR NSIG < 0 *
C      * *
C      * REFERENCE: S.L. MARPLE,Jr. "DIGITAL SPECTRAL ANALYSIS *
C      *              WITH APPLICATION". *
C      * *
C      *****
C
PARAMETER MAXU=100 ! Maximum Dimention of U Matrix
PARAMETER MAXV=100 ! Maximum Dimention of V,S Matrix
COMPLEX X(NAS,NAS),FB(MAXU,MAXV),U(MAXU,MAXU),V(MAXV,MAXV)
COMPLEX W(512),Z(512)
REAL PSD3d(512,512),S(MAXV),P(512)

C
PI2=8.*ATAN(1.)
NPSD = 2 ** NEXP
ISTAT=0
IF (NSIG .LT. IP .AND. NSIG .GE. 0) GO TO 5
ISTAT=1
RETURN
5  NP=N-IP
DO 140 INS = 1,NS
DO 20 I=1,NP
DO 10 K=1,IP
      FB(I,K)=X(I-K+IP,INS)
10      FB(I+NP,K)=CONJG(X(I+K,INS))
20 CONTINUE
NP2=2*NP

C
CALL CSVD (FB,MAXU,MAXV,NP2,IP,0,IP,IP,S,U,V)

C
C  INITIALIZE PSD BUFFER TO ZERO
C
DO 30 K=1,NPSD
30      P(K)=0.
DO 70 I=NSIG+1,IP
DO 40 K=1,IP
40      Z(K)=V(K,I) ! Copy The Components of Ith Eigenvector to Z
DO 50 K=IP+1,NPSD !Zero Padding The Data Vector
50      Z(K)=(0.,0.)

```



```

      INVRS=1
      CALL FFT (Z,NEXP,INVRS)
      DO 60 K=1,NPSD
          TEMP=REAL(Z(K))**2+AIMAG(Z(K))**2
          IF (METHOD .EQ. 1) P(K)=P(K)+TEMP
60      IF (METHOD .eq. 2) P(K)=P(K)+TEMP/S(I)
70      CONTINUE
      DO 80 K=1,NPSD
90      P(K)=1./P(K)
      DO 130 J=1,NPSD/2
          PSD3D(J+NPSD/2,INS)=P(J) ! Zero Frequency at The Center
130      PSD3D(J,INS) = P(J+NPSD/2)
140      CONTINUE
      RETURN
      END

```

```

SUBROUTINE FFT(X,M,INVRS)
C
C
C *****
C *
C * FUNCTION: THIS PROGRAM COMPUTES THE FAST
C * TRANSFORM OF AN ARRAY OF COMPLEX DATA*
C * USING DECIMATION-IN-TIME FFT
C *
C * INPUT PARAMETERS:
C *
C * X - COMPLEX ARRAY OF DIMENSION 2**Mx1 OF DATA
C * SAMPLES.
C * M - POWER OF TWO FOR FFT SIZE
C * INVRS - SET EQUAL TO:
C * 1 FOR FORWARD FOURIER TRANSFORM
C * -1 FOR INVERSE FOURIER TRANSFORM
C *
C * OUTPUT PARAMETERS:
C *
C * X - COMPLEX ARRAY OF DIMENSION 2**Mx1
C * OF FOURIER TRANSFORM VALUES
C *
C * REFERENCE: S.M. KAY: " MODERN SPECTRAL ESTIMATION*
C *
C *****
C
C
      COMPLEX X(512),Y(512),W,SAVE
      N=2**M
C      PUT THE DATA INTO BIT REVERSED FORM
      Y(1)=X(1)

```



```

DO 20 I=2,N
    NUM=I-1
    J=0
    L=N
DO 10 K=1,M
    L=L/2
    IF(NUM.LT.L)GO TO 10
    J=J+2**(K-1)
    NUM=NUM-L
10  CONTINUE
    Y(I)=X(J+1)
20  CONTINUE
C
C  BEGIN COMPUTATION OF FFT
C
    PI=4.*ATAN(1.)
    LDFT=1
    NDFT=N
    DO 50 K=1,M
        LDFT=2*LDFT
        NDFT=NDFT/2
    DO 40 I=1,NDFT
        DO 30 J=1,LDFT/2
C
C  DETERMINE TWIDDLE FACTOR
C
        ARG=-(2.*PI*INVRS)*(J-1.)/LDFT
        W=CMPLX(COS(ARG),SIN(ARG))
C
C  DETERMINE INDICES FOR NEXT BUTTERFLY TO BE COMPUTED
C
        NP=J+LDFT*(I-1)
        NQ=NP+LDFT/2
C
C  COMPUTE BUTTERFLY FOR THE ITH DFT IN STAGE K
C
        SAVE=Y(NP)+W*Y(NQ)
        Y(NQ)=Y(NP)-W*Y(NQ)
        Y(NP)=SAVE
30  CONTINUE
40  CONTINUE
50  CONTINUE
C
C  DIVIDE ARRAY VALUES BY N IF INVERSE FFT DESIRED
C
    DO 60 I=1,N
        IF(INVRS.EQ.1)X(I)=Y(I)
60  IF(INVRS.EQ.-1)X(I)=Y(I)/N
RETURN

```


END

SUBROUTINE CSVD (A,MMAX,NMAX,M,N,IP,NU,NV,S,U,V)

```
C
C
C *****
C *
C * FUNCTION: SINGULAR VALUE DECOMPOSITION OF AN *
C *      M × N COMPLEX MATRIX A, *
C *      WHERE M ≥ N. THE SINGULAR *
C *      VALUES ARE STORED IN THE VECTOR S. *
C *
C * REFERENCE: P.A. Businger and G.H. Golub, *
C *      "Singular Value Decomposition of a Complex Matrix," *
C *      Communications of the ACM, vol. 12, *
C *      pp. 564-565, October 1969. *
C *****
C
C      COMPLEX A(100,100),U(100,100),V(100,100),Q,R
C      REAL S(100),B(100),C(100),T(100)
C      DATA ETA,TOL/1.2E-7,2.4E-32/
C      NP=N+IP
C      N1=N+1
C
C      HOUSEHOLDER REDUCTION
C
C      C(1)=0.
C      K=1
10    K1=K+1
C
C      ELIMINATION OF A(I,K) , I=K+1,...,M
C
C      Z=0.
C      DO 20 I=K,M
20    Z=Z+REAL(A(I,K))**2+AIMAG(A(I,K))**2
C      B(K)=0.
C      IF (Z .LE. TOL) GO TO 70
C      Z=SQRT(Z)
C      B(K)=Z
C      W=CABS(A(K,K))
C      Q=(1.,0.)
C      IF (W .NE. 0.) Q=A(K,K)/W
C      A(K,K)=Q*(Z+W)
C      IF (K .EQ. NP) GO TO 70
C      DO 50 J=K1,NP
C      Q=(0.,0.)
```



```

DO 30 I=K,M
30  Q=Q+CONJG(A(I,K))*A(I,J)
    Q=Q/(Z*(Z+W))
    DO 40 I=K,M
40  A(I,J)=A(I,J)-Q*A(I,K)
50  CONTINUE
C
C    PHASE TRANSFORMATION
C
    Q=-CONJG(A(K,K))/CABS(A(K,K))
    DO 60 J=K1,NP
60  A(K,J)=Q*A(K,J)
C
C    ELIMINATION OF A(K,J) , J=K+2,...,N
C
70  IF (K .EQ. N) GO TO 140
    Z=0.
    DO 80 J=K1,N
80  Z=Z+REAL(A(K,J))**2+AIMAG(A(K,J))**2
    C(K1)=0.
    IF (Z .LE. TOL) GO TO 130
    Z=SQRT(Z)
    C(K1)=Z
    W=CABS(A(K,K1))
    Q=(1.,0.)
    IF (W .NE. 0.) Q=A(K,K1)/W
    A(K,K1)=Q*(Z+W)
    DO 110 I=K1,M
    Q=(0.,0.)
    DO 90 J=K1,N
90  Q=Q+CONJG(A(K,J))*A(I,J)
    Q=Q/(Z*(Z+W))
    DO 100 J=K1,N
100 A(I,J)=A(I,J)-Q*A(K,J)
110 CONTINUE
C
C    PHASE TRANSFORMATION
C
    Q=-CONJG(A(K,K1))/CABS(A(K,K1))
    DO 120 I=K1,M
120 A(I,K1)=A(I,K1)*Q
130 K=K1
    GO TO 10
C
C    TOLERANCE FOR NEGLIGIBLE ELEMENTS
C
140 EPS=0.
    DO 150 K=1,N
    S(K)=B(K)

```



```

      T(K)=C(K)
150    EPS=AMAX1(EPS,S(K)+T(K))
      EPS=EPS*ETA
C
C      INITIALIZATION OF U AND V
C
      IF (NU .EQ. 0) GO TO 180
      DO 170 J=1,NU
      DO 160 I=1,M
160    U(I,J)=(0.,0.)
170    U(J,J)=(1.,0.)
180    IF (NV .EQ. 0) GO TO 210
      DO 200 J=1,NV
      DO 190 I=1,N
190    V(I,J)=(0.,0.)
200    V(J,J)=(1.,0.)
C
C      QR DIAGONALIZATION
C
210    DO 380 KK=1,N
      K=N1-KK
C
C      TEST FOR SPLIT
C
220    DO 230 LL=1,K
      L=K+1-LL
      IF (ABS(T(L)) .LE. EPS) GO TO 290
      IF (ABS(S(L-1)) .LE. EPS) GO TO 240
230    CONTINUE
C
C      CANCELLATION OF B(L)
C
240    CS=0.
      SN=1.
      L1=L-1
      DO 280 I=L,K
      F=SN*T(I)
      T(I)=CS*T(I)
      IF (ABS(F) .LE. EPS) GO TO 290
      H=S(I)
      W=SQRT(F*F+H*H)
      S(I)=W
      CS=H/W
      SN=-F/W
      IF (NU .EQ. 0) GO TO 260
      DO 250 J=1,N
      X=REAL(U(J,L1))
      Y=REAL(U(J,I))
      U(J,L1)=CMPLX(X*CS+Y*SN,0.)

```



```

250    U(J,I)=CMPLX(Y*CS-X*SN,0.)
260    IF (NP .EQ. N) GO TO 280
        DO 270 J=N1,NP
            Q=A(L1,J)
            R=A(I,J)
            A(L1,J)=Q*CS+R*SN
270    A(I,J)=R*CS-Q*SN
280    CONTINUE
C
C    TEST FOR CONVERGENCE
C
290    W=S(K)
        IF (L .EQ. K) GO TO 360
C
C    ORIGIN SHIFT
C
        X=S(L)
        Y=S(K-1)
        G=T(K-1)
        H=T(K)
        F=((Y-W)*(Y+W)+(G-H)*(G+H))/(2.*H*Y)
        G=SQRT(F*F+1.)
        IF (F .LT. 0.) G=-G
        F=((X-W)*(X+W)+(Y/(F+G)-H)*H)/X
C
C    QR STEP
C
        CS=1.
        SN=1.
        L1=L+1
        DO 350 I=L1,K
            G=T(I)
            Y=S(I)
            H=SN*G
            G=CS*G
            W=SQRT(H*H+F*F)
            T(I-1)=W
            CS=F/W
            SN=H/W
            F=X*CS+G*SN
            G=G*CS-X*SN
            H=Y*SN
            Y=Y*CS
            IF (NV .EQ. 0) GO TO 310
            DO 300 J=1,N
                X=REAL(V(J,I-1))
                W=REAL(V(J,I))
                V(J,I-1)=CMPLX(X*CS+W*SN,0.)
300    V(J,I)=CMPLX(W*CS-X*SN,0.)

```



```

310    W=SQRT(H*H+F*F)
      S(I-1)=W
      CS=F/W
      SN=H/W
      F=CS*G+SN*Y
      X=CS*Y-SN*G
      IF (NU .EQ. 0) GO TO 330
      DO 320 J=1,N
      Y=REAL(U(J,I-1))
      W=REAL(U(J,I))
      U(J,I-1)=CMPLX(Y*CS+W*SN,0.)
320    U(J,I)=CMPLX(W*CS-Y*SN,0.)
330    IF (N .EQ. NP) GO TO 350
      DO 340 J=N1,NP
      Q=A(I-1,J)
      R=A(I,J)
      A(I-1,J)=Q*CS+R*SN
340    A(I,J)=R*CS-Q*SN
350    CONTINUE
      T(L)=0.
      T(K)=F
      S(K)=X
      GO TO 220

C
C    CONVERGENCE
C
360    IF (W .GE. 0.) GO TO 380
      S(K)=-W
      IF (NV .EQ. 0) GO TO 380
      DO 370 J=1,N
370    V(J,K)=-V(J,K)
380    CONTINUE
C
C    SORT SINGULAR VALUES
C
      DO 450 K=1,N
      G=-1.
      J=K
      DO 390 I=K,N
      IF (S(I) .LE. G) GO TO 390
      G=S(I)
      J=I
390    CONTINUE
      IF (J .EQ. K) GO TO 450
      S(J)=S(K)
      S(K)=G
      IF (NV .EQ. 0) GO TO 410
      DO 400 I=1,N
      Q=V(I,J)

```



```

      V(I,J)=V(I,K)
400    V(I,K)=Q
410    IF (NU .EQ. 0) GO TO 430
      DO 420 I=1,N
      Q=U(I,J)
      U(I,J)=U(I,K)
420    U(I,K)=Q
430    IF (N .EQ. NP) GO TO 450
      DO 440 I=N1,NP
      Q=A(J,I)
      A(J,I)=A(K,I)
440    A(K,I)=Q
450    CONTINUE
C
C    BACK TRANSFORMATION
C
      IF (NU .EQ. 0) GO TO 510
      DO 500 KK=1,N
      K=N1-KK
      IF (B(K) .EQ. 0.) GO TO 500
      Q=-A(K,K)/CABS(A(K,K))
      DO 460 J=1,NU
460    U(K,J)=Q*U(K,J)
      DO 490 J=1,NU
      Q=(0.,0.)
      DO 470 I=K,M
470    Q=Q+CONJG(A(I,K))*U(I,J)
      Q=Q/(CABS(A(K,K))*B(K))
      DO 480 I=K,M
480    U(I,J)=U(I,J)-Q*A(I,K)
490    CONTINUE
500    CONTINUE
510    IF (NV .EQ. 0) GO TO 570
      IF (N .LT. 2) GO TO 570
      DO 560 KK=2,N
      K=N1-KK
      K1=K+1
      IF (C(K1) .EQ. 0.) GO TO 560
      Q=-CONJG(A(K,K1))/CABS(A(K,K1))
      DO 520 J=1,NV
520    V(K1,J)=Q*V(K1,J)
      DO 550 J=1,NV
      Q=(0.,0.)
      DO 530 I=K1,N
530    Q=Q+A(K,I)*V(I,J)
      Q=Q/(CABS(A(K,K1))*C(K1))
      DO 540 I=K1,N
540    V(I,J)=V(I,J)-Q*CONJG(A(K,I))
550    CONTINUE

```


560 CONTINUE
570 RETURN
END

Appendix B

Assembly Program for Sequential Periodogram

```

;          512-Point , Non-In-Place FFT.
;
od          ident 1,0
            page 132,60,3,3,3
;          External data starts at address $1000
;          Internal data starts at address 0
;          Coefficient table starts at address $1200
;          Output data starts at $1400

Reset      equ          $40
Start      equ          $100      ;Starting address of program
Points     equ          512      ;512 points FFT
Scaling    equ          $60000    ;Scaling factor
Data       equ          $1000     ;Input data starting address
Coef       equ          $1200     ;Coefficient starting address
Odata      equ          $1400     ;Output data buffer
Idata      equ          $1600     ;Intermediate data buffer
Psd_data   equ          $1800     ;psd data starting address
Intdata    equ          $0        ;Internal data starting address
Bcr        equ          $ffe      ;Bus control register
Vipr       equ          $f000     ;Sci timer interrupt prioroty = 1
                                ;Ssi interrupt priority = 1
Ipr        equ          $fff      ;Interrupt priority register
Pcc        equ          $ffe1     ;Port C control register
Vpcc       equ          $0001ff   ;Bir pattern for PCC
Cra        equ          $ffec     ;Port C control register A
Vcra       equ          $004000   ;Word length = 16 bit
Crb        equ          $ffed     ;Port C control register B
Vcrb       equ          $00b200   ;Continuous and sync mode,Rx interrupt enable
Ssir       equ          $fee      ;SSI status register
Rx         equ          $fef      ;SSI RX data register
Tx         equ          $fef      ;SSI TX data register
Scr        equ          $ff0      ;SCI control register
Vscr       equ          $0000     ;Select timer interrupt
```



```

Vmr      equ      $fc      ;set interrupt level =0
Ssi_rxe_ira equ      $000e      ;Address of RX with excpt interrupt vector
Ssi_txe_ira equ      $0012      ;Address of TX with excpt interrupt vector
Ssi_rx_ira equ      $000c      ;Ssi rx interrupt address
Trigger  equ      $5        ;Trigger period for oscilloscope
Pbd      equ      $ffe4      ;Port B data register
Vpbd     equ      $fff      ;Port B output value
Pbaddr   equ      $ffe2      ;Port B data direction register
Pbc      equ      $ffe0      ;Port B control register
Vpbc     equ      $0        ;Set port B as parallel I/O
Vfft_s   equ      $fff      ;fft ready flag
Divisor  equ      240        ;divide the sampling frequency 48k → 200hz
Counter  equ      8          ;8 samples
IP        equ      $1300      ;Tempelory input buffer pointer
IPA       equ      $1380      ;Input buffer
Ptr_r0    equ      $1301      ;Input buffer pointer r0
S_nr      equ      $1302      ;Sample number
Skip_nr   equ      $1303      ;The kill number
FFT_s     equ      $1304      ;fft status
S_r0      equ      $1305
S_m0      equ      $1306
S_r1      equ      $1307
S_m1      equ      $1308
S_r4      equ      $1309
S_m4      equ      $130a
S_r7      equ      $130b
S_m7      equ      $130c
S_x0      equ      $130d
S_x1      equ      $130e
S_a1      equ      $130f
S_b1      equ      $1310
Disp_r1   equ      $1311
Rxe_a     equ      $1312
Txe_a     equ      $1313
Pi         equ      3.141592654
Freq       equ      2.0*Pi/@cvf(Points)

```

```

;*****

```

```

      org      x:Coef
count set      0
      dup      Points/2
      dc       -@cos(@cvf(count)*Freq)
count set      count+1
      endm

```

```

      org      y:Coef
count set      0

```



```

        dup    Points/2
        dc     -@sin(@cvf(count)*Freq)
count    set    count+1
        endm

        org    p:Reset
        jmp    Start

        org    p:Start
Begin
        movec   #0,sp           ;init stack pointer
        movec   #0,sr           ;Clear Loop flag
        jsr     Init
Loop
        move    x:FFT_s,x0       ;wait for fft ready
        move    #Vfft_s,a
        cmp     x0,a
        jne     Loop
        move    #0,x0           ;reset fft ready status
        move    x0,x:FFT_s
        jsr     Do_FFT
        jmp     Loop

;*****
;
;Subroutines

Init     movep   #0,x:Bcr        ;Set bcr to zero wait state

;Set up ssi for operation with DSP56ADC16EVB

        movep   #0,x:Pcc        ;Zero pcc to cycle it
        movep   #Vpcc,x:Pcc     ;Set port c to function as ssi
        movep   #Vcra,x:Cra     ;CRA pattern for word length: 16 bit
        movep   #$3200,x:Crb    ;CRB pattern for continuous clock, sync
                                ;but not enable the rxi yet

;Initialize address pointers

        move    #Data,r0        ;ro points to input data starting address
        move    r0,r4
        move    #Points-1,m0    ;Module 512
        move    m0,m4
        move    #0,y0          ;Init Input Imaginary Data

;Port B

        movep   #Vpbc,x:Pbc     ;set up port B as parallel IO
        movep   #$fff,x:Pbaddr  ;set data direction: o/p

```


;Initialize the counter values

```

move    #0,x0
move    x0,x:Skip_nr    ;Init the Skip_nr counter
move    x0,x:S_nr       ;init the sample number
move    #IPA,x1
move    x1,x:IP          ;Points to the temporary buffer
move    #Data,x0
move    x0,x:Ptr_r0      ;Points to the beginning of the buffer
move    #Psd_data,x0
move    x0,x:Disp_r1     ;Points to the begining of PSD

```

;Collect first 512 samples

```

move    #>Scaling,x1    ;get the scaling factor
do      #Points,End_real_data
Poll512 jclr    #7,x:Ssisr,Poll512;Store data in x memory
move    x:Rx,x0          ;get data
mpy     x0,x1,a          ;scaling
move    a,x:(r0)+        y0,y:(r4)+
End_real_data
move    #Vfft_s,x0       ;Init FFT_s to ready
move    x0,x:FFT_s
andi    #Vmr,mr          ;Enable interrupt
movep   #Vcrb,x:Crb      ;enable the rxi

```

=====

```

movep   #Vscr,x:Scr      ;Disable the SCI timer interrupt
movep   #Vipr,x:Ipr      ;Set the interrupr priority level=3
rts

```

=====

Do_FFT

```

move    x:Ptr_r0,r0      ;initialize input pointer
move    #Points/4,n0      ;initialize input and output pointers offset
move    n0,n4
move    n0,n6             ;initialize coefficient offset
move    #Points-1,m0      ;initialize address modifiers
move    m0,m1             ;for modulo addressing
move    m0,m4
move    m0,m5
move    #Idata,r2         ;initialize the intermediate pointer
move    n0,n2
move    m0,m2

```

;

; Do first and second Radix 2 FFT passes, combined as 4-point butterflies

;

```

move    x:(r0)+n0,x0

```



```

    tfr    x0,a      x:(r0)+n0,y1
    do     n0,_twopass
    tfr    y1,b      x:(r0)+n0,y0
    add    y0,a      x:(r0),x1      ;ar+cr
    add    x1,b      r0,r4          ;br+dr
    add    a,b      (r0)+n0        ;ar'=(ar+cr)+(br+dr)
    subl   b,a      b,x:(r2)+n2    ;br'=(ar+cr)-(br+dr)
    move   (r0)+n0
    tfr    x0,a      a,x0          y:(r0),b
    sub    y0,a      y:(r4)+n4,y0  ;ar-cr
    sub    y0,b      x0,x:(r2)+n2  ;bi-di
    add    a,b      y:(r0)+n0,x0  ;cr'=(ar-cr)+(bi-di)
    subl   b,a      b,x:(r2)+n2    ;dr'=(ar-cr)-(bi-di)
    tfr    x0,a      a,x0          y:(r4),b
    add    y0,a      y:(r0)+n0,y0  ;bi+di
    add    y0,b      x0,x:(r2)+n2  ;ai+ci
    move   (r0)+n0
    add    b,a      y:(r0)+,x0     ;ai'=(ai+ci)+(bi+di)
    subl   a,b      a,y:(r2)+n2   ;bi'=(ai+ci)-(bi+di)
    tfr    x0,a      b,y:(r2)+n2
    sub    y0,a      x1,b          ;ai-ci
    sub    y1,b      x:(r0)+n0,x0  ;dr-br
    add    a,b      x:(r0)+n0,y1   ;ci'=(ai-ci)+(dr-br)
    subl   b,a      b,y:(r2)+n2   ;di'=(ai-ci)-(dr-br)
    tfr    x0,a      a,y:(r2)+n2
    move   (r2)+
_twopass
;
; Perform all next FFT passes except last pass with triple nested DO loop
;

    move   #Points/8,n1      ;initialize butterflies per group
    move   #4,n2             ;initialize groups per pass
    move   #-1,m2            ;linear addressing for r2
    move   #0,m6             ;initialize C address modifier for
                                ;reverse carry (bit-reversed) addressing
    do     #@cvi(@log(Points)/@log(2)-2.5),_end_pass
                                ;6 passes for 512 pt. FFT
    move   #Idata,r0         ;initialize A input pointer
    move   r0,r1
    move   n1,r2
    move   r0,r4             ;initialize A output pointer
    move   (r1)+n1           ;initialize B input pointer
    move   r1,r5             ;initialize B output pointer
    move   #Coef,r6          ;initialize C input pointer
    lua    (r2)+,n0          ;initialize pointer offsets
    move   n0,n4
    move   n0,n5
    move   (r2)-             ;butterfly loop count
    move   x:(r1),x1      y:(r6),y0  ;lookup -sine and -cosine values

```



```

    move          x:(r6)+n6,x0  y:(r0),b      ;update C pointer, preload data
    mac   x1,y0,b                                y:(r1)+,y1
    macr  -x0,y1,b                                y:(r0),a
    do    n2,_end_grp
    do    r2,_end_bfy
    subl  b,a      x:(r0),b      b,y:(r4)      ;Radix 2 DIT butterfly kernel
    mac   -x1,x0,b  x:(r0)+,a      a,y:(r5)
    macr  -y1,y0,b  x:(r1),x1
    subl  b,a      b,x:(r4)+      y:(r0),b
    mac   x1,y0,b                                y:(r1)+,y1
    macr  -x0,y1,b  a,x:(r5)+      y:(r0),a
_end_bfy
    move   (r1)+n1
    subl  b,a      x:(r0),b      b,y:(r4)
    mac   -x1,x0,b  x:(r0)+n0,a  a,y:(r5)
    macr  -y1,y0,b  x:(r1),x1    y:(r6),y0
    subl  b,a      b,x:(r4)+n4    y:(r0),b
    mac   x1,y0,b  x:(r6)+n6,x0  y:(r1)+,y1
    macr  -x0,y1,b  a,x:(r5)+n5  y:(r0),a
_end_grp
    move   n1,b1
    lsr    b        n2,a1          ;divide butterflies per group by two
    lsl    a        b1,n1          ;multiply groups per pass by two
    move   a1,n2
_end_pass
;
; Do last FFT pass
;
    move   #2,n0          ;initialize pointer offsets
    move   n0,n1
    move   #Points/4,n4    ;output pointer A offset
    move   n4,n5          ;output pointer B offset
    move   #Idata,r0       ;initialize A input pointer
    move   #Odata,r4       ;initialize A output pointer
    move   r4,r2          ;save A output pointer
    lua    (r0)+,r1        ;initialize B input pointer
    lua    (r2)+n2,r5      ;initialize B output pointer
    move   #0,m4          ;bit-reversed addressing for output ptr. A
    move   m4,m5          ;bit-reversed addressing for output ptr. B
    move   #Coef,r6       ;initialize C input pointer
    move   (r5)-n5        ;predecrement output pointer
    move   x:(r1),x1  y:(r6),y0
    move   x:(r5),a  y:(r0),b
    do    n2,_lastpass
    mac   x1,y0,b  x:(r6)+n6,x0  y:(r1)+n1,y1  ;Radix 2 DIT butterfly kernel
    macr  -x0,y1,b  a,x:(r5)+n5  y:(r0),a      ;with one butterfly per group
    subl  b,a      x:(r0),b      b,y:(r4)
    mac   -x1,x0,b  x:(r0)+n0,a  a,y:(r5)
    macr  -y1,y0,b  x:(r1),x1    y:(r6),y0

```



```

        subl    b,a        b,x:(r4)+n4    y:(r0),b
_lastpass
        move    a,x:(r5)+n5
        move    #Odata,r0        ;points to real output buffer
        move    #Psd_data,r1    ;points to psd buffer
        move    #Points-1,m0
        move    m0,m1
        move    m0,m4
        move    r0,r4        ;points to imaginary buffer
        do      #Points,End_psd ;calculate the power spectrum density
            move    x:(r0)+,x0    y:(r4)+,y0    ;get fft data
            mpyr    x0,x0,a        ;square the real data
            mpyr    y0,y0,b        ;square the imaginary data
            add     b,a
            move    a,y1
            move    #0,y0
            jsr     Sqr
            move    b,x:(r1)+
        End_psd
        nop
        rts

```

;=====

```

Sqr      clr     b        #<40,x0        ;Init root and guess
        move    x0,x1        ;Init bit to test
        do      #23,End1
            mpy    -x0,x0,a    ;square and negate the guess
            add     y,a        ;compare to double precision input
            tge    x0,b        ;update root if input <= guess
            tfr     x1,a        ;get bit to test
            asr     a        ;shift to next bit to test
            add     b,a        a,x1        ;form new guess
            move    a,x0        ;save new guess
        End1
        rts

```

;*****

;interrupt Service Routine

```

Save     move    r0,x:S_r0
        move    m0,x:S_m0    ;save registers
        move    r1,x:S_r1
        move    m1,x:S_m1
        move    r4,x:S_r4
        move    m4,x:S_m4
        move    r7,x:S_r7
        move    m7,x:S_m7
        move    x1,x:S_x1

```



```

        move    x0,x:S_x0
        move    a,x:S_a1
        rts

Restore  move    x:S_r0,r0          ; restore registers
        move    x:S_m0,m0
        move    x:S_r1,r1
        move    x:S_m1,m1
        move    x:S_r4,r4
        move    x:S_m4,m4
        move    x:S_r7,r7
        move    x:S_m7,m7
        move    x:S_x0,x0
        move    x:S_x1,x1
        move    x:S_a1,a
        rts

ISR_SSI_RX
        jsr     Save
        move    x:Disp_r1,r1      ;Get the PSD nr.
        move    #Points-1,m1
        move    r1,a
        move    #Psd_data,x0
        cmp     x0,a              ;The first PSD ?
        jne     Disp              ;no
        do      #Trigger,E_trig   ;Yes,send the trigger pulse
        move    #Vpbd,x0
        move    x0,x:Pbd

E_trig
        move    #0,x0
        move    x0,x:Pbd          ;reset the trigger pulse

Disp    move    x:(r1)+,a          ;Get the PSD amplitude
        movep   a,x:Tx            ;Display the PSD
        move    r1,x:Disp_r1      ;Update the PSD nr.
        movep   x:Ssisr,a         ;always read the ssisr and rx
        movep   x:Rx,a            ;to avoid exceptions
        move    #-1,m7            ;linear addressing mode
        move    x:>Skip_nr,r7     ;Read the Skip_nr counter value
        move    #>Divisor,x0
        move    (r7)+             ;increment the counter
        move    r7,a
        cmp     x0,a              ;skip 240 samples ?
        jeq     Null              ;yes
        move    r7,x:>Skip_nr     ;update the Skip_nr value
        jmp     exit

Null    move    #0,r7
        move    #>Scaling,x1     ;scaling factor
        move    r7,x:Skip_nr      ;reset the Skip_nr counter
        move    #Counter-1,m4     ;modulus addressing mode

```



```

    movep    x:Rx,x0
    move     x:IP,r4          ;get the tempory i/p buffer pointer
    mpy      x0,x1,a
    move     x:>S_nr,r7       ;get the sample nr.
    move     a,x:(r4)+        ;store the data
    move     #>Counter,x0
    move     (r7)+            ;increment by 1
    move     r7,a
    cmp      x0,a             ; 8 samples ?
    jeq      Pre_fft         ;yes , jump
    move     r4,x:IP          ;no, update the IP buffer address
    move     r7,x:S_nr        ;update the sample nr.
    jmp      exit
Pre_fft     move     #Points-1,m0 ;modulus addressing mode
            move     x:Ptr_r0,r0 ;points to the tempory buffer
            do       #Counter,e_copy
            move     x:(r4)+,a
            move     a,x:(r0)+
e_copy
            move     #IPA,x0
            move     x0,x:IP    ;Init the tempory buffer pointer
            move     #0,x0      ;reset the sample number
            move     x0,x:S_nr
            move     r0,x:Ptr_r0 ;update the i/p buffer pointer
            move     #Vfft_s,x0
            move     x0,x:FFT_s ;set the fft status
exit        jsr      Restore
            rti

Rxcept      move     a,x:Rxe_a
            movep    x:Ssisr,a1 ;read the status register
            movep    x:Rx,a1    ;read the rx
            move     x:Rxe_a,a
            rti

Txcept      move     a,x:Txe_a
            movep    x:Ssisr,a1 ;read the status
            movep    a1,x:Tx    ;send data to tx
            move     x:Txe_a,a
            rti

;*****

;set up jsr's

            org      p:Ssi_rx_jra
            jsr      ISR_SSI_RX

            org      p:Ssi_rxe_jra

```


jsr	Rxcept
org	p:Ssi_txe_jra
jsr	Txcept
end	Begin

References

- [1] Bershad, N. J. *On the Optimum Gain Parameter in LMS Adaption*, IEEE Transactions on Acoustics, Speech, and Signal Processing vol. ASSP-35, no.7, July 1987, pp 1065-1068
- [2] Bievenu, G. and Kopp, L. *Adaptivity to Background Noise Spatial Coherence for High Resolution Passive Methods*, in ICASSP, pp. 307-310
- [3] Bienvenu, G. *Principles of High-Resolution Array Processing*, Chapter 5 in Signal Processing: Theory and Application
- [4] Blackman, R.B. and J.W. Tukey *The Measurements of Power Spectral from the Point of View of Communications Engineering*, Bell Syst. Tech. J. vol.37, pp 185-282, 485-569, 1958; also republished by Dover Publications, New York 1959
- [5] Burg, J.P. *Maximum Entropy Spectral Analysis*, in Proceedings 37th Meeting of Society of Exploration Geophysicists, Oklahoma City, Okla, Oct. 1967
- [6] Cadzow, J. A. *Singular Value Decomposition Approach to Time Series Modeling*, IEE Proc. vol.130, part-F, N0.3, April, 1983, pp.202-210
- [7] Cadzow, J. A. *Direction-of-Arrival Estimation Using signal Subspace Modeling*, IEEE Trans. on Aerospace and Electronic Systems, vol.28, no.1, Jan 1991, pp. 64-78
- [8] Chen, C. H. and A. H. Costa *An Interactive and Data-Adaptive High Resolution Spectral Analysis techniques*, Proceedings of the IEEE International Conference on System, Man and Cybernetics 8-12 Aug, 1988, Beijing, China, pp. 82-85
- [9] Chen, C. H. , A. H. Costa and C. Sue *New Results on Data Adaptive High Resolution Spectral Analysis Techniques*, Proceedings of the IEEE International Conference on System, Man, and Cybernetics. Aug, 1988, Beijing, China, pp.82-85

- [10] Chen, W. Y. and G. R. Stegen *Experiments with maximum Entropy Power Spectral of Sinusoids*, J. Geophys. Res. vol.79 pp-3019-3022, July 1974
- [11] Cooley, J.W. and J.W. Tukey *An Algorithm for the Machine Calculation of Complex Fourier Series*, Mathematics of Computation, vol.19, pp 297-301 April 1965
- [12] DeGroat, R.D. and R.A. Roberts *Efficient, Numerically Stabilized Rank-one Eigenstructure Updating*, ASSP, vol.38, no.2, Feb. 1990, pp. 301-316
- [13] Gabriel, W. F. *Using Spectral Estimation Techniques in Adaptive Processing Antenna Systems*, IEEE Trans. on Antenna and Propagation, vol.AP-34, no. 3, March 1986
- [14] Gaby, J. E. and M. H. Hayes *Artificial Intelligence Applied to Spectrum Estimation*, 1984 IEEE Int, Conf. ASSP, pp.13.5.1-13.5.4
- [15] Griffiths, L. J. *Rapid Measurement of Digital Instantaneous Frequency*, IEEE Transactions on Acoustics, Speech, and Signal Processing vol. ASSP-23, pp 207-222, April 1975
- [16] Haykin, S. *Radar Signal Processing*, IEEE ASSP Magazine April, 1985 pp. 2-18
- [17] Haykin, S. *Adaptive Filter Theory*, second ed. Prentice Hall, 1991.
- [18] Herring, R.W. *The Cause of Line Splitting in Burg Maximum-Entropy Spectral Analysis*, IEEE Trans. ASSP vol. ASSP-28, pp. 692-701, Dec. 1980
- [19] Haweel, T.I. and P.M. Clarkson, *A Class of Order Statistic LMS Algorithms*, IEEE Trans. ASSP, vol. 40, no.1, pp. 44-53, Jan 1992
- [20] Johnson, D.H. and S.R. DeGraaf *Improving the Resolution of Bearing in Passive Sonar Arrays by Eigenvalue Analysis*, ASSP, vol. ASSP-30, pp. 638-647, August 1982
- [21] Johnson, D. H. *The Application of Spectral Estimation Methods to Bearing Estimation Problems*, Proceedings of the IEEE vol.70, no.9, Sep. 1982, pp. 1018-1028
- [22] Kahunen, J. *Adaptive Algorithm for Estimating Eigenvector of Correlation Type Matrices*, in ICASSP, 1984

- [23] Kay, S.M. *Modern Spectral Estimation, Theory and Applications*, Prentice Hall 1988
- [24] Kaveh, M. and S.P. Bruzzone *Statistical Efficiency of Correlation-Based Methods for ARMA Spectral Estimation*, IEE Proc. vol.130, part F, April, 1983. pp.211-217
- [25] Kleman, V. C. and A. J. Laub *The Singular Value Decomposition: Its Computation and Some Applications*, IEEE Trans. on Automatic Control, vol.AC-25,no.2, april 1980, pp.164-176
- [26] Kumaresan,R. and Tufts,D.W. *Estimating the Angles of Arrival of Multiple Plane Wave*, IEEE Trans. on AES, vol. AES-19, 1983
- [27] Kung, S. Y. *VLSI Array Processors*, IEEE ASSP Magazine July, 1985
- [28] Larimore H.G. and R.J. Calvert *Convergence Studies of Thompson's Unbiased Adaptive Spectral Estimator*, in 14th ACCSC, Pacific Grove, CA, 1981
- [29] Marple Jr., S.L. *Digital Spectral Analysis*, Prentice Hall Inc. 1987
- [30] Motorola Inc., *DSP56000/DSP56001 Digital Signal Processor User's Manual*, Rev. 2, 1990
- [31] Nuthalapat, R. M. *High Resolution Reconstruction of ISAR Images*, IEEE Trans. on Aerospace and Electronic Systems, vol.28, no.2, April, 1992 pp. 462-472
- [32] Oritigueira, M.D. and M.A. Lagunas *A Recursive Algorithm for the Computation of Eigenvalues and Eigenvectors with Application to Array Processing*, in Proc. of the Latvian Signal Processing Intern. Conf., April 1990, Riga, pp.85-88
- [33] Oritigueira, M. D. *Eigendecomposition Versus singular Value Decomposition in Adaptive Array Signal Processing*, Signal Processing vol.25, no.1, oct. 1991. pp.35-49
- [34] Owsly,N.L. *Adaptive Data Orthogonalization*, in Proc. IEEE Int. Conf. ASSP 1978

- [35] Pisarenko, V.F. *The Retrival of Harmonics from Covariance Functions*, Geophys. J Roy. Astron. Soc. 1973
- [36] Robinson, E. A. *A Historical Prospective of Spectrum Estimation*, proceedings of The IEEE vol.70 no.9 Sept, 1982 pp. 885-907
- [37] Schlindwein, F. S. *A Real-Time Autoregressive Spectrum Analyzer for Doppler Ultrasound Signals*, Ultrasound in Med. and Biol. vol.15, no.3, pp. 263-272, 1989
- [38] Schmidt, R.O. *A Signal Subspace Approach to Multiple Emitter Location and Spectral Estimation*, Ph.D. dissertation, Dept. of E.E. Stanford Univ., Stanford, Calif. Nov. 1981
- [39] Schmidt, R.O. *Multiple Emitter Location and Signal Parameter Estimation*, in Proc. RAC Spectrum Estimation workshop NY OCT. 1979
- [40] Schreiber, R. *Implementation of Eigenvector Methods*, SPIE vol.495 Real Time Signal Processing VII 1984, pp.3-6
- [41] Schuster, Sir Arthur, *On the Investigation of Hidden Periodicities with Application to a supposed Twenty-Six-Day Period of Meteorological Phenomena*, Terrestrial Magnetism and Atmospheric Electricity vol.3, no.1, pp 13-41, March 1898
- [42] Schreiber, R. *Implementaion of daptive Array Algorithms*, ASSP, vol. ASSP-34, oct., 1986
- [43] Sharman, K.C. *Adaptive Algorithms for Estimating the Complete Covariance Eigenstructure*, in ICASSP, 1986, pp. 1401-1403
- [44] Thompson, P.A. *An Adaptive Spectral Analysis Technique for Unbiased Frequency Estimation in the Presence of White Noise*, in Proc. 13th asilomar conf. Circuits, Syst. Comput. Pacific Grove, CA 1980
- [45] Van Veen, B.D. and K.M. Buckley *Beamforming: A Versatile Approach to Spatial Filtering*, IEEE ASSP Magazine April 1988, pp.4-24
- [46] Wesolowski, K. , C. M. Zhao and W. Rupprecht *Adaptive LMS Transversal Filters with Controlled Length*, IEE Proceedings-F vol.139, no.3, June, 1992, pp.233-238

- [35] Pisarenko, V.F. *The Retrival of Harmonics from Covariance Functions*, Geophys. J Roy. Astron. Soc. 1973
- [36] Robinson, E. A. *A Historical Prospective of Spectrum Estimation*, proceedings of The IEEE vol.70 no.9 Sept, 1982 pp. 885-907
- [37] Schlindwein, F. S. *A Real-Time Autoregressive Spectrum Analyzer for Doppler Ultrasound Signals*, Ultrasound in Med. and Biol. vol.15, no.3, pp. 263-272, 1989
- [38] Schmidt, R.O. *A Signal Subspace Approach to Multiple Emitter Location and Spectral Estimation*, Ph.D. dissertation, Dept. of E.E. Stanford Univ., Stanford, Calif. Nov. 1981
- [39] Schmidt, R.O. *Multiple Emitter Location and Signal Parameter Estimation*, in Proc. RAC Spectrum Estimation workshop NY OCT. 1979
- [40] Schreiber, R. *Implementation of Eigenvector Methods*, SPIE vol.495 Real Time Signal Processing VII 1984, pp.3-6
- [41] Schuster, Sir Arthur, *On the Investigation of Hidden Periodicities with Application to a supposed Twenty-Six-Day Period of Meteorological Phenomena*, Terrestrial Magnetism and Atmospheric Electricity vol.3, no.1, pp 13-41, March 1898
- [42] Schreiber, R. *Implementaion of daptive Array Algorithms*, ASSP, vol. ASSP-34, oct., 1986
- [43] Sharman, K.C. *Adaptive Algorithms for Estimating the Complete Covariance Eigenstructure*, in ICASSP, 1986, pp. 1401-1403
- [44] Thompson, P.A. *An Adaptive Spectral Analysis Technique for Unbiased Frequency Estimation in the Presence of White Noise*, in Proc. 13th asilomar conf. Circuits, Syst. Comput. Pacific Grove, CA 1980
- [45] Van Veen, B.D. and K.M. Buckley *Beamforming: A Versatile Approach to Spatial Filtering*, IEEE ASSP Magazine April 1988, pp.4-24
- [46] Wesolowski, K. , C. M. Zhao and W. Rupprecht *Adaptive LMS Transversal Filters with Controlled Length*, IEE Proceedings-F vol.139. no.3, June, 1992, pp.233-238

- [47] Widrow, B. et. al. *Adaptive Noise Cancelling: Principles and Applications*. Proceedings of the IEEE, vol.63, pp 1692-1716, Dec 1975
- [48] Widrow, B. et. al. *Adaptive Signal Processing*, Prentice-Hall Inc. 1985
- [49] Wiener, Norbert *Generalized Harmonic Analysis*, Acta Math. vol.55, pp117-258 1930
- [50] Yang,J.F. *Adaptive Eigenspace Algorithms for Direction or Frequency Estimation and Tracking*, IEEE Trans. on Acoustic,Speech,and Signal Processing vol. 36 no. 2 Feb. 1988
- [51] Yu, K. B. *Recursive Updating the Eigenvalue Decomposition of a Covariance Matrix*, ASSP, vol.39, no.5, May 1991, pp. 1136-1145
- [52] Yu, K. B. *Efficient,Parallel Adaptive Eigenbased Techniques for Direction of Arrival Estimation and Tracking* Proceeding Fifth ASSP Workshop Spectrum Estimation Modeling, Oct. 1990, pp.347-351
- [53] Yule,G *On a Method of Investigating Periodicities in Distributed Series, with Special Reference to Wolfer's Sunspot Numbers*, Philospherical Trans. of the Royal Society of London, ser.A, vol.266, pp. 267-298, July 1927
- [54] Zeidler, J. R. *Performance Analysis of LMS Adaptive Prediction Filters*, Proceedings of the IEEE, vol.78, no.12, December 1990, pp 1781-1806