E C G DATA COMPRESSION USING TRANSFORMATIVE TECHNIQUES

A DISSERTATION

submitted in partial fulfilment of the requirements for the award of the degree of

MASTER OF ENGINEERING

in

ELECTRICAL ENGINEERING

(With Specialization in Measurement and Instrumentation)

By

SANJAY PAL SINGH

DEPARTMENT OF ELECTRICAL ENGINEERING
UNIVERSITY OF ROORKEE
ROORKEE-247 667 (INDIA)

MARCH, 1999

CANDIDATE'S DECLARATION

I hereby certify that the work being presented in this dissertation entitled, **"ECG DATA COMPRESSION USING TRANSFORMATIVE TECHNIQUES"** in partial fulfilment of the requirement of the award of the degree of **"MASTER OF ENGINEERING"** in **"MEASUREMENT AND INSTRUMENTATION"** submitted in the department of Electrical Engineering of University of Roorkee, Roorkee is an authentic record of my own work carried for the period from July 1998 to March 1999, under the supervision and guidance of Dr. S.C. Saxena, Professor & Head, Department of Electrical Engineering and Dr. R.S. Anand, Assistant Professor, Department of Electrical Engineering, University of Roorkee, Roorkee.

The matter embodied in this dissertation has not been submitted by me for the award of any other degree.

SP Singh

Dated : 27-3-1999 **(SANJAY PAL SINGH)**

Place : Roorkee

This is to certify that the above statement made by the candidate is correct to the best of our knowledge.

30.3.99

(DR. R.S. ANAND)
Assistant Professor
Department of Electrical Engg.
University of Roorkee,
Roorkee - 247667 (U.P.)

27/3/99

(DR. S.C. SAXENA)
Professor and Head,
Department of Electrical Engg.
University of Roorkee,
Roorkee - 247 667 (U.P.)

ACKNOWLEGEMENT

I acknowledge the almightily who paved the way for the completion of this work. I express deep sense gratitude towards my supervisors Dr. S.C. Saxena, Professor and Head, Department of Electrical Engineering and Dr. R.S. Anand, Assistant Professor, Department of Electrical Engineering, University of Roorkee, Roorkee, for their invaluable guidance, support and continued thought providing discussions at every juncture in carrying out this work.

My sincere thanks are due to all professors of Measurement and Instrumentation Group of Department of Electrical Engineering for their invaluable suggestions from time to time.

Special thanks are due to Dr. S.Mukerjee, Assoc.Professor, Dr. Vinod Kumar, Professor and Dr. H.K.Verma, Professor in Department of Electrical Engineering, for providing the necessary facilities to carry out this work. I am also thankful to Mr. Rajnesh Tyagi, Mr. Z.A.Khan, Mr. V.P.Singh, Mr. S.K.Chakrabarti, Mr. L.M.Wagmare, Mr. G. Parmar and Mr. V.K. Nimesh for helping me from time to time during my dissertation work.

I also express deep sense of gratitude to Shri Jogeshwar Prasad who deserved words of appreciation for his assistance.

I am also grateful to G.M., Shri Chandra Prakash, Director Telecom Maintenance Dehradun Shri Jai Singh, D.E.T., Shri Lallu Ram, Shri C.L.Verma and Co-axial staff of Telephone Exchange Roorkee for giving me full support in completing this work.

Last but not the least I wish to thank my wife Mrs. Reeta and other family members who encouraged me at crucial junctures during all this work. As always I owe more to my mother and father than words can express.

SP Singh.

Dated : 27-3-99
Place : Roorkee

(SANJAY PAL SINGH)

ABSTRACT

This dissertation deals with the application of transformative techniques for the compression of ECG data obtained from the cardiac patients and other subjects. Two techniques, namely Fast Fourier Transform (FFT) and Fast Walsh Transform (FFT) have been selected to carryout the work in this dissertation.

After general description of the ECG signal in Chapter 2 and overview of various techniques for ECG data compression in Chapter 3, the details of the implementation aspect of FFT algorithm and FWT algorithm are presented in Chapter 4. The procedural steps of to implement FFT, inverse FFT, FWT and inverse FWT algorithms are given and also various aspects of their software implementation are discussed. Besides these, calculation of PRD, visual inspection of reconstructed waveforms, and detection of P, Q, R, S and T wave peaks have also been carried out. These algorithms have been tested on large amount of ECG data taken from the CSE data base. The results with discussions on performance aspects are presented for different cases. The conclusion of the present work and scope of future work are also given in the last chapter of the dissertation.

Overall, the work contained in this dissertation adds a further step in the area of ECG data compression.

CONTENTS

CHAPTER-1
INTRODUCTION

1.1 INTRODUCTION

The computerisation of Electrocardiogram (ECG) processing system alongwith the increased performance requirements demands reliable, accurate and efficient data compression techniques. The practical importance of ECG data compression has become evident in many aspects of computerized electrocardiography which includes (a) enhanced storage capacity of ECG's as data bases for subsequent comparison, (b) feasibility of transmission of real-time ECG's over the public telephone network [32], (c) improved functionality of ambulatory ECG monitors & recorders, and (d) implementation of cost effective real time system algorithms.

The interpretation of electrocardiography assisted by computers has become a common tool for cardiologists in diagnosing cardiac disorders. It is necessary for the maximum utilization of computers that ECG's be transmitted to a central computing facility from different locations such as ambulatory monitor, Holter monitor, etc. for further processing & diagnosis. Data compression also improves the rate of transmission of ECG data to the central computing centre apart from saving memory for subsequent storage. Data compression techniques are being used in vast spectrum of communication areas such as speech, image & telemetry transmission.

1.2 GENERAL CONCEPTS :

The methods used in data compression are mainly classified into following three major catagories:

(a) Direct data compression

(b) Transformative methods

(c) Parameter extraction

Data compression by the transformation or the direct data compression methods contain transformed or actual data from the original signal. Whereby, the original data are reconstructed by an inverse process. The direct data compressors base their detection of redundancies on direct analysis of the actual signal samples. The transformative compression methods mainly utilise spectral and energy distribution analysis for redundancy detection, while, the parameter extraction method is an irreversible process with which a particular characteristic or parameter of the signal is extracted. These extracted parameters (e.g., measurement of the probability distribution)are subsequently utilized for classification based on a prior knowledge of the signal features [46].

1.3 OBJECTIVE OF THE PRESENT WORK

In the present work, the transformative methods are used for the purpose of analysising the ECG signal. There are many transformative techniques. Present work deals with the Fast Fourier Transform (FFT) and Fast Walsh Transform (FWT) for compression of ECG data signal and comparision of both the methods for data compression.

1.4 ORGANIZATION OF DISSERTATION

The present work is divided into six chapters.

Chapter I, the literature review based on the work done by different researcher is briefly reported.

The chapter II is concerned with the concept of electrocardiography, flow of blood in heart, mechanical and electrical activity of heart, and ECG waveforms. Further different lead systems are discussed and a brief description of standard data base is also given in this chapter.

In Chapter III, various techniques of data compression have been discussed in detail and their brief comparison is also presented.

In Chapter IV, the Transfarmative techniques i.e. FFT and FWT are discussed in detail.

Chapter V deals with ECG data compression using FFT and FWT. Their performances have been tested for both the methods of compression and have been compared w.r.t. PRD, compression ratio, visual inspection and peak detection.

Chapter 6 presents the canclusions drawn from the presented work along with the scope for future improvemnts.

CHAPTER - 2

ELECTROCARDIOGRAPHY

2.1 INTRODUCTION

Electrocardiogram (ECG) has become a very popular and important diagnostic tool next to stethoscope and blood pressure measuring instrument iin the present time. The main advantage of ECG is its simplicity and non-invasive characteristics. The ECG provides faithful representation of the function of the heart.

2.2 THE HEART

Heart is one of the most important organ of the body. The basic function of the heart is to circulate the blood throughout the body.

The heart is divided into four chambers, namely right and left atria and right and left ventricles. The right ventricle pumps the blood through lungs, where it is oxygenated. This oxygen enriched blood enters the left atrium, and pumped into left ventricle. The left ventricle pumps the blood into the arteries to circulate it throughout the body. For proper functioning of heart, the atria and ventricles must operate in proper co-ordination and time relationship. The complete circulation cycle of the blood is as follows:

Left Atrium $\xrightarrow[\text{Valve}]{\text{Through mitral}}$ Left Ventricle $\xrightarrow[\text{Valve}]{\text{Through Arotic}}$ Legs, Internal Organs, Arms, Heads $\xrightarrow[\text{Vena Cava}]{\text{Through Superior}}$ Right Atrium $\xrightarrow[\text{Valve}]{\text{Tricuspid}}$ Right Ventricle $\xrightarrow[\text{Valve}]{\text{Through Semilunar}}$ Lungs $\xrightarrow[\text{blood}]{\text{Oxygenated}}$ Left Atrium (Fig.2-1).

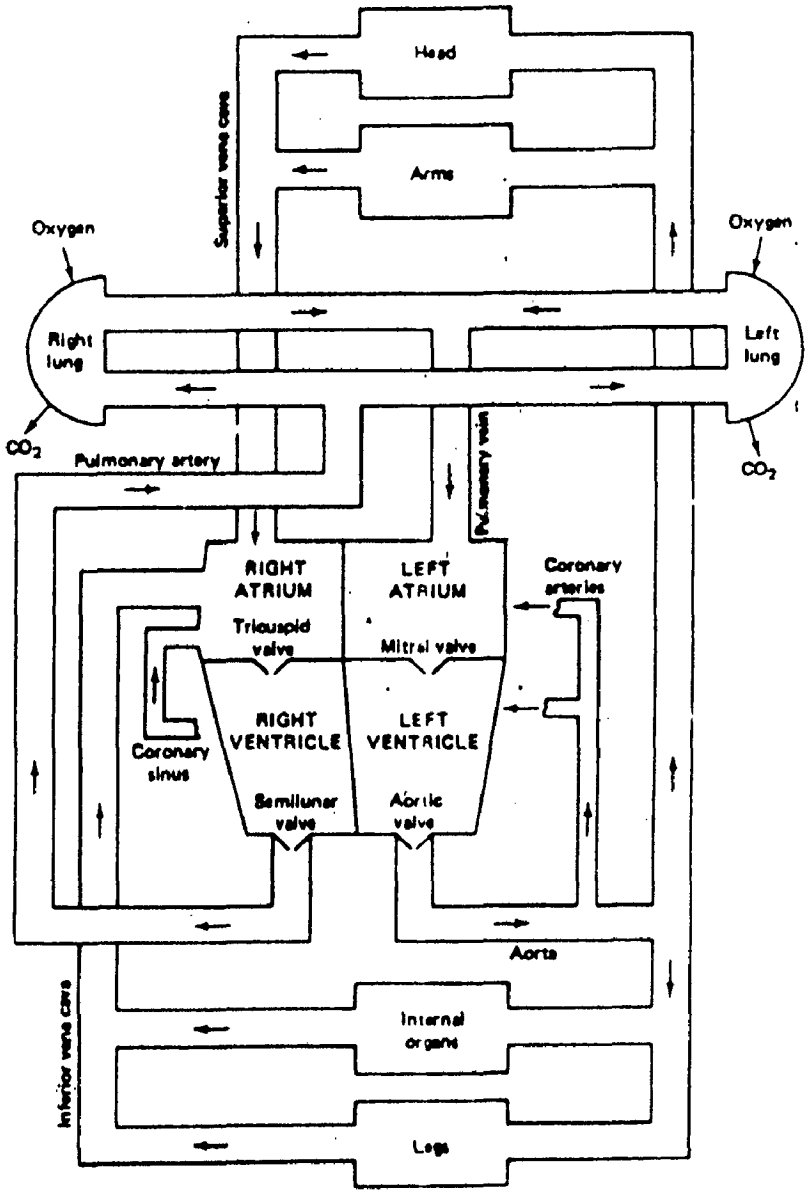


Fig. 2.1 : CARDIOVASCULAR CIRCULATION [12]

2.3 ELECTRICAL ACTIVITY OF HEART

The electrical activity of the heart orignates at the sinoatrial (SA) node [12]. The action potentials generated by the pacemaker or SA node propagate in all directions along the surface of atria towards the junction of the atria and ventricles and terminates at a point near the heart i.e. at atrioventricular (AV) node. At this node, some special fibres act as "delay line" to provide coordination between atria and ventricles. After passing through the delay line, the electrical excitation rapidly spreads to all parts of ventricles by the bundle of His. The fibres in the bundle called Purkinje fibres are further divided into branches in respective ventricles (Fig. 2-2).

2.4 ELECTROCARDIOGRAM (ECG)

The electrocardiogram (ECG or EKG) is a graphic recording or display of the time-variant voltages produced by the mycardium during the cardiac cycle. Fig 2-3, shows the basic waveform of the normal electrodiogram.

The electrical activity during the cardiac cycle is characterized by five separate segments of deflection designated as P,Q,R,S and T waves.

A normal ECG consists of a series of the following five successive waves :

P Wave - This is the deflection produced by the atrial depolarisation.

Q Wave - The initial negative deflection resulting from vertical depolarisation is Q-wave.

R Wave - The positive deflection during ventricular depolorisation is R-wave.

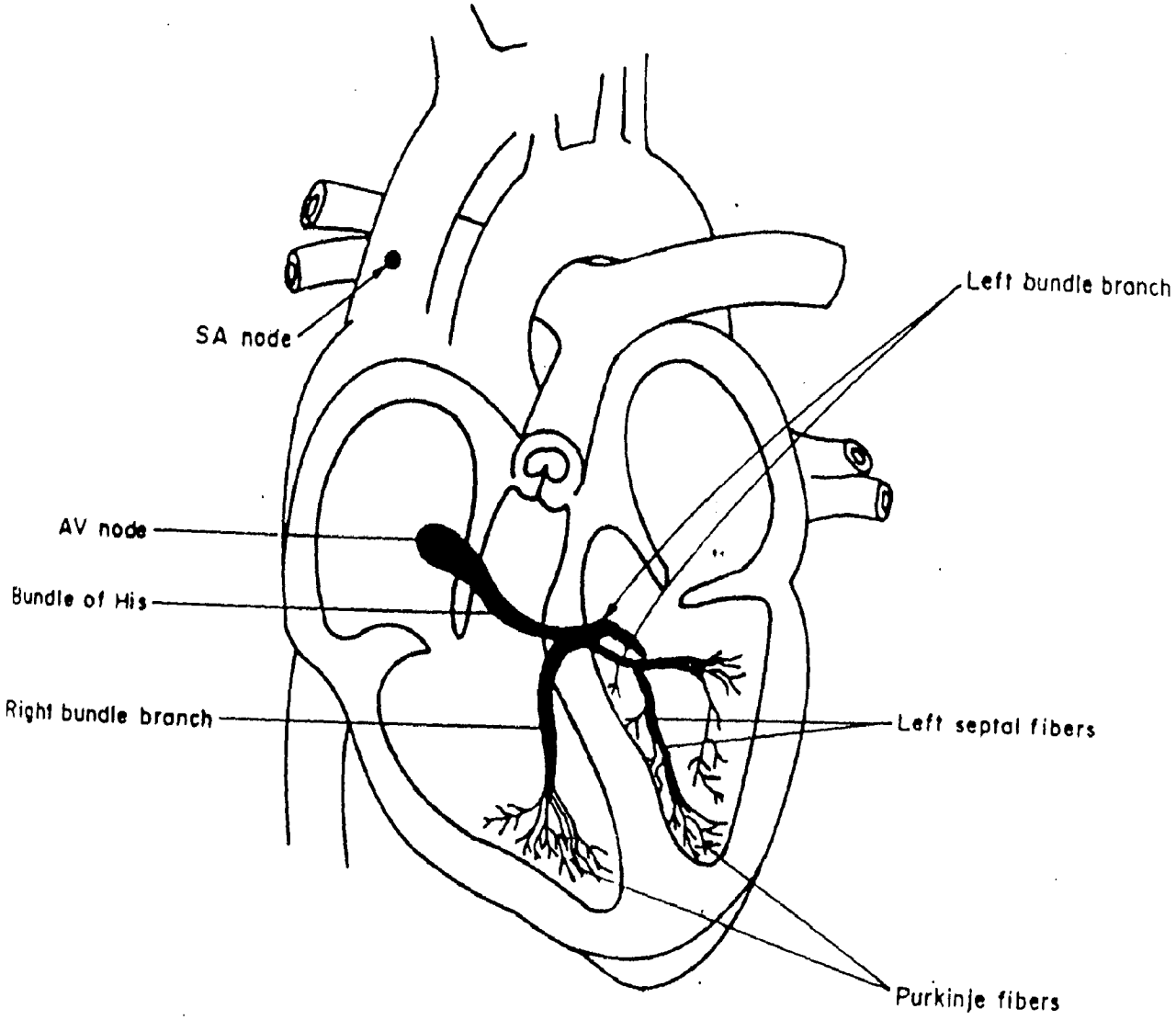


Fig. 2.2 : ELECTRICAL CONDUCTION OF THE HEART [12]

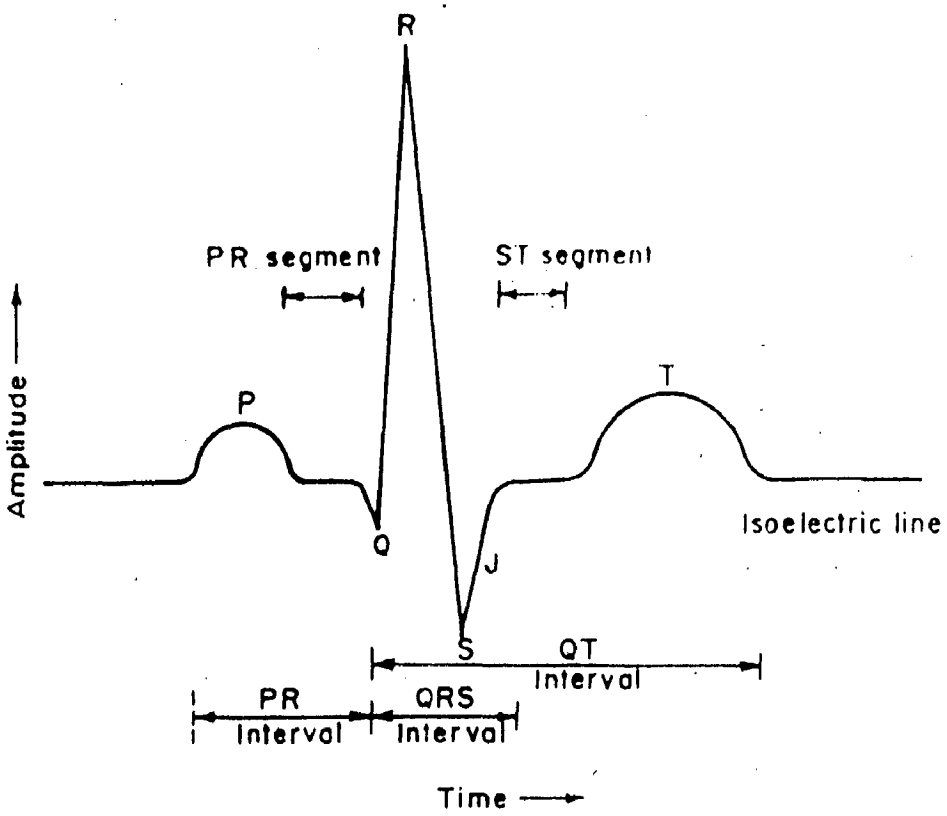


Fig. 2.3 : NORMAL CARDIAC CYCLE ON THE ECG [12]

S Wave - The first negative deflection of ventricular depolarisation, that follows the first positive deflection (R), is S-wave.

T Wave - The deflection produced by ventricular repolarisation is T-wave.

U Wave - It is present between the T wave and next P wave. It is a result of slow repolarisation of the intraventricular conduction.

The electrical potential and there corresponding frequencies and durations are given below [12]:

Wave	Amplitude(mv)	Maximum freq.(Hz)	Segment	Duration
P	0.25	10	PR interval	0.2 Sec.
R	1.6	20-30	QT interval	0.45 Sec.
Q	25% of R Wave	20-30	ST Segment	0.15 Sec.
S	upto 2	20-30	P Wave duration	0.16 Sec.
T	0.5	10	QRS duration	0.11 Sec.

2.5 RECORDING OF ECG

To record an ECG signal in 12 lead system, 5 number of electrodes are used which are fixed on the body of the patient. These are fixed on the following locations:

Right Arm - RA

Left Arm - LA

Right leg - RL

Left leg - LL

Chest - V_1-V_6

To record the ECG,different types of lead system as given below are used:

(i) Bipolar standard limb leads

(ii) Unipolar leads

(iii) Unipolar chest leads for precordial leads

(iv) Orthogonal leads

2.5.1 BIPOLAR STANDARD LIMB LEADS

The three bipolar limb leads I,II and III are the original lead selected by Einthoven to record electric potential in the frontal plane as shown in Fig 2-4.

The three bipolar limb leads first introduced by Einthoven [12] are :

Lead I = LA - RA

Lead II = LL - RA

Lead III = LL - LA

RL is grounded and is called reference or ground lead.

These leads are bipolar leads. In each of these lead positions, the QRS of a normal heart is such that the R wave is positive.

Out of the three limb leads, Lead II produces the greatest R-wave potential.

2.5.2 UNIPOLAR LEADS

Fig 2-5. shows the unipolar augmented lead aVR, aVL and aVF introduced by Wilson in 1944.

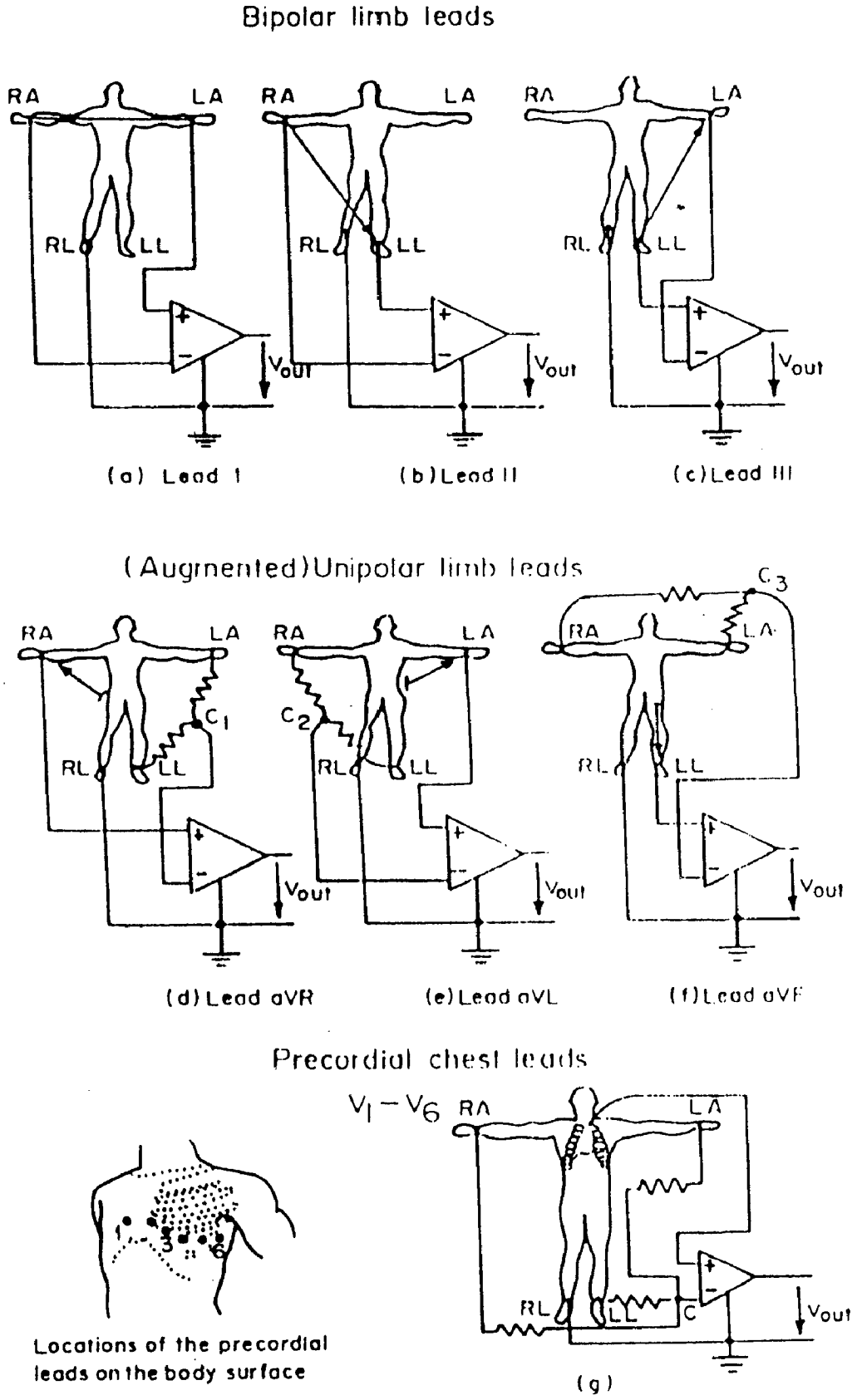


Fig. 2.4 : LEAD CONFIGURATION OF ECG [12]

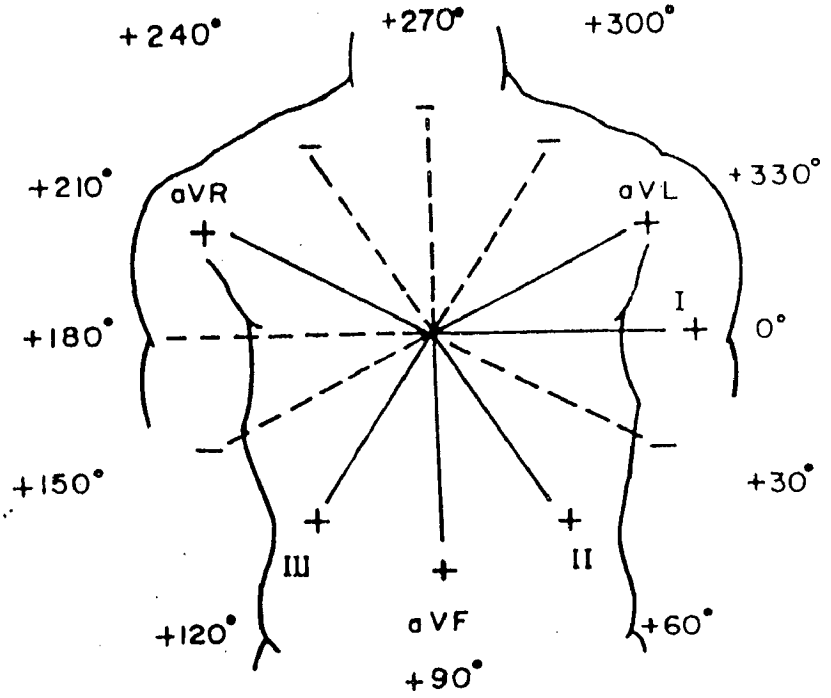


Fig. 2.5 : HEXAXIAL REFERENCE SYSTEM FOR SIX FRONTAL PLANE LEADS [12]

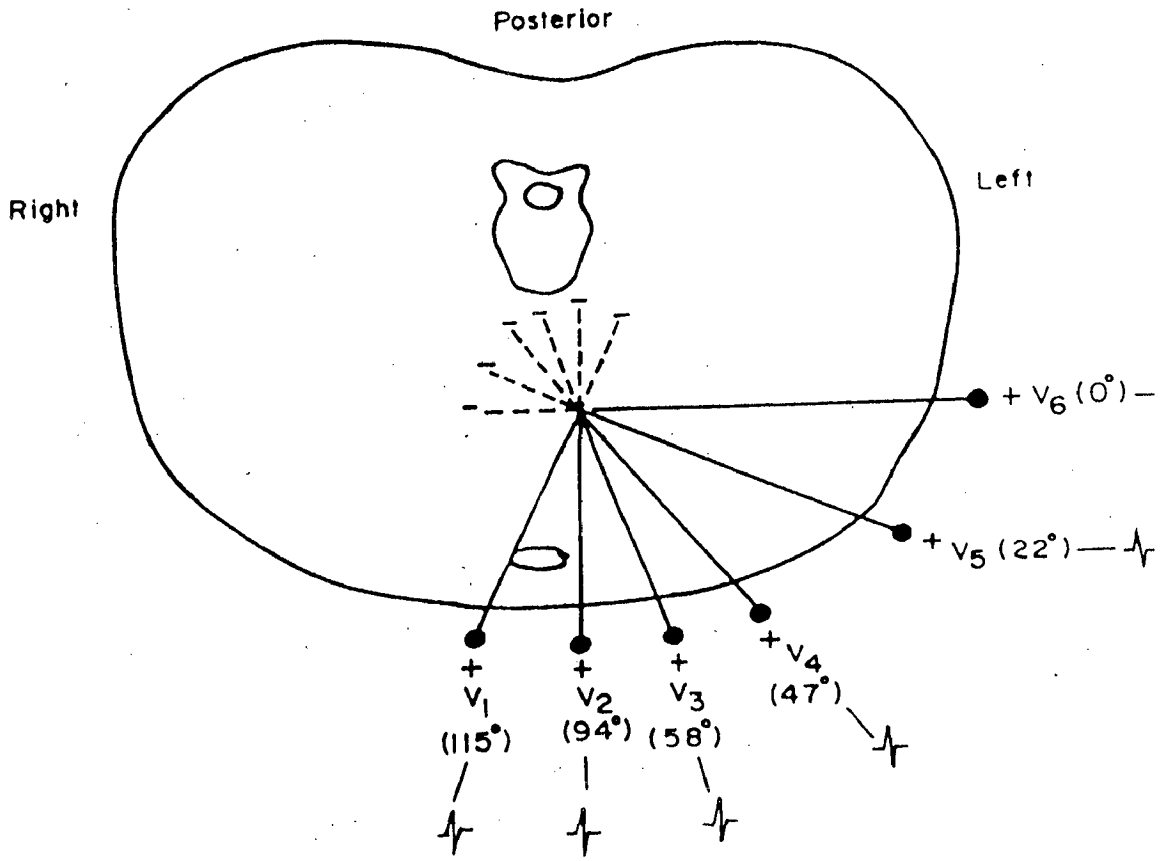


Fig. 2.6 : CHEAST LEADS IN HORIZONTAL PLANE [12]

$$\text{Lead aVR} = \text{RA} - \left[\frac{\text{LL} + \text{LA}}{2}\right]$$

$$\text{Lead aVL} = \text{LA} - \left[\frac{\text{RA} + \text{LL}}{2}\right]$$

$$\text{Lead aVF} = \text{LL} - \left[\frac{\text{RA} + \text{LA}}{2}\right]$$

2.5.3 UNIPOLAR CHEST LEADS OR PRECORDIAL LEADS

The chest leads V_1 - V_6 are shown in Fig 2-6. The location of electrodes for recording chest leads V_1, V_2, V_3, V_4, V_5 and V_6 are defined as below :

V_1 - Fourth intercostal space, at right sternal margin

V_2 - Fourth intercostal space, at left sternal margin

V_3 - Midway between V_2 and V_4

V_4 - Fifth intercostal space, at mid-clavicullar line.

V_5 - Same level as V_4, an anterior auxiliary line

V_6 - Same level as V_4, an mid-auxiliary line

2.5.4 ORTHOGONAL LEADS

An ideal lead sysetm for recording ECG and vector cardiogram (VCG) essentially consists of three leads with the following characteristics:

(a) The leads must be perpendicular to each other and also to the horizontal, vertical and sagittal axis of the body.

(b) The amplitudes of the three leads would be equal from vectorial stand point.

(c) All three leads would have the same strength and direction for all points in the heart where electromotive forces are generated. By convention X,Y and Z are referred to as, horizontal, vertical and sagittal axis and hence these are usually referred to as X,Y and Z leads [21].

2.6 ECG DATA BASE

The ECG data used in this work is sampled at 500Hz with a resolution of at least 10 bits. It has been taken from CSE data base [56]. This data base contains data for I,II,III, aVR, aVL, aVF, V_1, V_2, V_3, V_4, V_5, V_6, X, Y and Z leads.

CHAPTER 3

DIFFERENT TECHNIQUES OF COMPRESSION

3.1 INTRODUCTION

Computerized electrocardiogram (ECG) processing systems have been widely used in clinical practice [59]. In view of the vast amount of ECG data generated each year, efficient storage and data compression is very important. [2,31,60]. The main problem faced in the ambulatory ECG monitor is the efficient handling of large quantities of data. Thus system stores or transmits in real-time a large volumes of ECG data. The use of suitable data compression techniques can give better utilization of memory or storage space in them and also can save on the data transmission time. This demands the use of accurate, reliable and efficient ECG data compression techniques without loosing clinical information of the ECG signal. The application of compression include the transmission of ECG's via telephone on mobile radio and the storage of the ECG on a medical smart card.

The compression of digital ECG has received great attention, since the computer application in electrocardiography, for the following three reasons :

1) There is need for increased efficiency in computerized rhythm analysis systems.
2) There is need for efficient utilization of memory space for permanent storage of digitized ECG's.
3) There is need to transmit multi channel digitized electrocardiogram to an ECG center over telephone lines.

3.2 INFORMATION THEROY AND DATA COMPRESSION

The mode of operation of a data compression system is a digital communication link, where the source output is assumed to be a digital signal as shown in Fig.3-1. The transmission of the data is accomplished by means of pulse code modulation (PCM) requiring in general a very large bandwidth. The number of pulses per second to be transmitted is a function of both of the number of samples and of the number of bits necessary to represent each sample. To reduce the large number of pulses per second, the data compression or data transformation techniques are employed [29]. In Fig.3-1, the channel encoder is useful after source encoder because the compressed data are in general more sensitive to the communication channel noise than the non-compressed data, due to fact that at receiving end,the number of samples per second are less. An error in compressed data will generally introduce a considerable amount of distortion. Because of this reason, the channel encoder is used. The direct data compression can be broadly divided into two category : Classical Direct data compression methods and Direct data compression methouds.

3.3 CLASSICAL DIRECT DATA COMPRESSION METHODS

The direct data transmission methods mostly rely on utilizing prediction or interpolation algorithms. A prediction algorithms utilizes a prior knowledge of some previous samples while an interpolation algorithm employs a prior knowledge of both the previous and future samples. The theoretical analysis of such compression techniques can be found in literature [13,16,17,38,50].

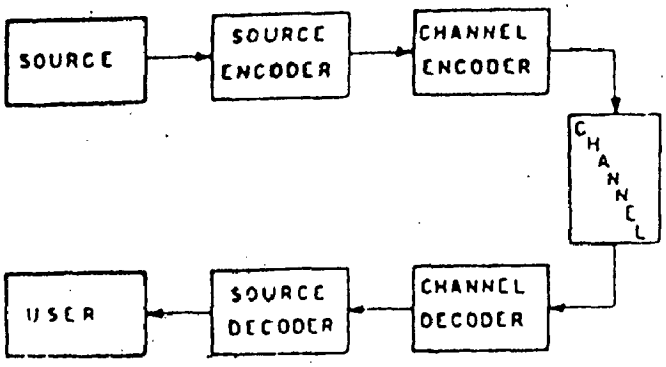


Fig. 3.1 : A DIGITAL COMMUNICATION SYSTEM WITH SOURCE AND CHANNEL ENCODING [4]

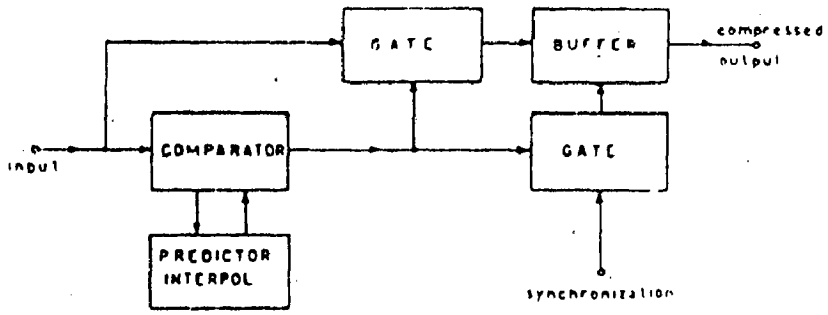


Fig. 3.2 : A TYPICAL DATA COMPRESSION SYSTEM WITH PREDICTION OR INTERPOLATION [4]

We can classify the direct data compression in three categories viz. tolerance comparison data compression, differential pulse code modulation (DPCM), and entropy coding methods.

3.3.1 TOLERANCE-COMPARISON DATA COMPRESSION TECHNIQUES

Most of the tolerance-comparison data compression techniques employ polynomial predictors and interpolations. The basic idea behind these techniques is to eliminate samples, from a data sequence, that can be implied by examining preceding and succeeding samples. The implementation of these techniques is executed by setting a preset error threshold centered around on actual sample point [31].

Description of tolerance-comparison compression techniques based on polynomial predictors/interpolates has been discussed in Special issue on redundancy reduction [53]. In comparison to speech data compression, polynomial compressors are widely used in ECG data compression[3,27,39].

In Fig.3-2, the block diagram of a typical data compression system with prediction or interpolation is shown. The non-redundant samples (i.e. sample for predication/interpolation) are fed into a buffer to be recognized at constant time intervals with the time position identification which is required for reconstruction.

(a) Polynomial Predictors:

Predicted data are obtained by extrapolating the polynomial one sample point at a time. The polynomial predictor is [39,58]

$$\hat{y}_n = y_{n-1} + \Delta y_{n-1} + \Delta^2 y_{n-1} + \text{---} + \Delta^k y_{n-1} \qquad (3.1)$$

Where

$\hat{y}_n$ = Predicted sample point at time t_n.

y_{n-1} = Sample value at one sample period prior to t_n.

$$\Delta y_{n-1} = y_{n-1} - y_{n-2} \tag{3.2}$$

$$\Delta^k y_{n-1} = \Delta^{k-1} y_{n-1} - \Delta^{k-1} y_{n-2} \tag{3.3}$$

The value of K represent the order of the polynomial prediction algorithm.

Zero-Order predictor (ZOP)

The ZOP is a polynomial with k=o.

$$\hat{y}_n = y_{n-1} \tag{3.4}$$

Here the predicted value is merely the previous data point. The efficient ZOP technique uses a floating operator wherein a tolerance band of $\pm\varepsilon$ is centered around the last saved data point as shown in Fig.3-3. The tolerance band actually floats with saved data points. The ZOP has proven to be very efficient for step-like data [3,4,39].

First-Order Predictor (FOP)

The FOP is an implementation of eqn (3.1) with K = 1. This gives a first-order polynomial of the form [3,4,39],

$$\hat{y}_n = 2y_{n-1} - y_{n-2} \tag{3.5}$$

The predicted value is a point on the straight line drawn between last two data points (y_{n-1} and y_{n-2}). In this case, the signal reconstruction

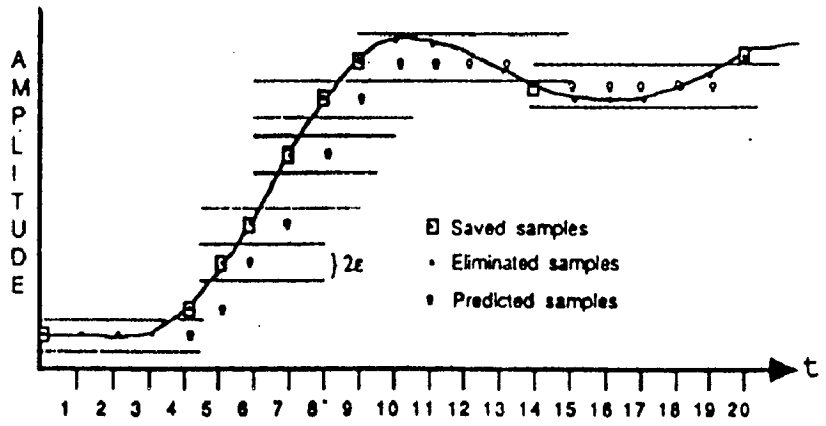


Fig. 3.3 : ILLUSRATION OF ZOP LOATING APERTURE [31]

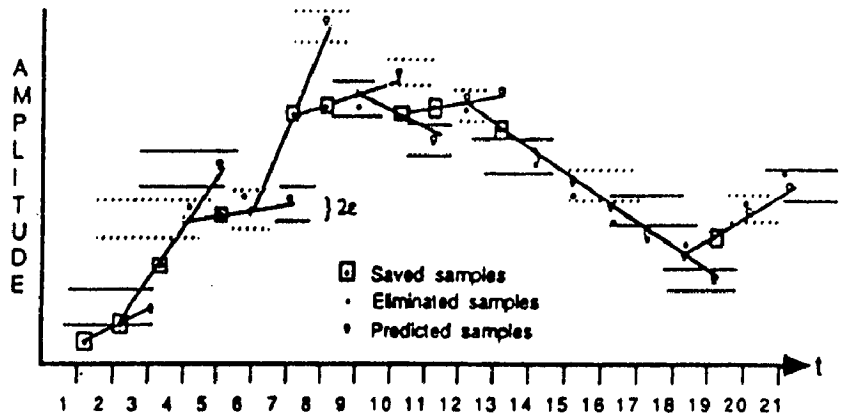


Fig. 3.4 : ILLUSRATION OF FOP-FLOATING APERTURE [31]

requires the saved sample values along with the corresponding time (Fig. 3-4).

(b) Polynomial Interpolations

This algorithm is different from the predictor ones as it utilizes the past and future values to decide whether or not the actual sample is redundant. Low - order polynomial interpolations have been found efficient in ECG data compression [6,15,22,33,48].

Zero-order Interpolation [ZOI]

This scheme is similar to ZOP with a difference that the redundant sample is selected by past and future data signal.The saved sample is compared as the average between the minimum and maximum sample values in the set. The ZOI is shown is Fig.3-5.

First-Order Interpolator (FOI)

The first-order Interpolator with two degrees of freedom (FOI-2DF) has been found to be the most efficient compression scheme among the FOI[3,33].The functional operation of the FOI-2DF is shown in Fig. 3-6.

The algorithm starts with retaining the first data point. A line is drawn between the retained point and the third sample point value (the first sample after saved one).If it is within a tolerance of $\pm \varepsilon$ of the interpolated value, then a straight line is drawn between the saved point and fourth point. The interpolated values of second and third points are now checked to examine if they are within preset error tolerance of the actual values. If the value is greater, than the process is repeated. The

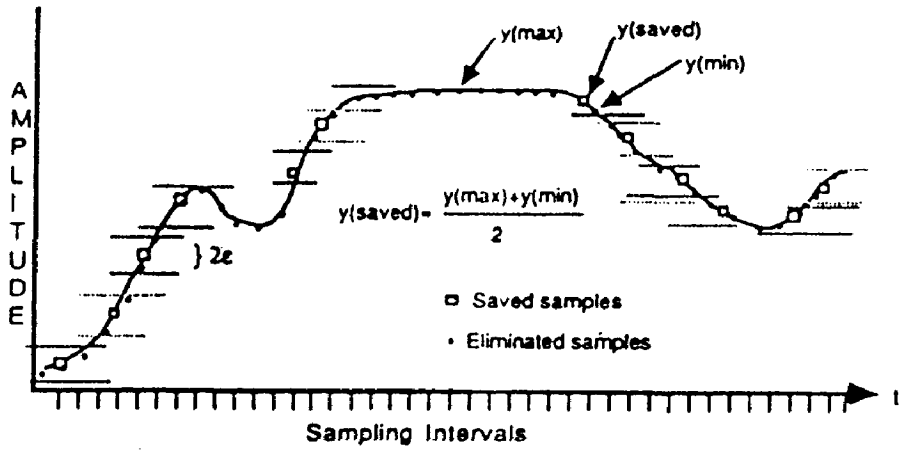


Fig. 3.5 : ILLUSRATION OF ZERO-ORDER INTERPOLATOR [31]

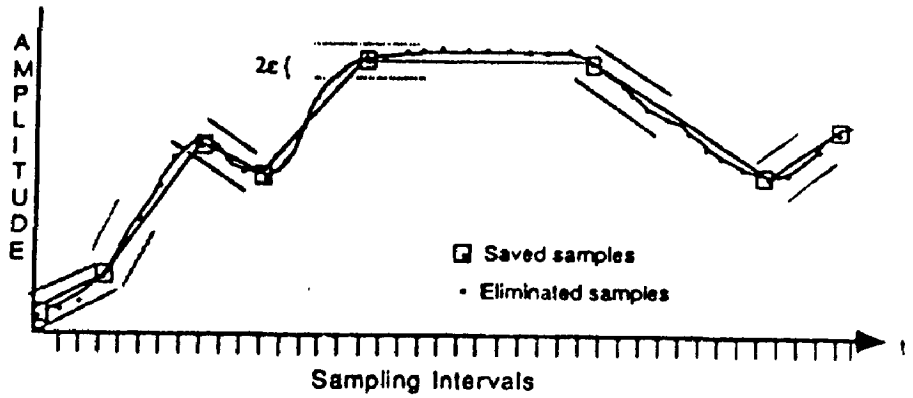


Fig. 3.6 : PRINCIPAL OPERATION OF FOI-2DF [31]

waveform is reconstructed by connecting the non redundant saved samples with straights lines. The FOI - DF is some times called "Two point projection" method [22]. This is due to the fact that the interpolated sample values are projected on the straight line drawn between sample points (interpolation straight line).

3.3.2 DATA COMPRESSION BY DIFFERENTIAL PULSE CODE MODULATION

The basic idea behind the differential pulse code modulation (DPCM) is that when data samples are estimated, the error (residual) between the actual sample and the estimated sample value ($e_n = y_n - \hat{y}_n$) is quantized and transmitted or stored [19,41]. Since waveform redundancy by DPCM coders is basically achieved by representing the actual correlated signal in terms of an uncorrelated signal namely, the estimation error signal. Thus, since the estimation error sequence is saved in place of the actual data sequence, upon reconstruction the original signal is preserved without loss of information.

The variance of the estimation error signal is smaller than the variance of original signal provided that the correlation of the input signal is high and the coefficients of estimator are correctly chosen. For a specified signal - to - quatization noise ratio(SNR), DPCM coding of a correlated waveform will result in bit rate reduction over PCM coding.

The gain (G) in the SNR of DPCM with respect to PCM can be expressed by,

$$G = \frac{E[y_n^2]}{E[e_n^2]} = \frac{E[y_n^2]}{E[y_n - \hat{y}_n)^{-2}]} \qquad (3.6)$$

Where $E[y_n^2]$ is the variance of y_n .

The Block diagram for differential pulse code modulation is shown in Fig.3-7.

In the DPCM technique, a predicted sample $\hat{y}_n$ is evaluated through a linear weghting of the M past samples y_n.

$$\hat{y}_n = \sum_{i}^{M} i\, a_i\, y_{n-1} \qquad (3.7)$$

The predicted sample can be obtained using any of the prediction algorithms like ZOP,FOP,ALP, etc.

Delta Modulation

The delta modulation technique (DMT) can be considered as a DPCM with a 1-digit code [18,54]. In this technique, the changes in the signal amplitude between consecutive sampling instants are transmitted in place of the absolute signal amplitude. There changes are sent in the form of binary pules whose sign (+or-) depends on the sign of the change in amplitude, Fig. 3-8 depicts the block diagram of delta modulation system. In classical delta modulation, a single pulse is transmitted at each sampling period instead of a complete codeword as in PCM as increase in the error of the reconstructed data is however to be seen in two effects.

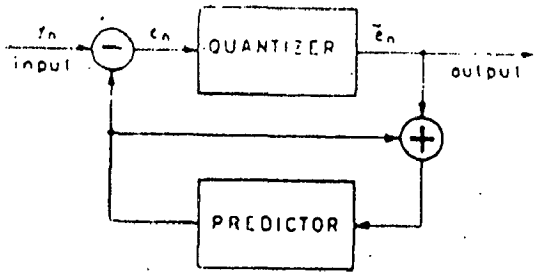


Fig. 3.7 : BLOCK DIAGRAM OF DIFFERENTIAL PULSE CODE MADULATION SYSTEM [4]

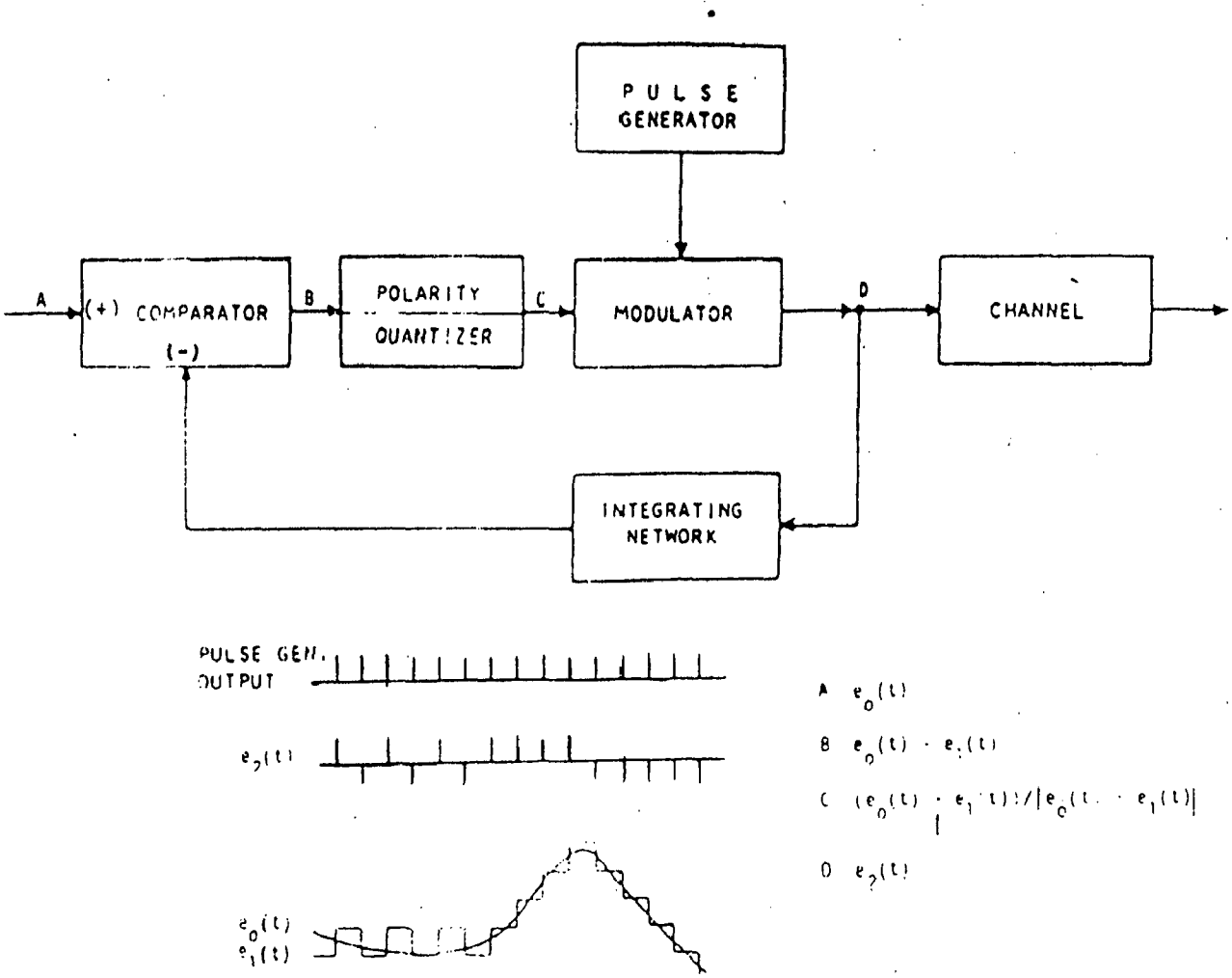


Fig. 3.8 : BLOCK DIAGRAM OF DELTA MODULATION SYSTEM [4]

(a) The approximation of the signal to a step function (granular or quantization noise).

(b) Delta modulation cannot accurately follow the quick variations of the signal i.e. the variations for which the gradient of the sampled data exceeds the limit value.

$$G = \Delta r$$

Where

Δ = change in the amplitude that a transmitted pulse represents

r = rate of the pulse transmission (slope-over load distortion)

3.3.3 ENTROPY CODING

Data compression through entropy coding is obtained by mean of assigning variable - length code words to a given occurrence. This compression method attempts to remove signal redundancy that arises with equal probability.

The method used in Huffman coding [26] which provides a method for the assignment of code words for quantizer outputs, with average wordlengths ranging from 1 to $|\log_2 L|$, based on the signal amplitude probability distribution values occurring with higher probability are assigned shorter code lengths compared to less probable ones. This results in minimization of the mean code length and is said to be the optimum code.

3.4 DIRECT ECG COMPRESSION SCHEMES

Following are the direct ECG compression schemes are:

3.4.1THE AZTEC TECHNIQUE

The amplitude zone time epoch coding (AZTEC) algorithm is a widely acknowledged data reduction algorithm for ECG Monitors and databases with an achieved compression ratios of 10 :1 [8,9]. It converts raw ECG sample points into plateaus to slopes. The AZTEC plateau (horizontal lines) are produced by utilizing zero-order interpolation. The stored values for each plateau are the amplitude value of the line and its length. The AZTEC slope produced whenever the number of samples needed to form a plateau is less than three. Signal reconstruction is achieved by expanding the AZTEC plateaus and slopes into a discrete sequence of data points.

3.4.2 THE TURNING POINT TECHNIQUE

The Turning Point (TP) data reduction algorithm is used for the purpose of reducing the sampling frequency of a ECG signal from 200 to 100 Hz without diminishing the elevation of large amplitude QRS's. This algorithm processes three data points at a time; a reference point (X_0) and two consecutive data point (X_1 and X_2). Either X_1 or X_2 is retained depending on which point preserves the slope of original three points. The TP algorithm produces a fixed compression rates of 2:1 where by the reconstructed signal resembles the original signal with some distortion. The main disadvantage of the TP method is that the saved points do not represent equally spaced time intervals.

3.4.3THE CORTES SCHEME

The coordinate reduction time encoding system (CORTES) algorithm is hybrid of AZTEC and TP algorithms. CORTES applies the TP algorithm to the high frequency regions, whereas, it applies AZTEC algorithm to isoelectric regions of the ECG signal. Wherever an AZTEC line is produced. The decision based on the length of the line as used to determine whether the AZTEC data or the TP data is to saved. If the line is longer than the empirically determined threshold, the AZTEC line is saved, otherwise, TP data are saved. Only AZTEC plateaus are generated, no slopes are produced. The reconstruction is achieved by expanding the AZTEC plateaus into discrete data points and then interpolating between each pair of the TP data. Parabolic smoothing is applied to AZTEC portions of the reconstructed CORTES signal to reduce distortion.

3.4.4 FAN AND SAPA TECHNIQUES

The Fan Algorithm :- Fan is a method of implementing the FOI-2DF without requiring the storage of all the actual data points between the last transmitted point and the present point during program execution. It draws the longest possible line between the starting point and the ending point so that all intermediate samples are within specified error tolerance. It guarantees the error between the line, joining any two permanent sample points and any actual (redundant) sample along the line is less than or equal to the magnitude of the present error tolerance. This method provides the best performance both w.r.t. compression ratio and signal fidelity [22].

The SAPA-2 Algorithm :- The basic idea behind SAPA (Scan-along polygonal approximation) is that the deviation between the straight-lines (Approximated signal) and the original signal is never more than the present error tolerance. It uses the center slope criterion, for verifying whether the sample is permanent or redundant, instead of actual sample value criterion [28].

3.4.5CYCLE-TO-CYCLE COMPRESSION

In this method we substitute a periodic signal by one cycle period and a count of the total number of cycles that occur in the signal. This approach is only applicable to periodic signals with the constraint that all the signal cycles are exactly the same. The ECG is quasi-periodic signal which does not change appreciably in morphology except as a result of the changes in the heart function. The cycle-to cycle ECG compression technique can result in a high compression ratio when applied to Holter ECG monitoring[14,25,30].

3.5 COMPRARISION

The comparision of different data compression techniques is given in table 3.1 below.

Table 3.1 Comparison of different data compression techniques for 100 ECG cycles

Method	Compression ratio	Sampling frequency (Hz)	PRD (%)
1. AZTEC [8]	10.0	500	28.0
2. TP [40]	2.0	200	5.3
3. CORTES [1]	4.8	200	7.0
4. FAN/SAPA [22]	3.0	250	4.0
5. Entropy coding of second difference ECG [11]	2.8	250	-
6. Peak-Picking (Spline) with Entropy Coding [27]	10.0	500	14.0
7. DPCM-Delta with threshold [55]	4.0	300	-
8. DPCM-linear Prediction [34]	2.5	250	-
9. DPCM-linear prediction, interpolation and entrophy coding [48]	7.8	500	3.5
10. Orthogonal transform (CT, KLT, HT) [2]	3.0	250	-
11. Dual application of K-L transform [60]	12.0	250	-
12. Fourier descriptors [46]	7.4	250	7.0
13. ANN method [49]	16.15	500	-

There is no standard data available for clinically acceptable deviations in papers between original and reconstructed signals. This issue

has been discussed with a group of cardiologists, according to their view the deviation of 15% in 'R' peaks,10% in P and Q peaks and 12% in S and T peaks is acceptable [36].

Detection of R-peaks and computation of R-R interval of an ECG record is an important requirement.Many methods, both hardware & software, have been developed for the detection of QRS complex [42,43,45,57].

The application of discrete Karhumen-Loeve series is discussed by Womble et al. and compression ratio of better then 12:1 for ECG is obtained for multilead ECG[60].

The ECG data contains noise such as base line wander, power-line frequency components. To remove this filtering is done.

The data compression by high-degree polynomial approximation is discussed by Philips et al. and is compared with discrete cosine transform and is found to yield a significant higher data compression for a given signal quality [44].

CHAPTER - 4

ECG DATA COMPRESSION USING TRANSFORMATIVE TECHNIQUES

4.1 INTRODUCTION

Similar to direct data compression techniques, most of the transformative compression techniques have been employed in multilead ECG compression and are useful for ECG wave detection.

The transformative techniques involve preprocessing the input signal by means of a linear orthogonal transformation and properly encoding the transformed output and reducing the amount of data needed to adequately represent the original signal. Upon signal reconstruction, an inverse transformation is performed and the original signal is recovered with a certain degree of error. There are many techniques to analyze degree of error and the same are discussed in next chapter.

The block diagram to obtain transform and its inverse is shown in Fig. 4-1. Many transformation techniques have been used for the compression of ECG signal .The Fourier Transform [FT], Orthogonal Exponentials, KL Transform (KLT), Haar Transform (HT), Walsh Transform (WT) and Cosine Transform (CT) have been successfully used for data compression. [2,7,24,37,61]. In all the above techniques, the signal is transformed into coefficients of respective transformations and the effective length of the coefficients is retained. The quality of retrieved signal is not very good because of the truncation of the number of coefficients Among the ECG

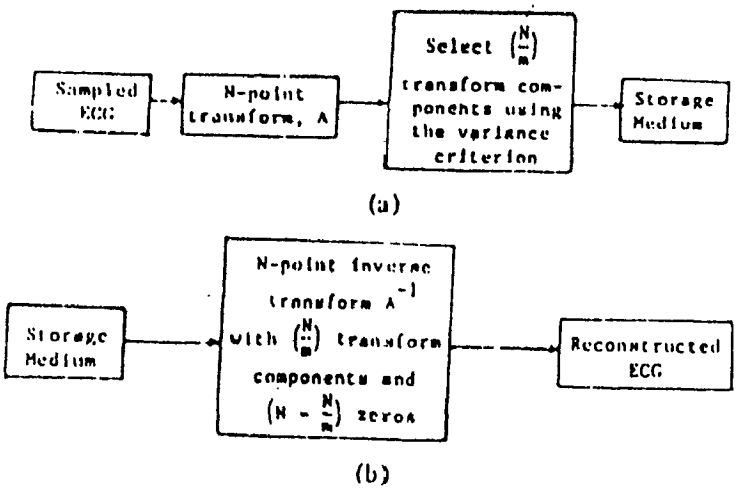


Fig. 4.1 : PERTAINING TO AN m:1 DATA COMPRESSION.
(a) STORAGE (b) RETRIEVEL [2]

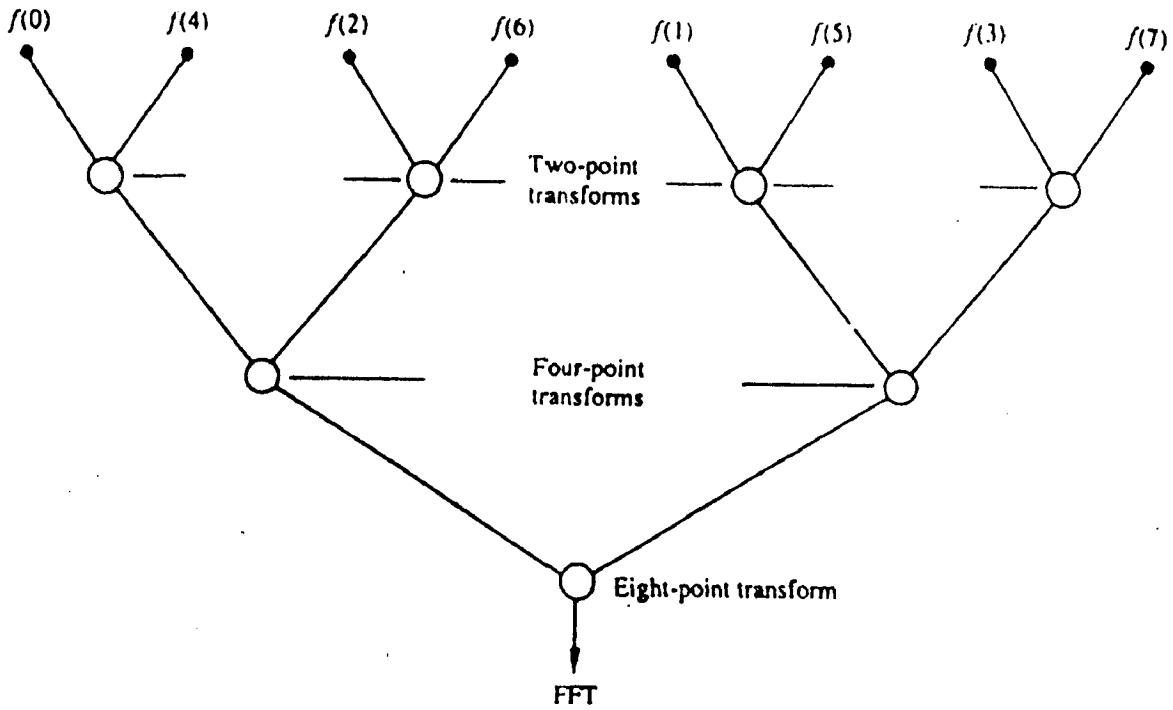


Fig. 4.2 : ORDERED INPUT ARRAY AND ITS USE IN SUCCESIVE-DOUBLING METHOD [23]

transformation techniques, the highest compression ratio for multilead ECG data has been reported for the KLT technique [2].

Irrespective of being direct type, these techniques do not eliminate the interference and noise signals from the real ECG signal. These also have dynamic adaptability and result in enhanced quality of retrieved signal.

4.2 FAST FOURIER TRANSFORM

The number of complex multiplications and addition required to implement the Fourier Transform is proportional to N^2. The Fourier and Inverse Fourier is given by

$$F(u) = \frac{1}{N} \sum_{x=o}^{N-1} f(x) \exp[-j2\pi ux/N] \qquad (4.1)$$

and

$$f(x) = \sum_{u=o}^{N-1} F(u) \exp[j2\pi ux/N] \qquad (4.2)$$

For each of the N values, expansion of the summation requires N complex multiplications of f(x) by exp[-j2πux/N) and N-1 additions of the results. The term of exp[-j2πux/N] can be computed once and stored in a table for all subsequent applications. Fourier Transform equation can make the number of multiplication and addition operations proportional to $N\log_2 N$. The implementation procedure is called the Fast Fourier Transform (FFT) algorithm. The reduction proportionality from N^2 to $N\log_2 N$ operations

represents a significant saving in computation effort.The comparison of N^2(Direct FT) versus $NLog_2N$ (FFT) is given in table 4.1 below for various values of N[23].

Table 4.1 A Comparison of N^2 versus N Log_2N for Various Values of N

N	N^2 (Direct FT)	N log_2N (FFT)	Computational Advantage (N/Log_2N)
2	4	2	2.00
4	16	8	2.00
8	64	24	2.67
16	256	64	4.00
32	1,024	160	6.40
64	4,096	384	10.67
128	16,384	896	18.29
256	65,536	2,048	32.00
512	262,144	4,608	56.89
1024	1,048,576	10,240	102.40
2048	4,194,304	22,528	186.18
4096	16,777,216	49,158	341.33
8192	67,108,864	106,496	630.15

4.2.1 FFT ALGORITHM

The FFT algorithm discussed in this chapter is based on the so called successive doubling method. The FFT is given by [23].

$$F(u) = \frac{1}{N} \sum_{x=o}^{N-1} f(x) W_N^{ux} \tag{4.3}$$

where

$$W_N = \exp[-J2\pi/N] \tag{4.4}$$

and N is said to be of the form

$$N = 2^n \tag{4.5}$$

Where n is a positive integer. So N can be expressed as 2M. (N = 2M). M is also a positive integer. By substituting value of N in Eqn.(4.3) we get

$$F(u) = \frac{1}{2M} \sum_{x=o}^{2M-1} f(x) W_{2M}^{ux} \tag{4.6}$$

$$= \frac{1}{2} \left[\frac{1}{M} \sum_{x=o}^{M-1} f(2x) W_{2M}^{u(2x)} + \frac{1}{M} \sum_{x=o}^{M-1} f(2x+1) W_{2M}^{u(2x+1)} \right] \tag{4.7}$$

From Eq.(4.4), $W_{2M}^{2ux} = W_M^{x}$,and so Eq.(4.7) may be expressed in the form

$$F(u) = \frac{1}{2} \left[\frac{1}{M} \sum_{x=o}^{M-1} f(2x) W_M^{ux} + \frac{1}{M} \sum_{x=o}^{M-1} f(2x+1) W_M^{ux} W_{2M}^{u} \right] \tag{4.8}$$

Defining $F_{even}(u) = \frac{1}{M} \sum_{x=o}^{M-1} f(2x) W_M^{ux}$ (4.9)

for u = 0,1,2,.....M-1,and

$$F_{odd}(u) = \frac{1}{M} \sum_{x=o}^{M-1} f(2x+1)\, W_M^{ux} \qquad (4.10)$$

for u = 0,1,2 .... M-1, reduces Eq. (4.8) to

$$F(u) = \frac{1}{2} \left[F_{even}(u) + F_{odd}(u) W_{2M}^{u}\right] \qquad (4.11)$$

Also, since $W_M^{u+M} = W_{2M}^{u}$ and $W_2^{u+M} = -W_{2M}^{u}$; Eqs. (4.9) and (4.10) give

$$F(u + M) = \frac{1}{2} \left[F_{even}(u) - F_{odd}(u) W_{2M}^{u}\right] \qquad (4.12)$$

Implementation of Eqn.(4.9) to (4.12) gives the successive doubling FFT algorithm.

4.2.2 THE INVERSE FFT

The Inverse FFT is obtained by implementing the forward transform by minor modification to the input. The forward Fourier Transform is given as [23]:

$$F(u) = \frac{1}{N} \sum_{x=o}^{N-1} f(x) \exp\left[-j2\pi ux/N\right] \qquad (4.13)$$

and inverse Fourier Transform is given by.

$$F(x) = \sum_{u=o}^{N-1} f(u) \exp\left[j2\pi ux/N\right] \qquad (4.14)$$

Taking the complex conjugate of Eqn.(4.14), and dividing both sides by N, yields.

$$\frac{1}{N} f^{*}(x) = \frac{1}{N} \sum_{u=o}^{N-1} f^{*}(u)\exp[-j2\pi ux/N] \qquad (4.15)$$

Comparing this result with Eqn. (4.13) shows that the right hand side of Egn. (4.15) is the same as for the forward transform. Thus by putting $F^{*}(u)$ into an algorithm designed to compute the forward transform gives the quality $F^{*}(x)/N$. Taking the complex conjugate and multiplying by N yields the desired inverse f(x).

4.2.3 SOFTWARE IMPLEMENTATION

The flow charts and Programs in "C" for implementation of FFT are given in Appendix (I) and (II) respectively.

The Basic consideration in calculating the FFT is that the input data must be arranged in the order required for successive applications of Egn. (4.9) and (4.10). For example we want to compute FFT of an eight-point function. {f(0),f(1) ----, f(7)} Eqn.(4.9) uses the samples with even argument. {f(0),f(2),f(4),f(6)} and Eqn.(4.10) for odd arguments {f(1),f(3),f(5),f(7)}. However each four-point is computed as two two-point transforms, i.e. even point {f(0),f(4)} and odd part {f(2),f(6)} and so on. No further rearrangement is required as each two element set is considered as having one even and one odd element. The successive doubling algorithm operates on this array in the manner as shown in Fig. 4-2.

Table 4.2 summarizes the procedure for N = 8. By examine table 4.2 given below, we say that an input array follow as a simple bit reversal.

Table-4.2 Example of Bit Reversal and Reordering of Array for Input into FFT Algorithm

Original Argument			Original Array	Bit-Reversed Argument			Recorded Array
0	0	0	f(0)	0	0	0	f(0)
0	0	1	f(1)	1	0	0	f(0)
0	1	0	f(2)	0	1	0	f(0)
0	1	1	f(3)	1	1	0	f(0)
1	0	0	f(4)	0	0	1	f(0)
1	0	1	f(5)	1	0	1	f(0)
1	1	0	f(6)	0	1	1	f(0)
1	1	1	f(7)	1	1	1	f(0)

In the graphs, we used following notations:

'1' Graph of original ECG data

'12' Reconstructed graph of 1:2 compression ratio

'11' Reconstructed graph of 1:4 compression ratio

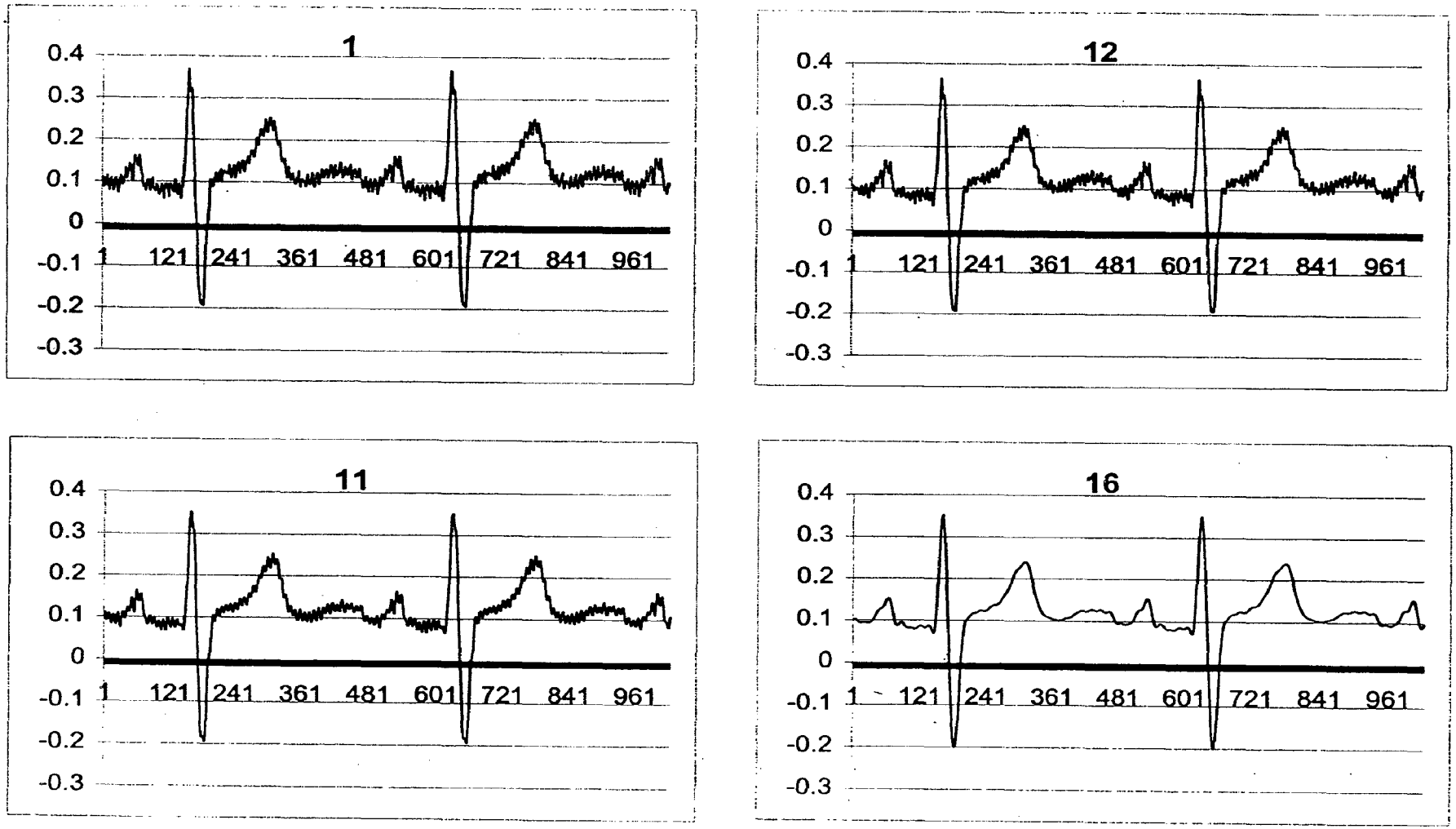
'16' Reconstructed graph of 1:8 compression ratio

All the graphs are plotted in between for amplitude verses sample number. Figs. 4.3 to 4.62 shows the software results of FFT analysis of ECG signal. The comments on the results are given in Chapter 5.

4.3 FAST WALSH TRANSFORM

The 1-D discrete Fourier Transform in general can be expressed as

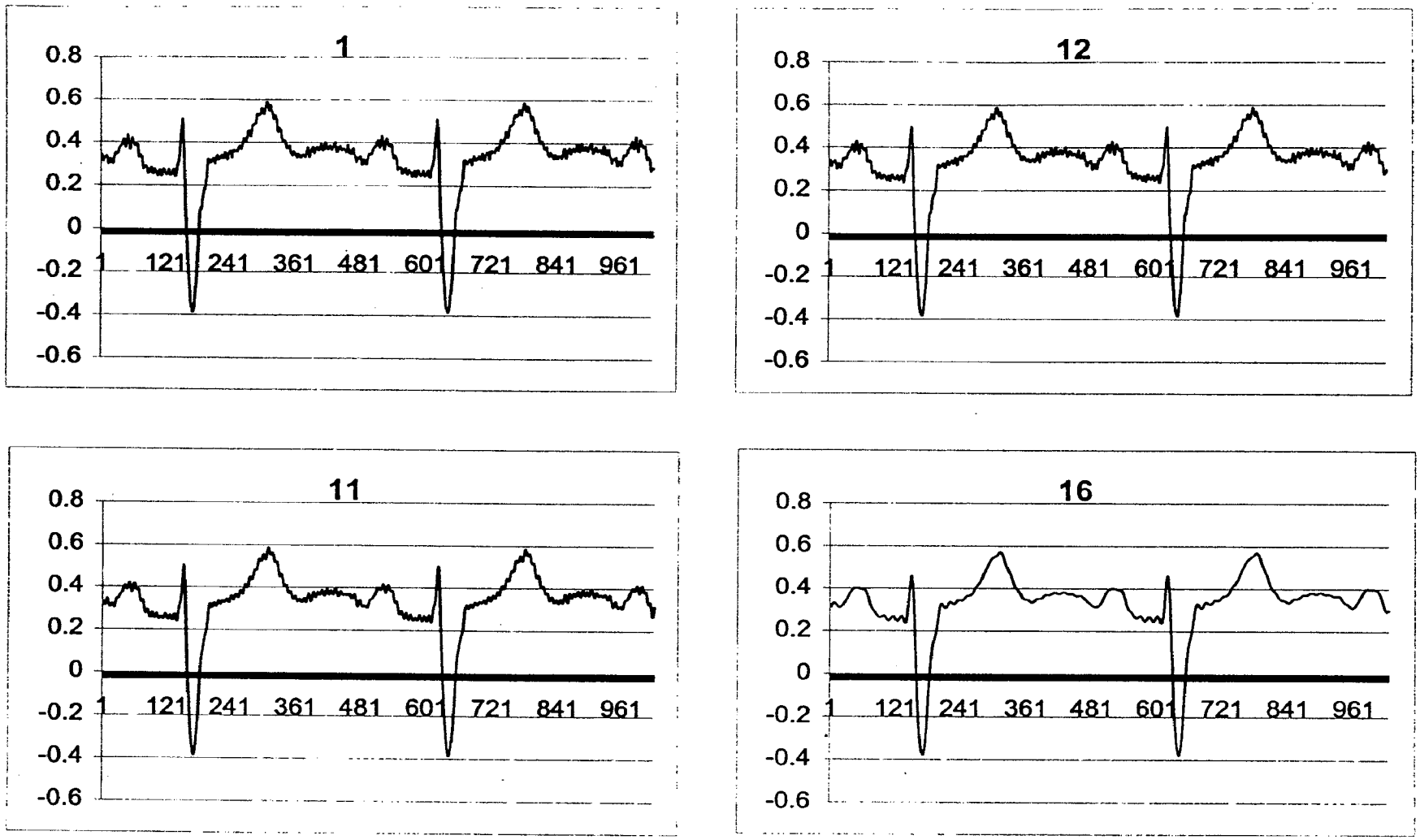
FOURIER ANALYSIS OF 1L1 ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

Fig. 4.3 FOURIER ANALYSIS OF 1L1 ECG DATA

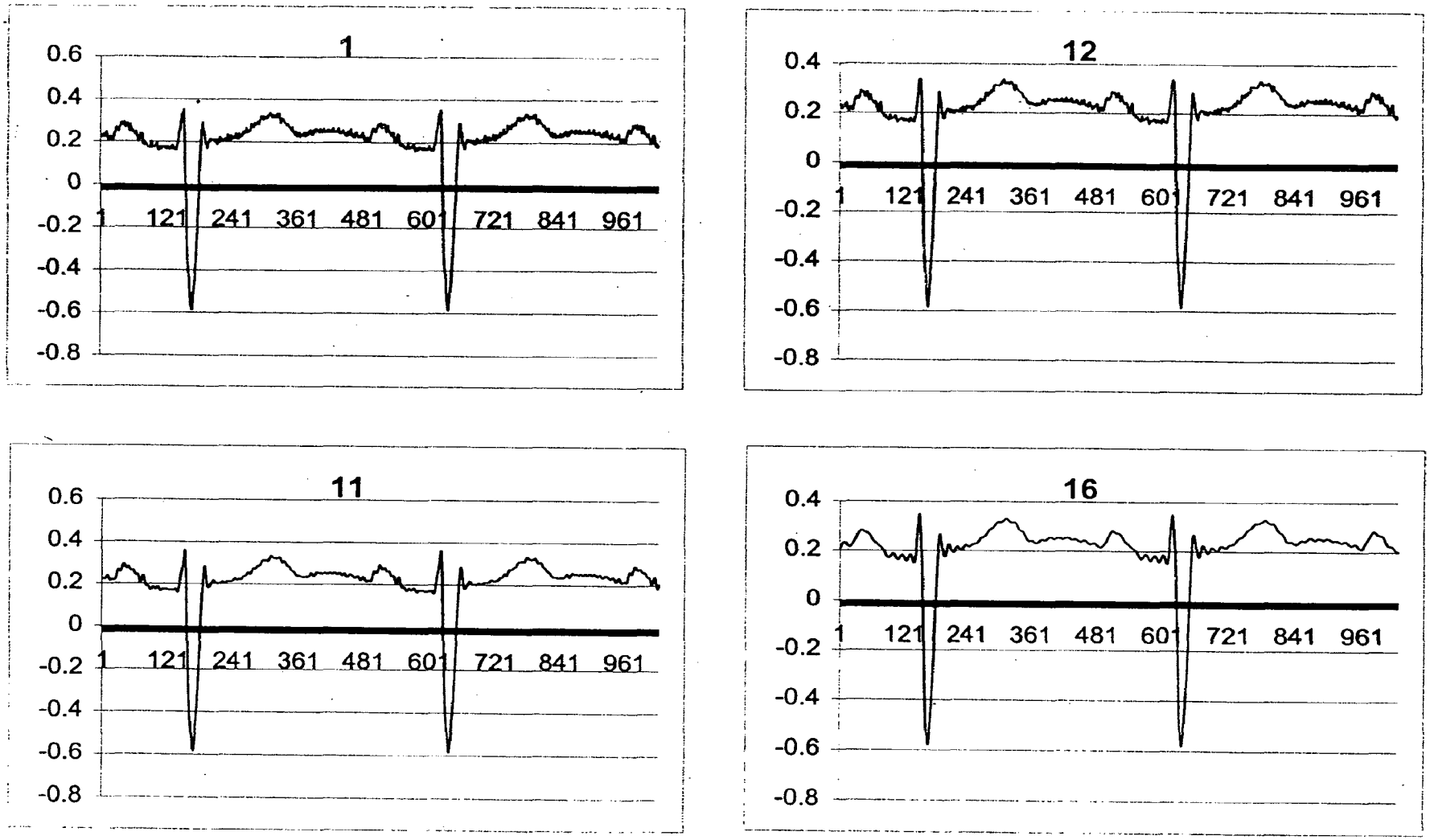
FOURIER ANALYSIS OF 1L2 ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

Fig. 4.4 FOURIER ANALYSIS OF 1L2 ECG DATA

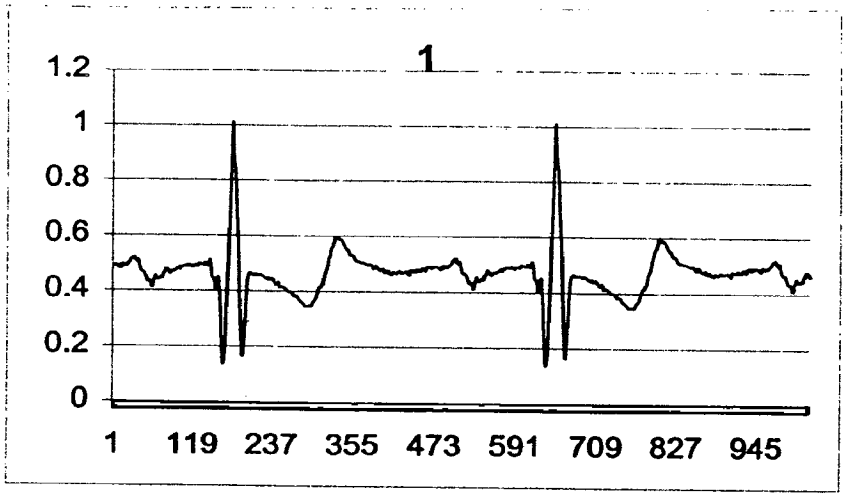
FOURIER ANALYSIS OF 1L3 ECG DATA

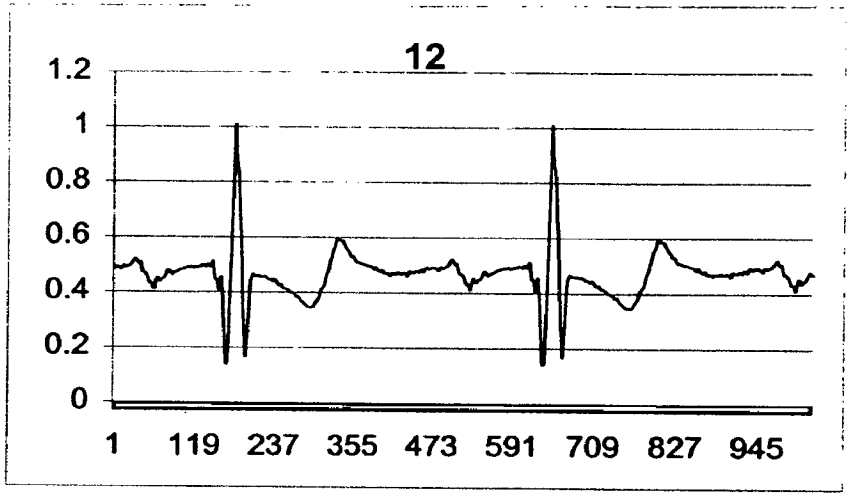


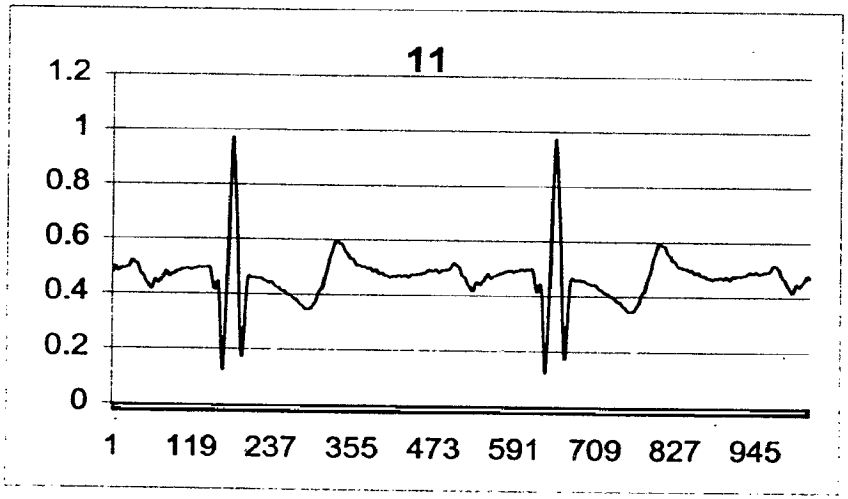
'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

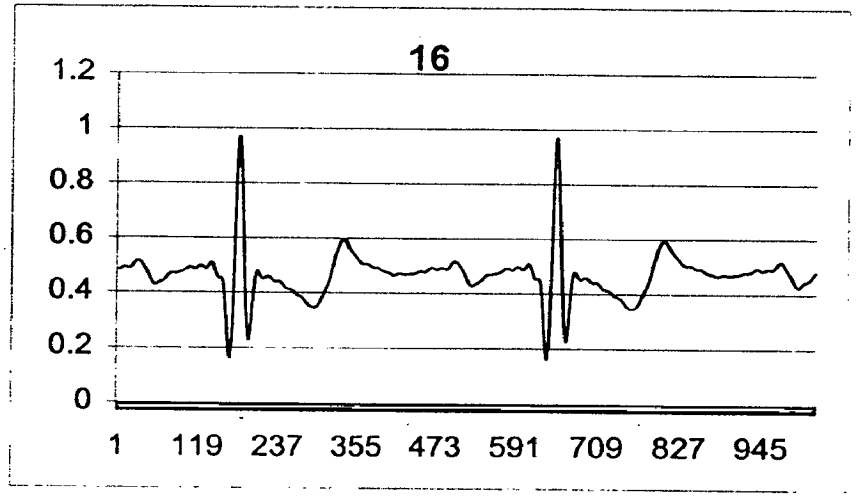
Fig. 4.5 FOURIER ANALYSIS OF 1L3ECG DATA

FOURIER ANALYSIS OF 1V1 ECG DATA





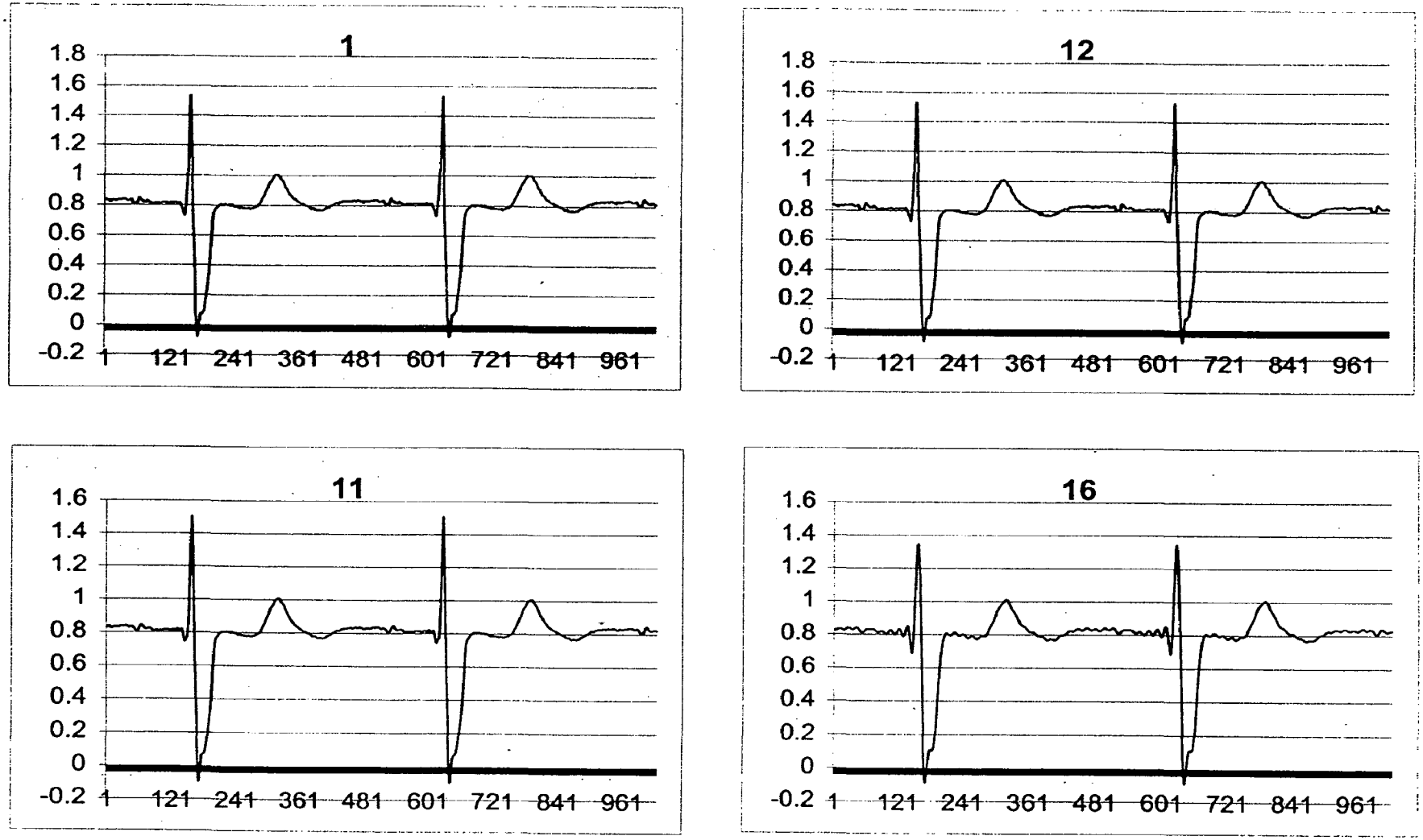




'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

Fig. 4.6 FOURIER ANALYSIS OF 1V1ECG DATA

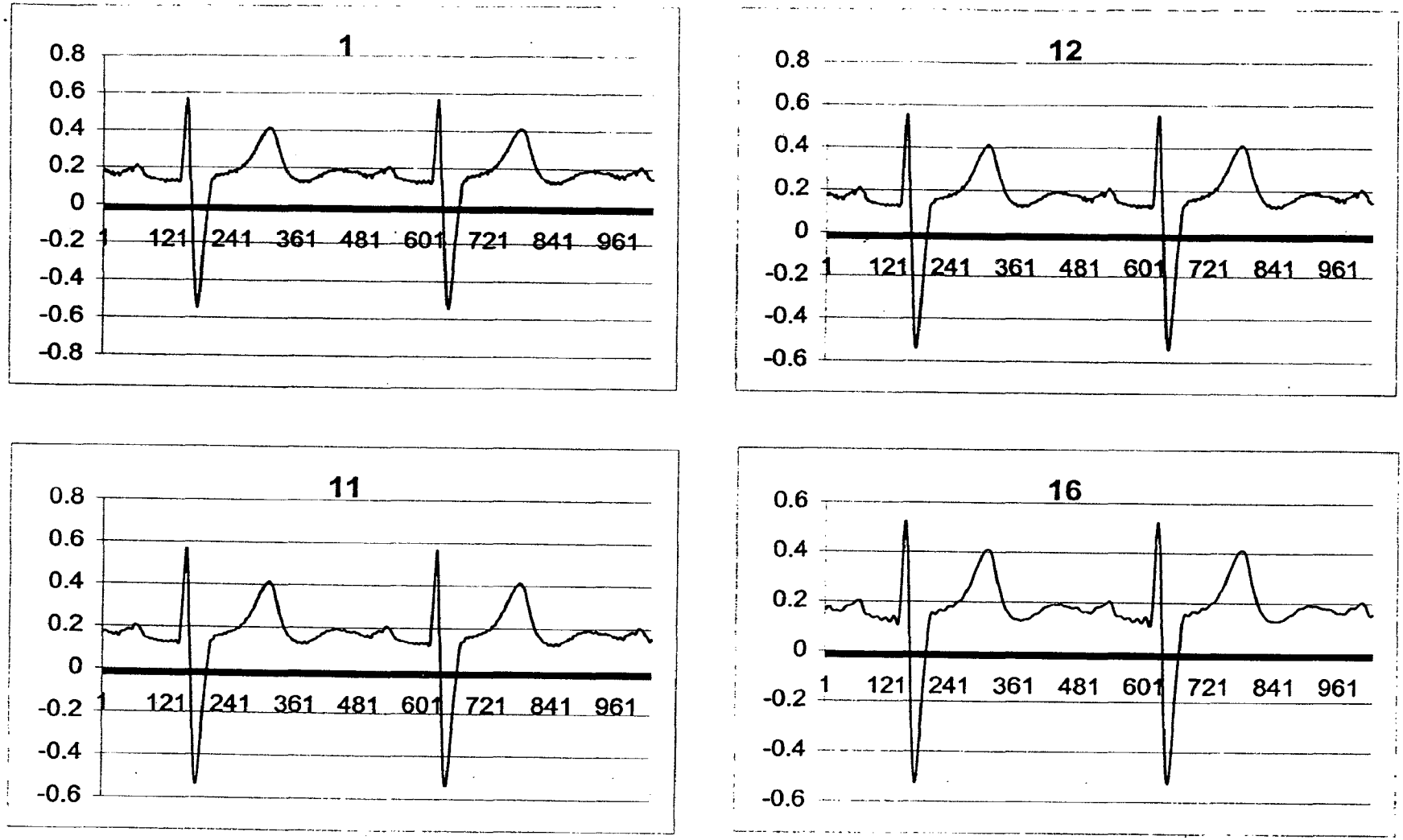
FOURIER ANALYSIS OF 1V2 ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

Fig. 4.7 FOURIER ANALYSIS OF 1V2ECG DATA

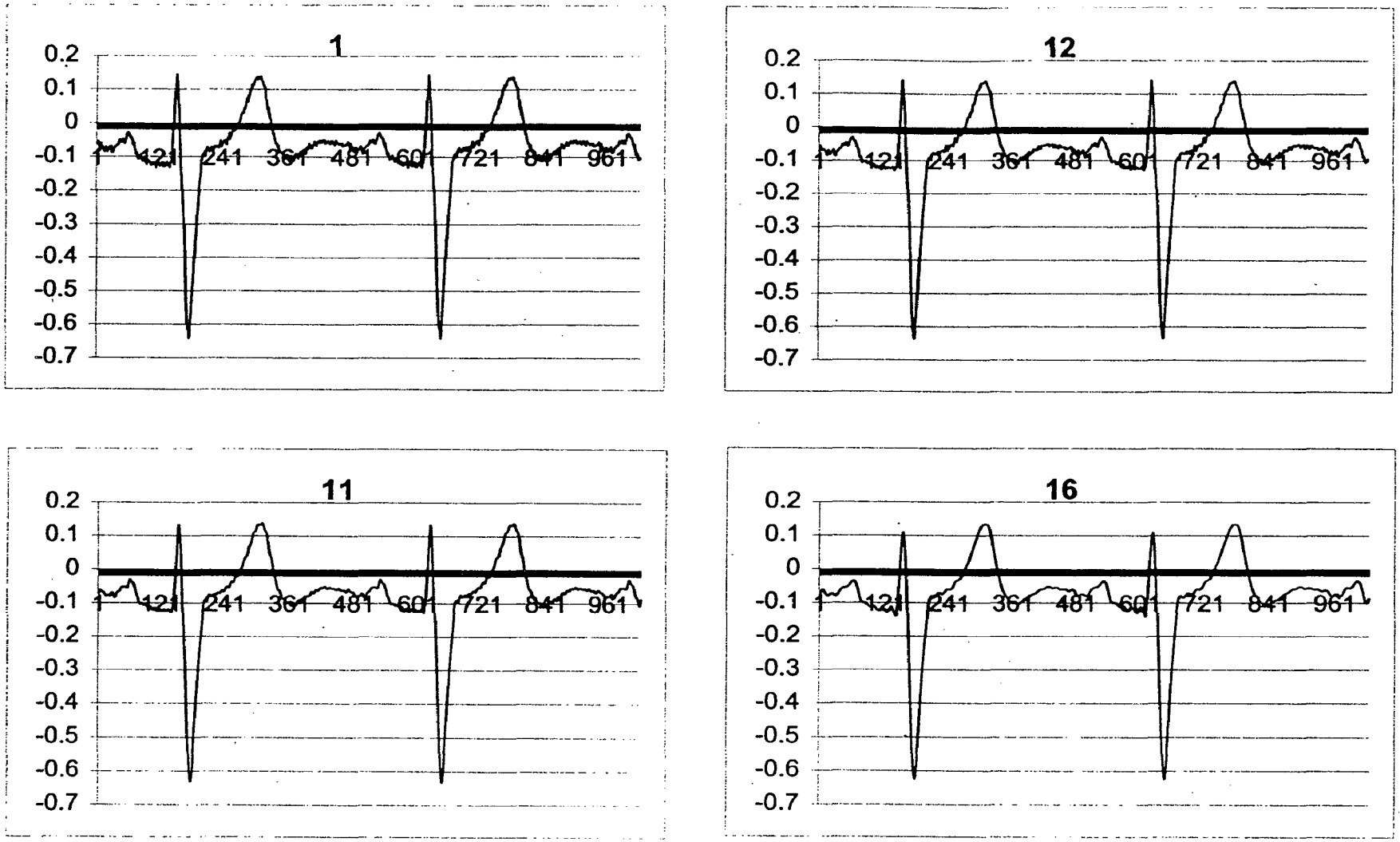
FOURIER ANALYSIS OF 1V5 ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

Fig. 4.10 FOURIER ANALYSIS OF 1V5 ECG DATA

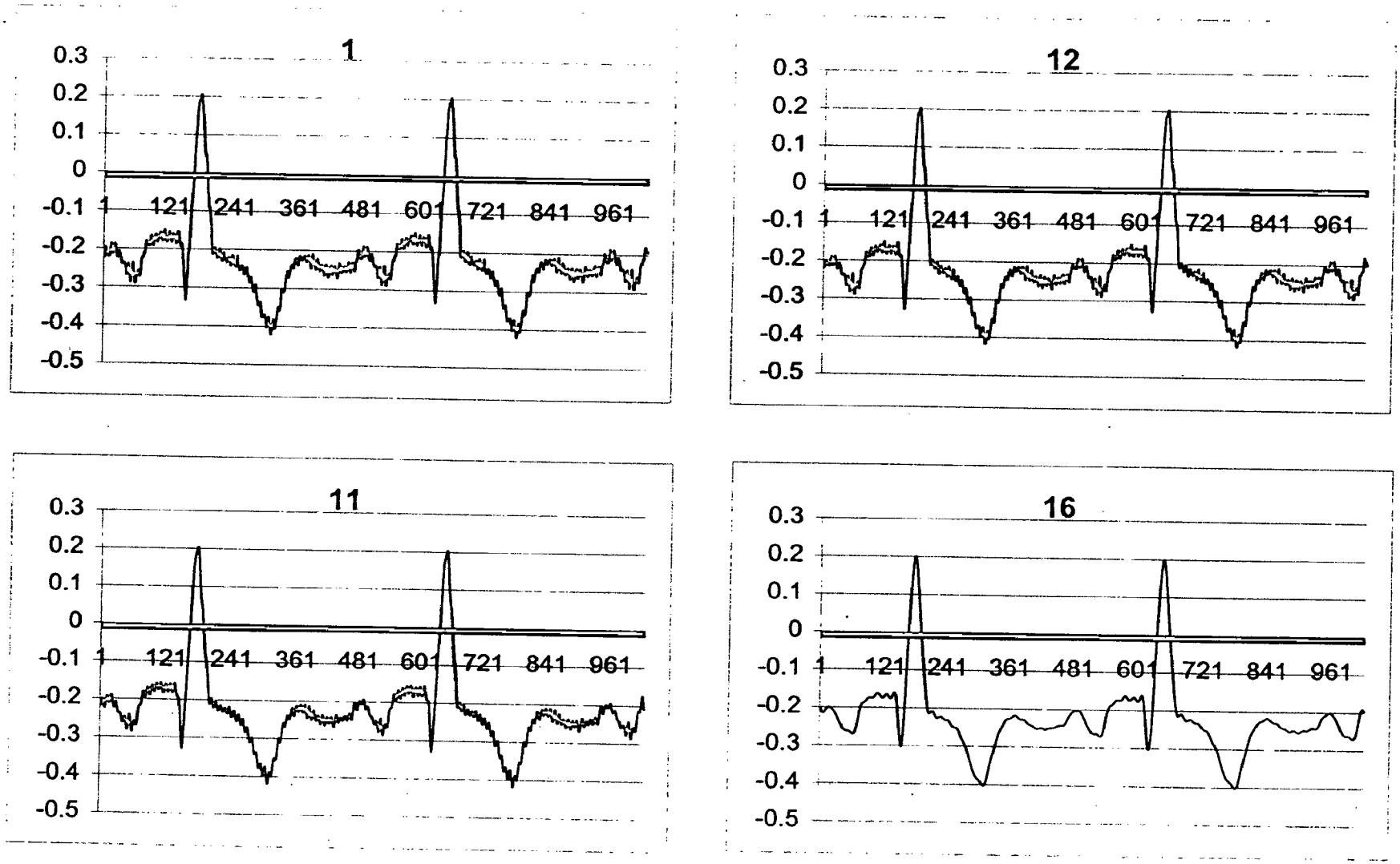
FOURIER ANALYSIS OF 1V6 ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

Fig. 4.11 FOURIER ANALYSIS OF 1V6 ECG DATA

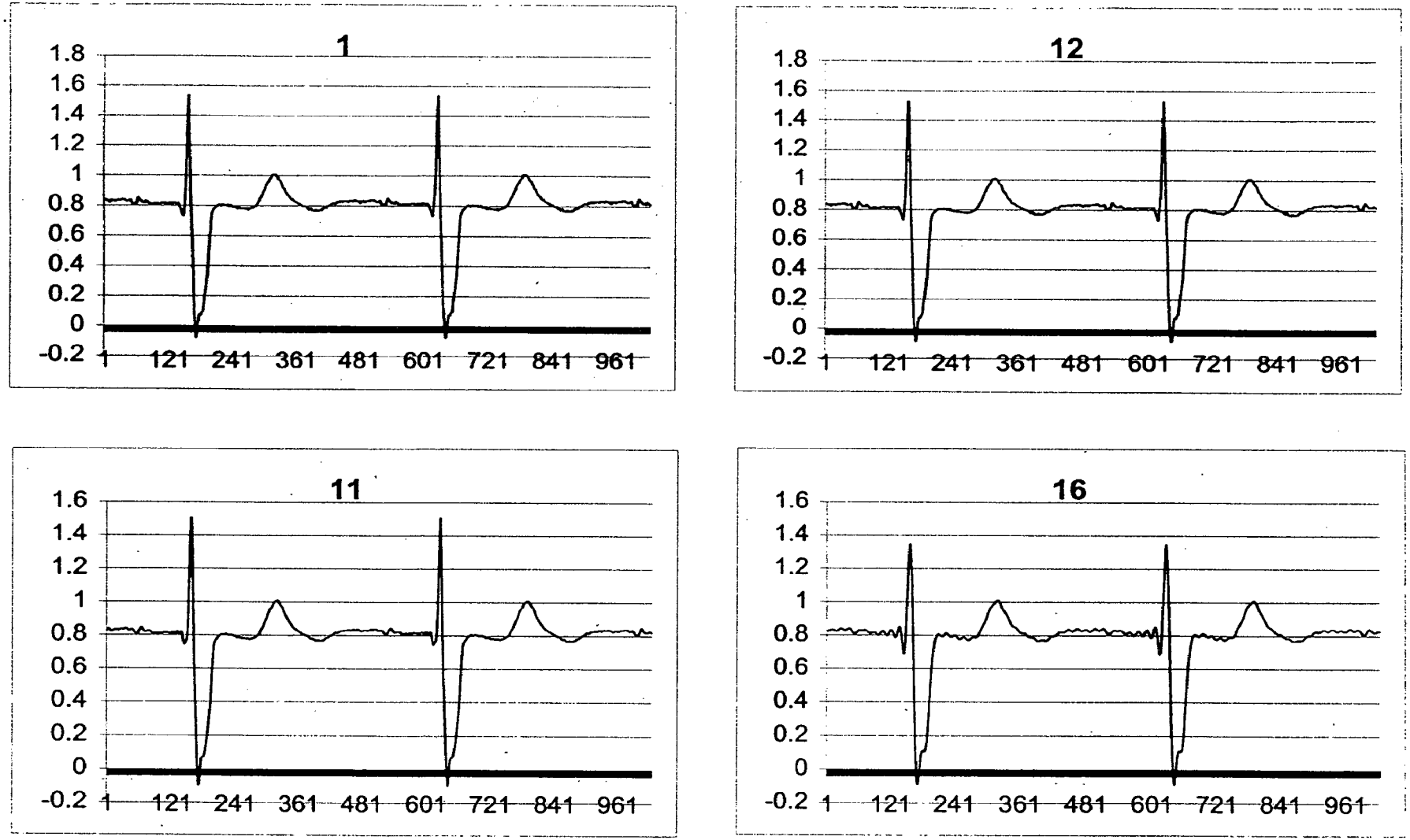
FOURIER ANALYSIS OF 1AR ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

Fig. 4.12 FOURIER ANALYSIS OF 1AR ECG DATA

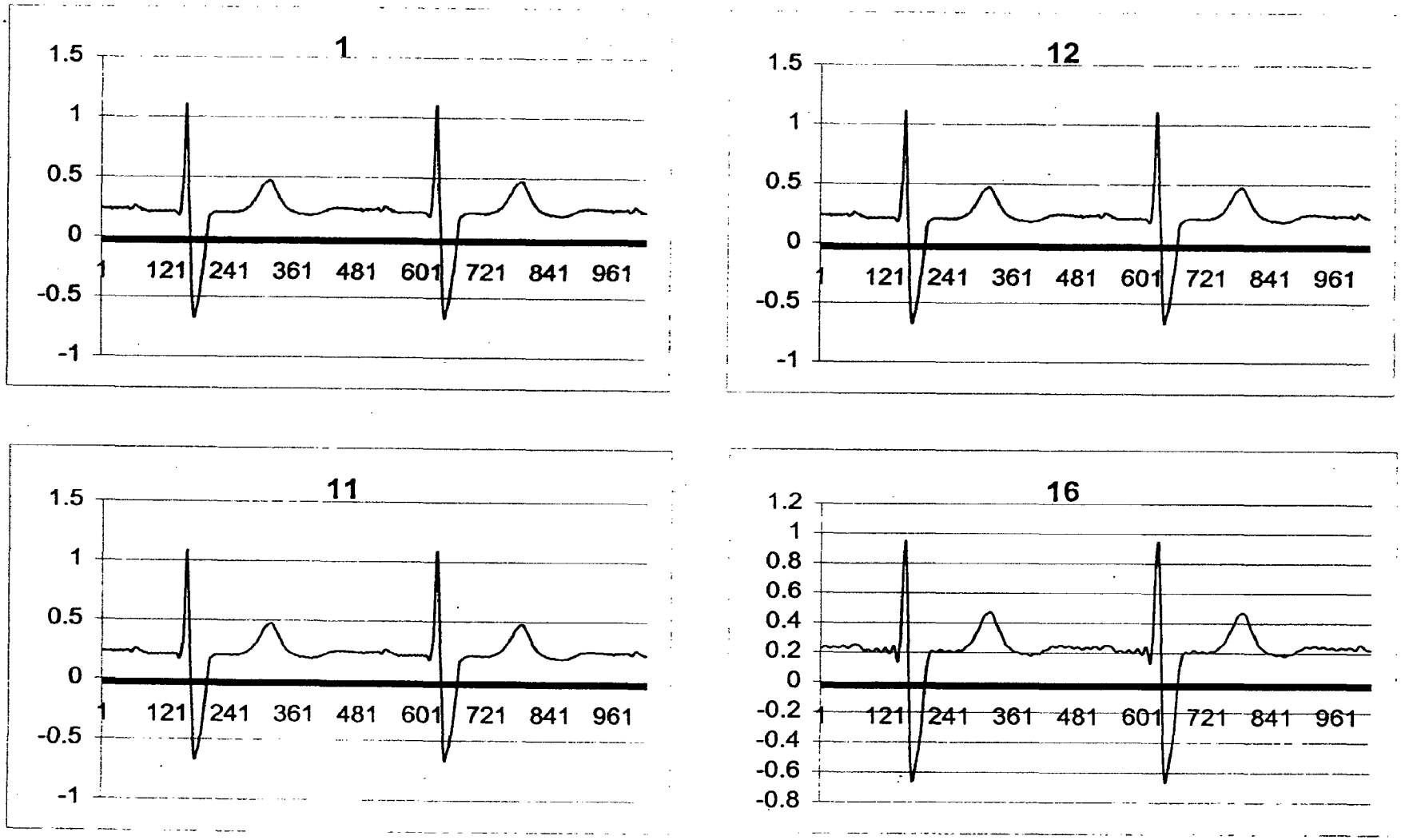
FOURIER ANALYSIS OF 1V2 ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

Fig. 4.7 FOURIER ANALYSIS OF 1V2ECG DATA

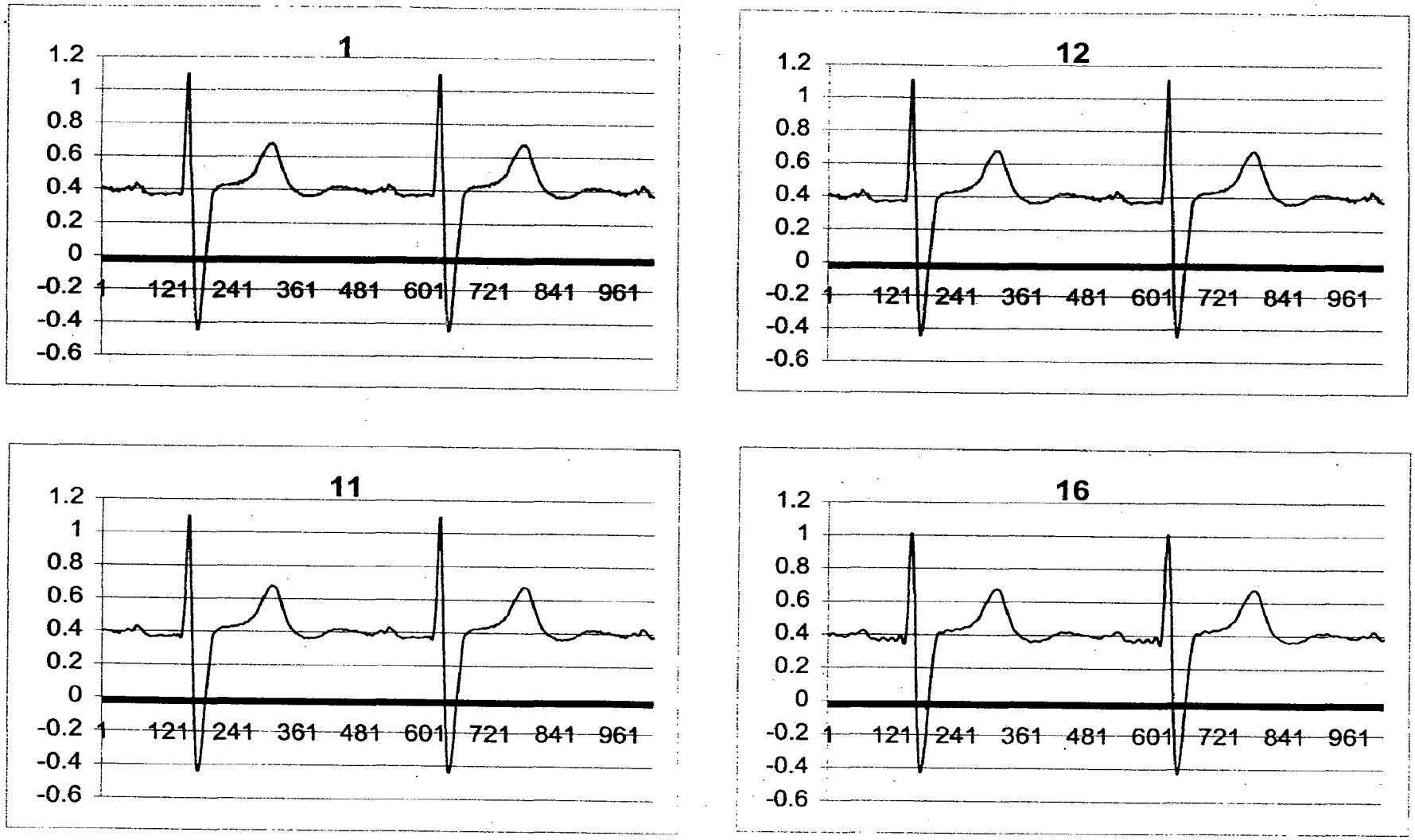
FOURIER ANALYSIS OF 1V3 ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

Fig. 4.8 FOURIER ANALYSIS OF 1V3 ECG DATA

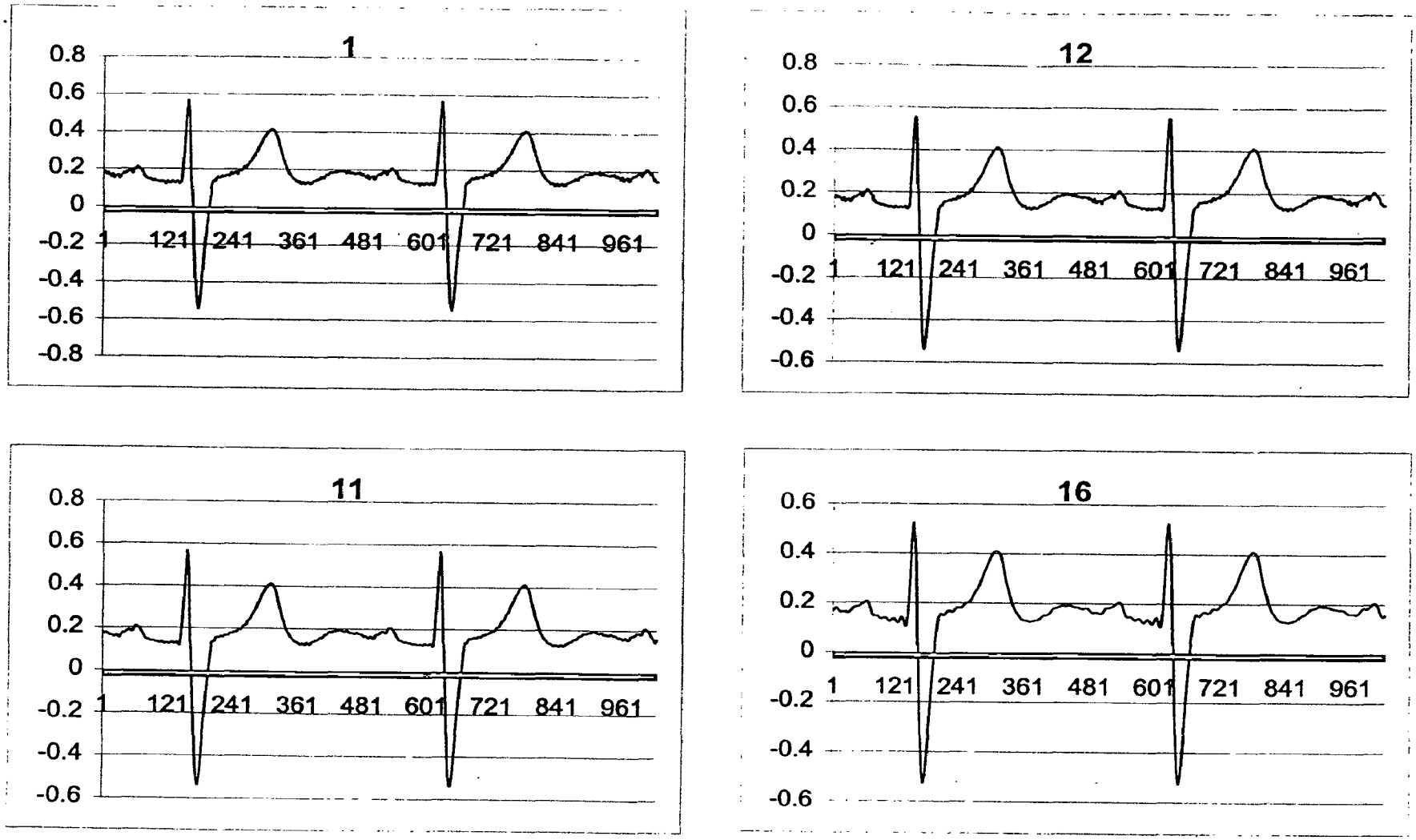
FOURIER ANALYSIS OF 1V4 ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

Fig. 4.9 FOURIER ANALYSIS OF 1V4 ECG DATA

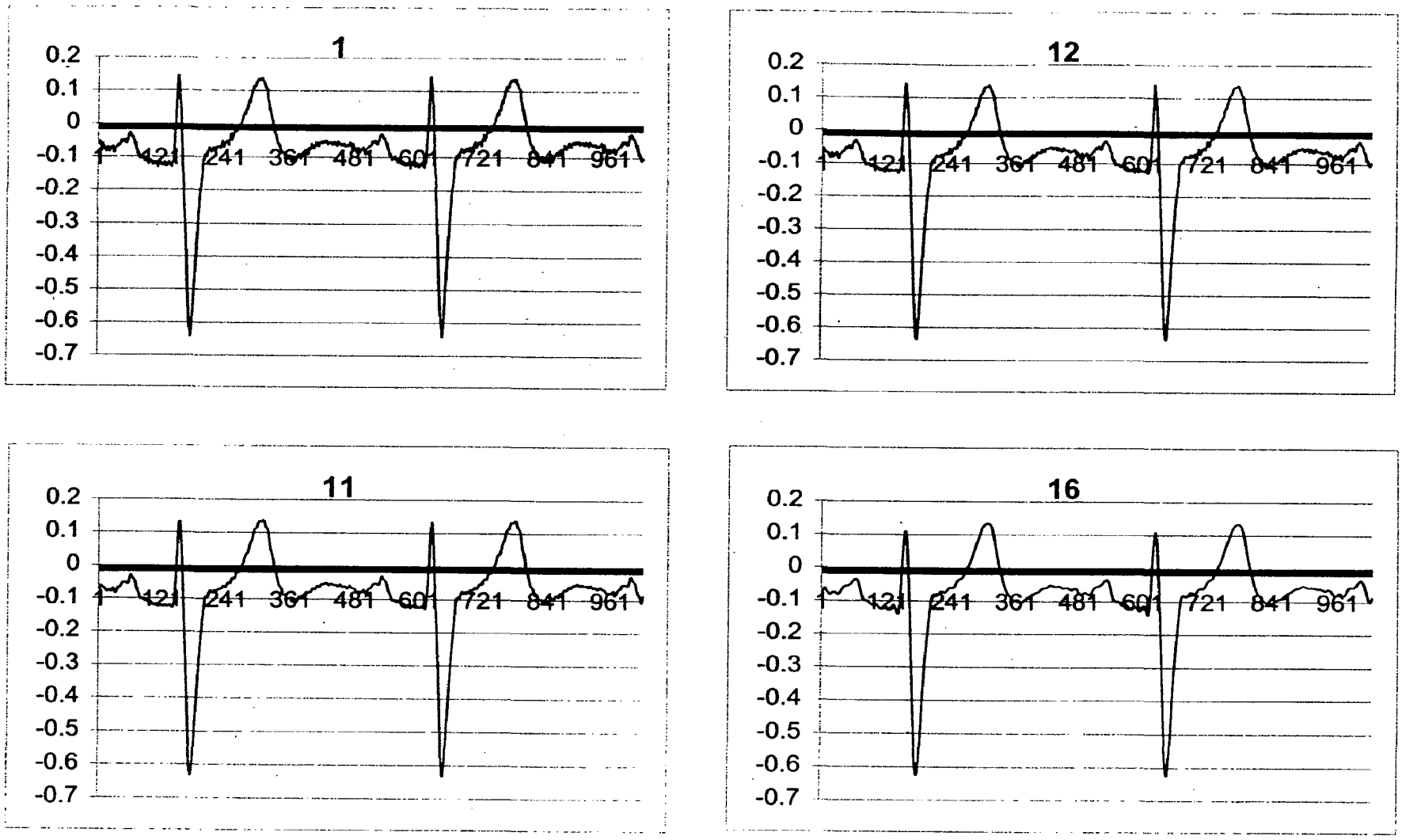
FOURIER ANALYSIS OF 1V5 ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

Fig. 4.10 FOURIER ANALYSIS OF 1V5 ECG DATA

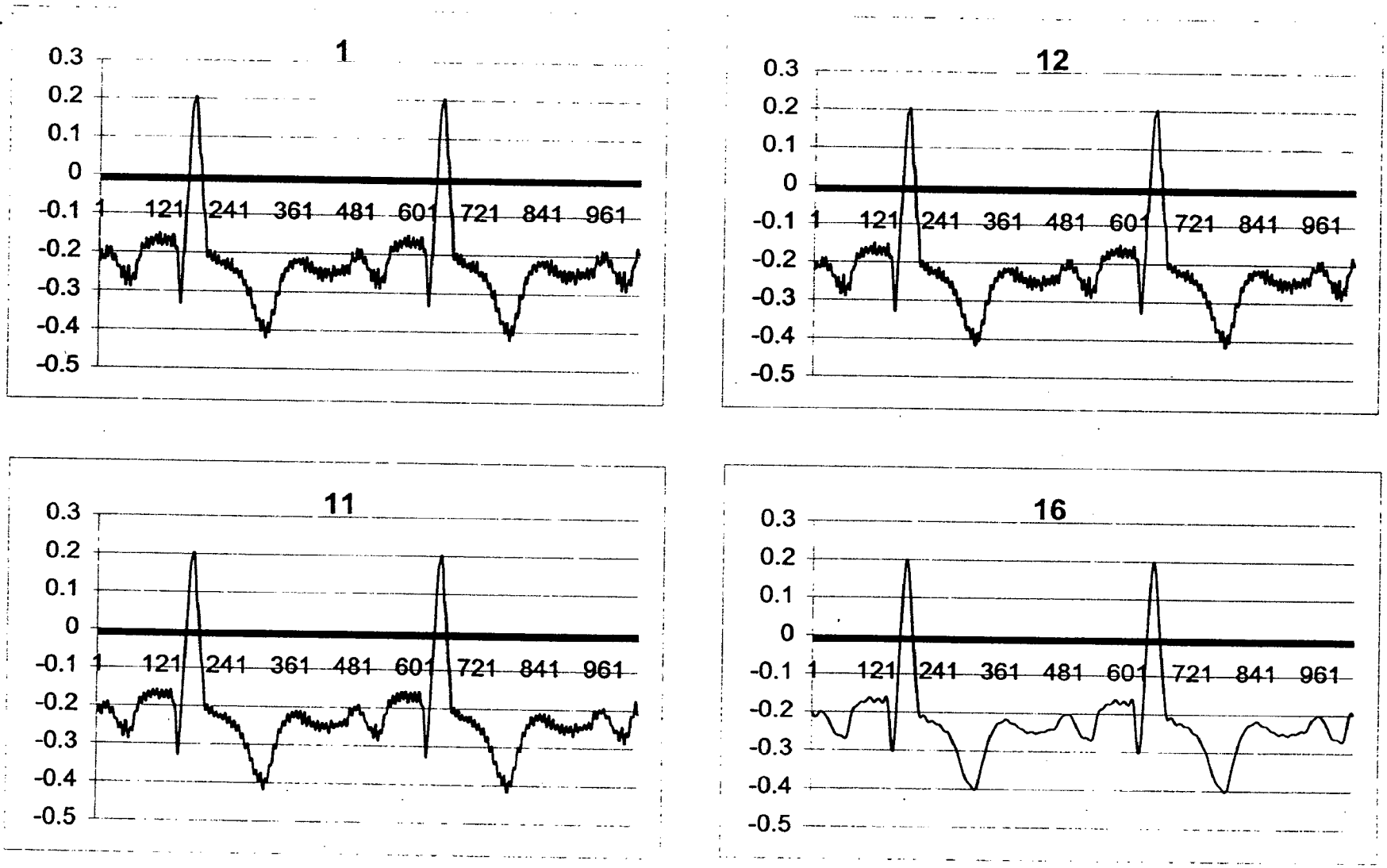
FOURIER ANALYSIS OF 1V6 ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

Fig. 4.11 FOURIER ANALYSIS OF 1V6 ECG DATA

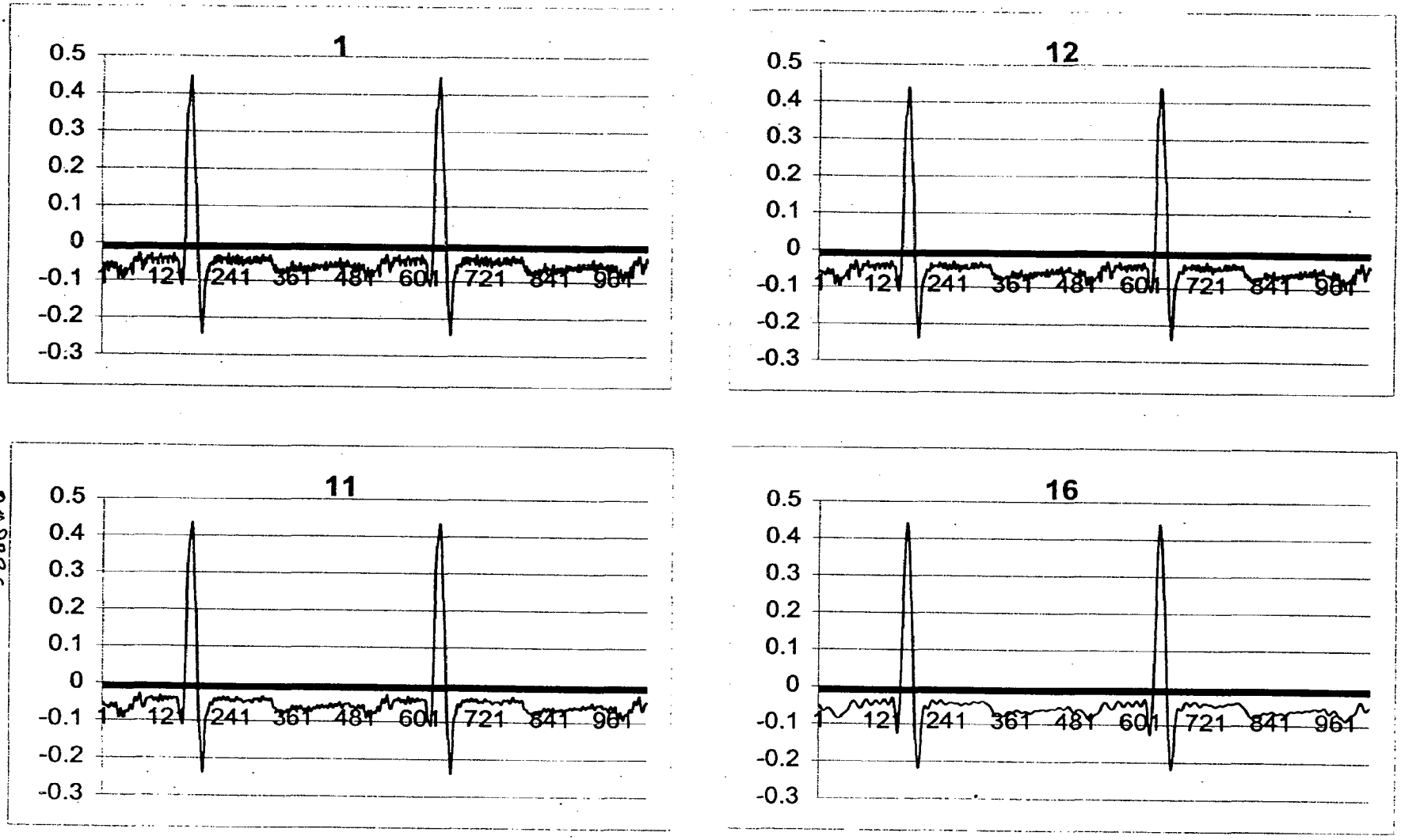
FOURIER ANALYSIS OF 1AR ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

Fig. 4.12 FOURIER ANALYSIS OF 1AR ECG DATA

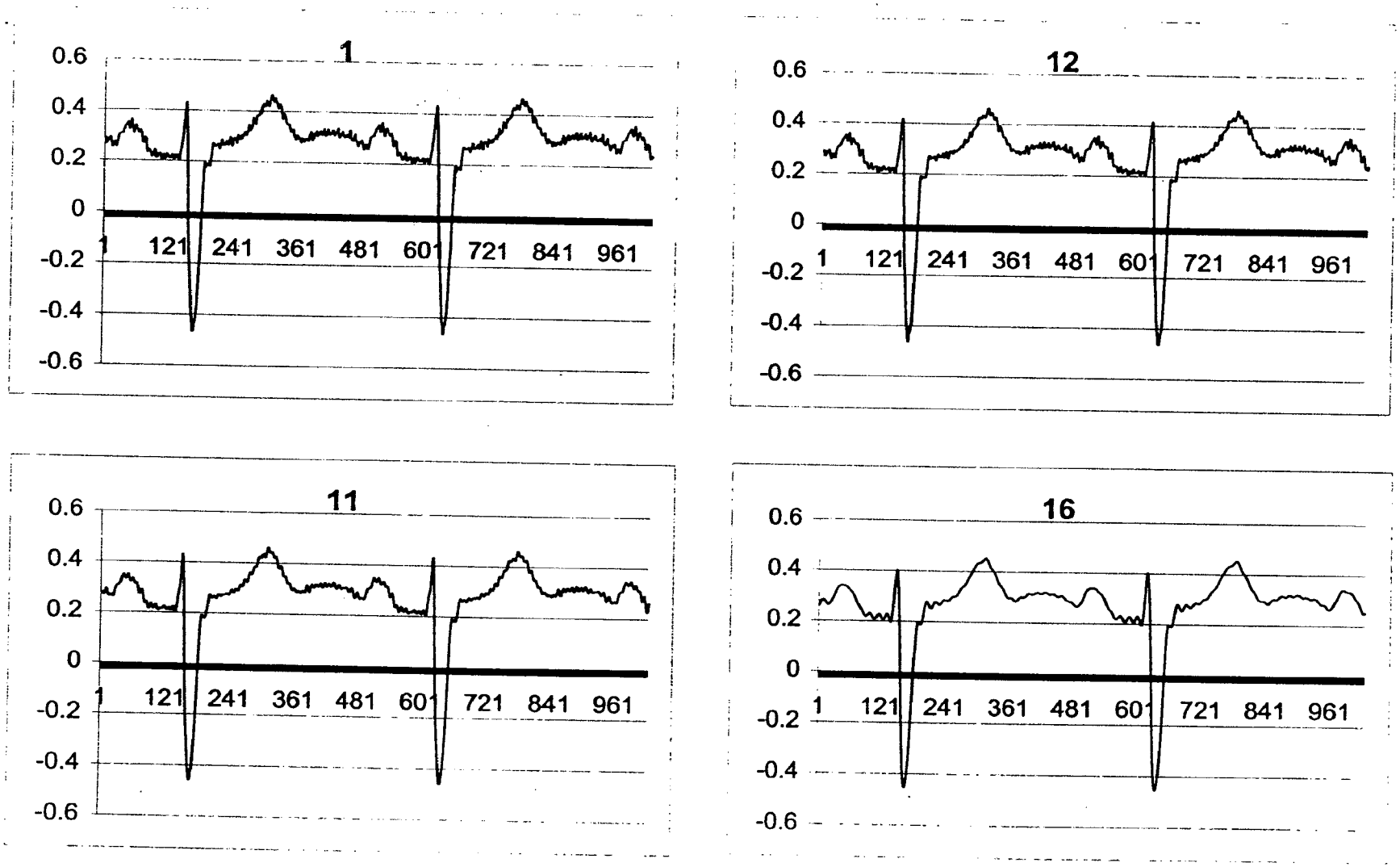
FOURIER ANALYSIS OF 1AL ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

Fig. 4.13 FOURIER ANALYSIS OF 1AL ECG DATA

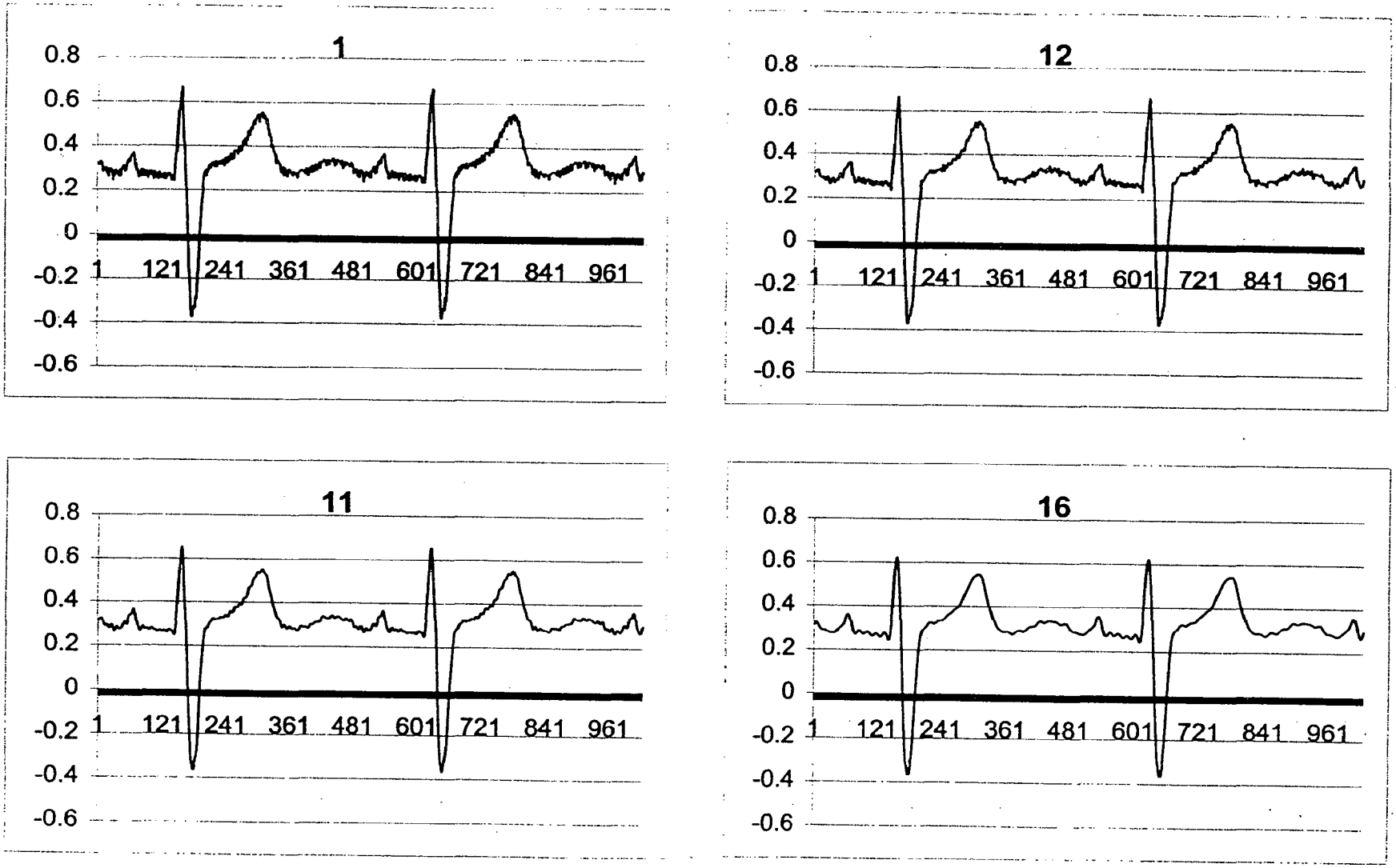
FOURIER ANALYSIS OF 1AF ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

Fig. 4.14 FOURIER ANALYSIS OF 1AF ECG DATA

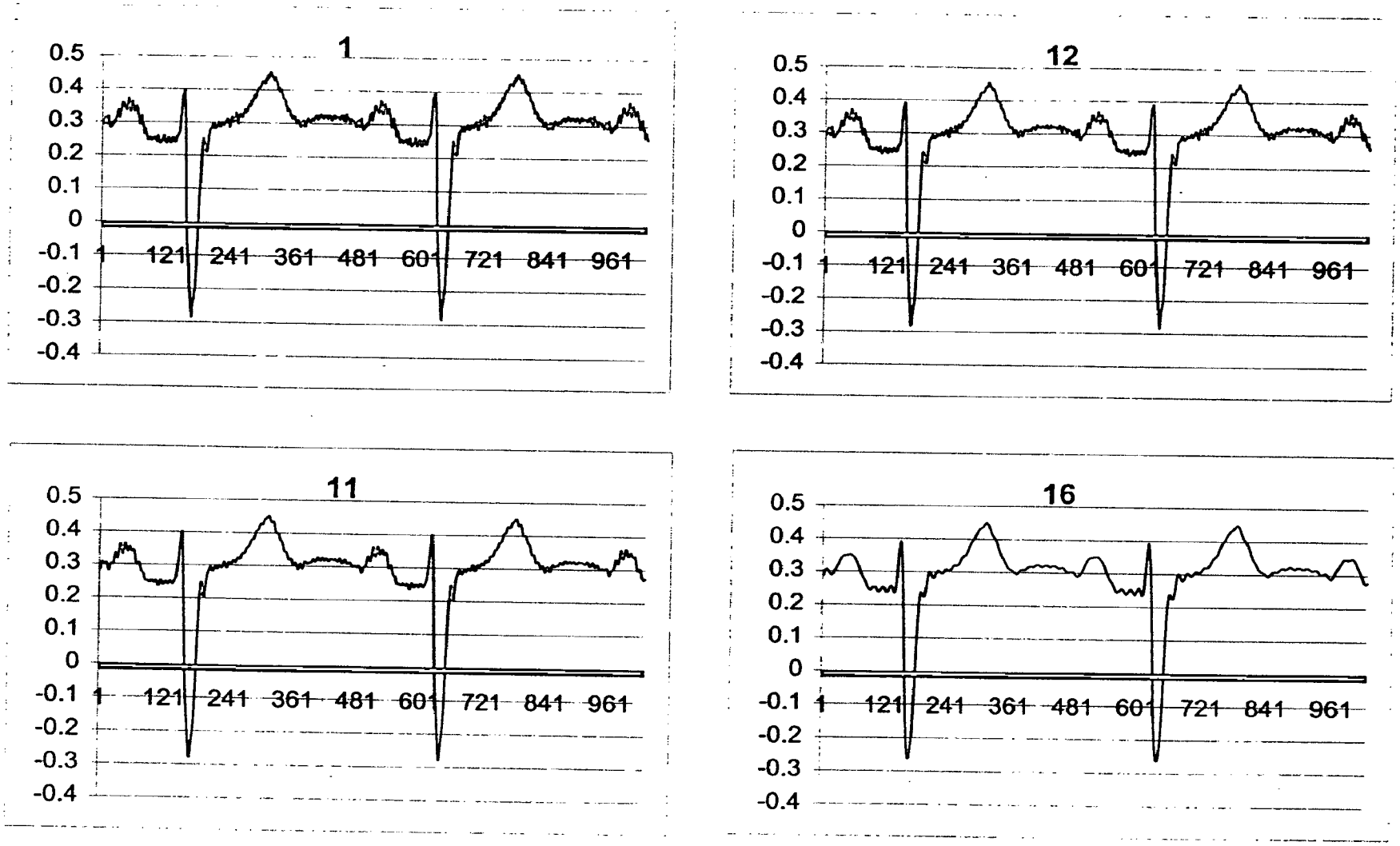
FOURIER ANALYSIS OF 1X ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

Fig. 4.15 FOURIER ANALYSIS OF 1X ECG DATA

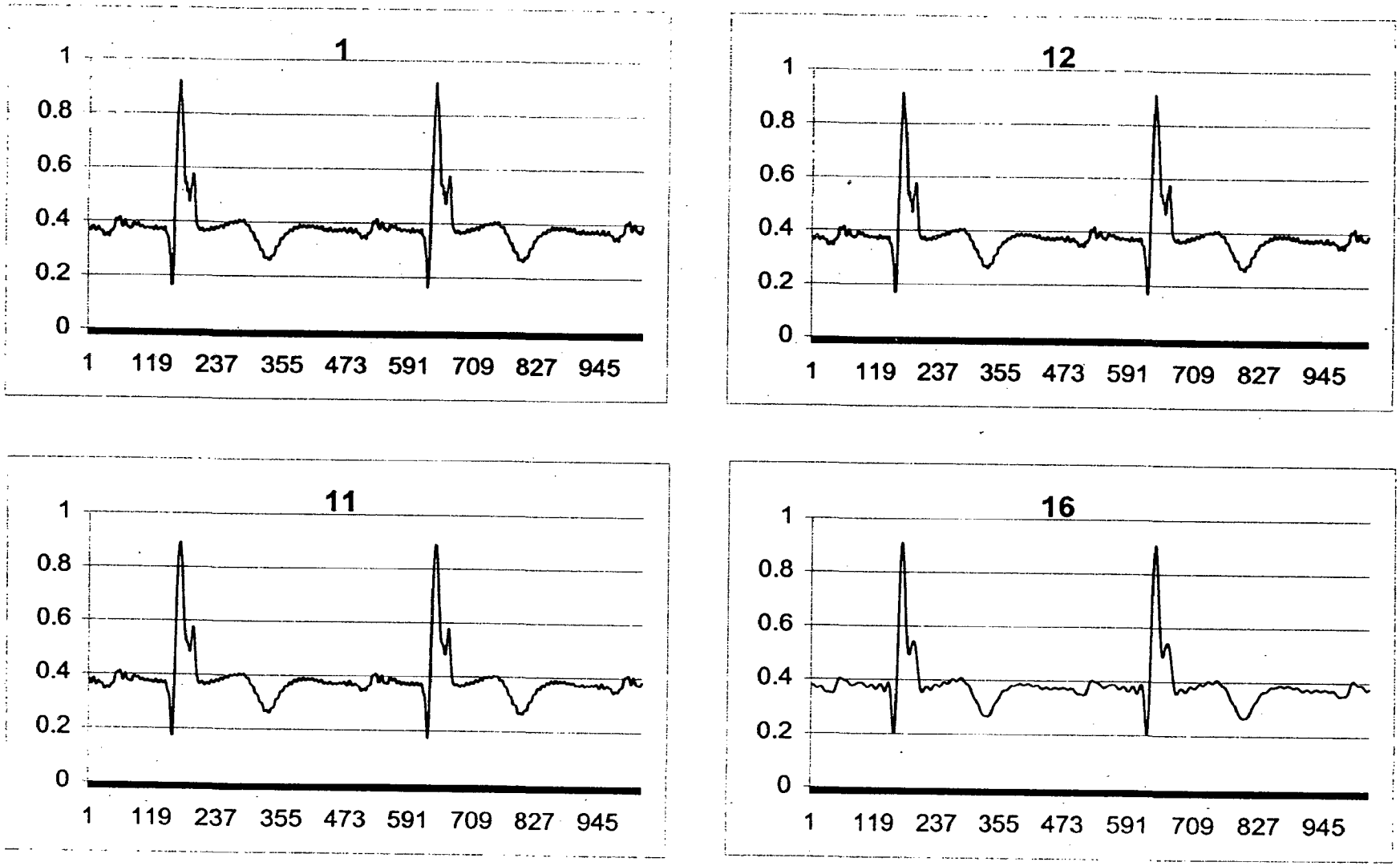
FOURIER ANALYSIS OF 1Y ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

Fig. 4.16 FOURIER ANALYSIS OF 1Y ECG DATA

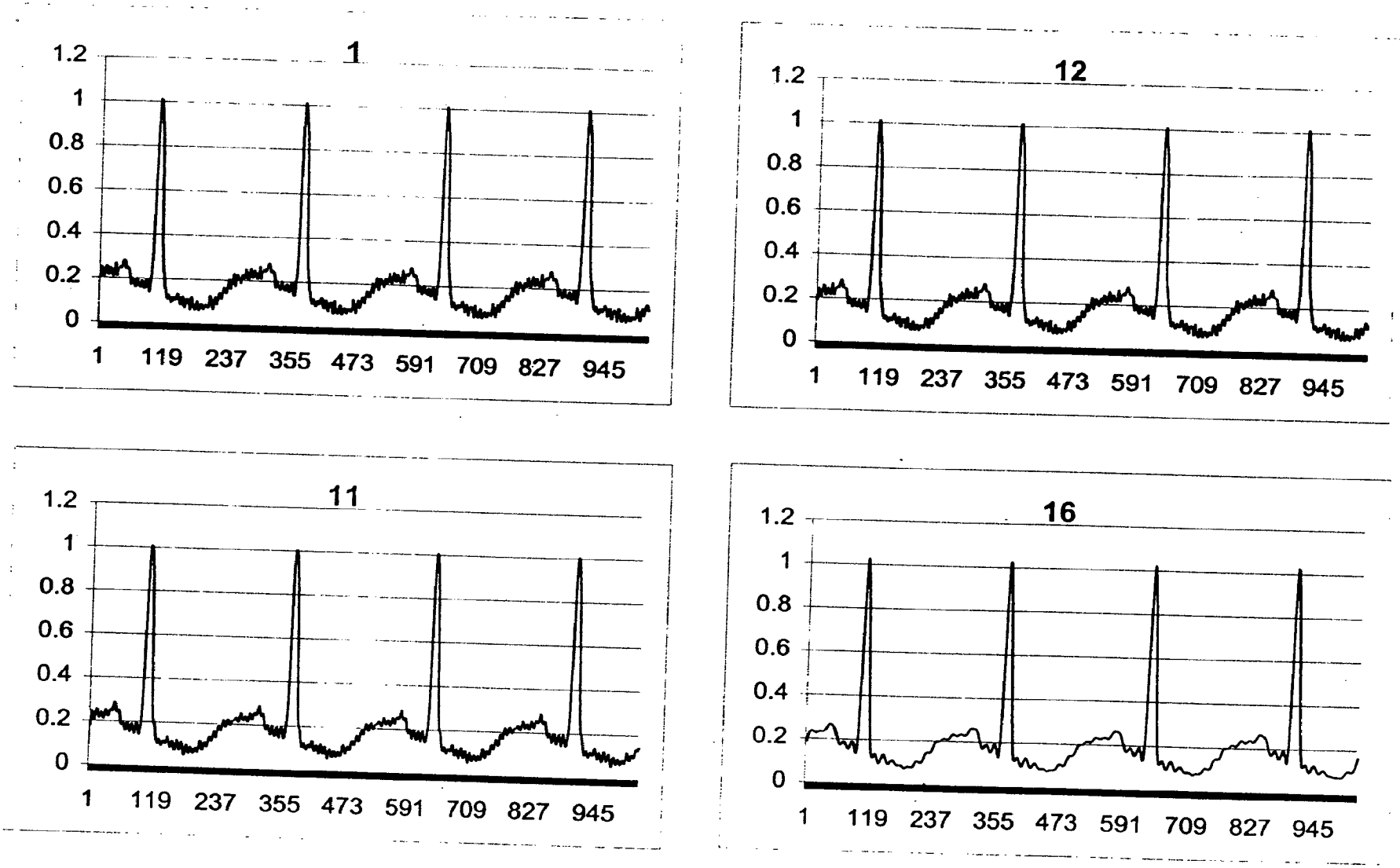
FOURIER ANALYSIS OF 1Z ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

Fig. 4.16 FOURIER ANALYSIS OF 1Z ECG DATA

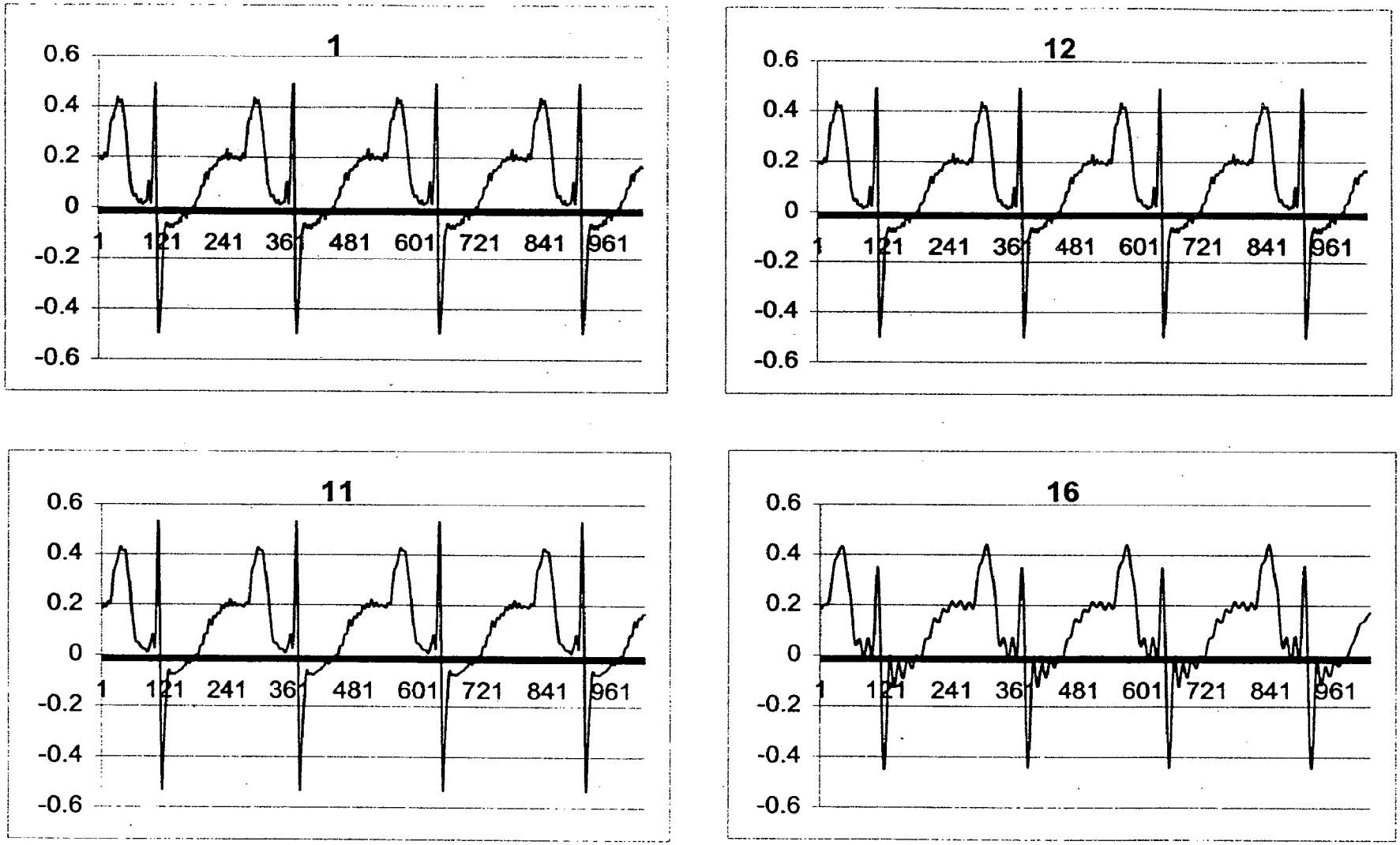
FOURIER ANALYSIS OF 2L1 ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

Fig. 4.18 FOURIER ANALYSIS OF 2L1 ECG DATA

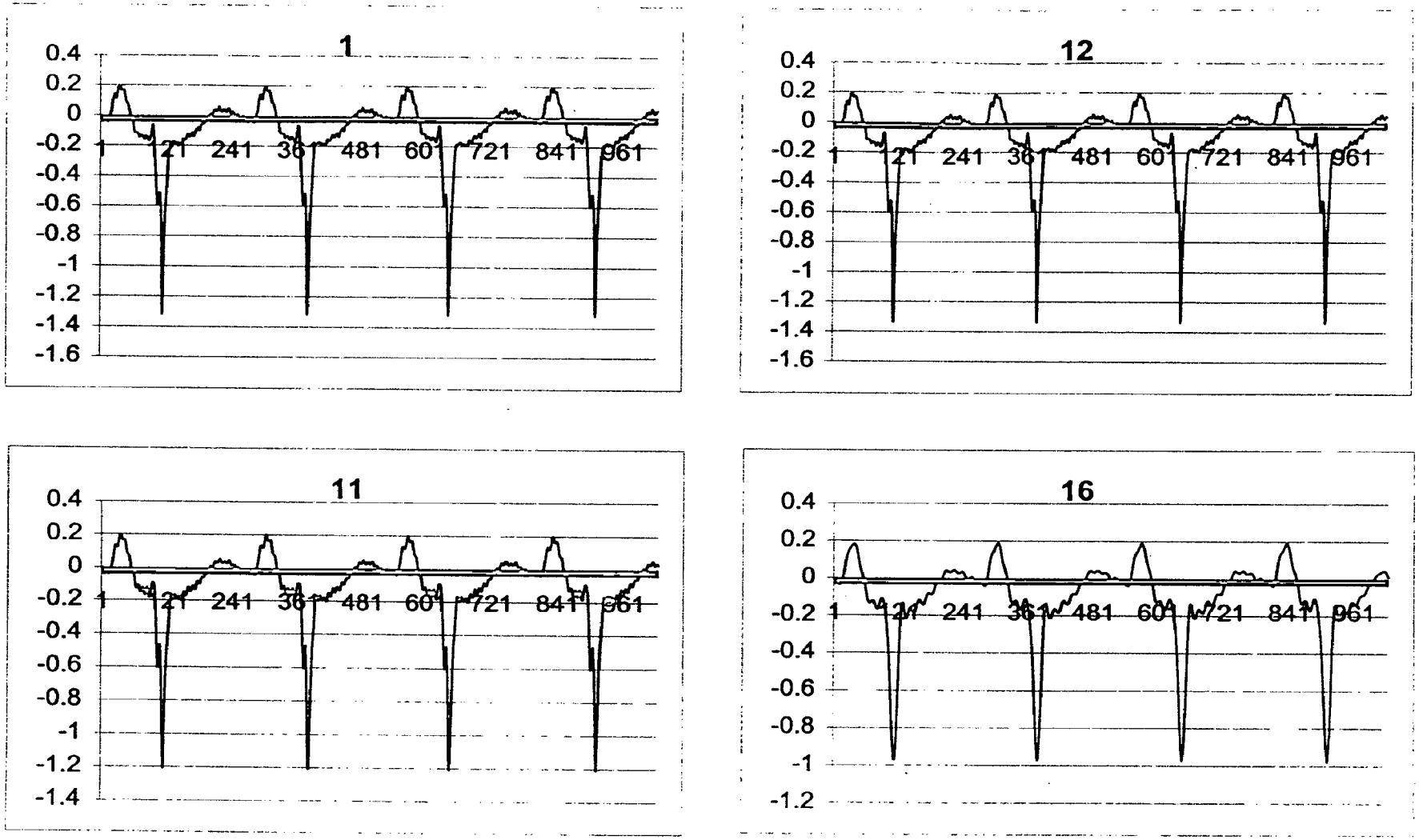
FOURIER ANALYSIS OF 2L2 ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

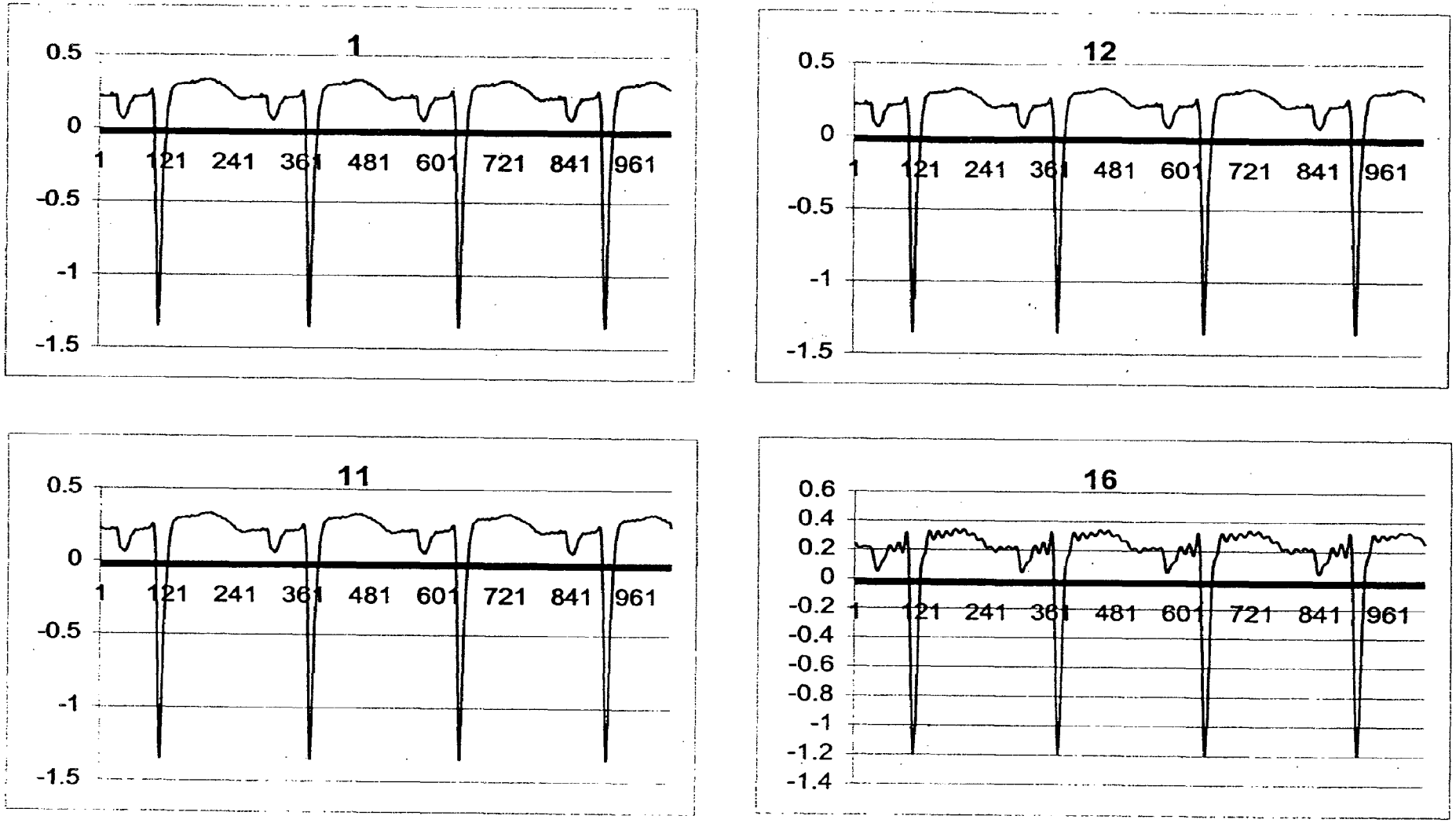
Fig. 4.19 FOURIER ANALYSIS OF 2L2 ECG DATA

FOURIER ANALYSIS OF 2L3 ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

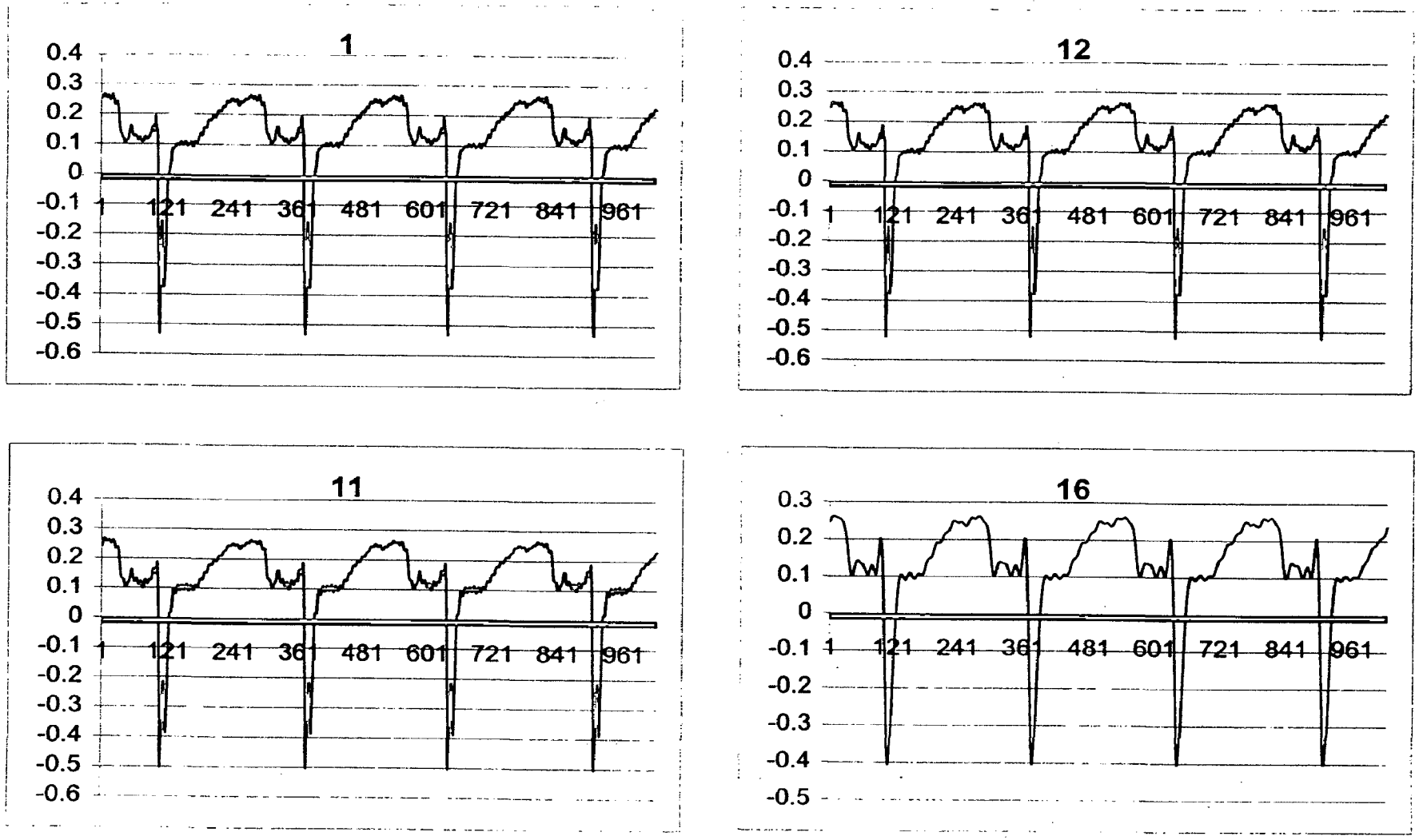
Fig. 4.20 FOURIER ANALYSIS OF 2L3 ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

Fig. 4.21 FOURIER ANALYSIS OF 2V1 ECG DATA

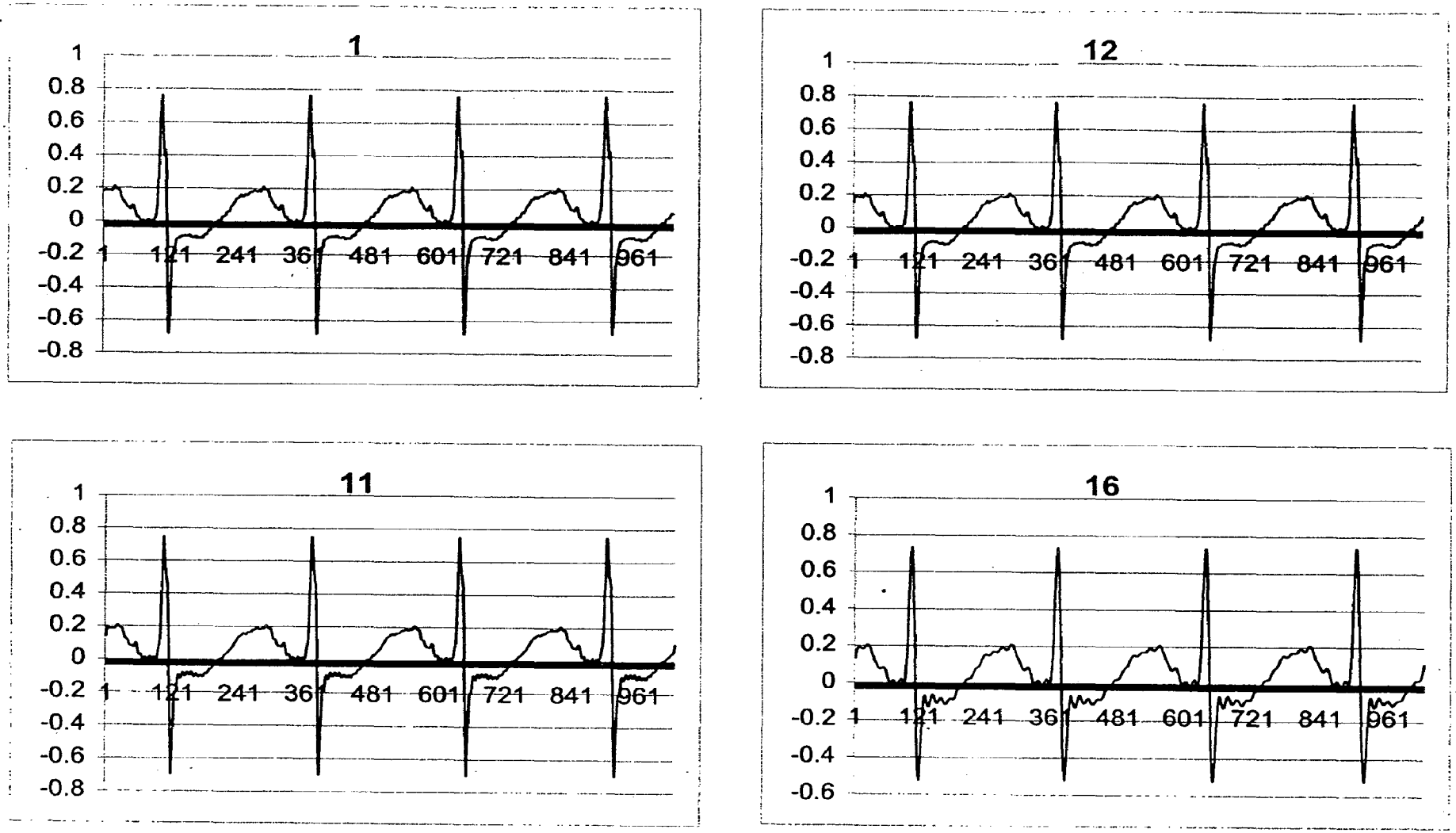
FOURIER ANALYSIS OF 2V2 ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

Fig. 4.22 FOURIER ANALYSIS OF 2V2 ECG DATA

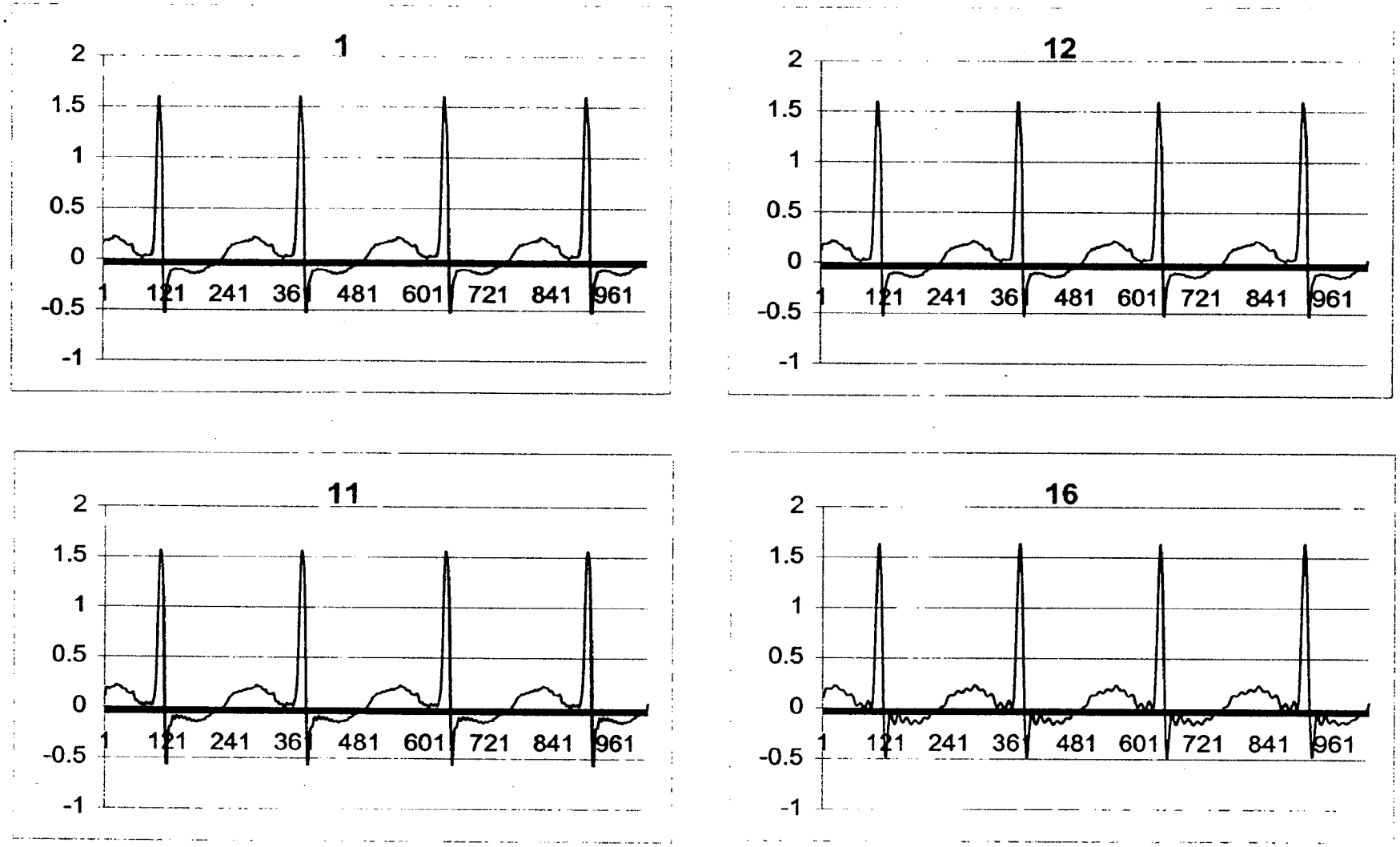
FOURIER ANALYSIS OF 2V3 ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

Fig. 4.23 FOURIER ANALYSIS OF 2V3 ECG DATA

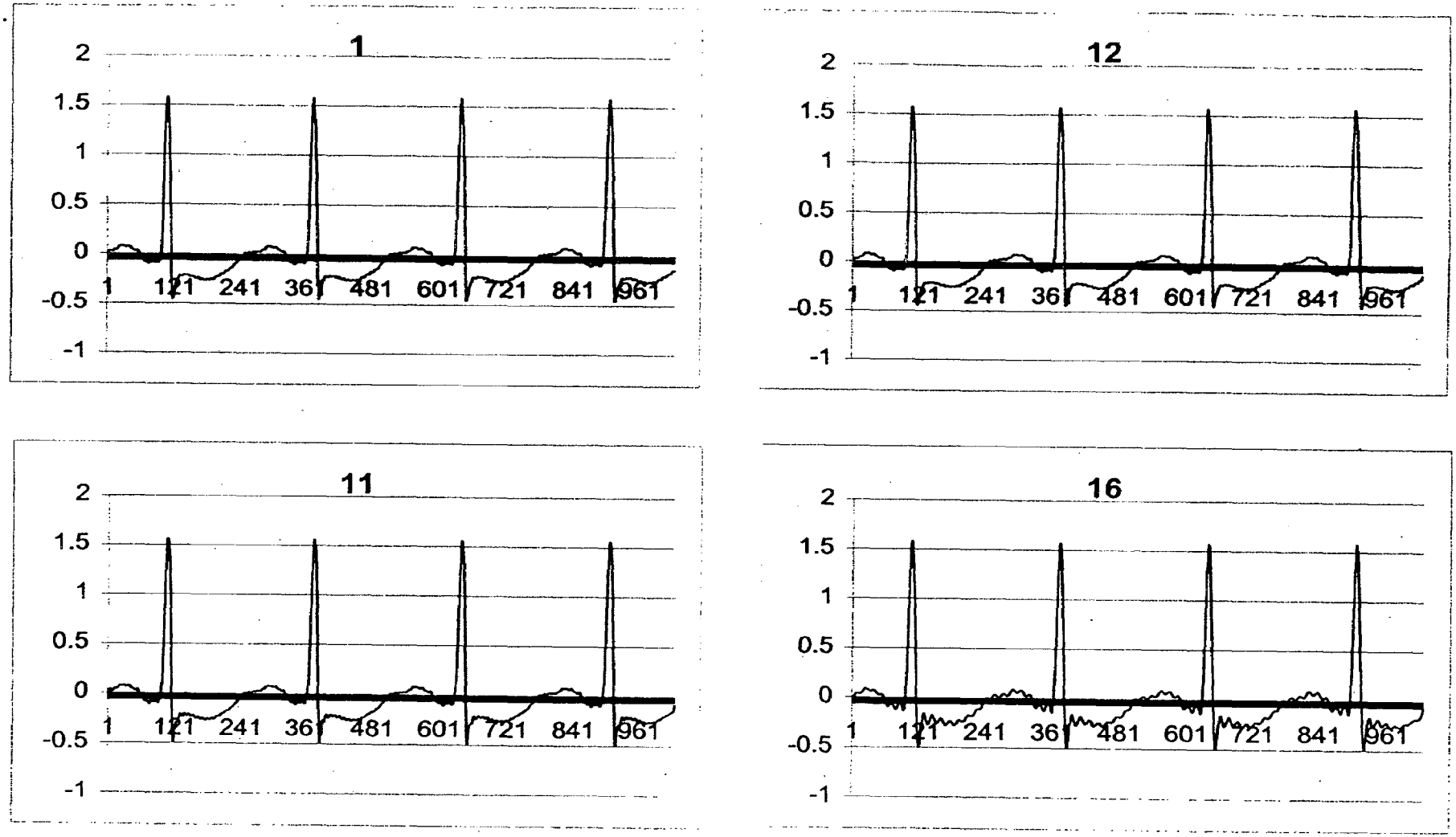
FOURIER ANALYSIS OF 2V4 ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

Fig. 4.24 FOURIER ANALYSIS OF 2V4 ECG DATA

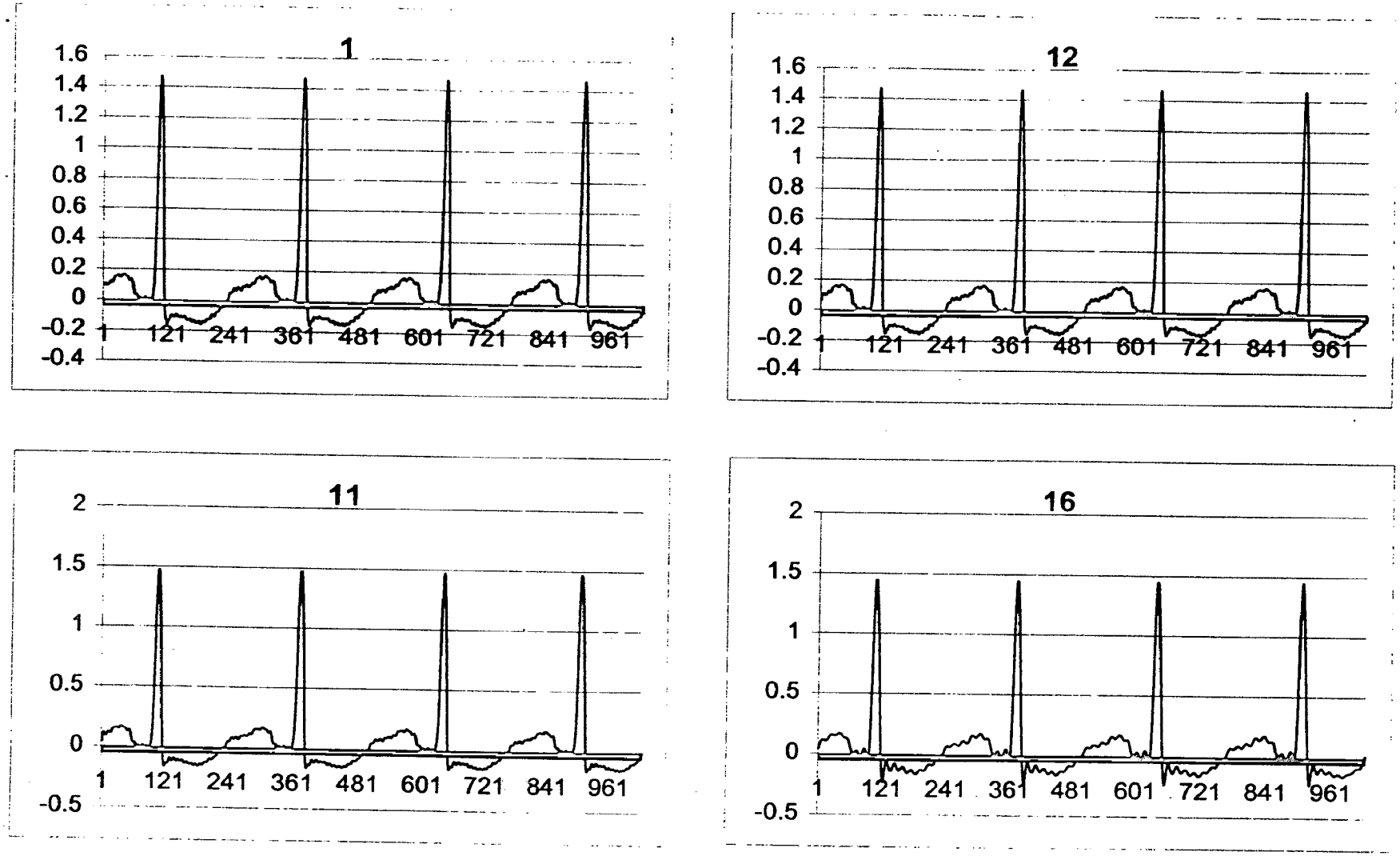
FOURIER ANALYSIS OF 2V5 ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

Fig. 4.25 FOURIER ANALYSIS OF 2V5 ECG DATA

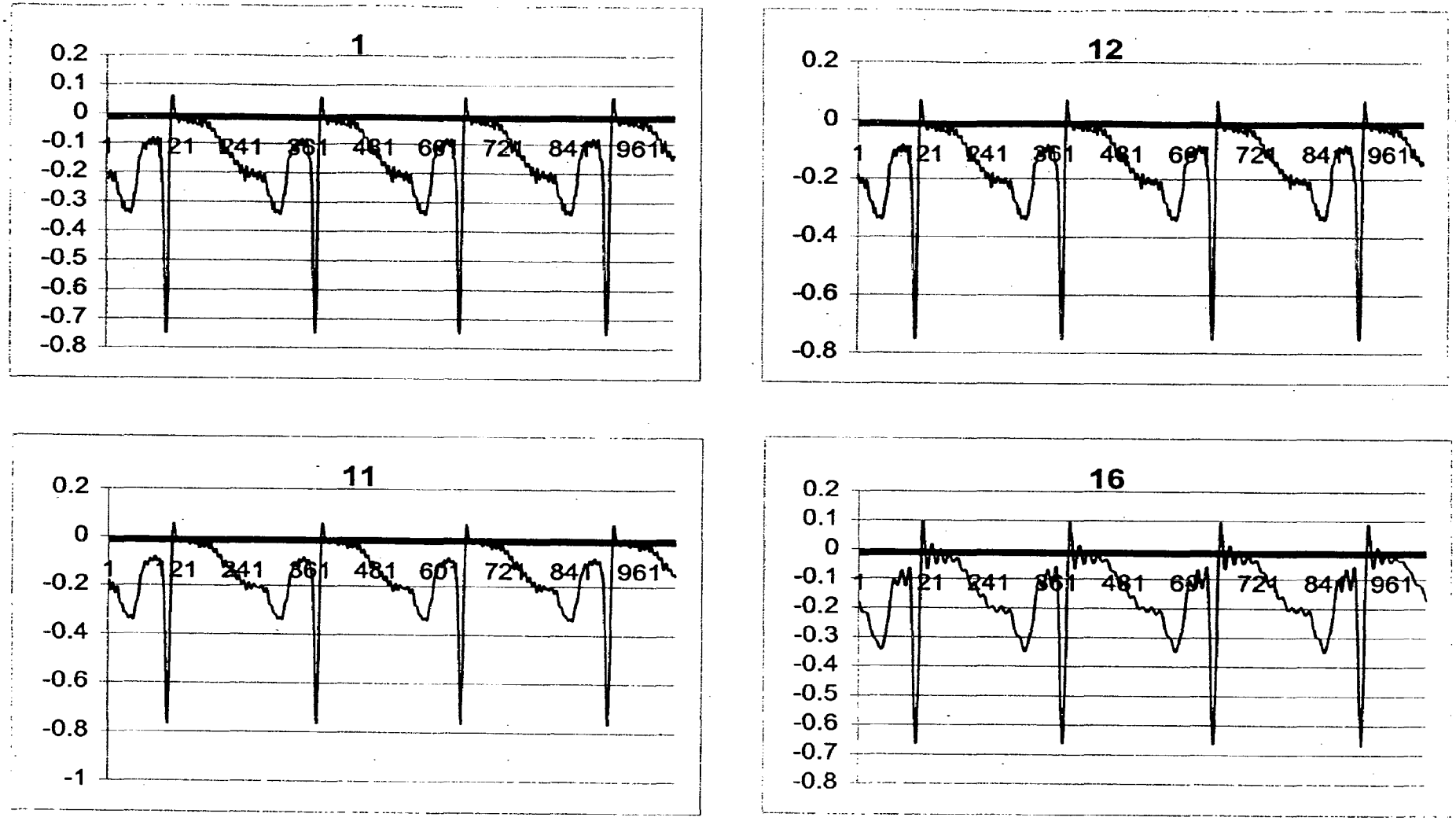
FOURIER ANALYSIS OF 2V6 ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

Fig. 4.26 FOURIER ANALYSIS OF 2V6 ECG DATA

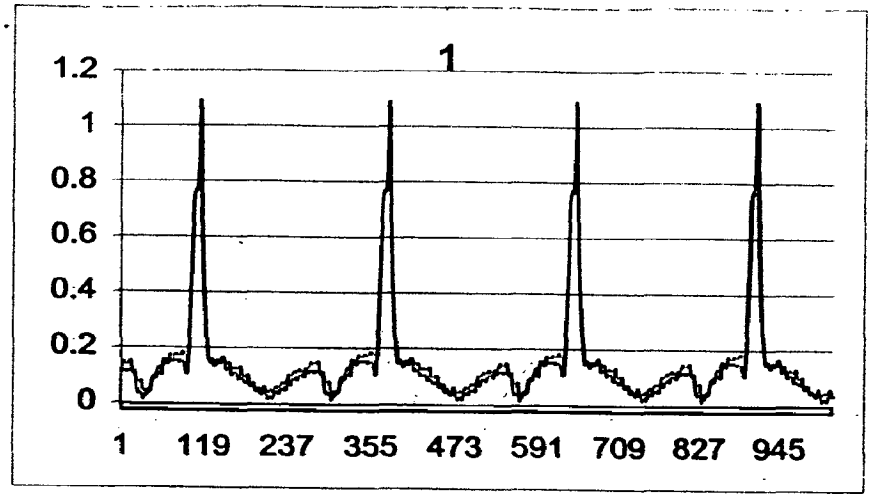
FOURIER ANALYSIS OF 2AR ECG DATA

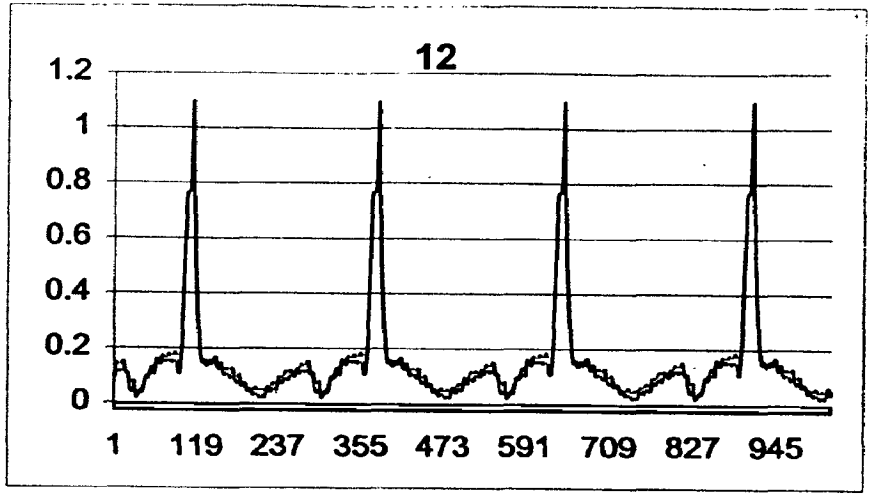


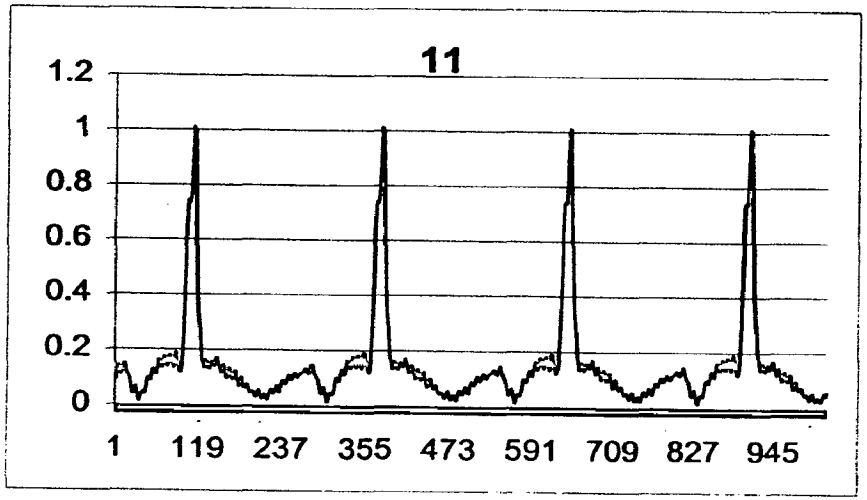
'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

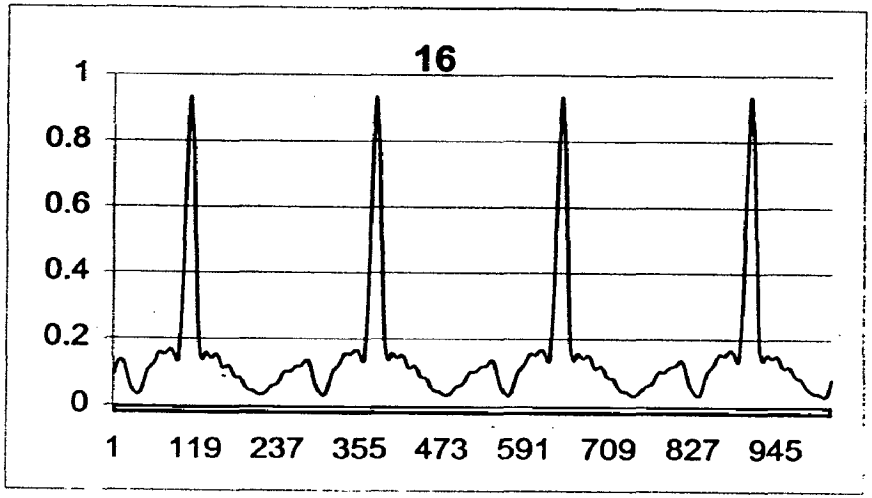
Fig. 4.27 FOURIER ANALYSIS OF 2AR ECG DATA

FOURIER ANALYSIS OF 2AL ECG DATA



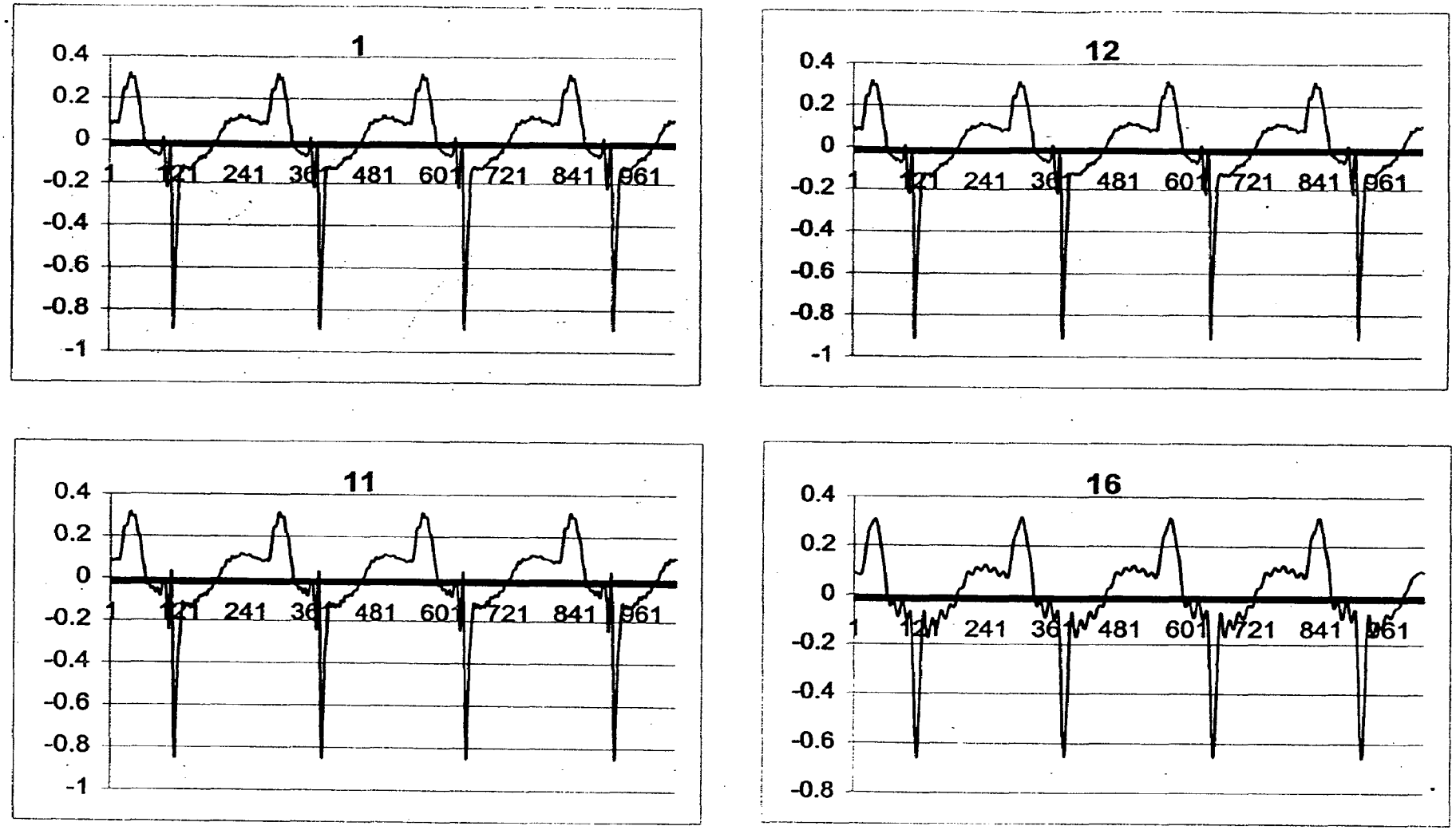






'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

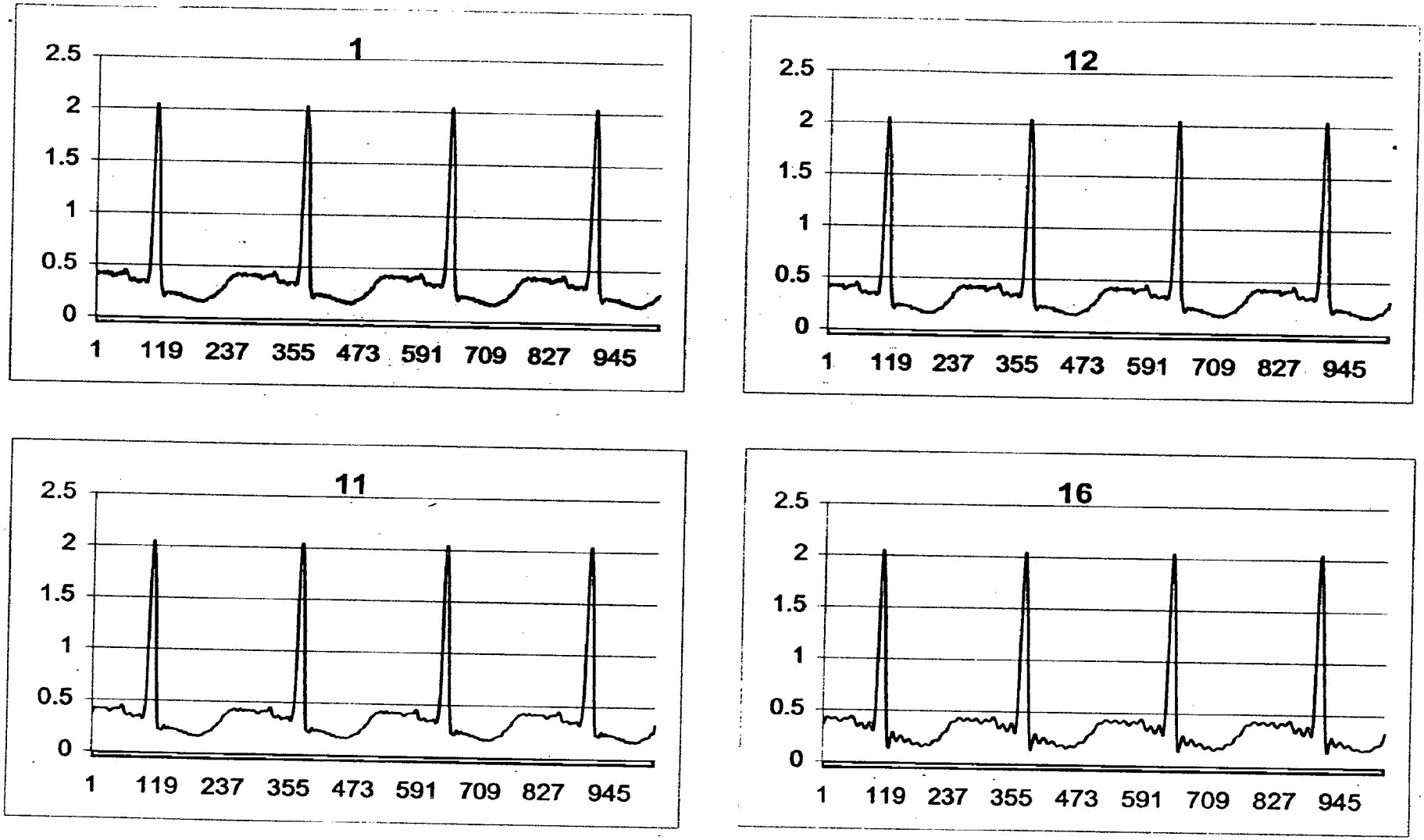
FOURIER ANALYSIS OF 2AF ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

Fig. 4.29 FOURIER ANALYSIS OF 2AF ECG DATA

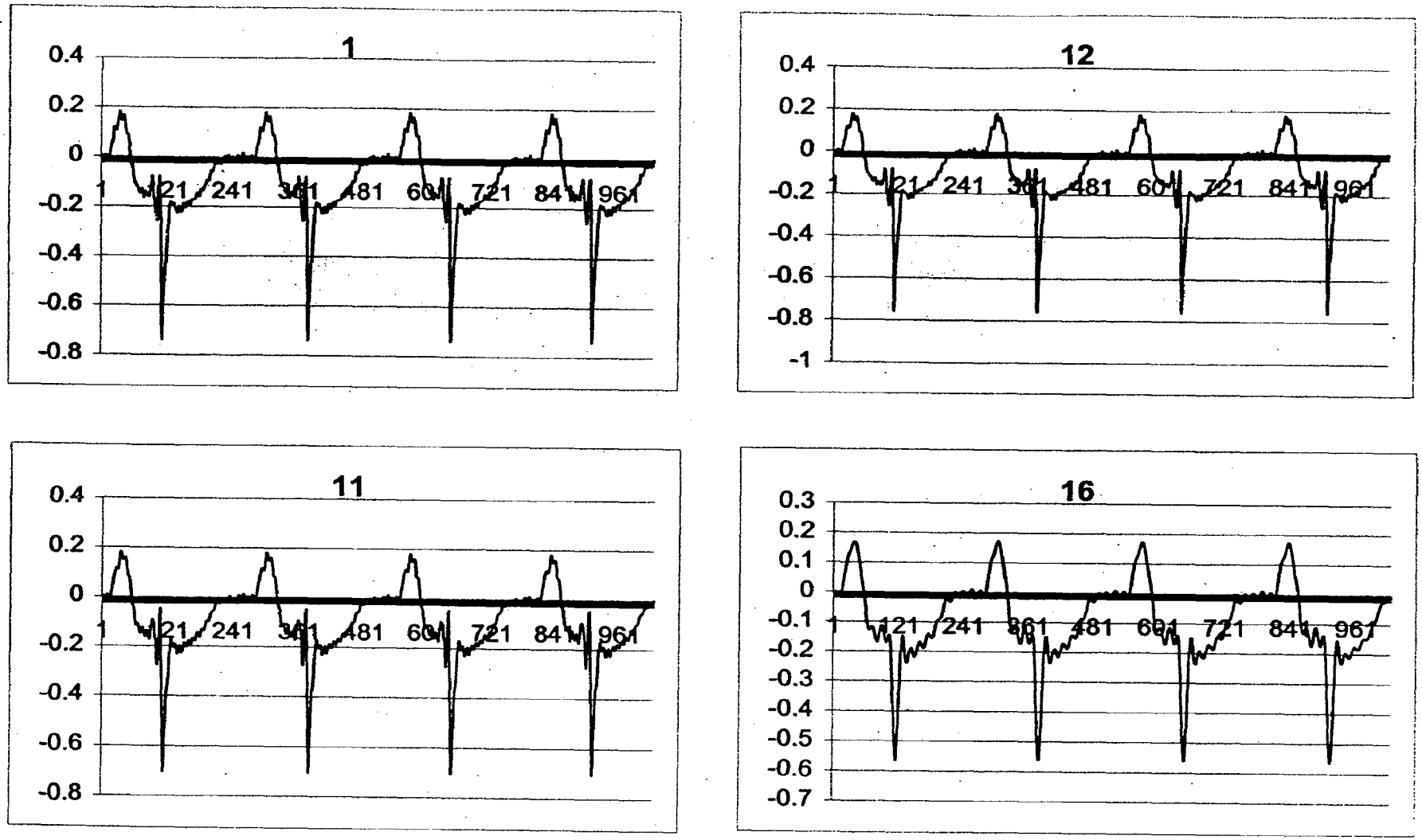
FOURIER ANALYSIS OF 2X ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

Fig. 4.30 FOURIER ANALYSIS OF 2X ECG DATA

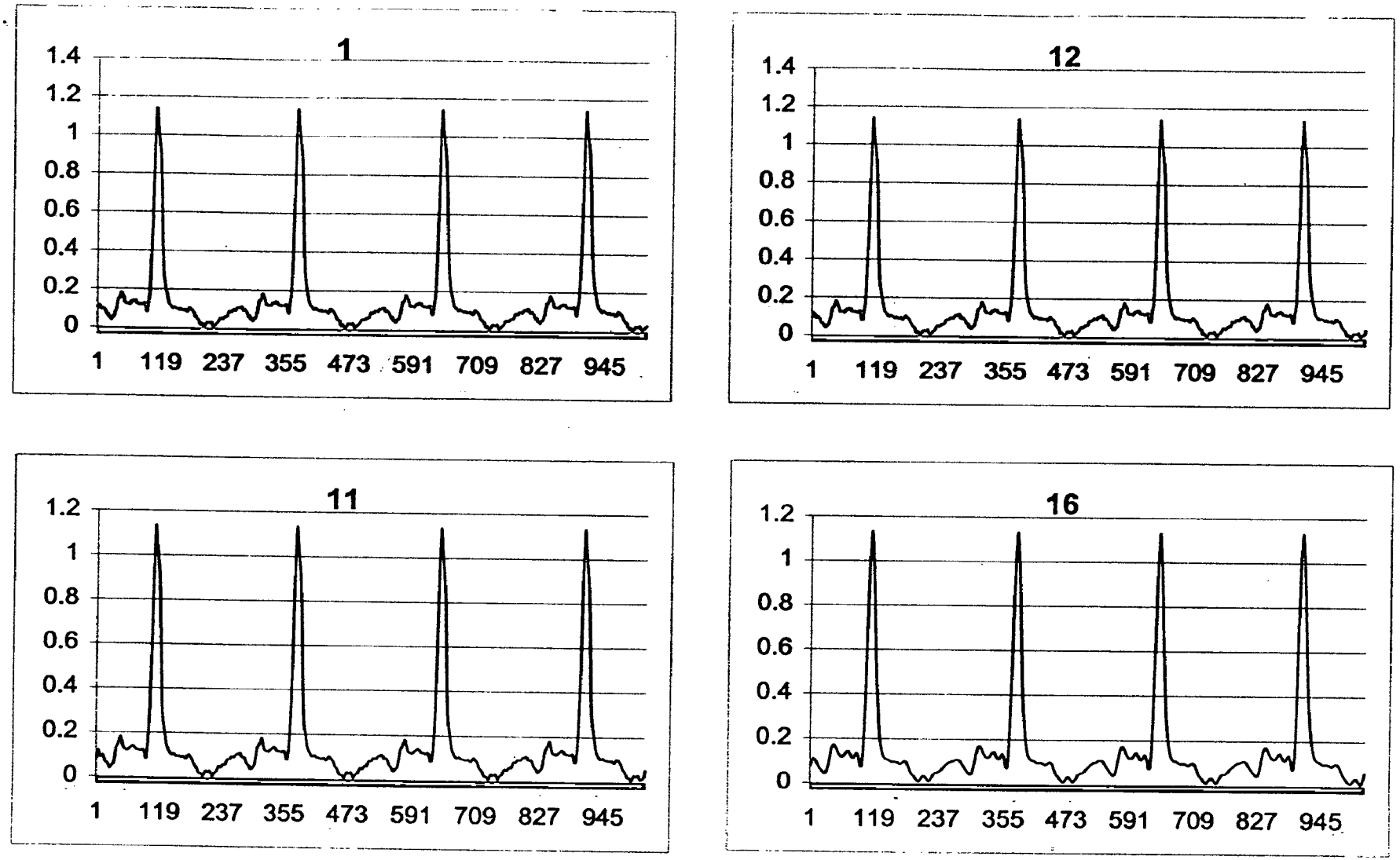
FOURIER ANALYSIS OF 2Y ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

Fig. 4.31 FOURIER ANALYSIS OF 2Y ECG DATA

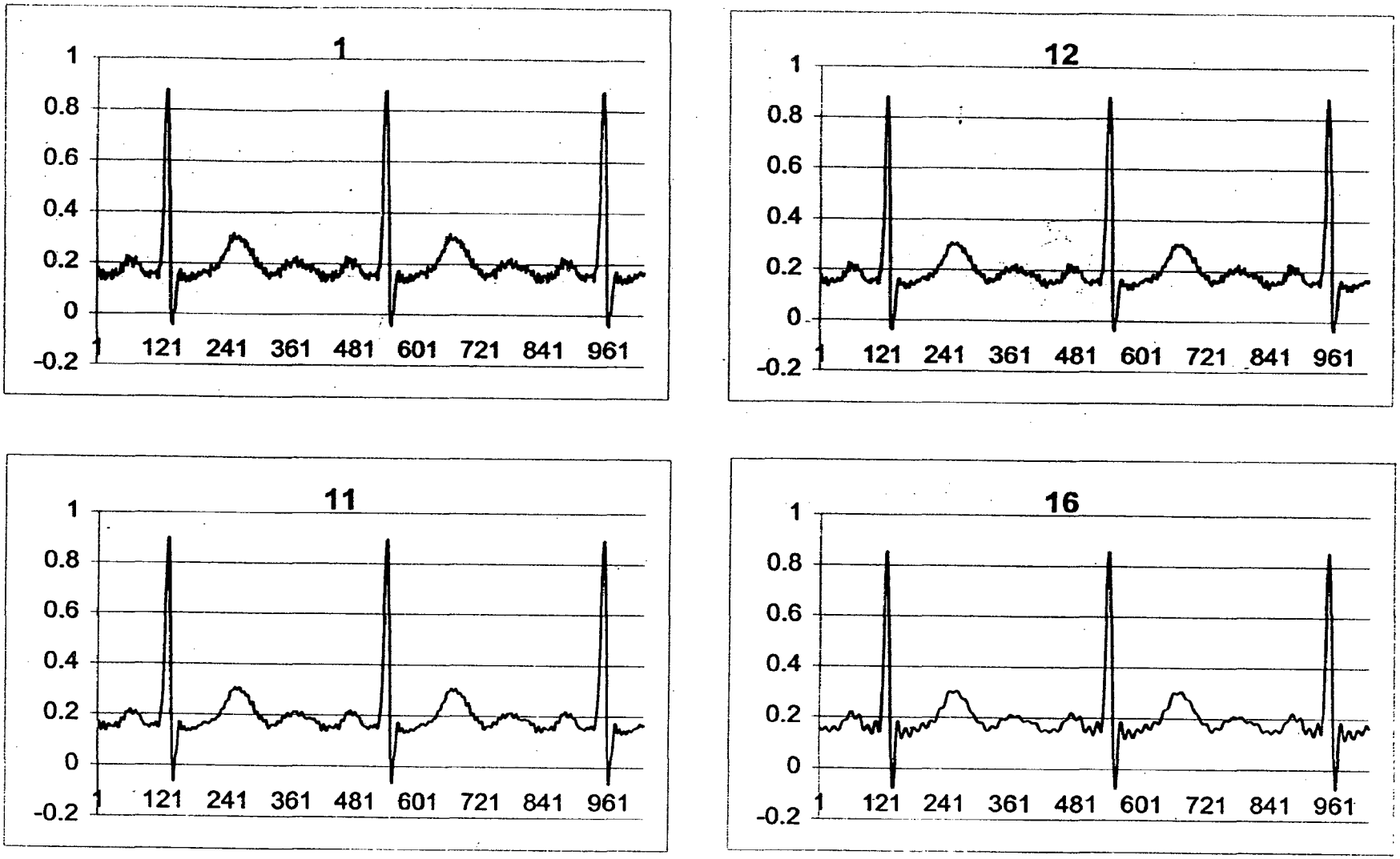
FOURIER ANALYSIS OF 2Z ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

Fig. 4.32 FOURIER ANALYSIS OF 2Z ECG DATA

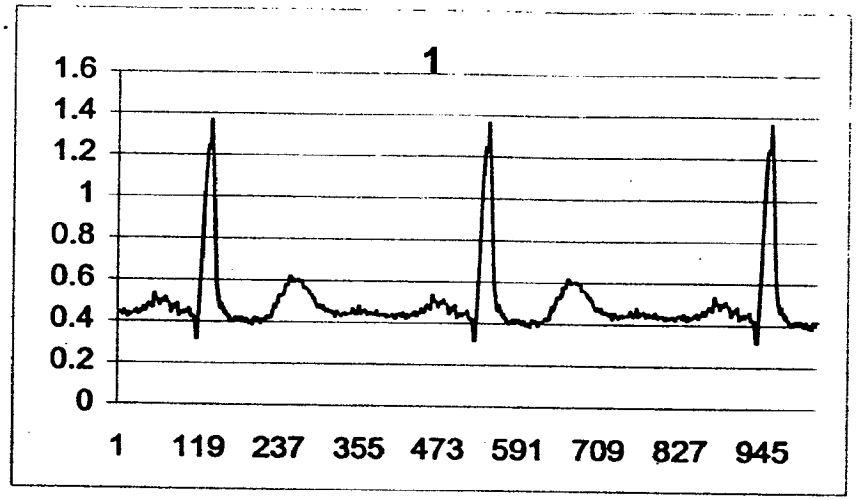
FOURIER ANALYSIS OF 4L1 ECG DATA

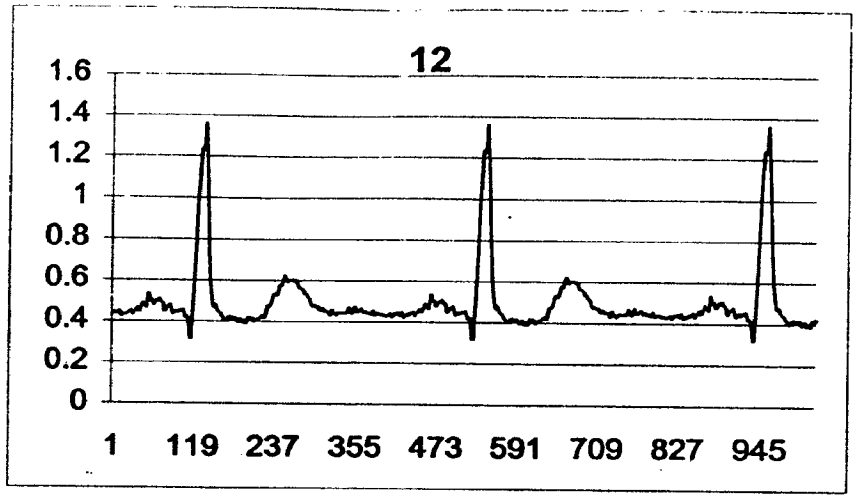


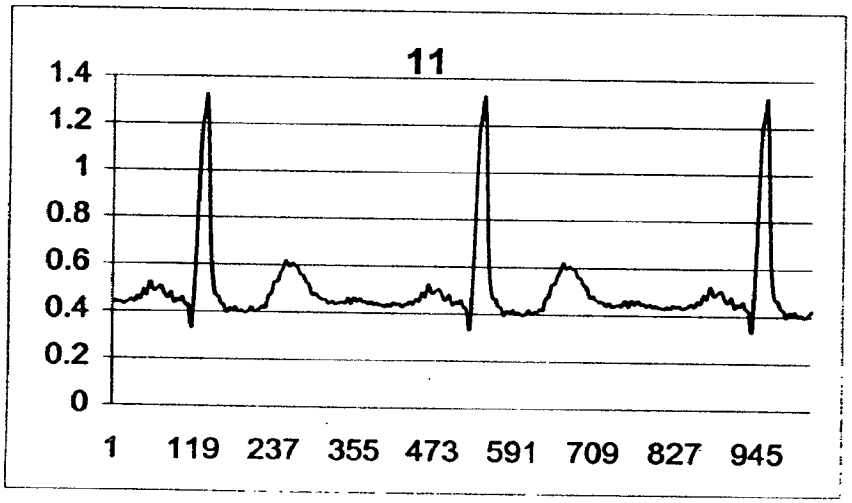
'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

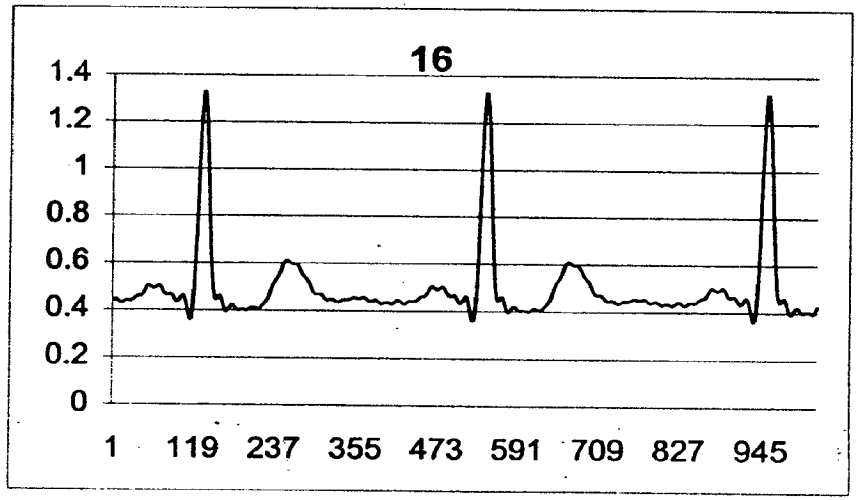
Fig. 4.33 FOURIER ANALYSIS OF 4L1 ECG DATA

FOURIER ANALYSIS OF 4L2 ECG DATA





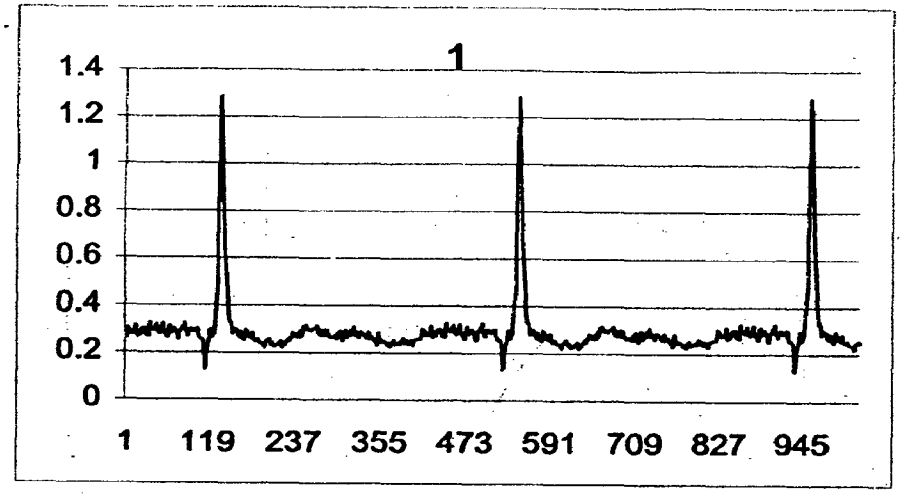


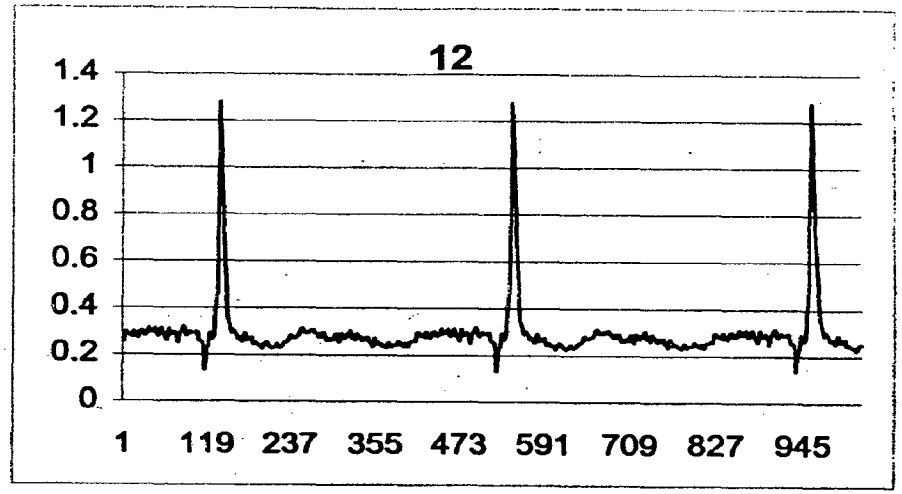


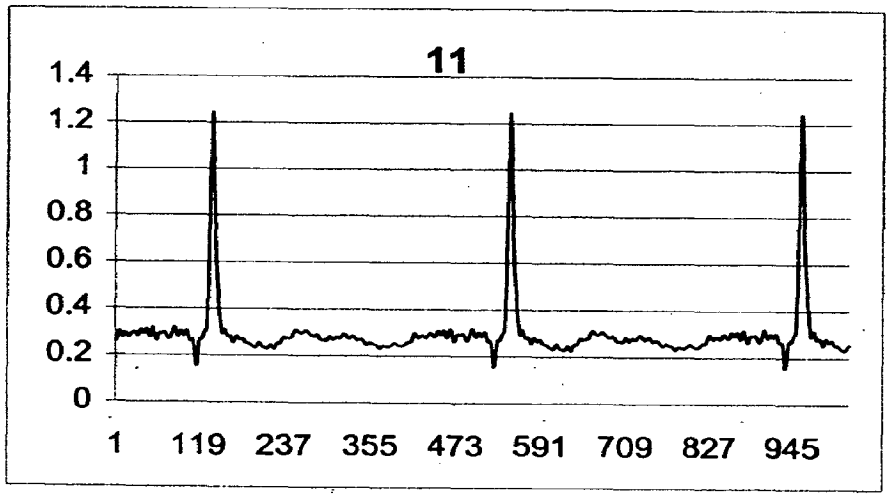
'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

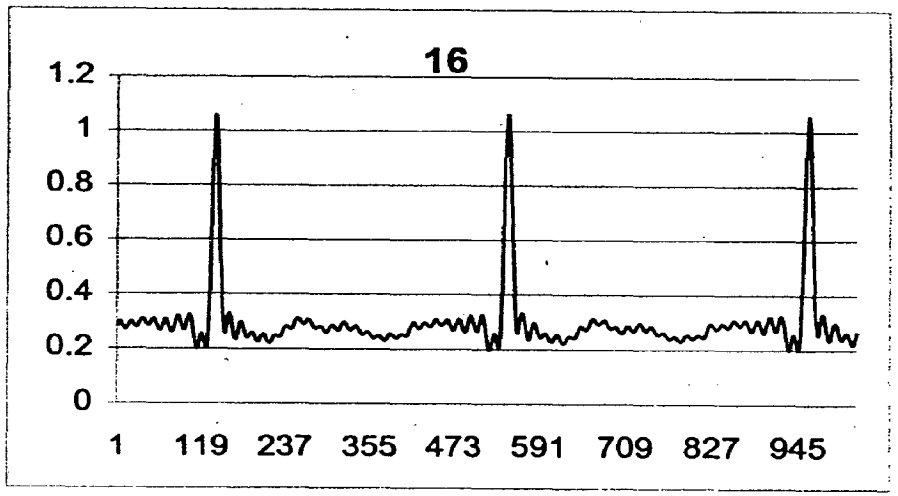
Fig. 4.34 FOURIER ANALYSIS OF 4L2 ECG DATA

FOURIER ANALYSIS OF 4L3 ECG DATA





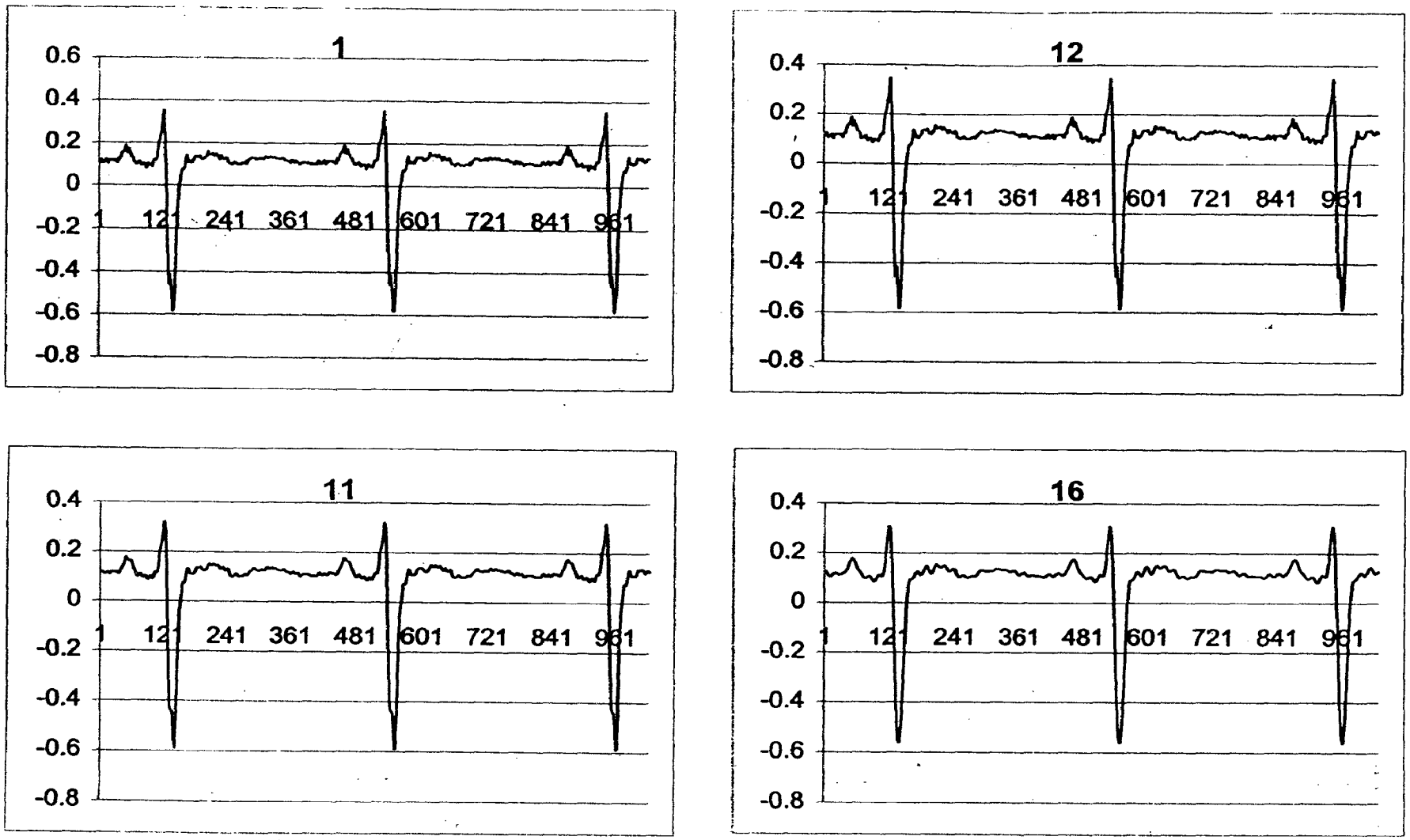




'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

Fig. 4.35 FOURIER ANALYSIS OF 4L3 ECG DATA

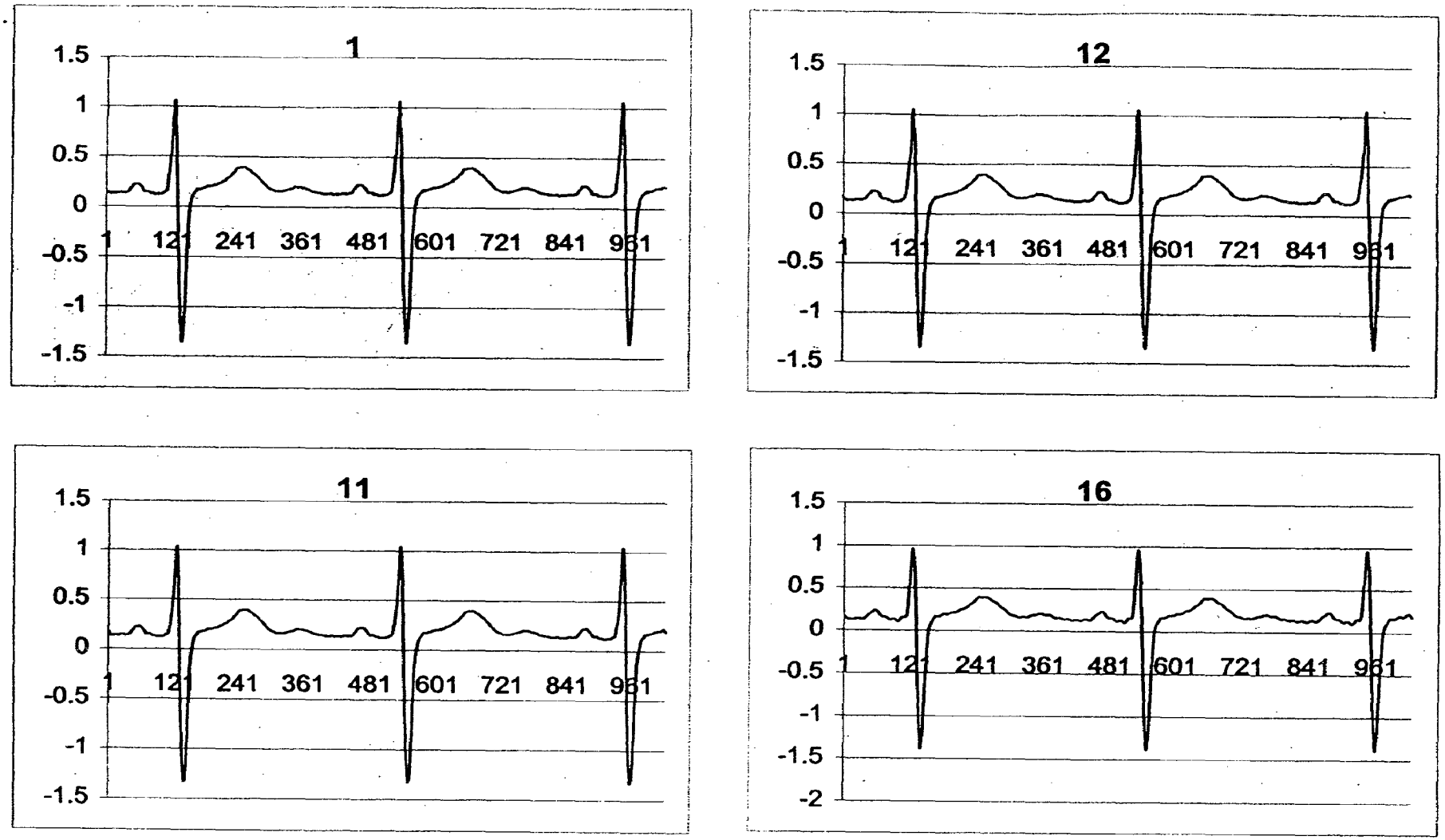
FOURIER ANALYSIS OF 4V1 ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

Fig. 4.36 FOURIER ANALYSIS OF 4V1 ECG DATA

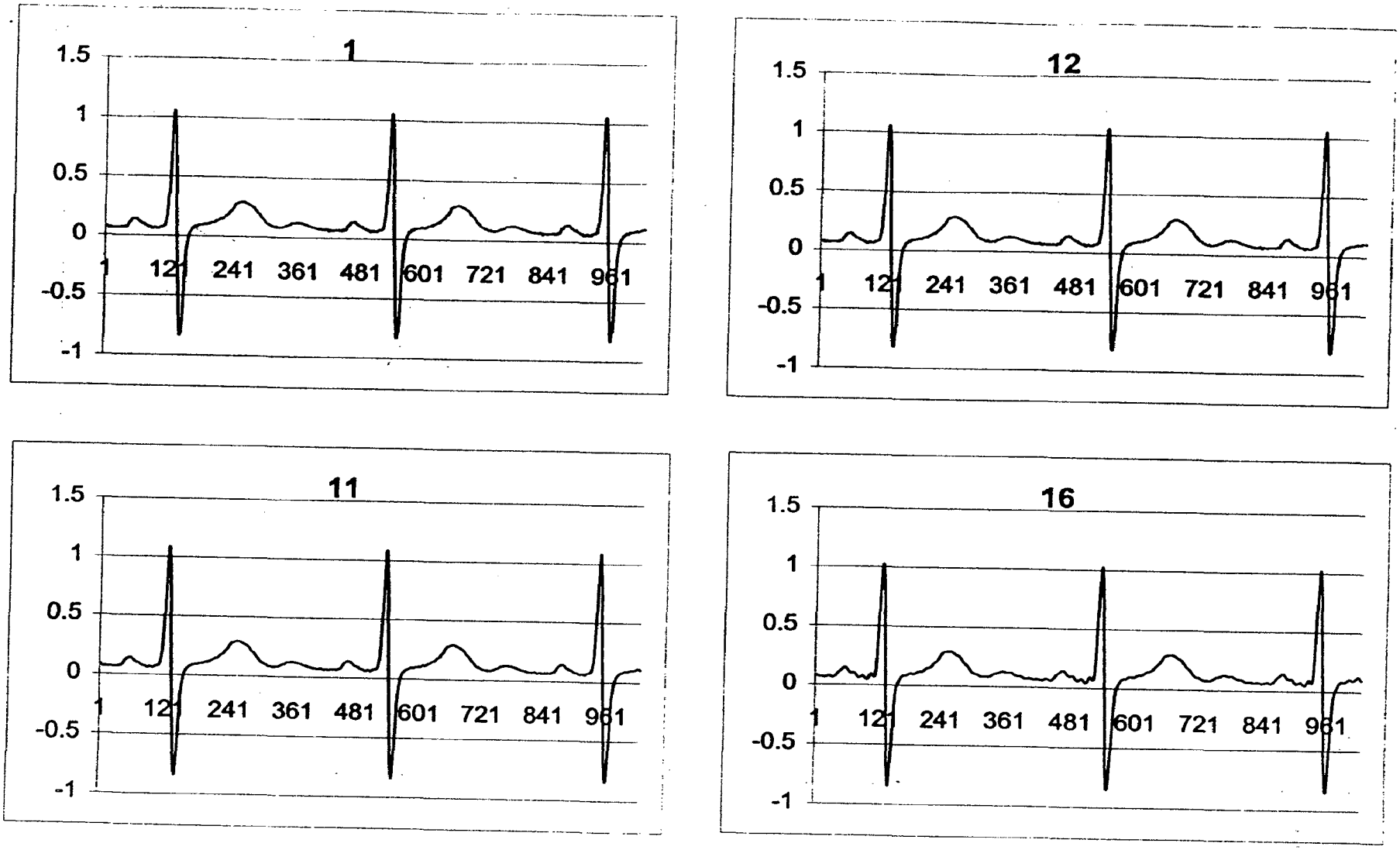
FOURIER ANALYSIS OF 4V2 ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

Fig. 4.37 FOURIER ANALYSIS OF 4V2 ECG DATA

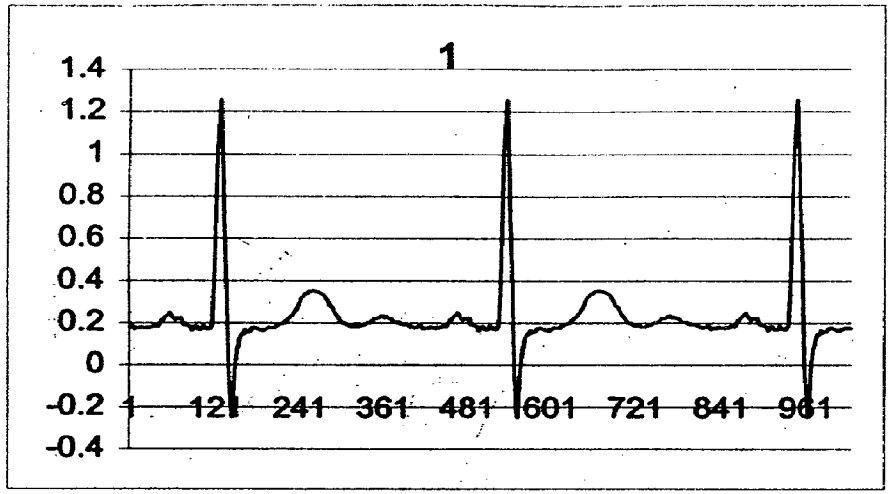
FOURIER ANALYSIS OF 4V3 ECG DATA

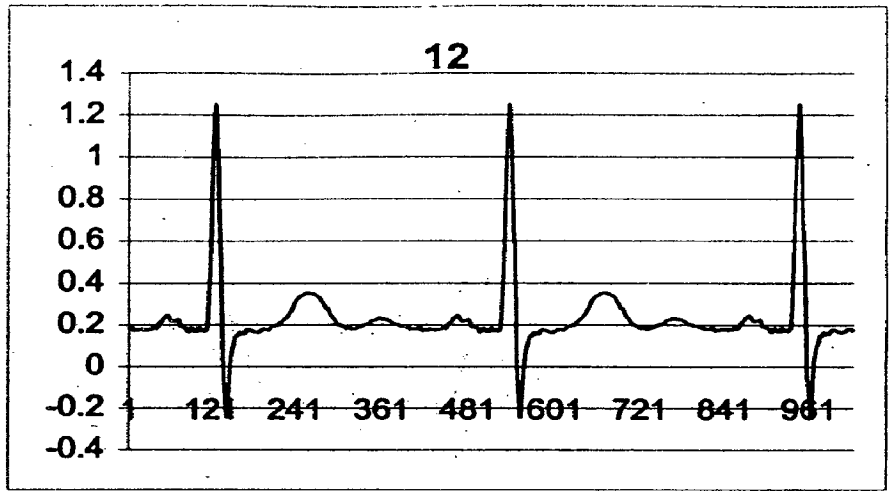


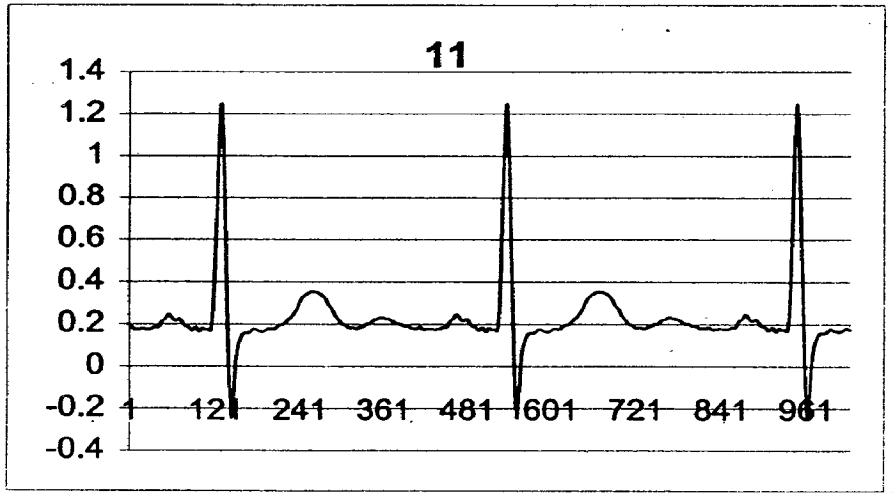
'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

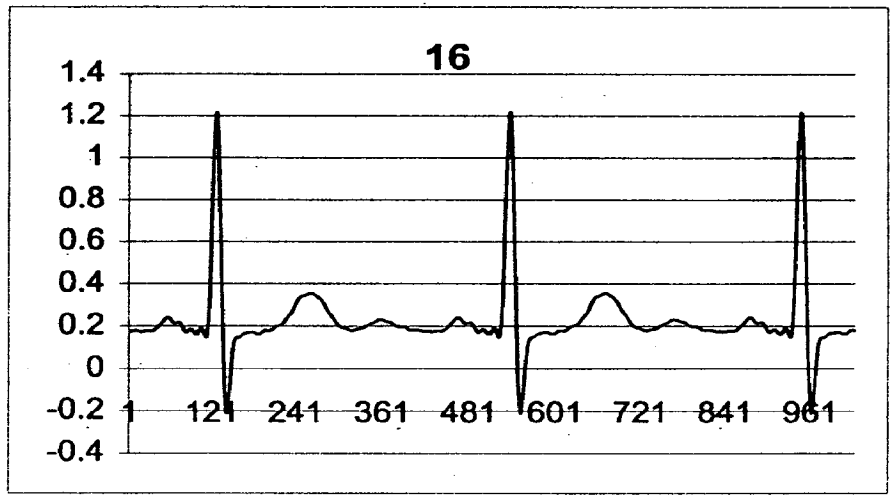
Fig. 4.38 FOURIER ANALYSIS OF 4V3 ECG DATA

FOURIER ANALYSIS OF 4V4 ECG DATA





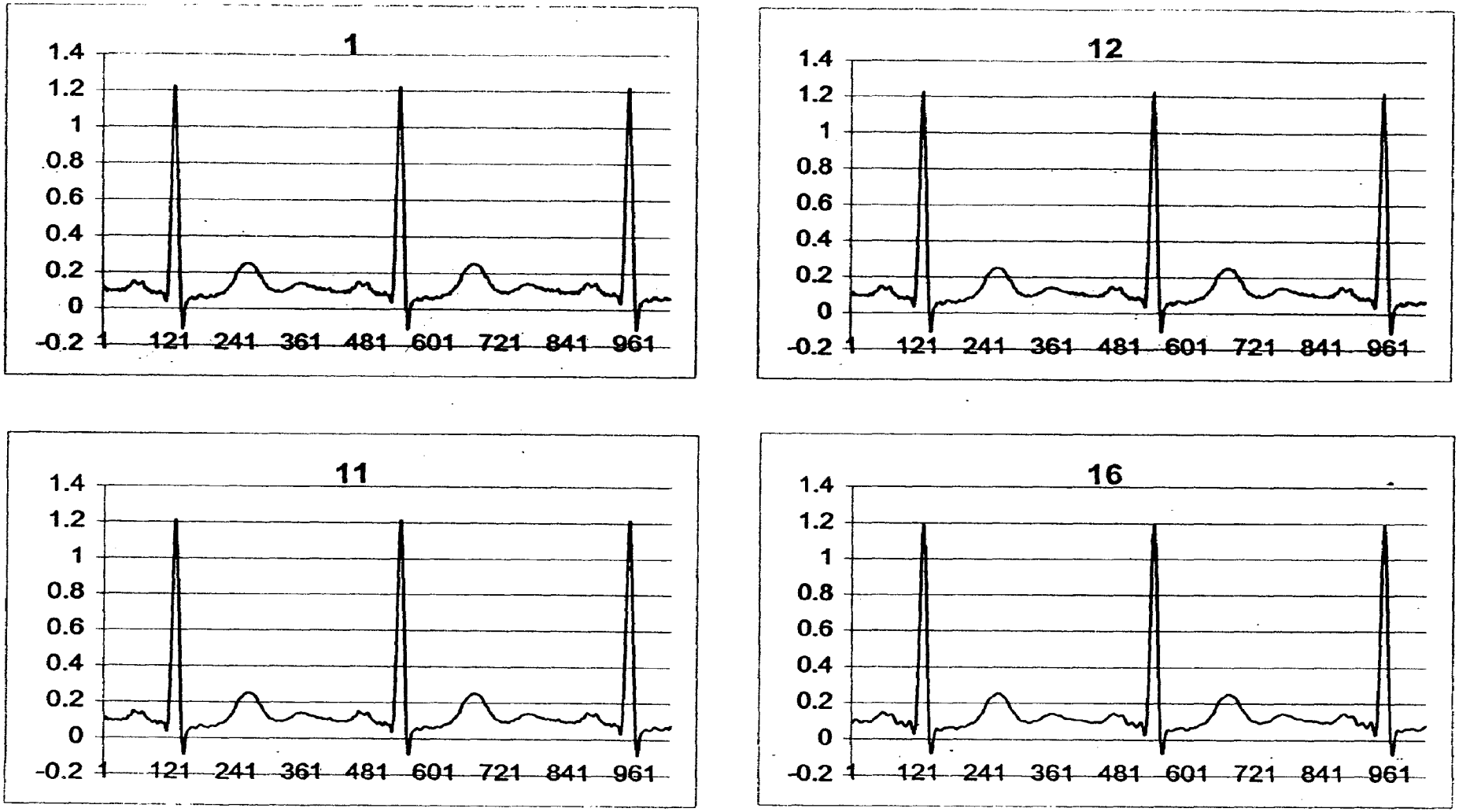




'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

Fig. 4.39 FOURIER ANALYSIS OF 4V4 ECG DATA

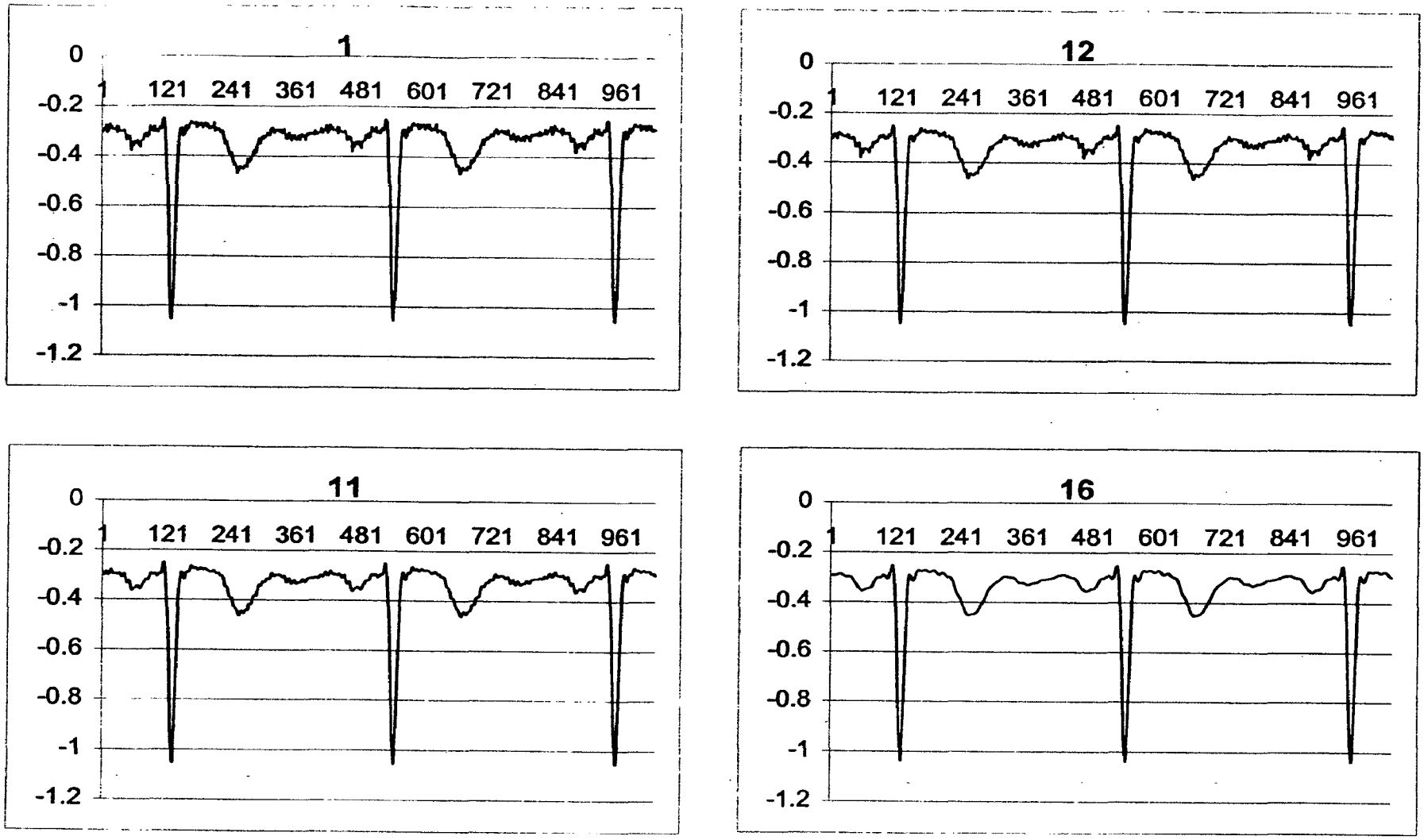
FOURIER ANALYSIS OF 4V5 ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

Fig. 4.40 FOURIER ANALYSIS OF 4V5 ECG DATA

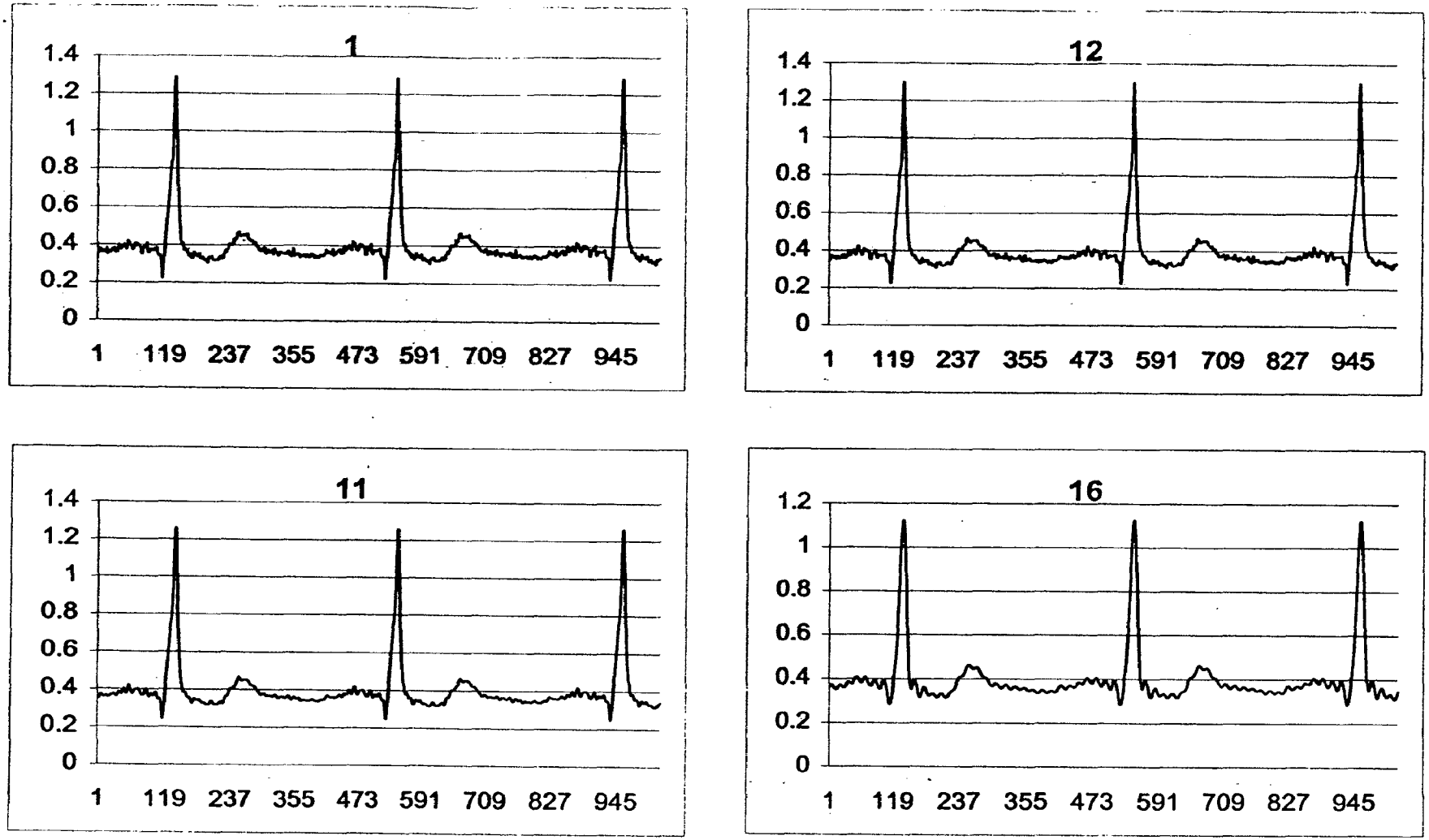
FOURIER ANALYSIS OF 4AR ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

Fig. 4.42 FOURIER ANALYSIS OF 4AR ECG DATA

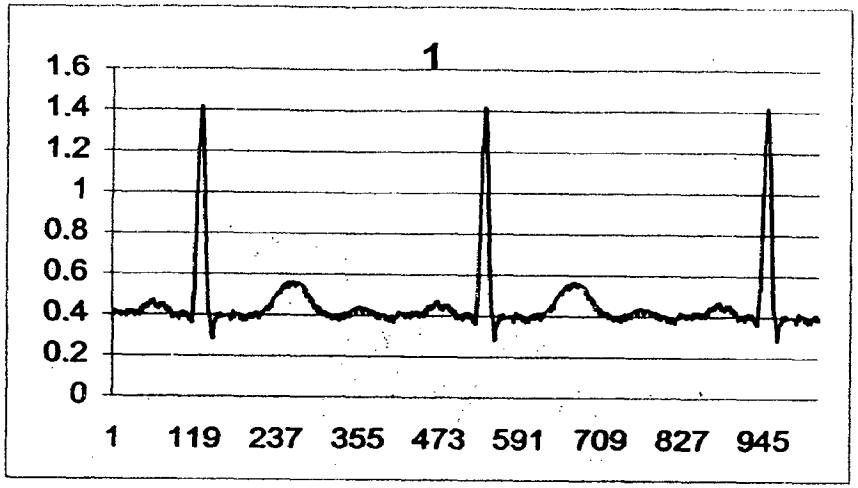
FOURIER ANALYSIS OF 4AF ECG DATA

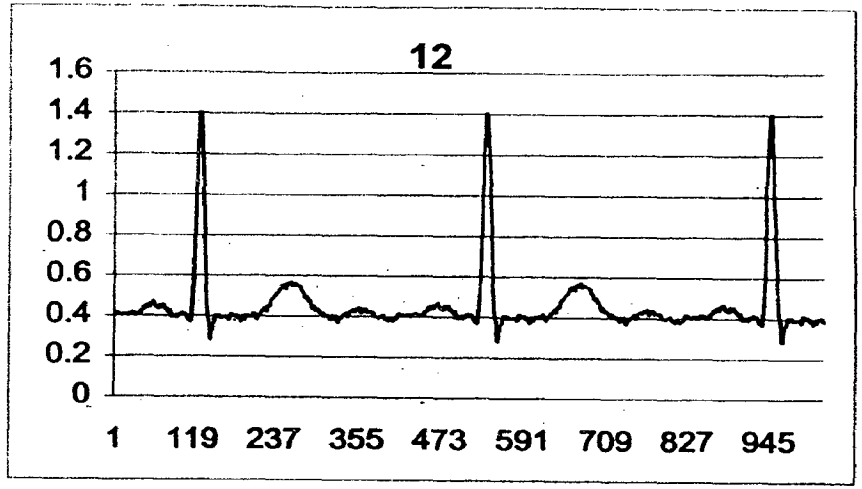


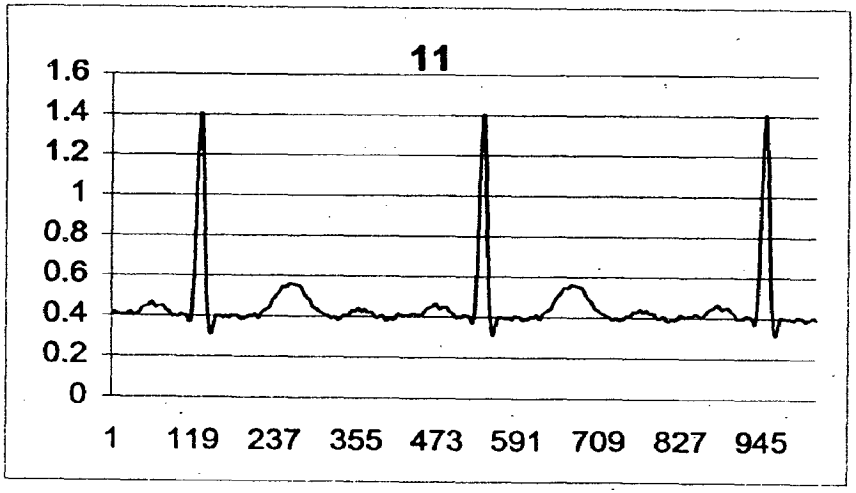
'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

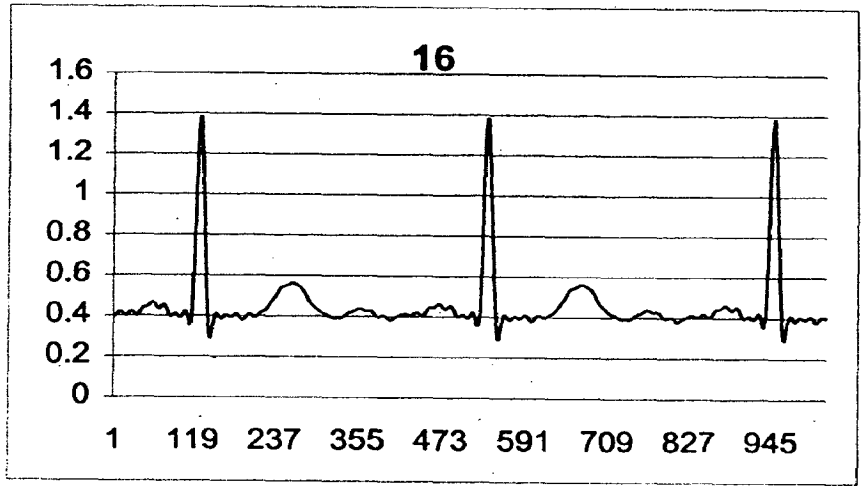
Fig. 4.44 FOURIER ANALYSIS OF 4AF ECG DATA

FOURIER ANALYSIS OF 4X ECG DATA





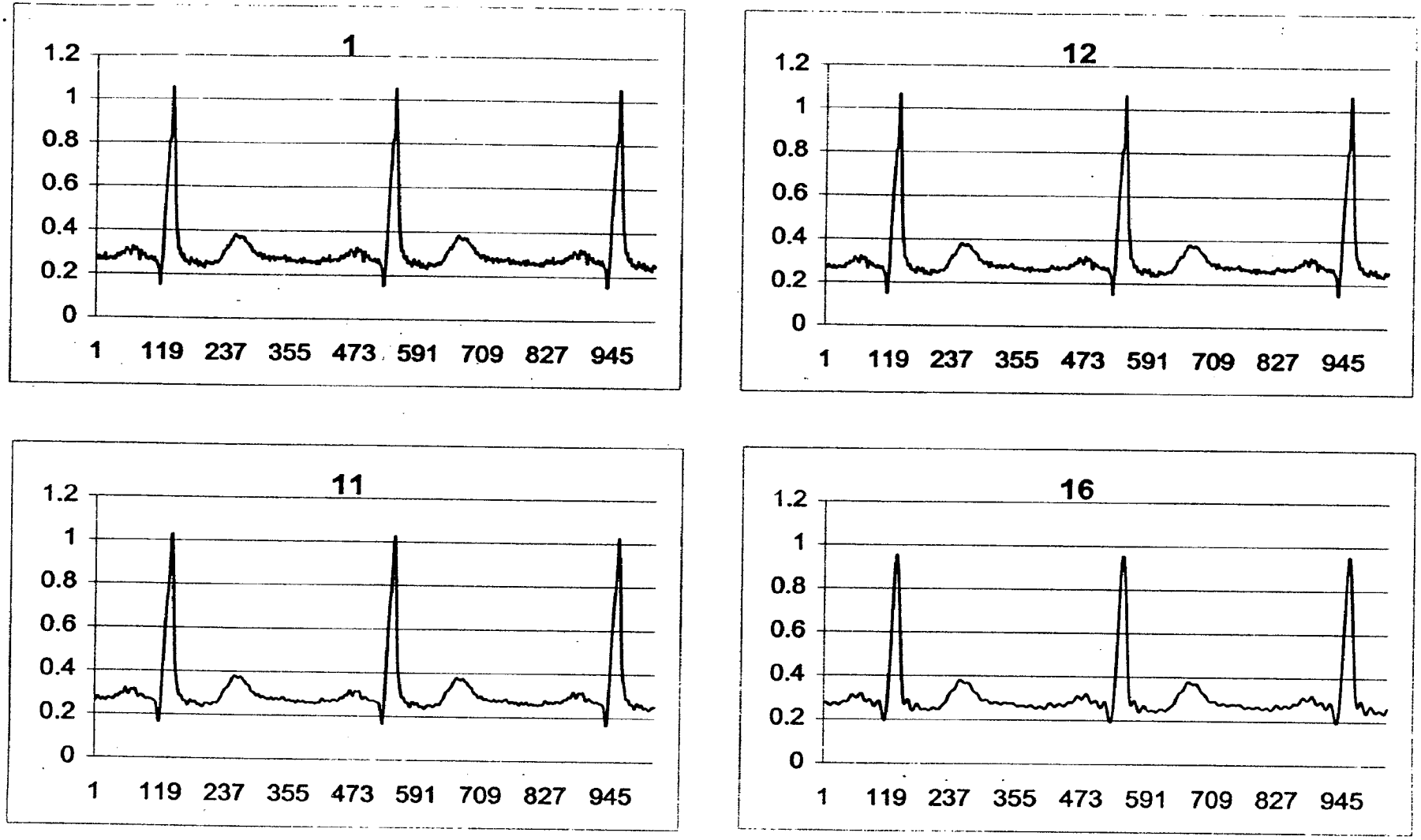




'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

Fig. 4.45 FOURIER ANALYSIS OF 4X ECG DATA

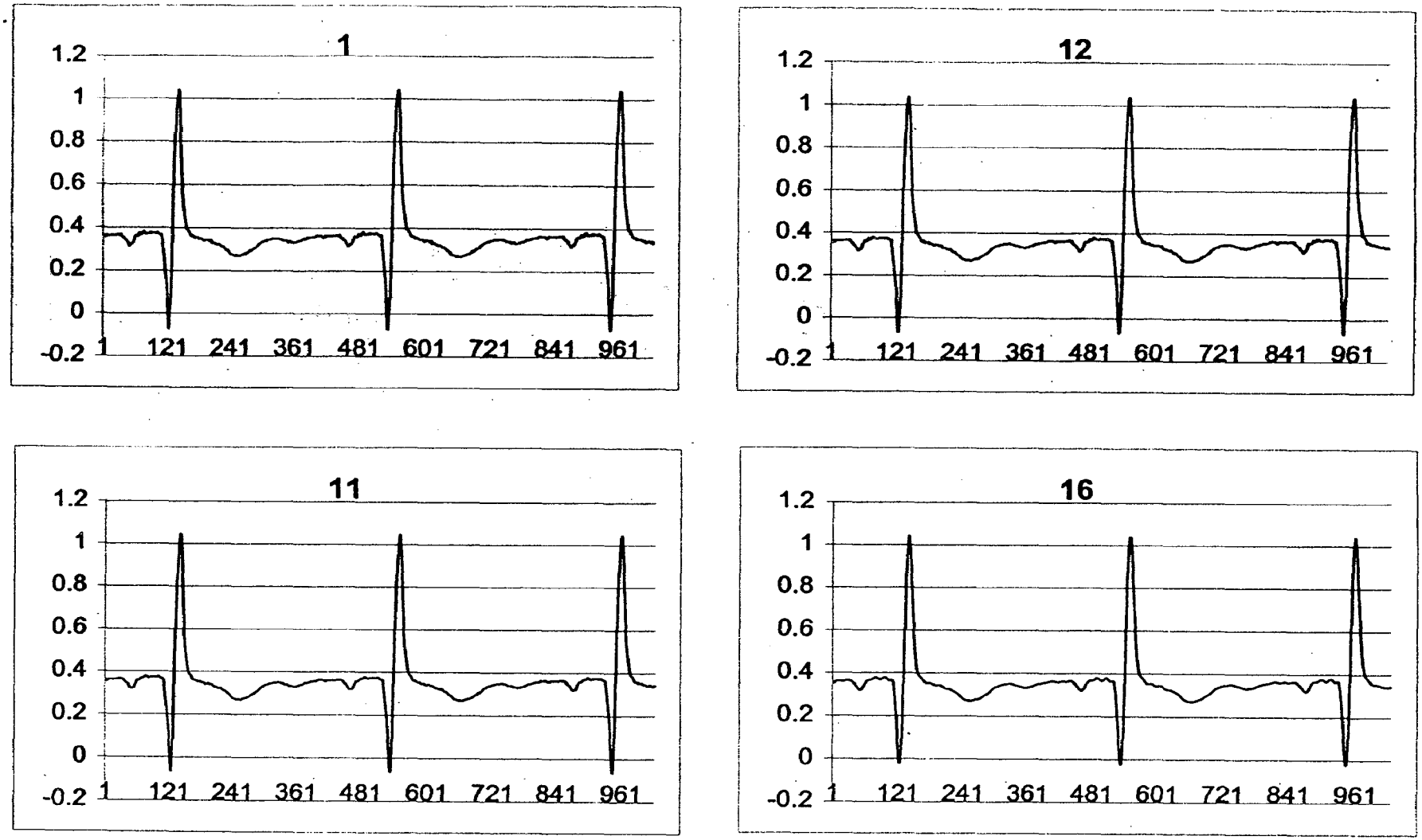
FOURIER ANALYSIS OF 4Y ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

Fig. 4.46 FOURIER ANALYSIS OF 4Y ECG DATA

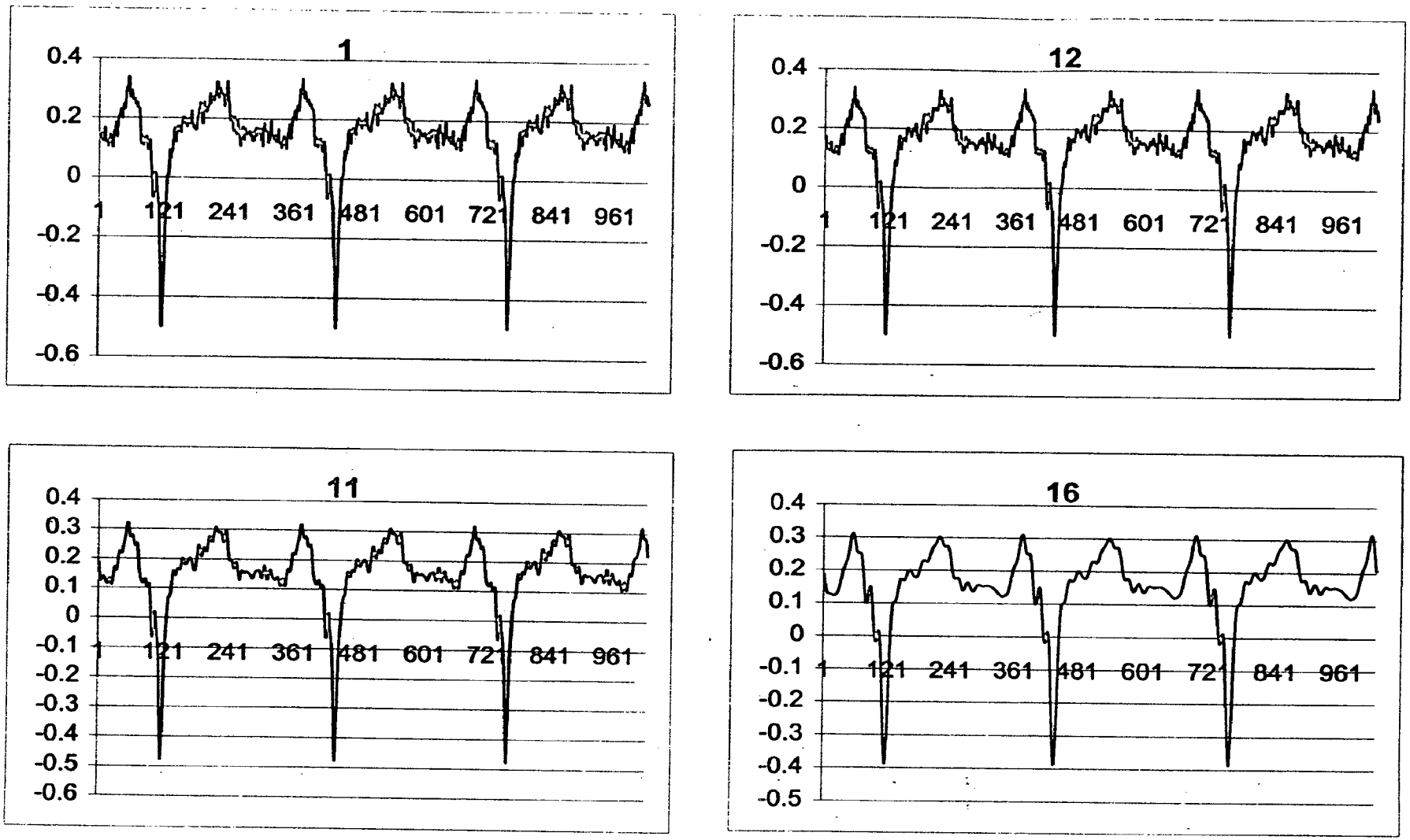
FOURIER ANALYSIS OF 4Z ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

Fig. 4.47 FOURIER ANALYSIS OF 4Z ECG DATA

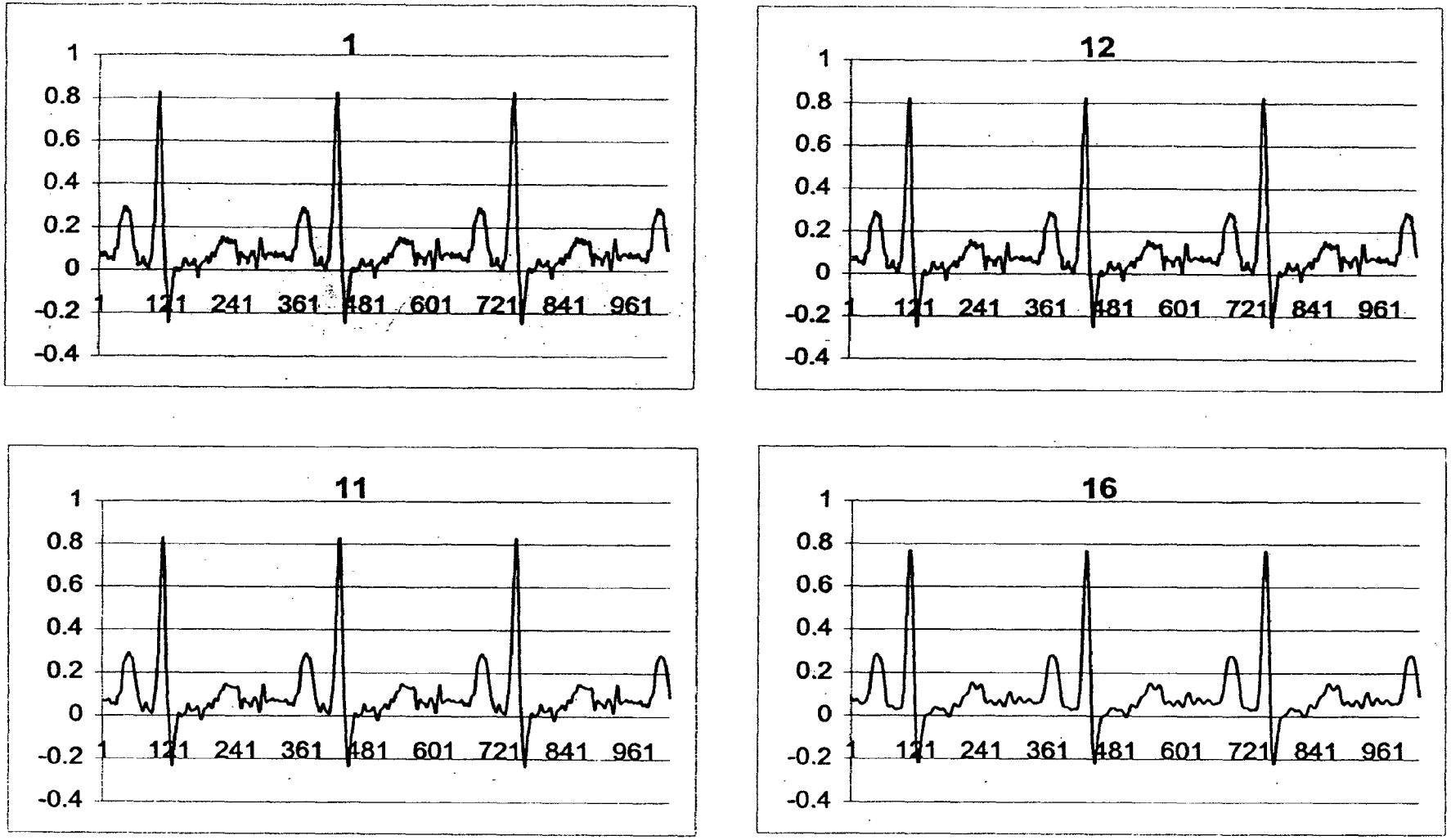
FOURIER ANALYSIS OF 5L1 ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

Fig. 4.48 FOURIER ANALYSIS OF 5L1 ECG DATA

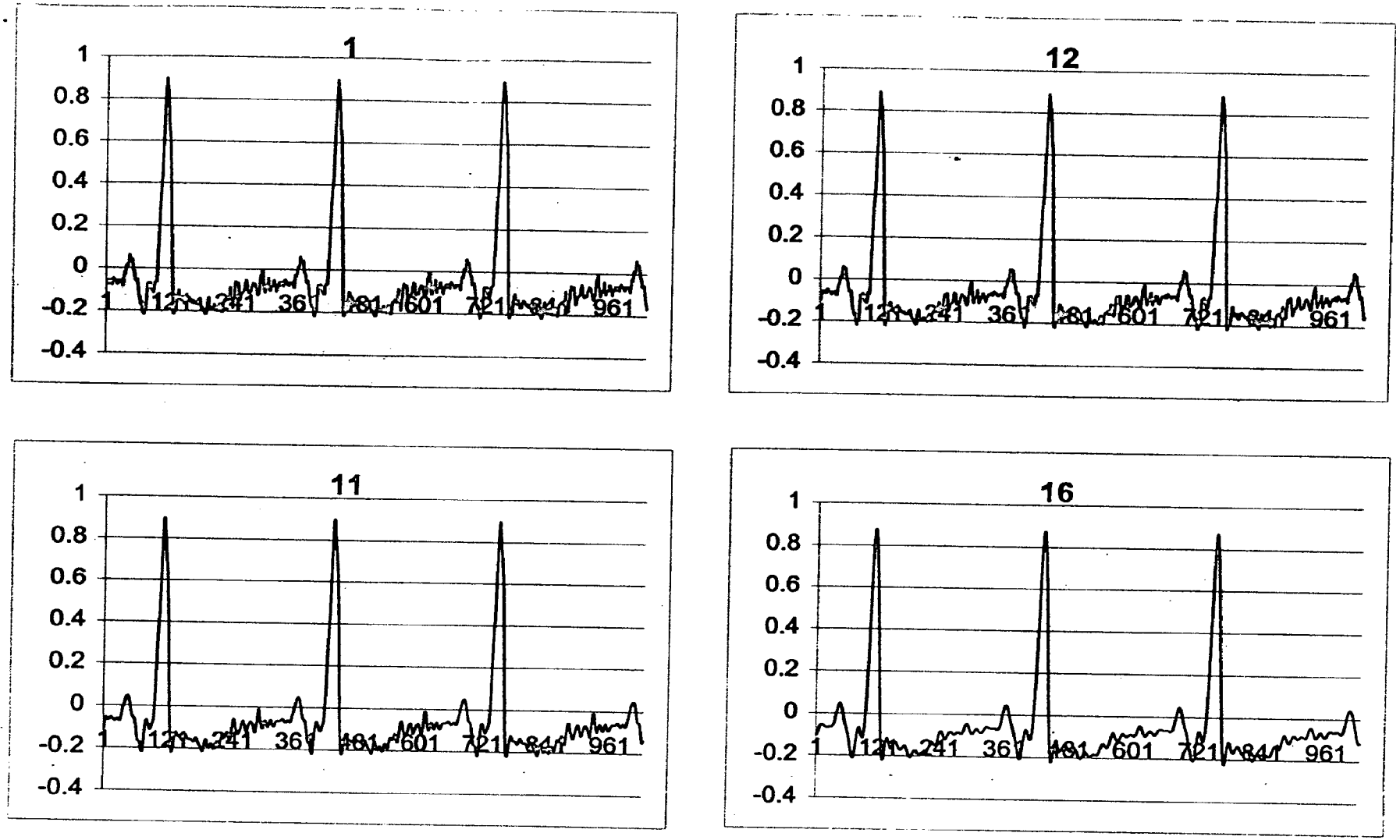
FOURIER ANALYSIS OF 5L2 ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

Fig. 4.49 FOURIER ANALYSIS OF 5L2 ECG DATA

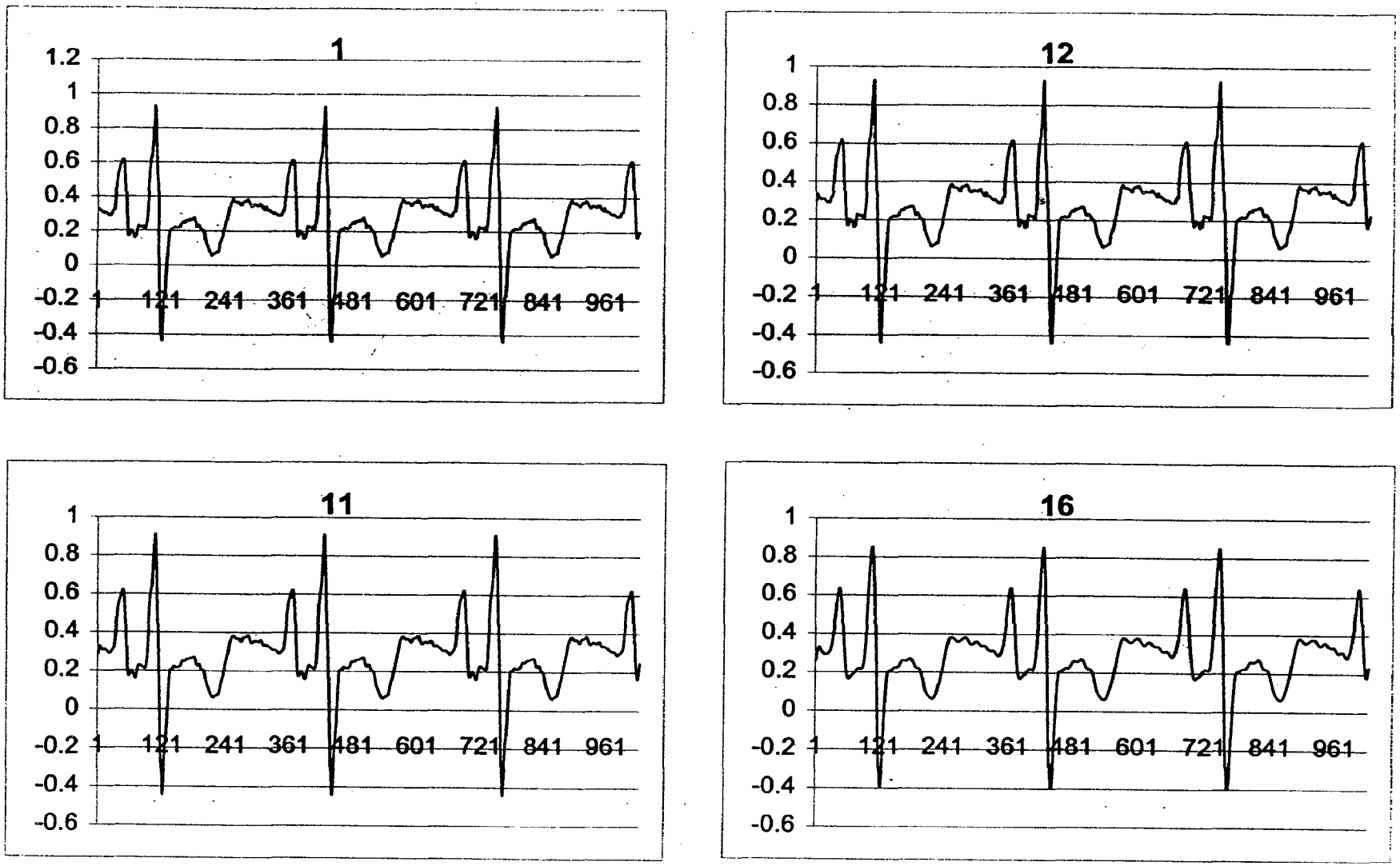
FOURIER ANALYSIS OF 5L3 ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

Fig. 4.50 FOURIER ANALYSIS OF 5L3 ECG DATA

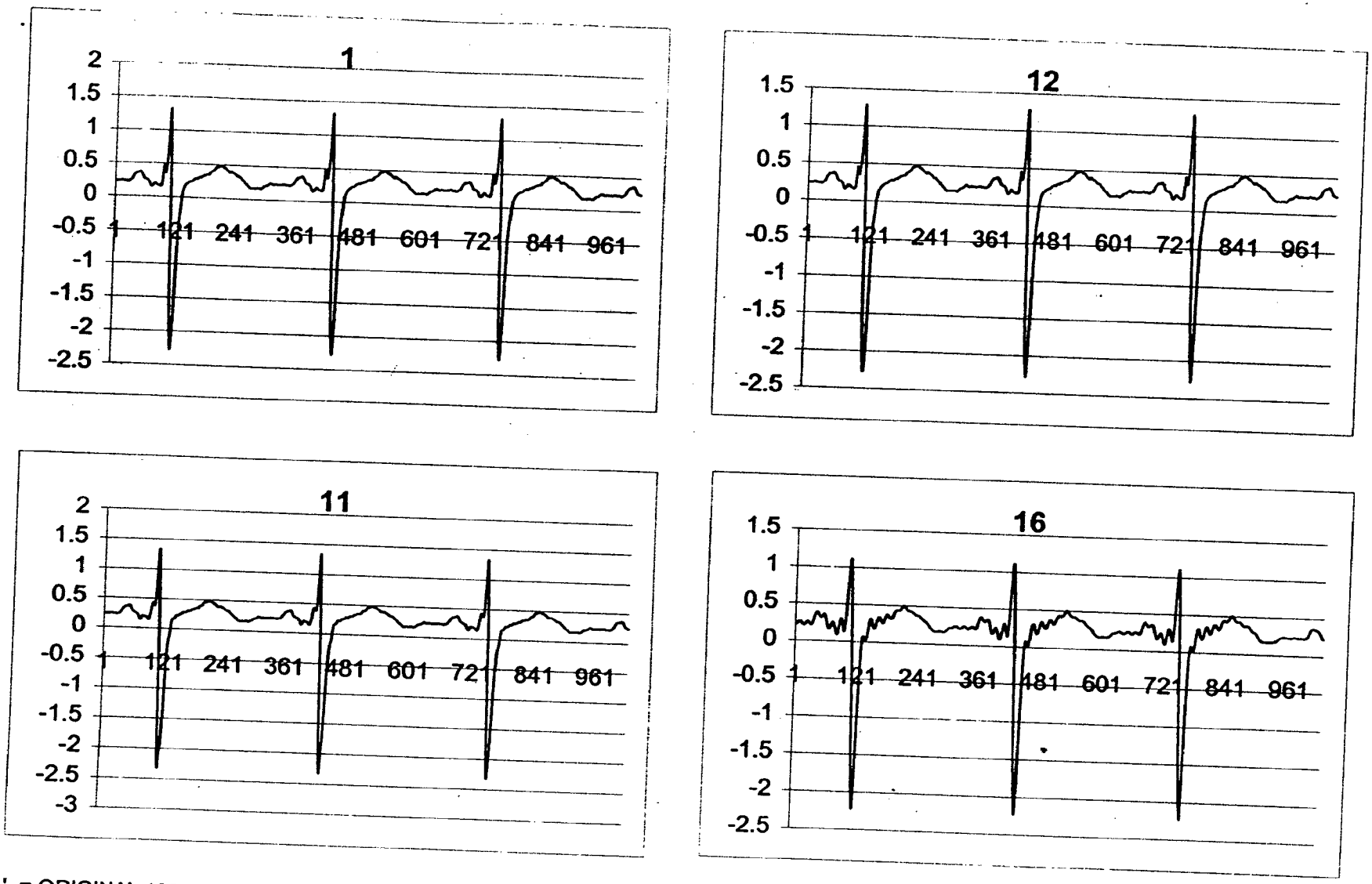
FOURIER ANALYSIS OF 5V1 ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

Fig. 4.51 FOURIER ANALYSIS OF 5V1 ECG DATA

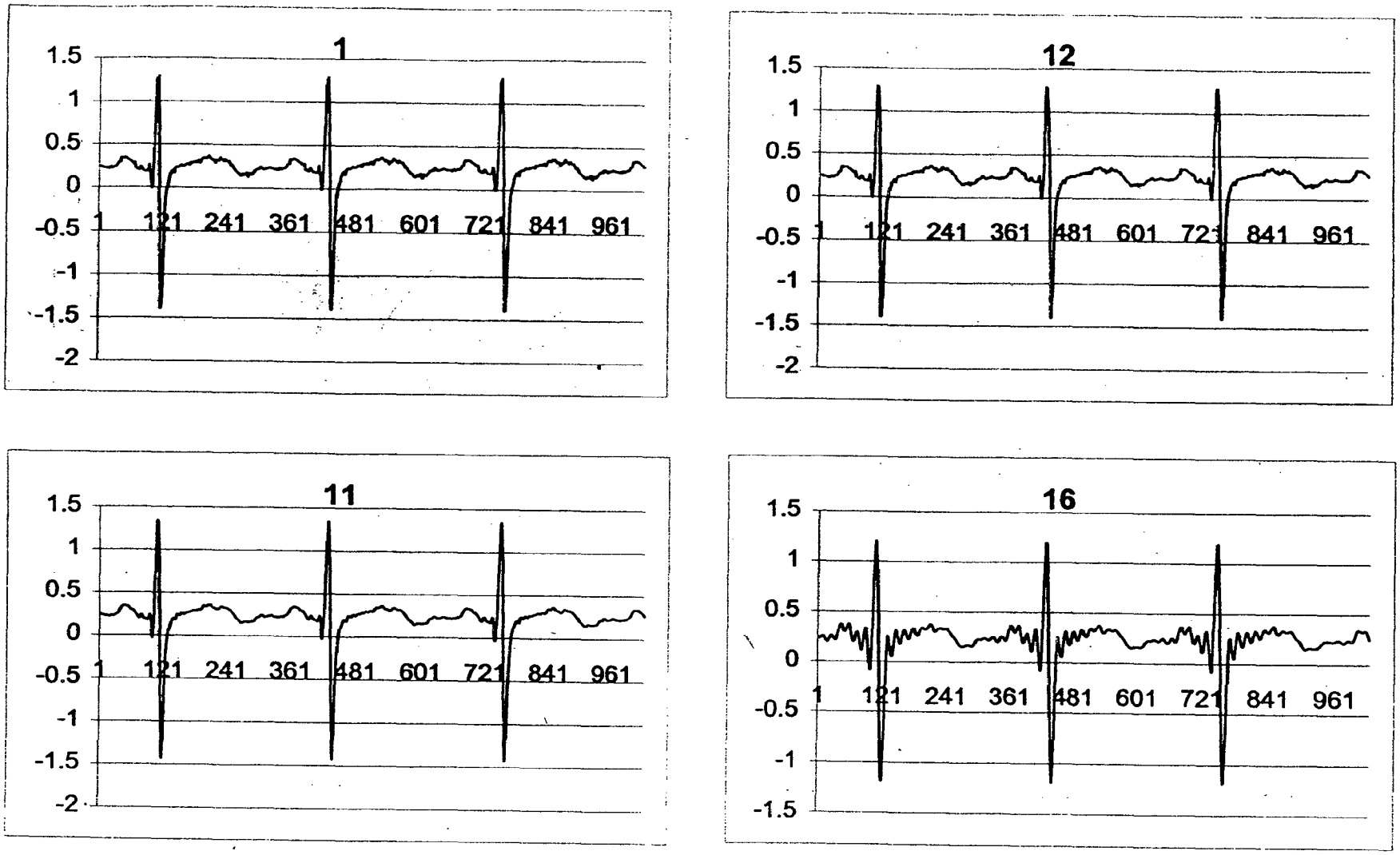
FOURIER ANALYSIS OF 5V2 ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

Fig. 4.52 FOURIER ANALYSIS OF 5V2 ECG DATA

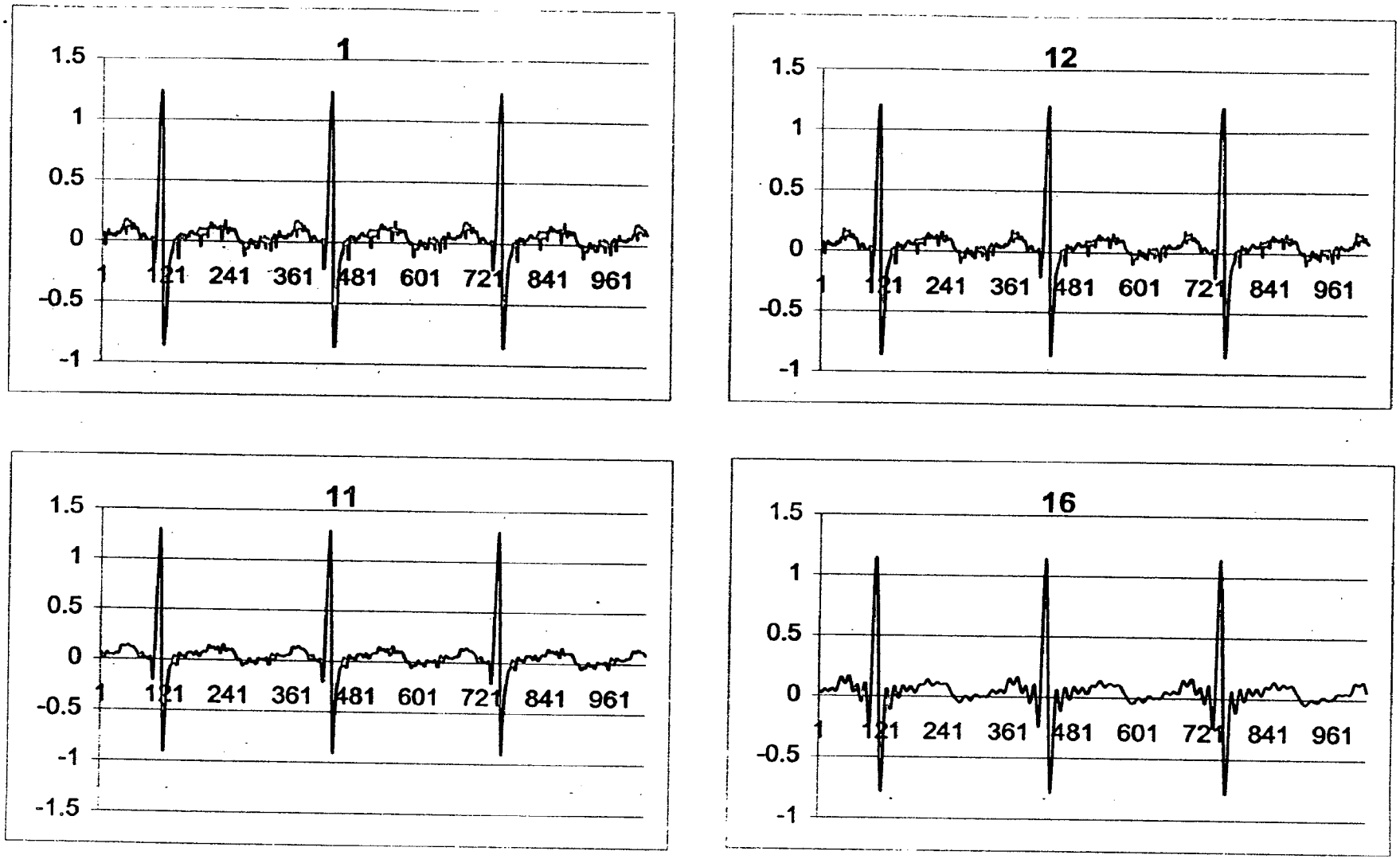
FOURIER ANALYSIS OF 5V3 ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

Fig. 4.53 FOURIER ANALYSIS OF 5V3 ECG DATA

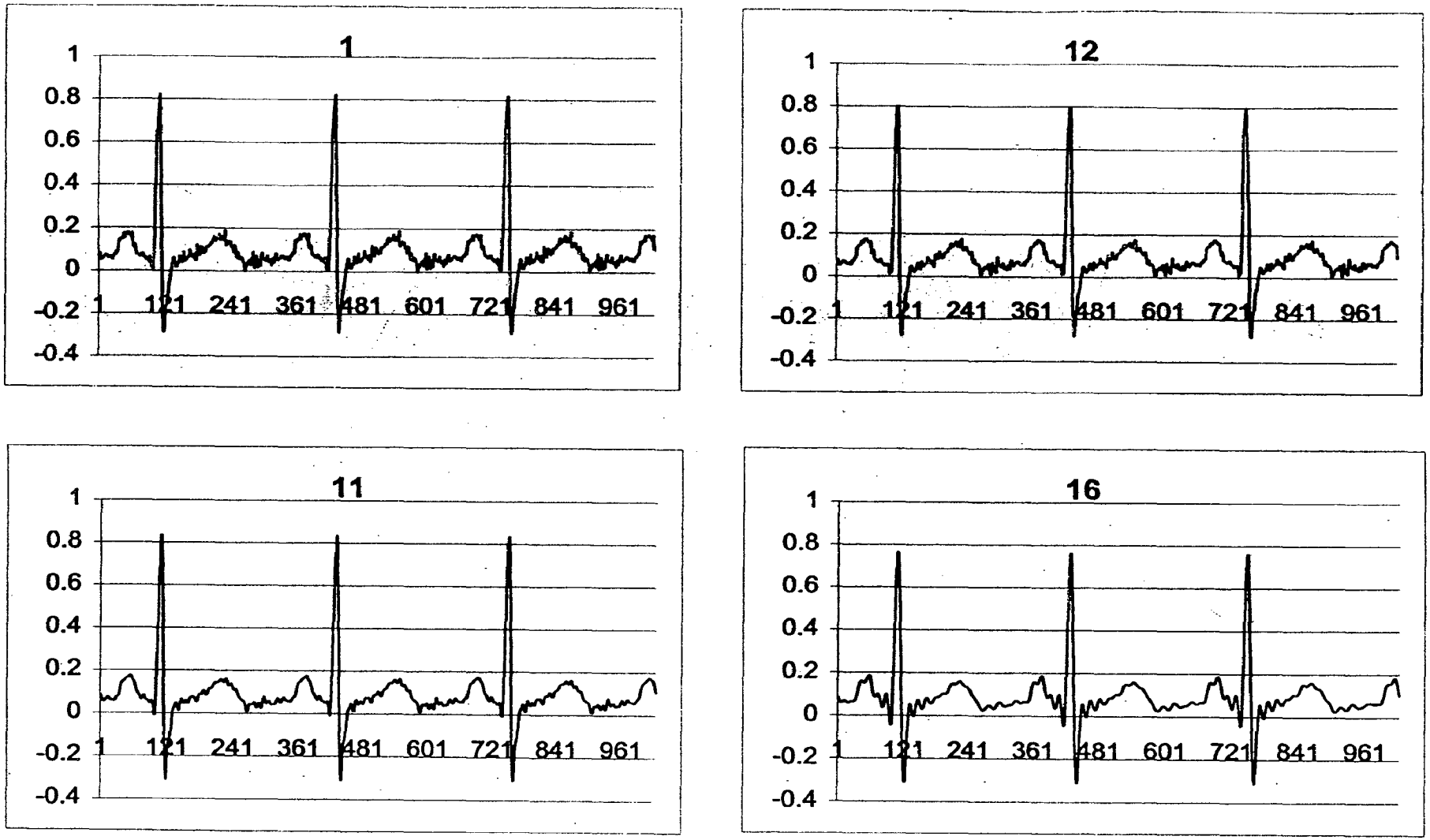
FOURIER ANALYSIS OF 5V4 ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

Fig. 4.54 FOURIER ANALYSIS OF 5V4 ECG DATA

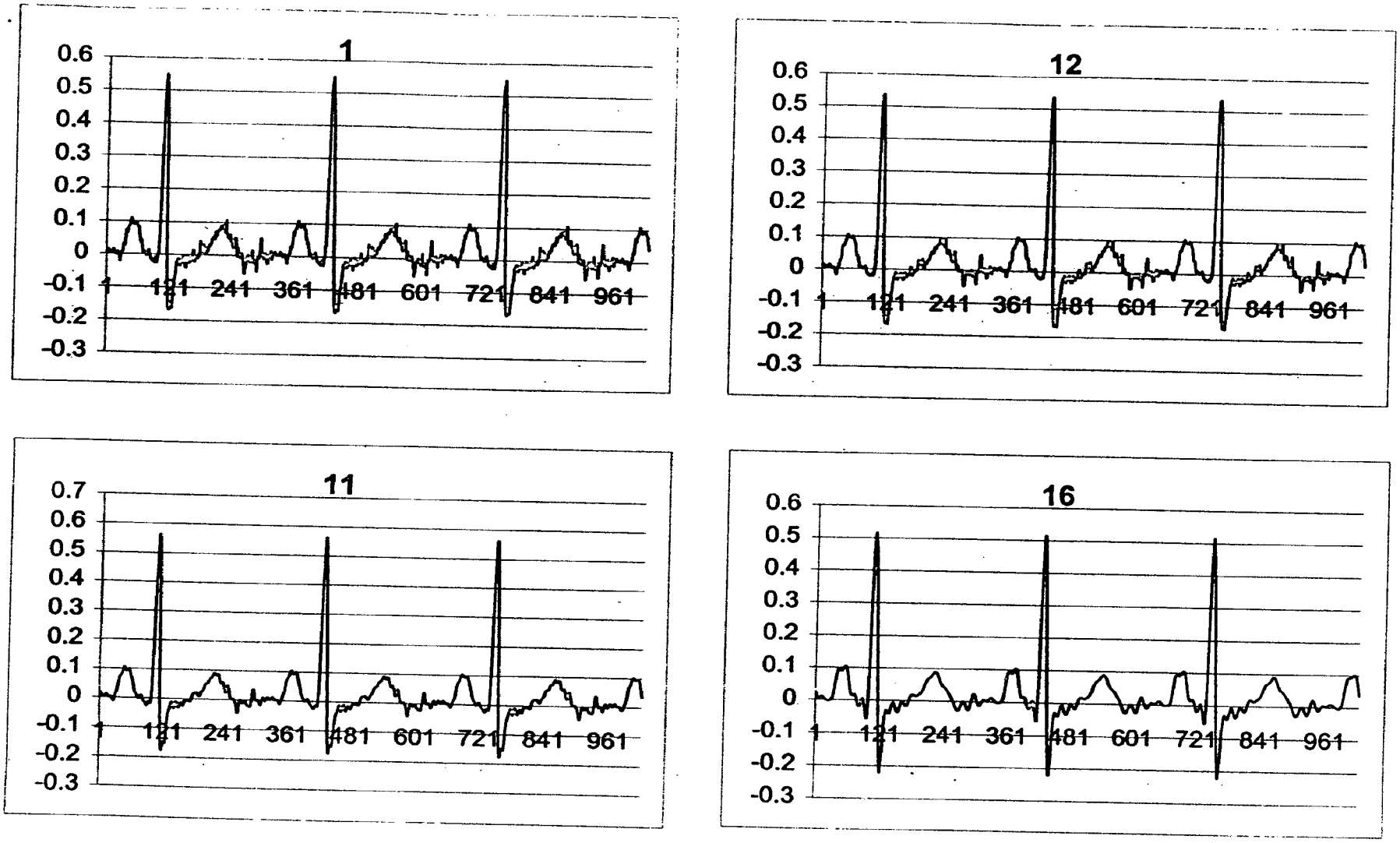
FOURIER ANALYSIS OF 5V5 ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

Fig. 4.55 FOURIER ANALYSIS OF 5V5 ECG DATA

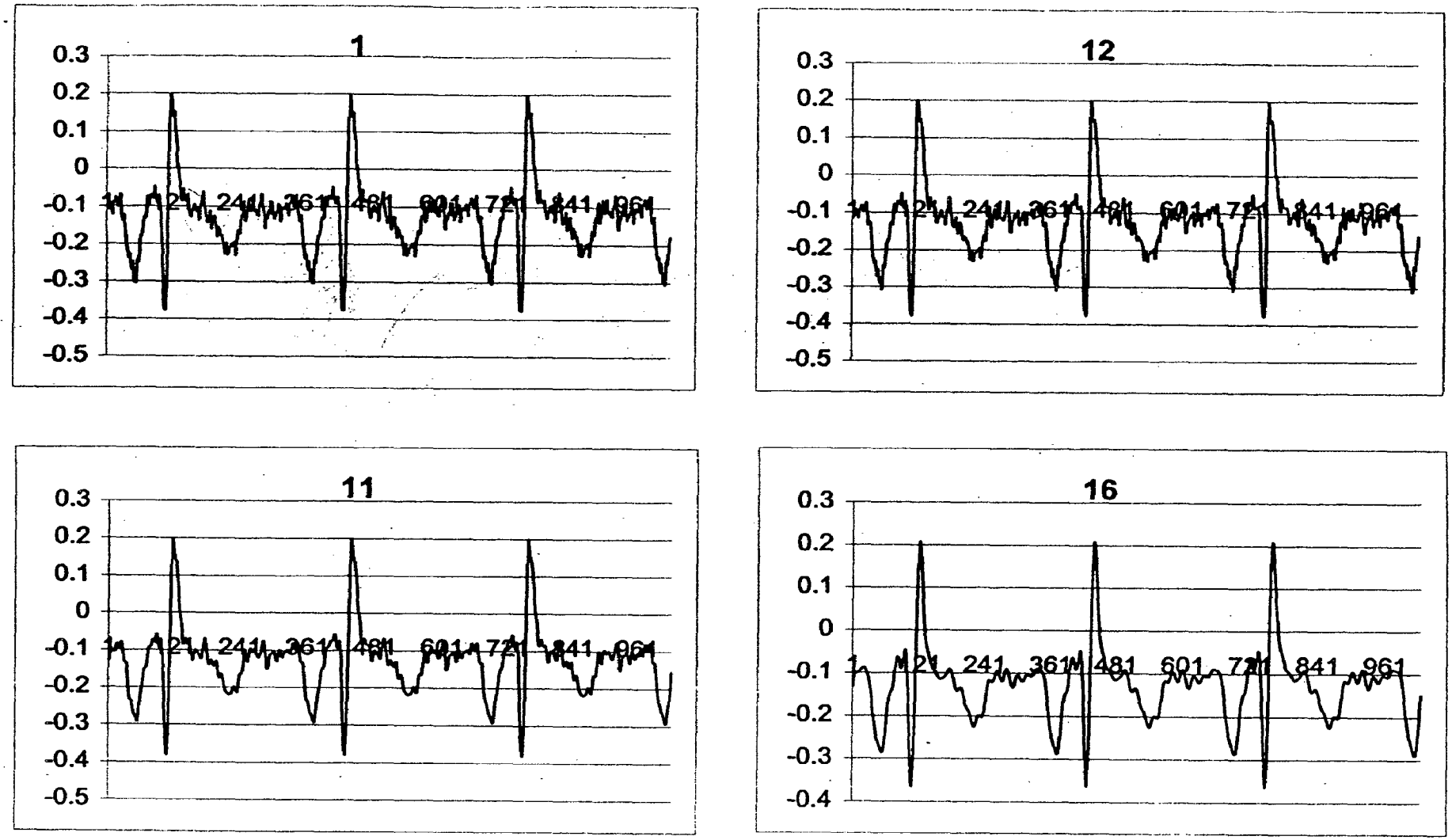
FOURIER ANALYSIS OF 5V6 ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

Fig. 4.56 FOURIER ANALYSIS OF 5V6 ECG DATA

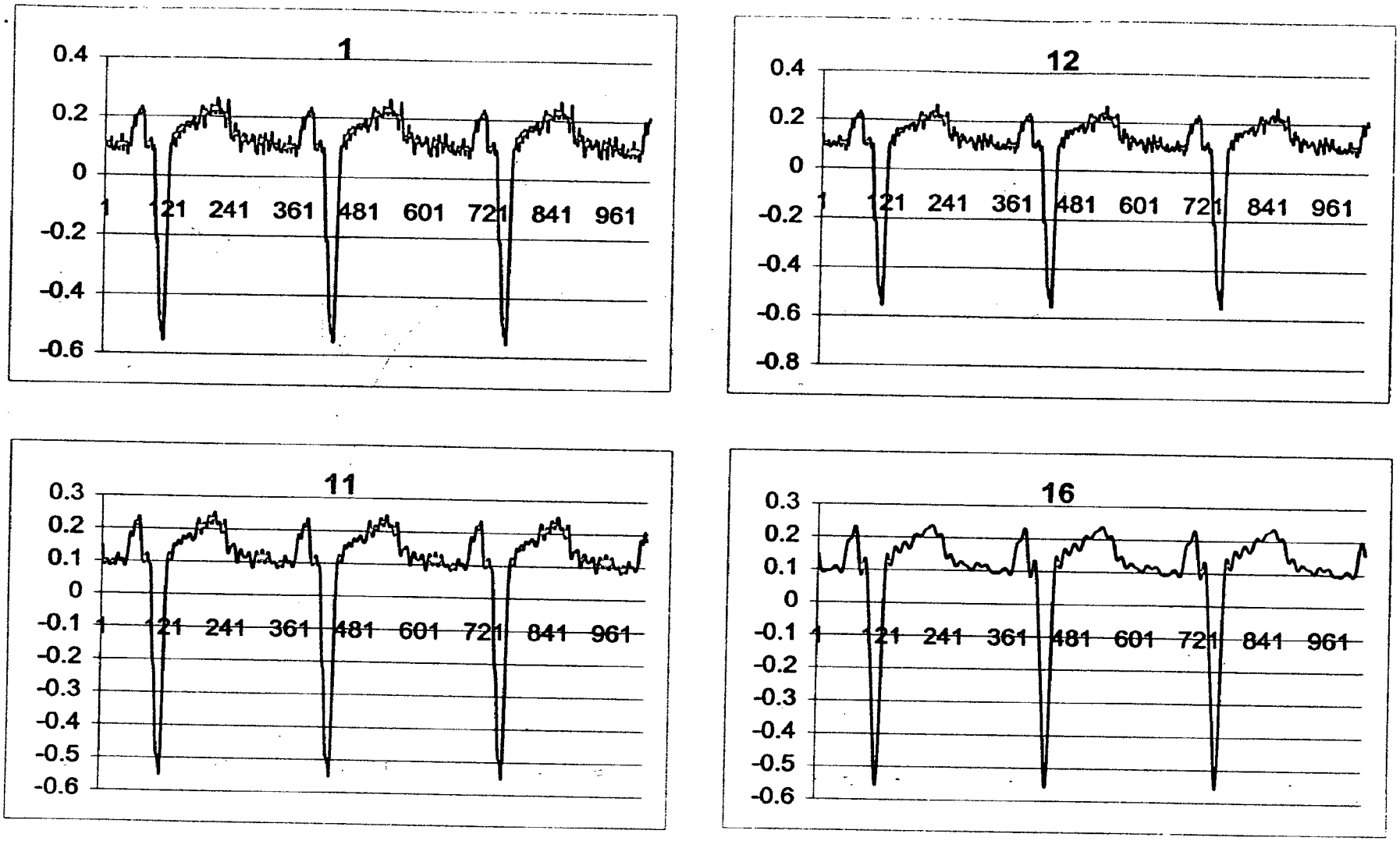
FOURIER ANALYSIS OF 5AR ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

Fig. 4.57 FOURIER ANALYSIS OF 5AR ECG DATA

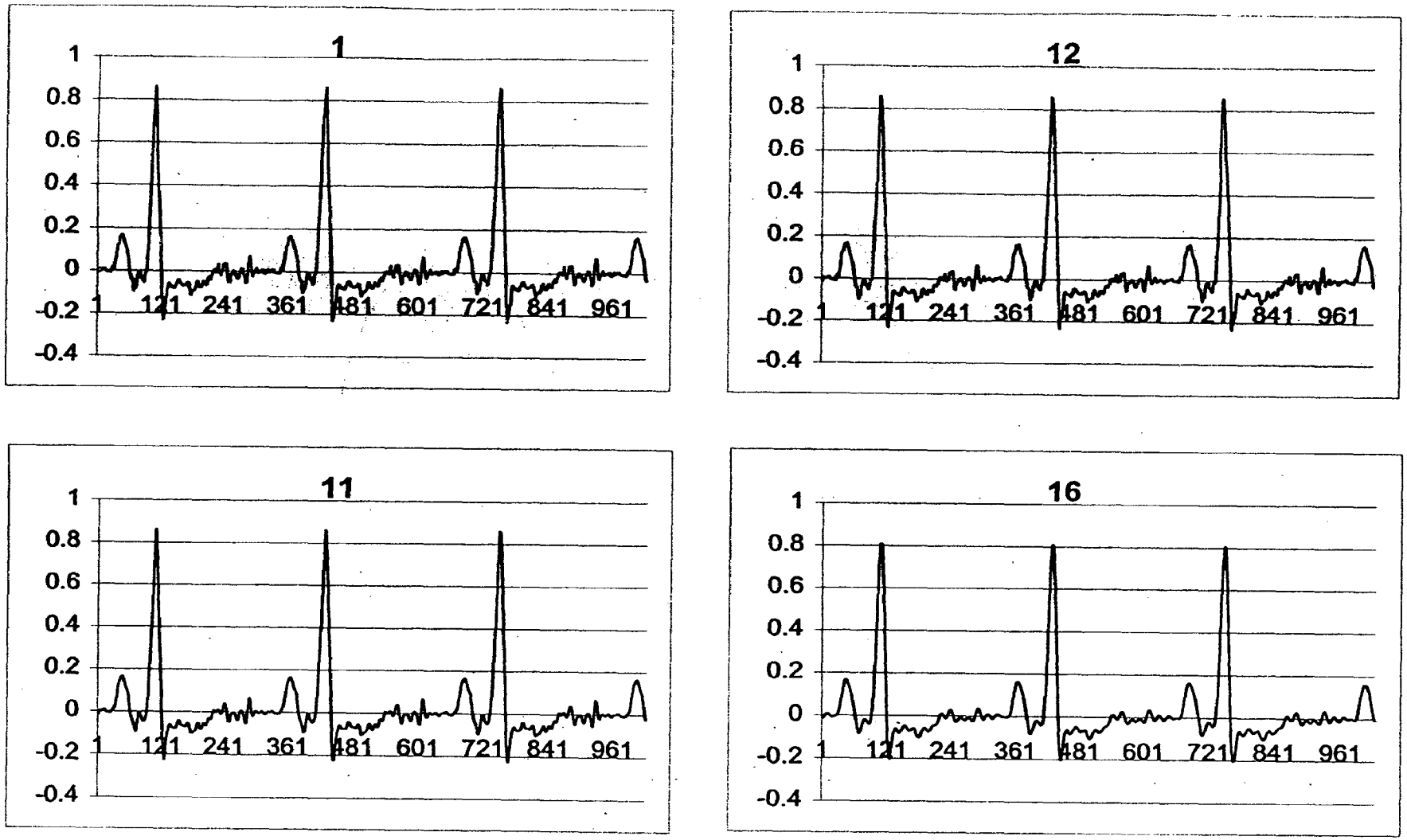
FOURIER ANALYSIS OF 5AL ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

Fig. 4.58 FOURIER ANALYSIS OF 5AL ECG DATA

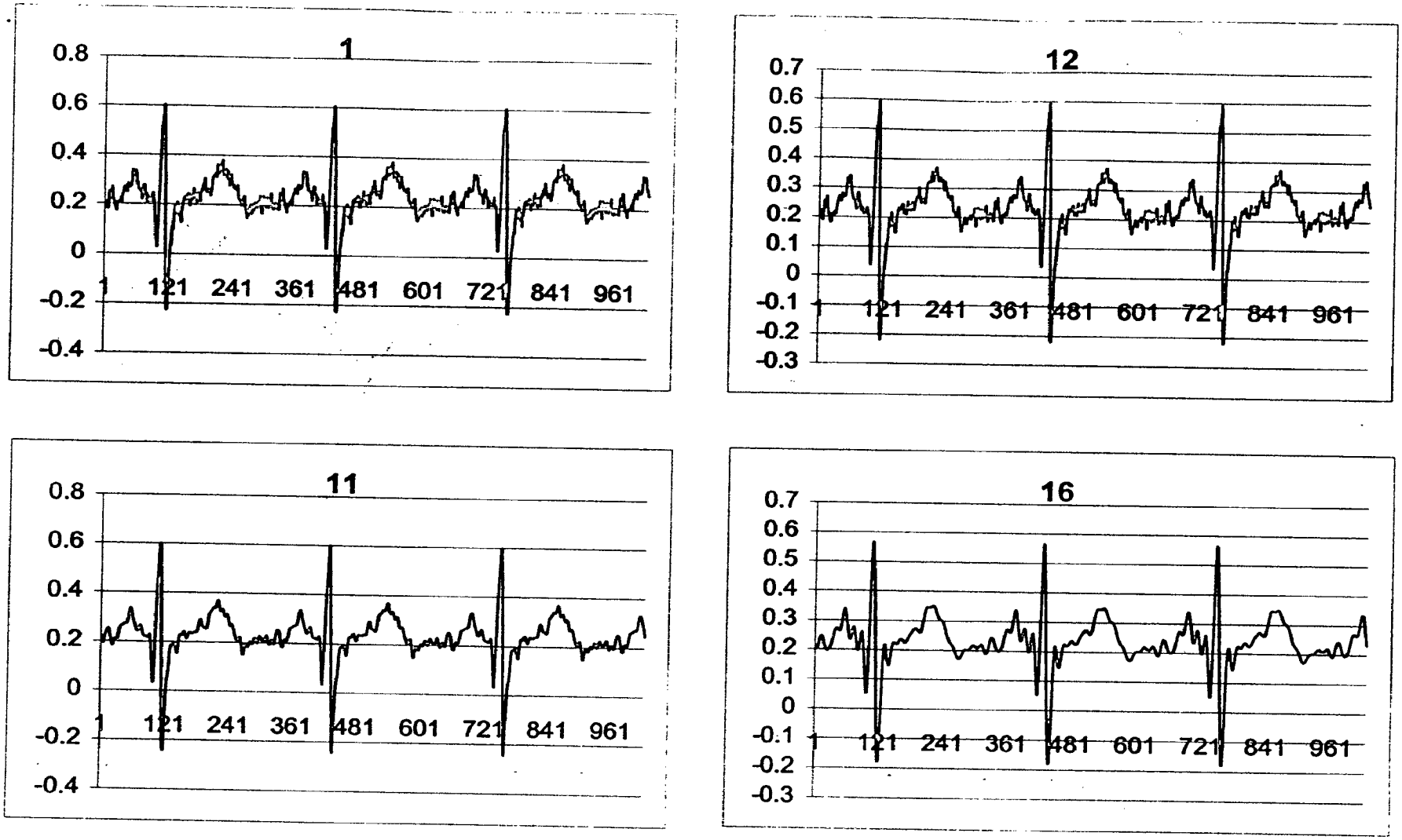
FOURIER ANALYSIS OF 5AF ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

Fig. 4.59 FOURIER ANALYSIS OF 5AF ECG DATA

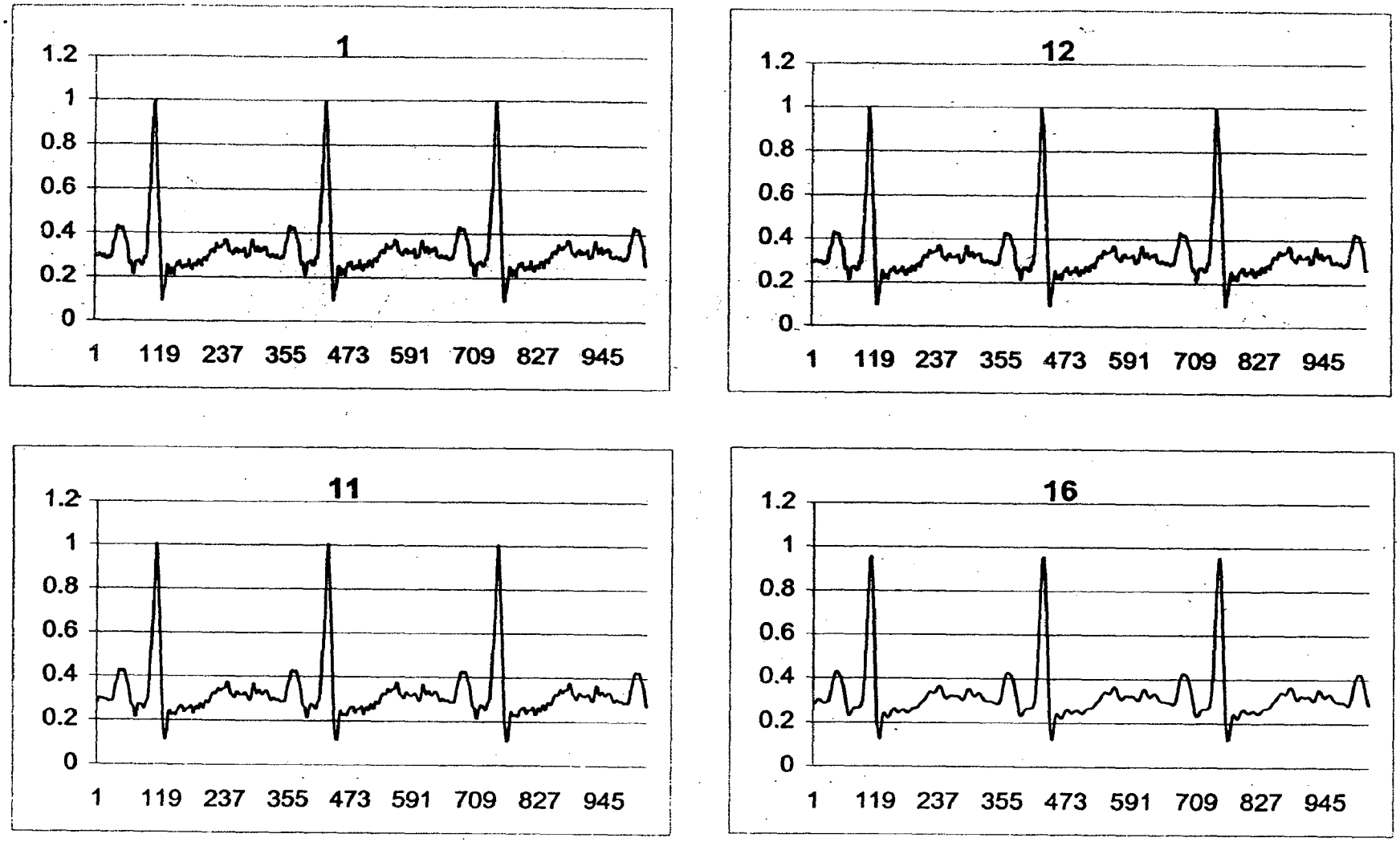
FOURIER ANALYSIS OF 5X ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

Fig. 4.60 FOURIER ANALYSIS OF 5X ECG DATA

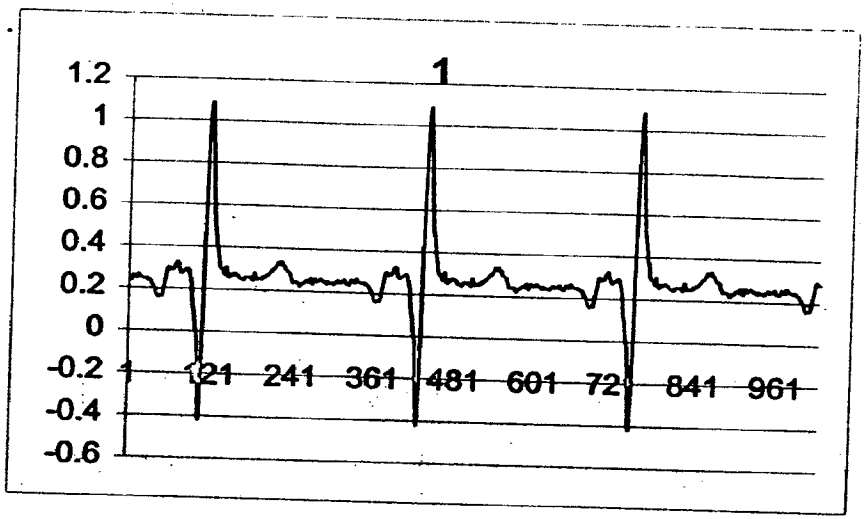
FOURIER ANALYSIS OF 5Y ECG DATA

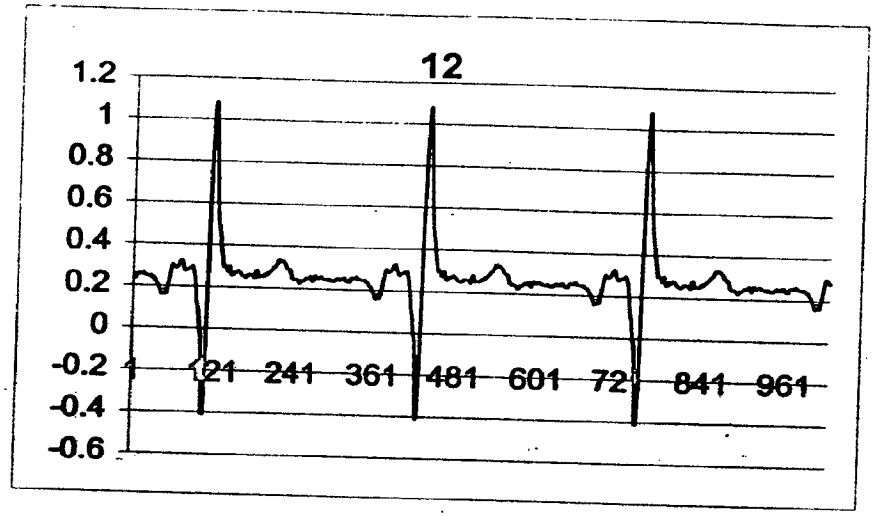


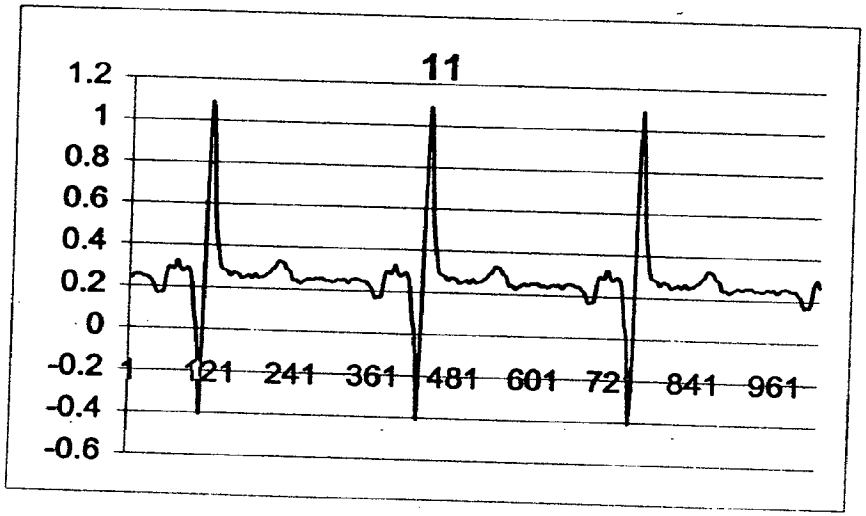
'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

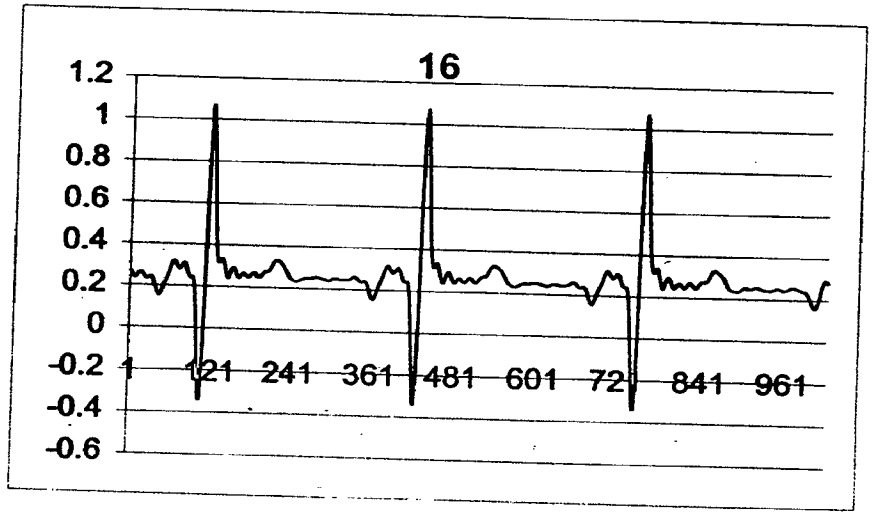
FIG. 4.61 FOURIER ANALYSIS OF 5Y ECG DATA

FOURIER ANALYSIS OF 5Z ECG DATA









'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

FIG. 4.62 FOURIER ANALYSIS OF 5Z ECG DATA

$$T(u) = \sum_{x=o}^{N-1} f(x)\, g(x,u) \qquad (4.16)$$

Where T(u) is the transform of f(x), g(x,u) is the forward transformation kernel and x has values in the range 0, 1, ..., N-1.

When $N = 2^n$, the discrete Walsh Transform of a function f(x), denoted by W(u) is obtained by substituting the Kernel [23],

$$g(x,u) = \frac{1}{N} \sum_{i=o}^{n-1} (-1)^{b_i(x)b_{n-1-i}(u)} \qquad (4.17)$$

into Eqn. (4.16),

$$W(u) = \frac{1}{N} \sum_{i=o}^{N-1} f(x) \prod_{i=o}^{n-1} (-1)^{b_i(x)b_{n-1-i}(u)} \qquad (4.18)$$

Where $b_k(z)$ is the Kth bit in the binary representation of Z. If n=3 and Z =6, $b_0(Z) = 0$, $b_1(z) = 1$ and $b_2(Z) = 1$. The value of g(x,u) is given for N = 8 in the Figs. 4.63 and 4.64.

4.3.1 FWT ALGORITHM

The Walsh Transform may be computed by a fast algorithm nearly identical in form to the successive doubling as for the FFT. The only difference is that all exponential terms W_N are set equal to 1 in the case of the Fast Walsh Transform (FWT). The basic relation leading to the FFT then becomes [12]

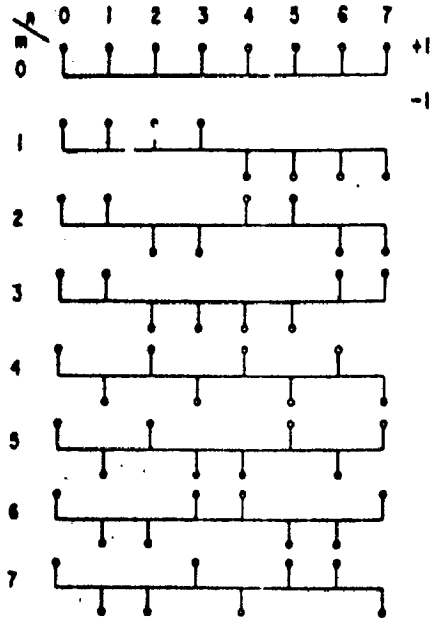


Fig. 4.63 : THE EIGHT WALSH FUNCTIONS OF LENGTH 8. [51]

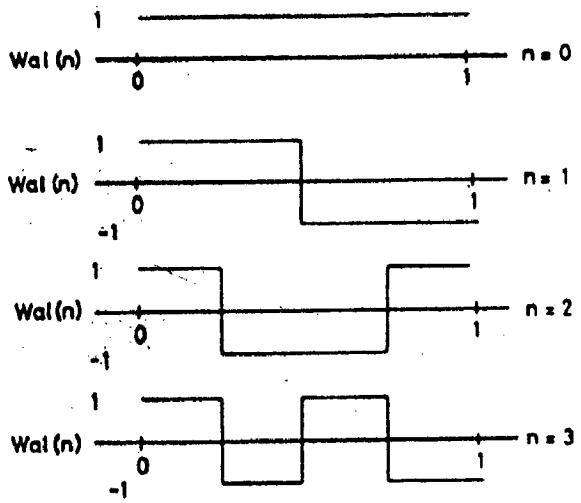


Fig. 4.64 : SEQUENCE-ORDERED CONTINUOUS WALSH FUNCTIONS [51]

$$W(u) = \frac{1}{2}[W_{even}(u) + W_{odd}(u)] \quad (4.19)$$

$$W(u + M) = \frac{1}{2}[W_{even}(u) - W_{odd}(u)] \quad (4.20)$$

Where M = N/2, u = 0,1....., M-1, and W(u) denotes the 1D Walsh Transform [23,47,51,52].

4.3.2 THE INVERSE FWT

The inverse transform is the relation

$$f(x) = \sum_{u=o}^{N-1} T(u)h(x,u) \quad (4.21)$$

Where h(x,u) is the inverse transformation kernel and x has values in the range 0,1,..., N-1.

The array formed by the walsh transformation kernel is a symmetric matrix having orthogonal rows and columns. This leads to an inverse kernel identical to forward Kernel except for a constant multiplicative factor of $\frac{1}{N}$, that is

$$h(x,u) = \prod_{i=o}^{n-1} (-1)^{b_i(x)b_{n-1-i}(u)} \quad (4.22)$$

Thus the Inverse Walsh Transformation is

$$f(x) = \sum_{u=o}^{N-1} W(u) \prod_{i=o}^{n-1} (-1)^{b_i(x)b_{n-1-i}(u)}$$

The Walsh Transform consists of a series expansion of basis functions whose value are +1 or -1, [23,51].

4.3.3 SOFTWARE IMPLEMENTATION

The flow charts and programs in "C" for implementation of FWT are given in Appendix-I and II respectively. The basic idea in calculating the FWT is same as that of FFT. Flow diagram of Fast Walsh Transform algorithm for N = 8 is shown in Fig. 4.65. Typical ECG and corresponding Walsh spectrum is also shown in Fig. 4.66.

Figs. 4.67 to 4.96 show the software results of FWT analysis of ECG signal. The comment on the results are given in chapter V.

4.4 METHODS USED IN ANALYSIS OF FFT AND FWT

The technique used to compress the data consist of using only a fraction of Fourier Spectrum/Walsh Spectrum to reconstruct an ECG in which reconstructed ECG signals are produced using from 1/8 to the entire Fourier/Walsh Spectrum. In doing so, the mid section of the Fourier Spectrum is set to zero and rest section are retained while in Walsh the lowest spectrum are retained and rest section are set to zero. And then inverse of spectrum is done by FFT/FWT.

In both the transformative techniques the comparision of original and reconstructed signal is done by the folloiwng three methods.

4.4.1 CALCULATION OF PERCENT ROOT MEAN SQUARE DIFFERENCE (PRD)

The PRD value is given by comparing original and reconstructed signal [31]

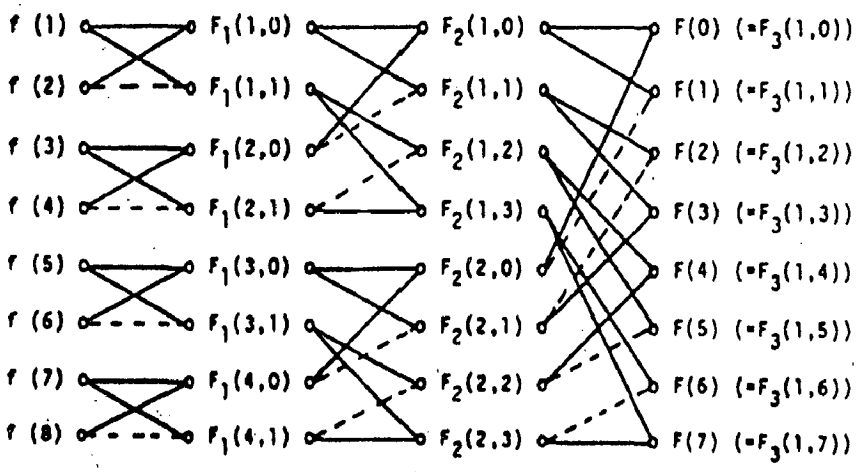


Fig. 4.65 : FLOW DIAGRAM OF FAST WALSH TRANSFORM ALGORITHM FOR N = 8 ---------- SUBTRACT ———— ADD [35]

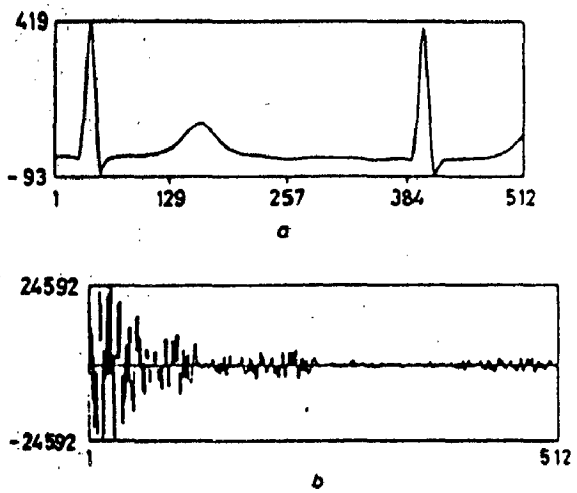
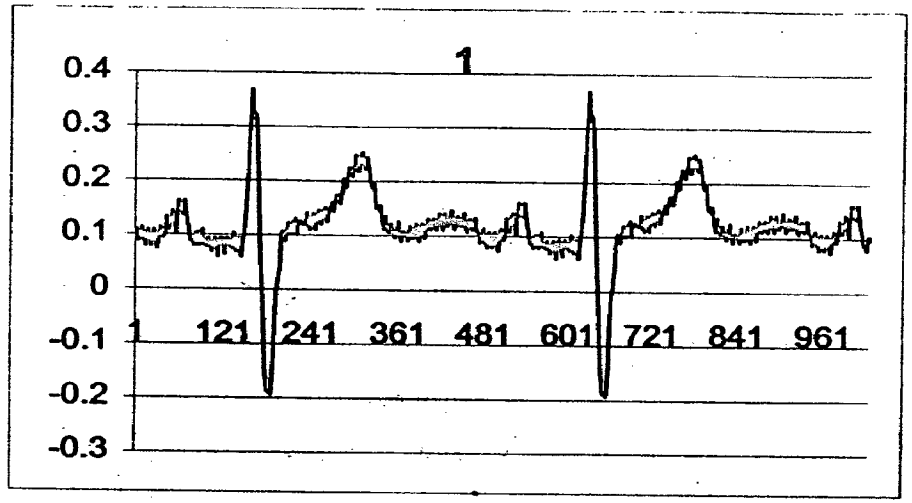
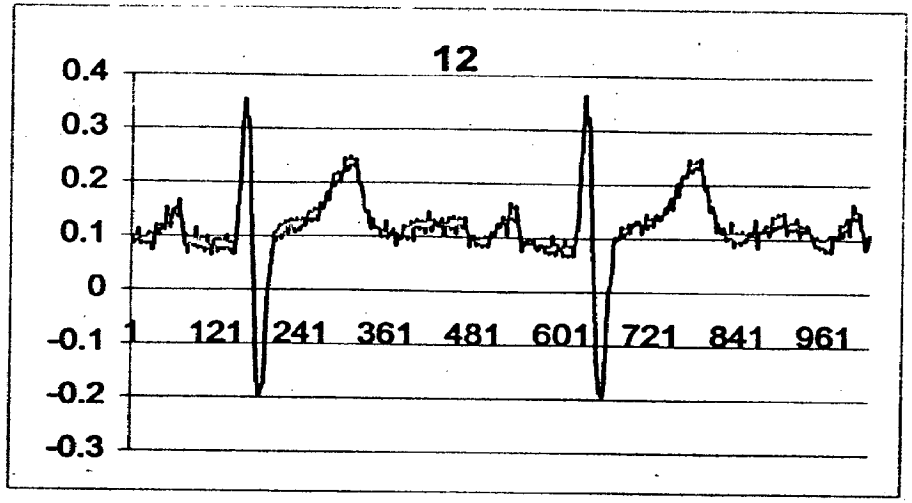
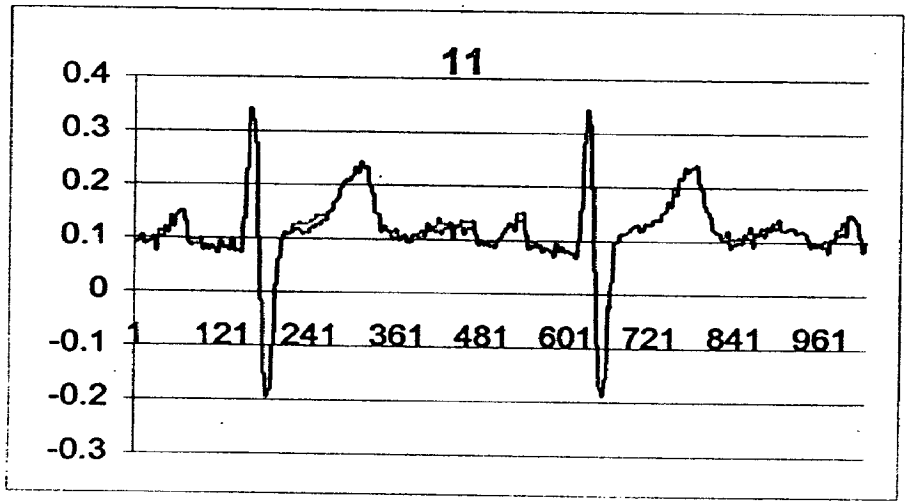


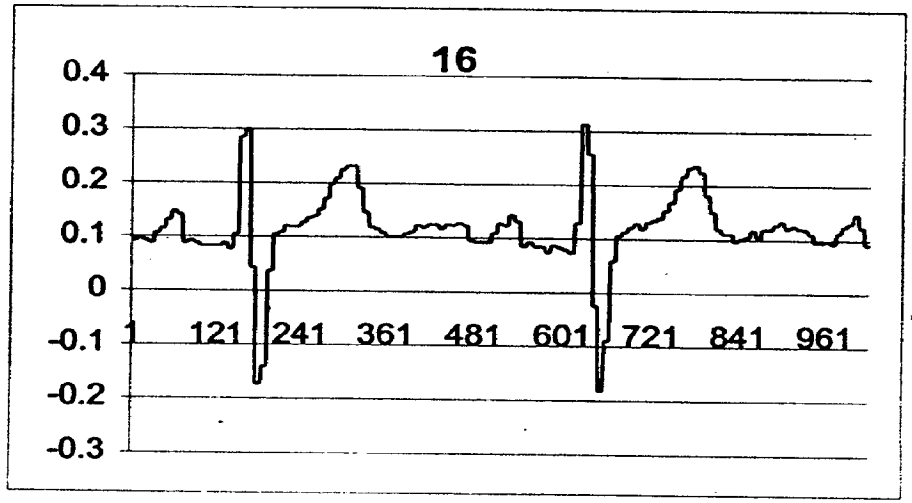
Fig. 4.66 : TYPICAL ECG AND CORRESPONDING SEQUENCE ORDERED WALSH SPECTRUM [35]

WALSH ANALYSIS OF 1L1 ECG DATA





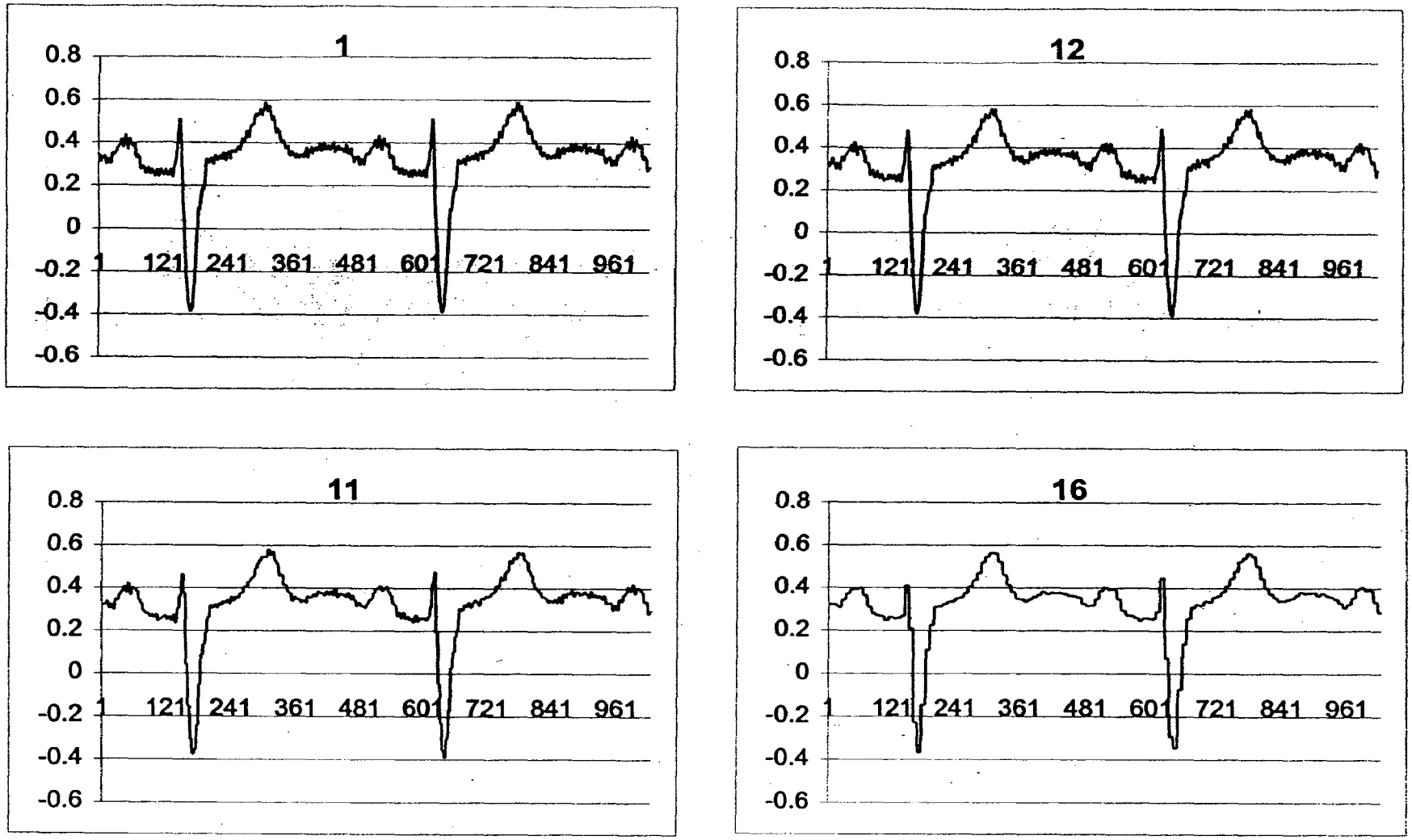




'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

FIG. 4.67 WALSH ANALYSIS OF 1L1 ECG DATA

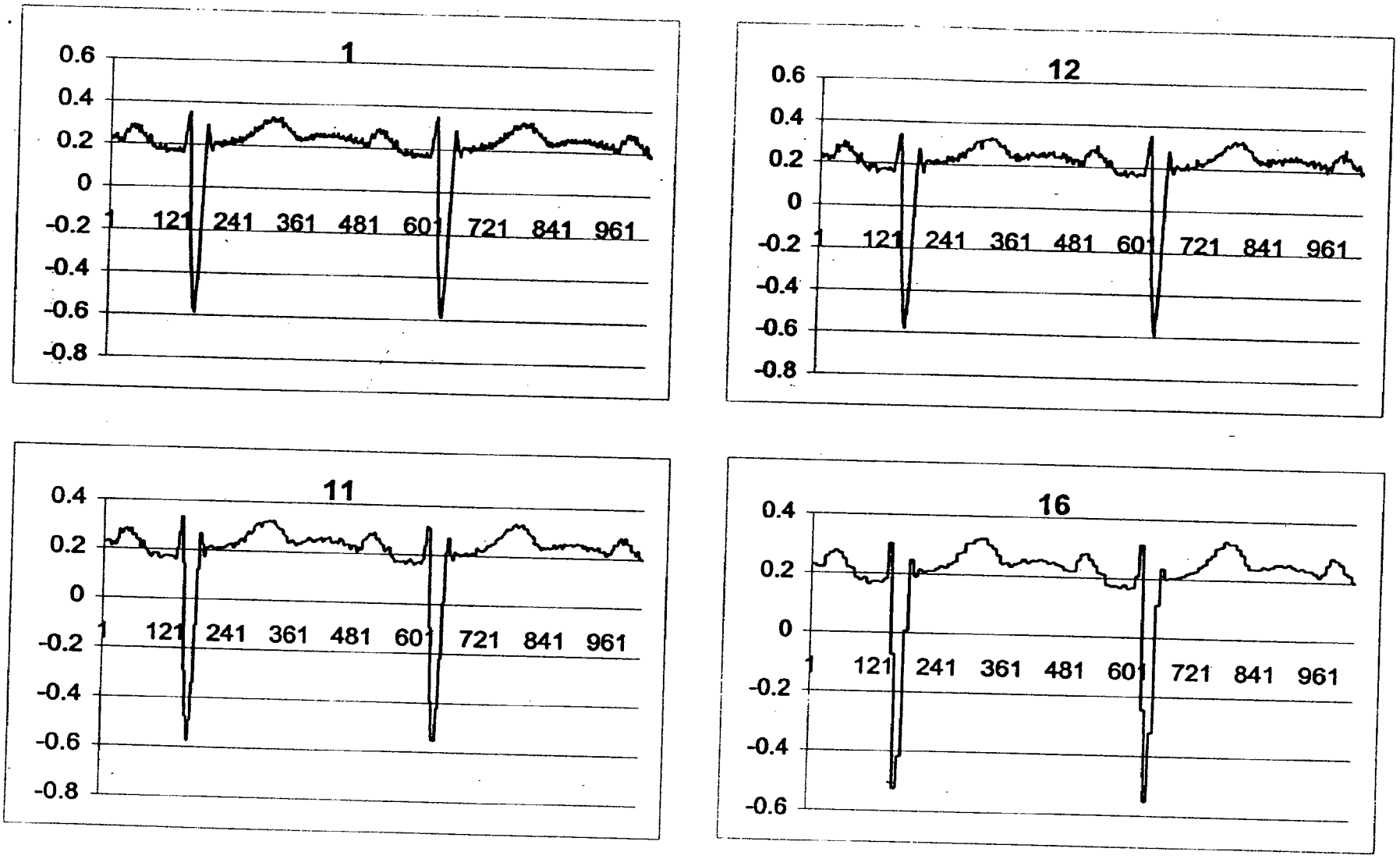
WALSH ANALYSIS OF 1L2 ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

FIG. 4.68 WALSH ANALYSIS OF 1L2 ECG DATA

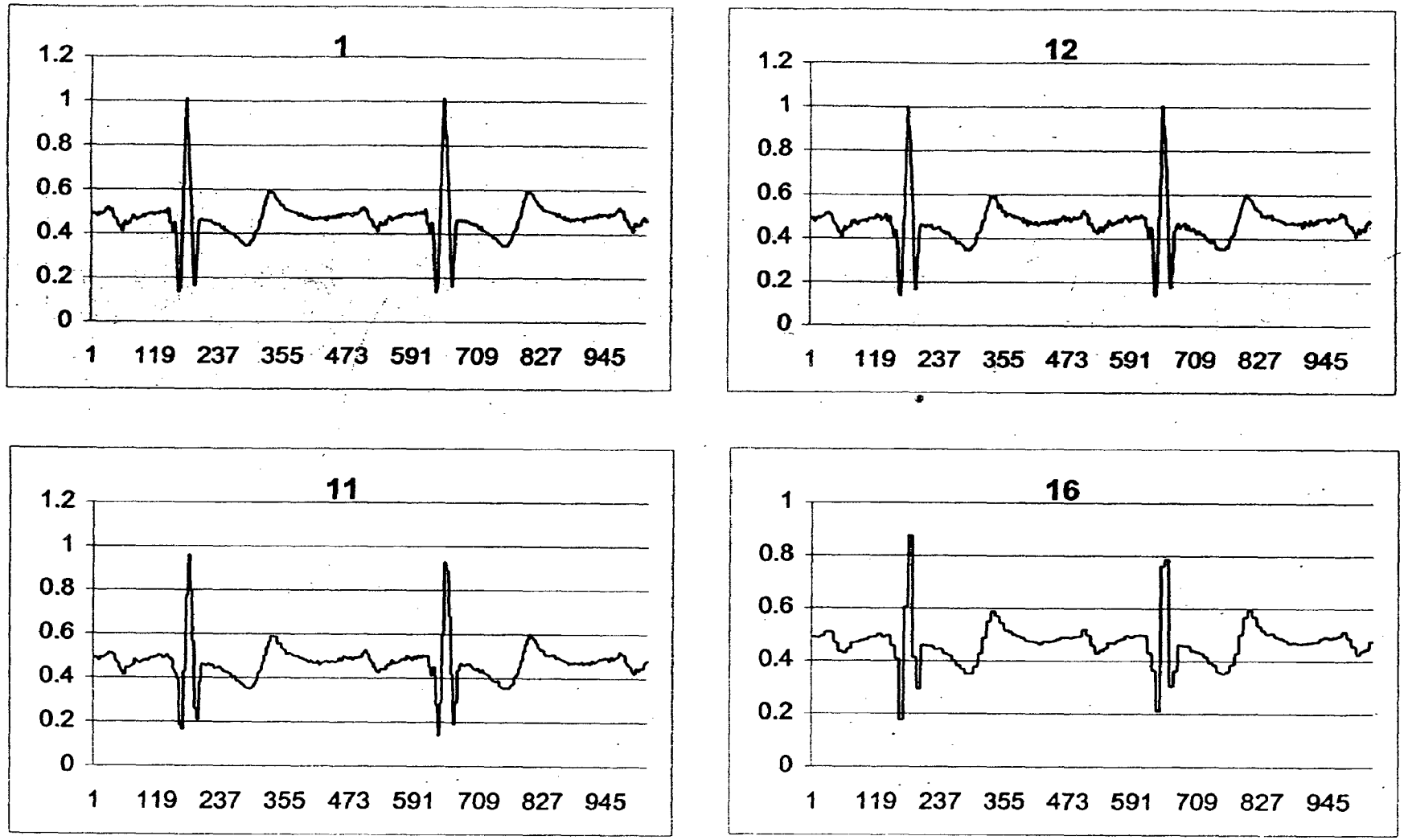
WALSH ANALYSIS OF 1L3 ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

FIG. 4.69 WALSH ANALYSIS OF 1L3 ECG DATA

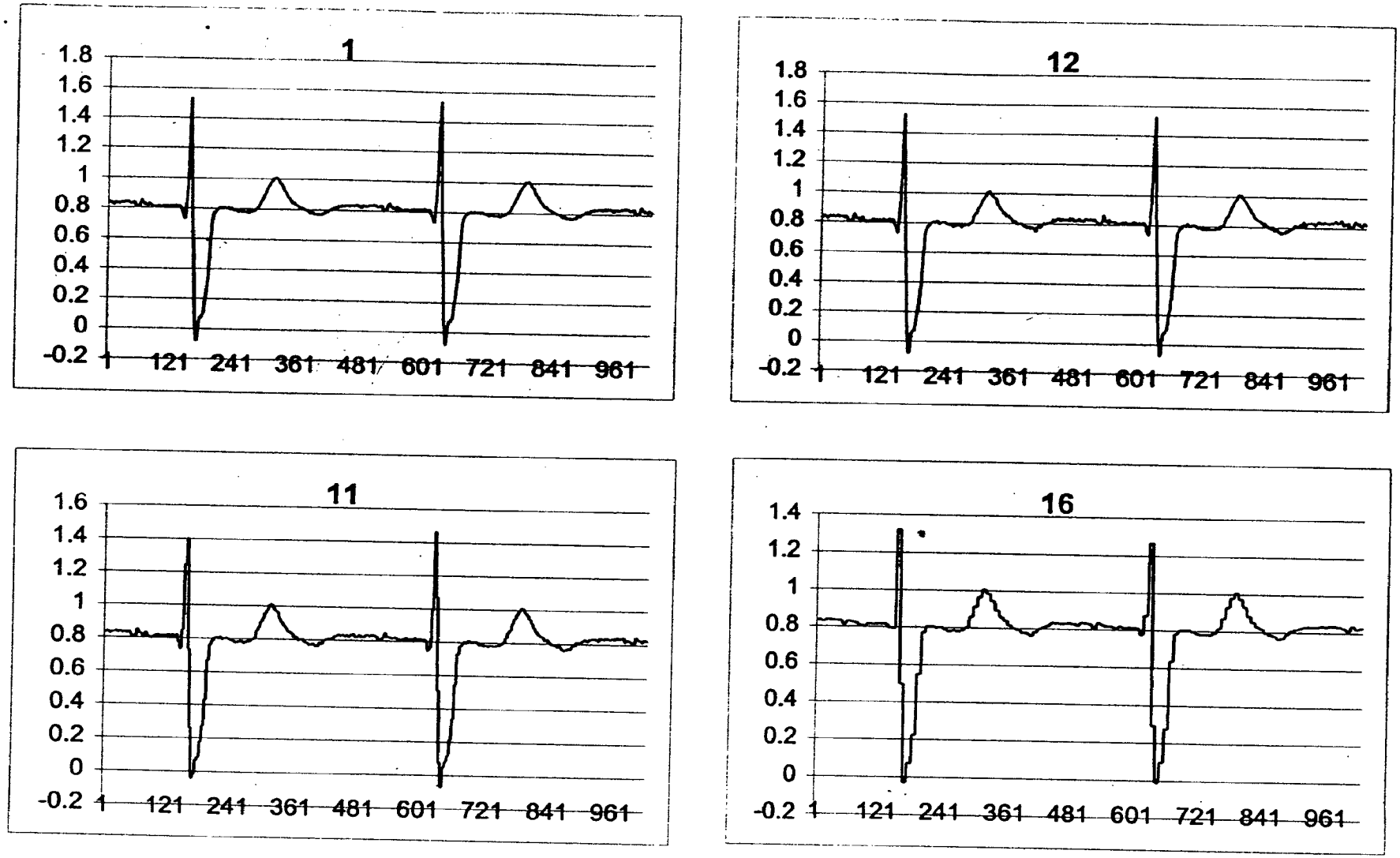
WALSH ANALYSIS OF 1V1 ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

FIG. 4.70 WALSH ANALYSIS OF 1V1 ECG DATA

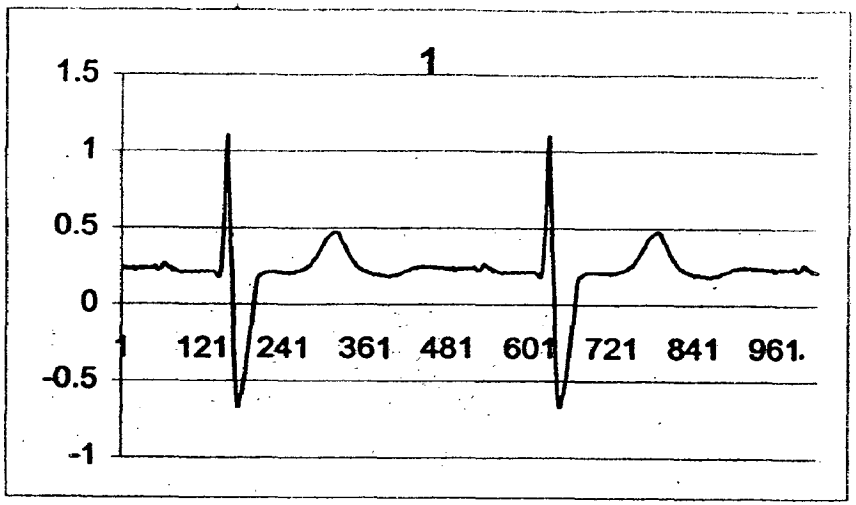
WALSH ANALYSIS OF 1V2 ECG DATA

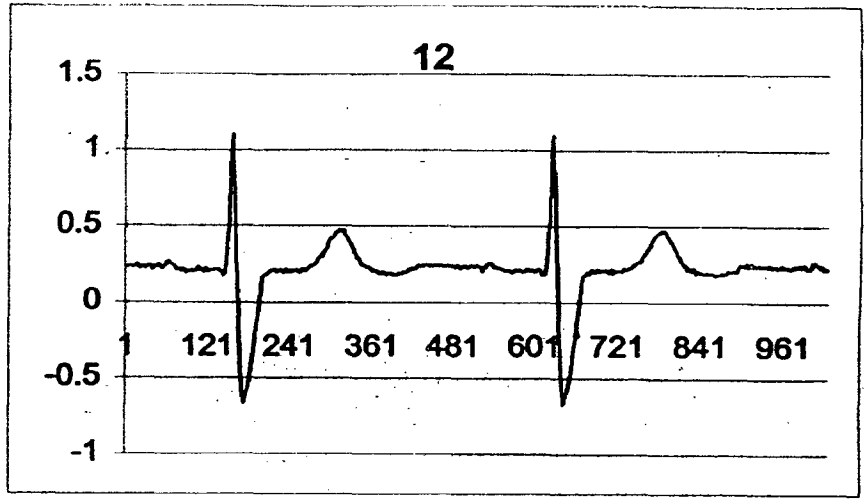


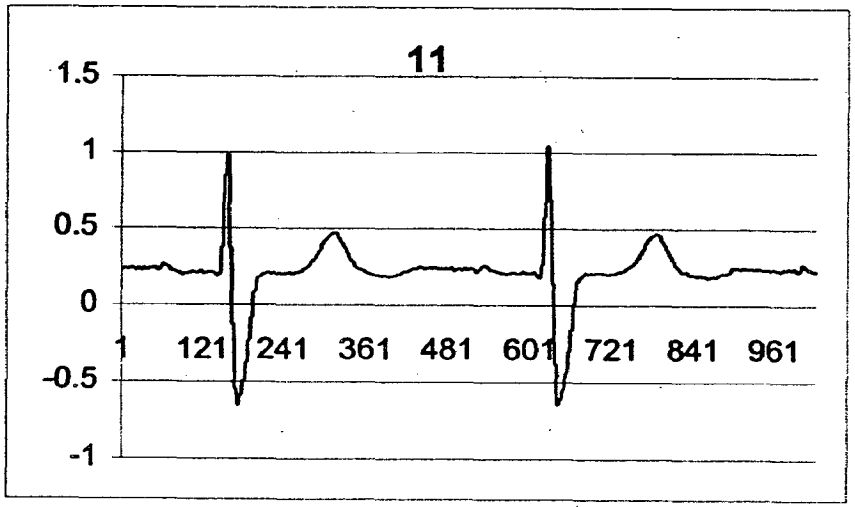
'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

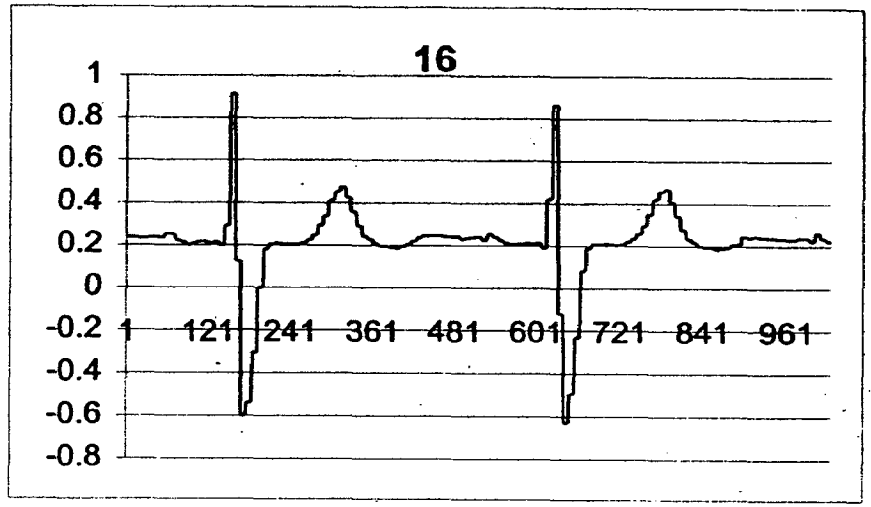
FIG. 4.71 WALSH ANALYSIS OF 1V2 ECG DATA

WALSH ANALYSIS OF 1V3 ECG DATA





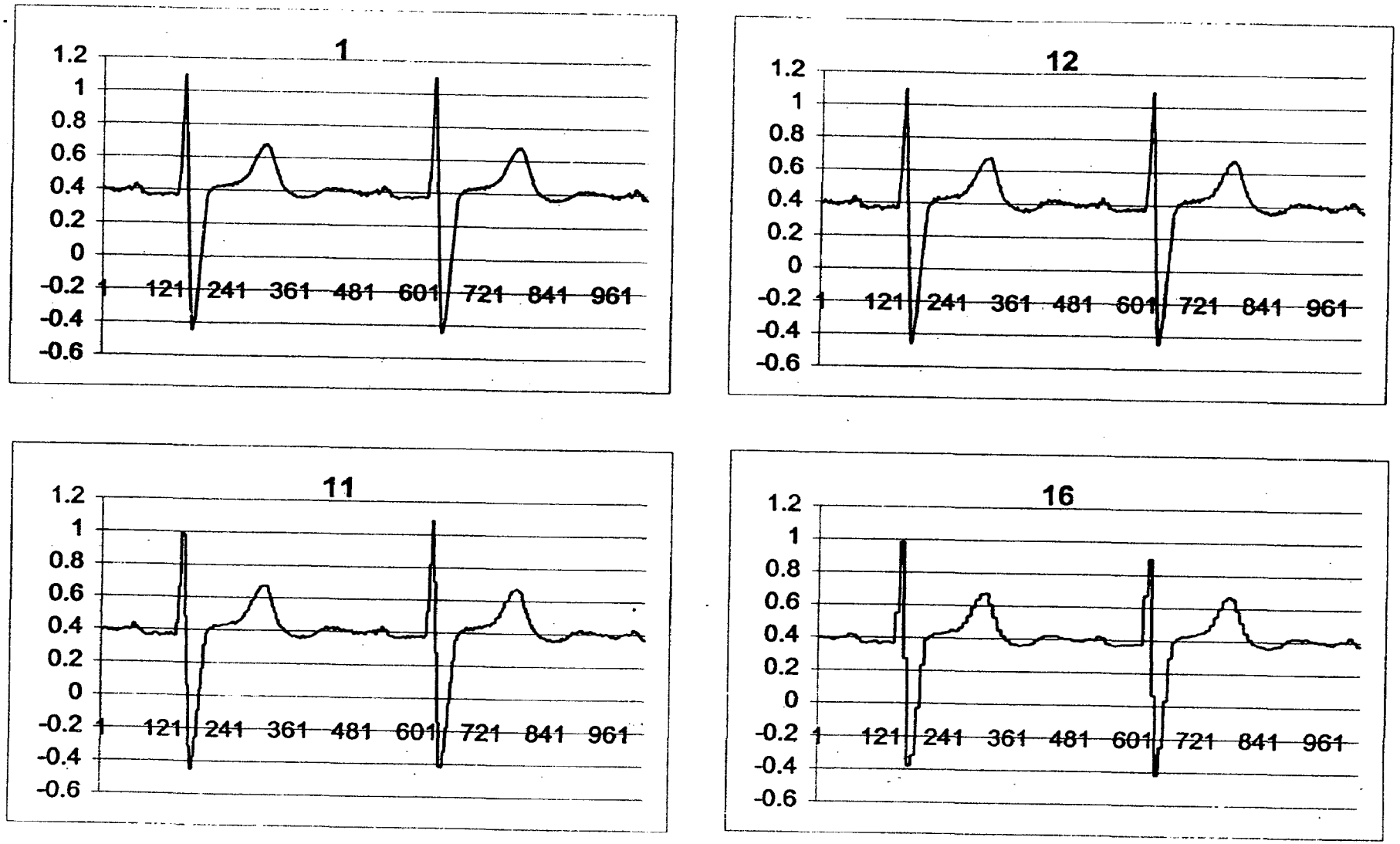




'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

FIG. 4.72 WALSH ANALYSIS OF 1V3 ECG DATA

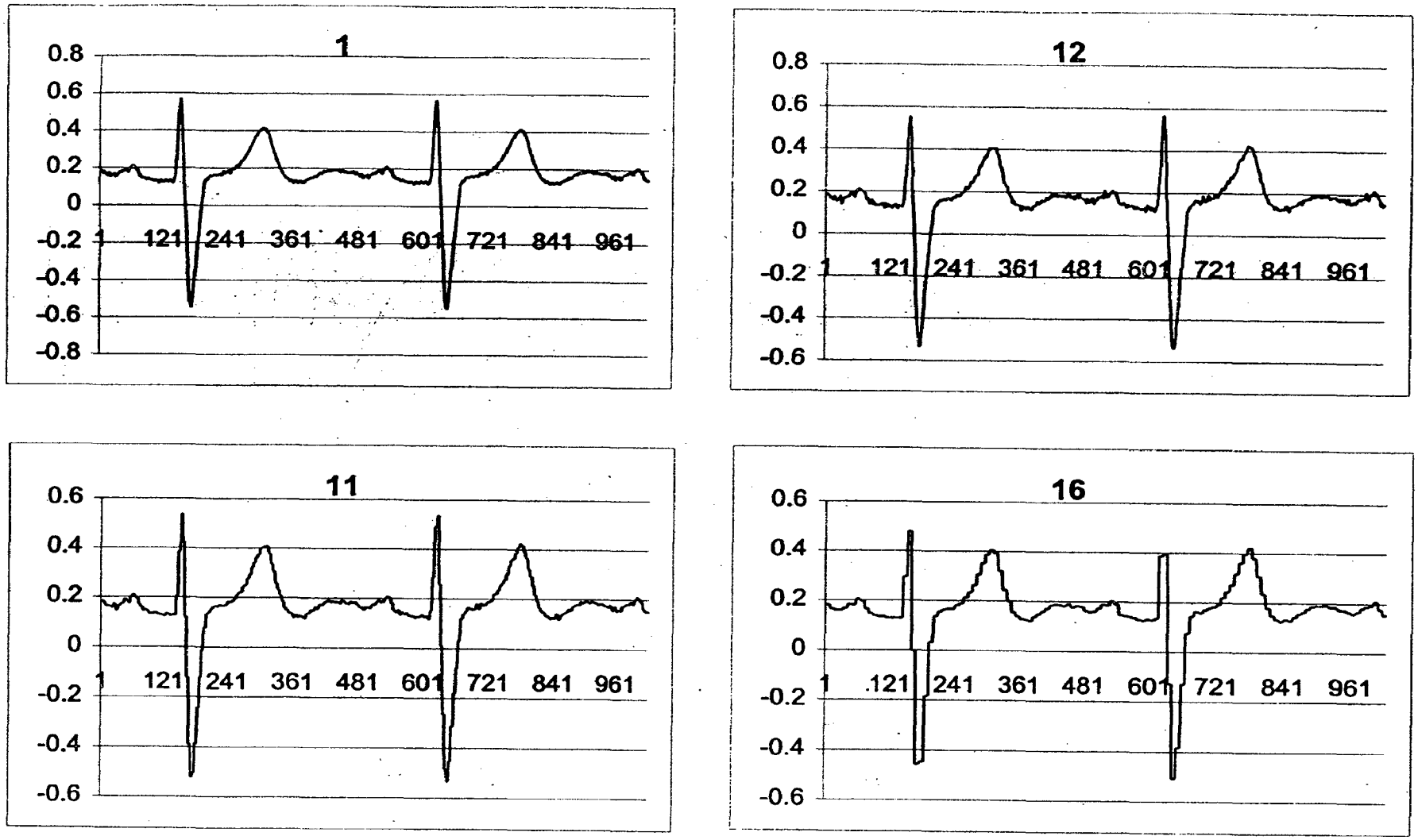
WALSH ANALYSIS OF 1V4 ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

FIG. 4.73 WALSH ANALYSIS OF 1V4 ECG DATA

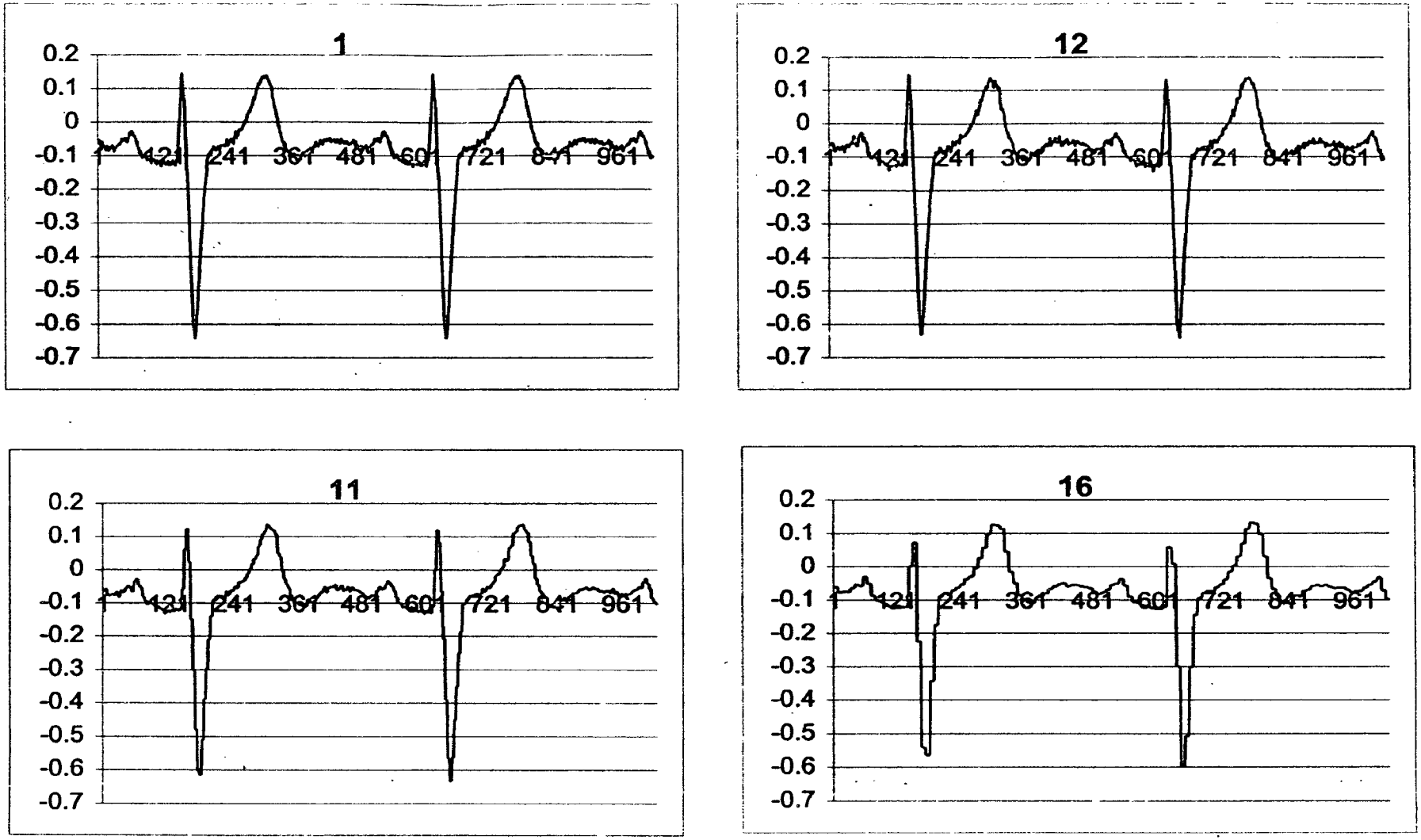
WALSH ANALYSIS OF 1V5 ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

FIG. 4.74 WALSH ANALYSIS OF 1V5 ECG DATA

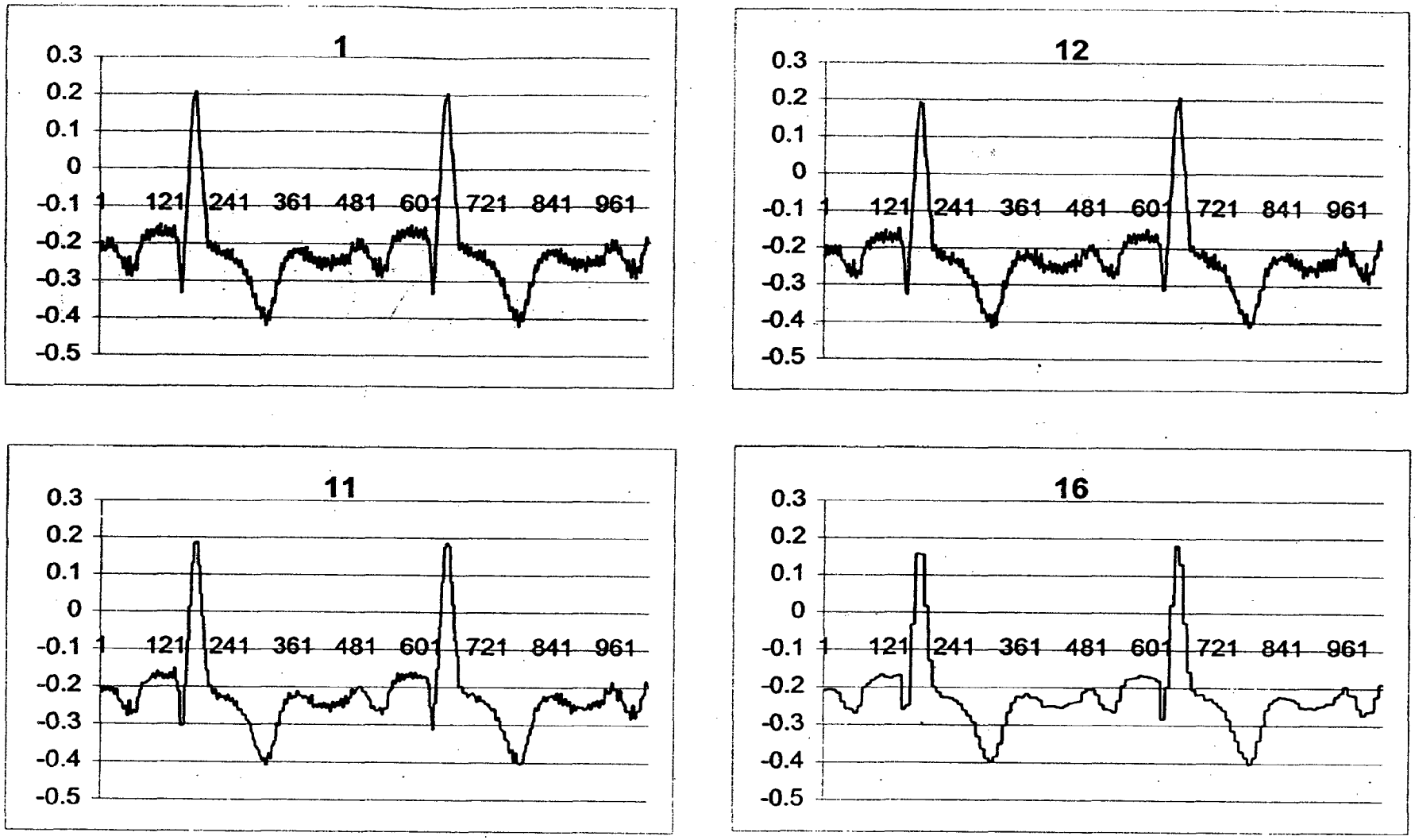
WALSH ANALYSIS OF 1V6 ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

FIG. 4.75 WALSH ANALYSIS OF 1V6 ECG DATA

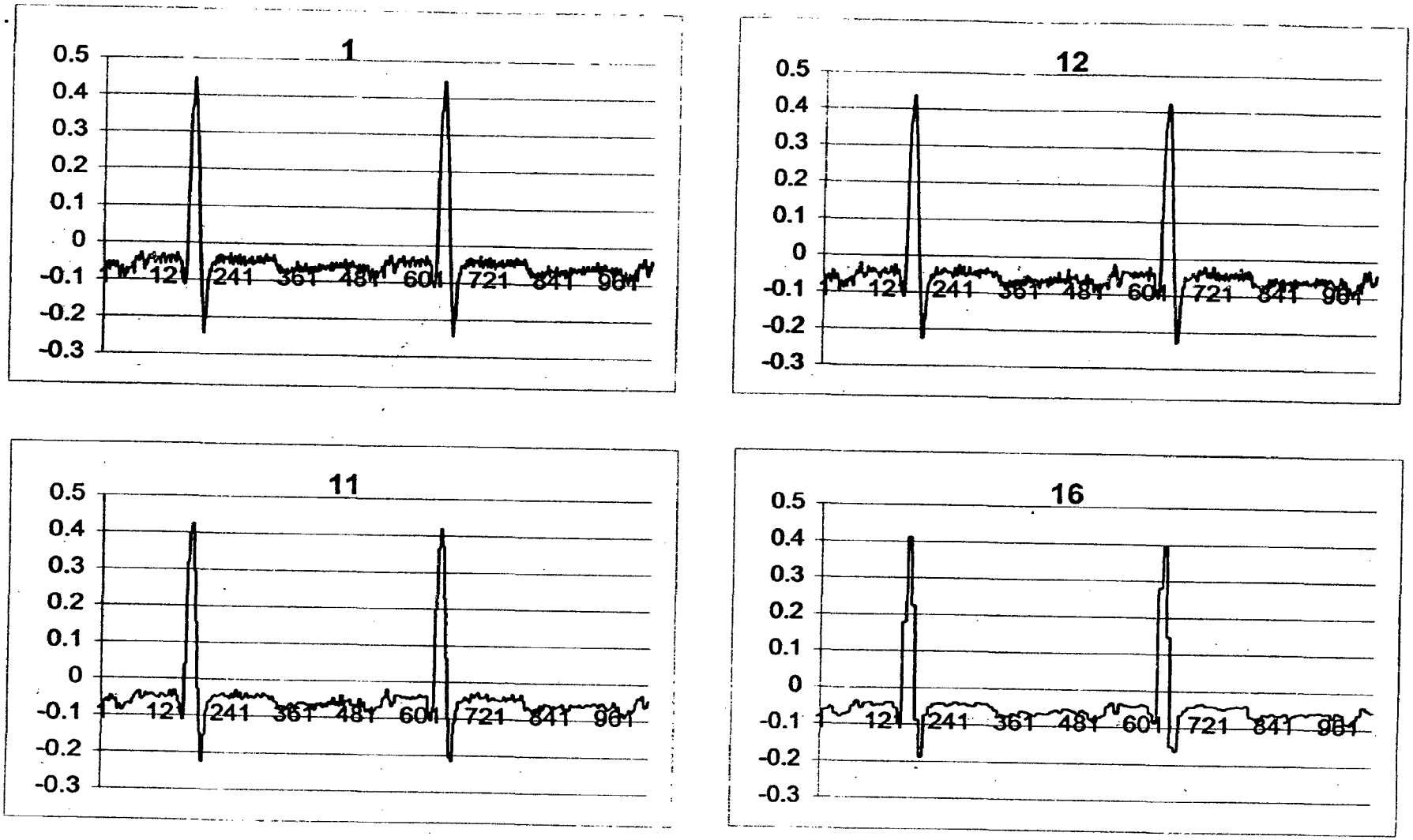
WALSH ANALYSIS OF 1AR ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

FIG. 4.76 WALSH ANALYSIS OF 1AR ECG DATA

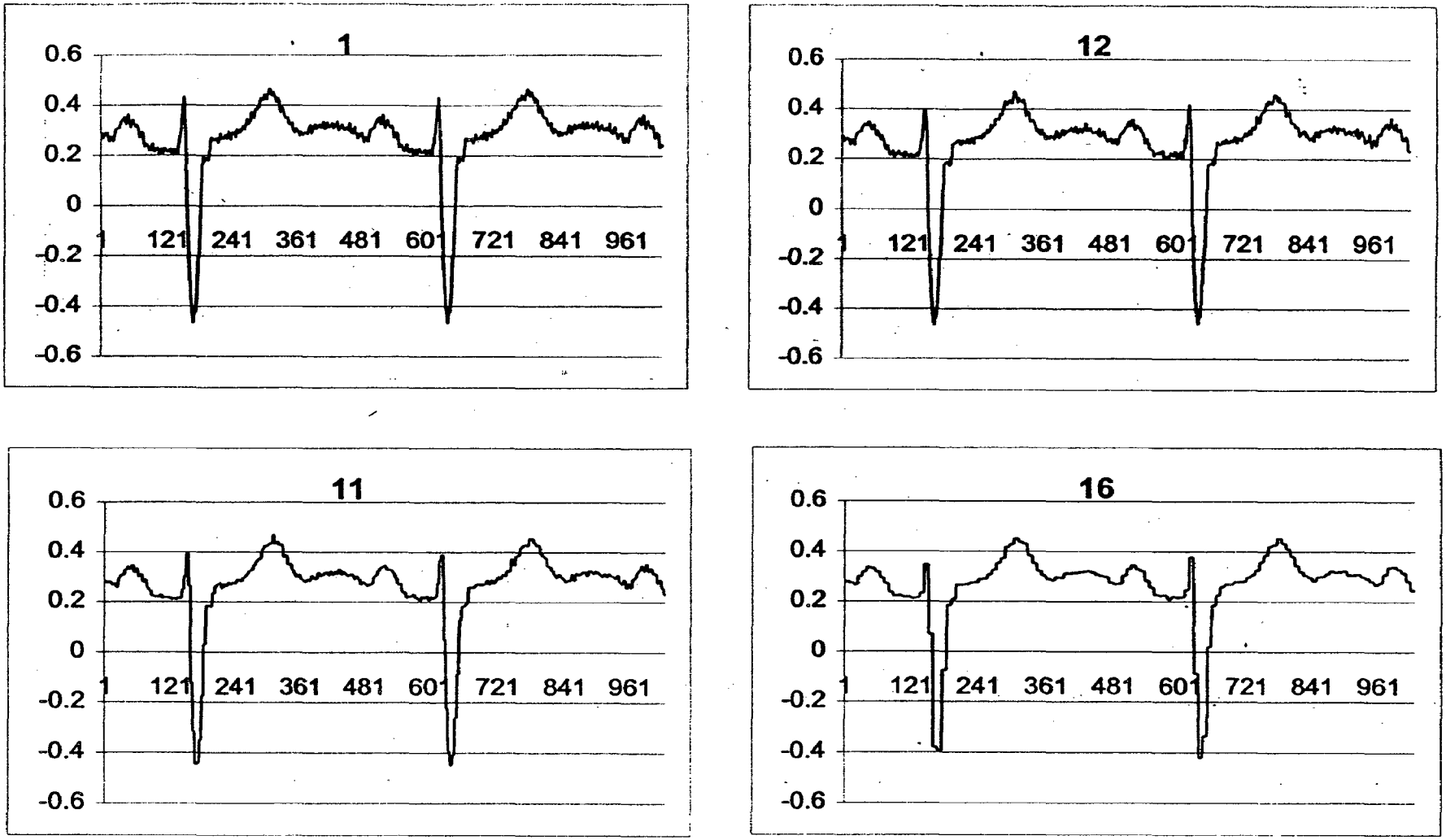
WALSH ANALYSIS OF 1AL ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

FIG. 4.77 WALSH ANALYSIS OF 1AL ECG DATA

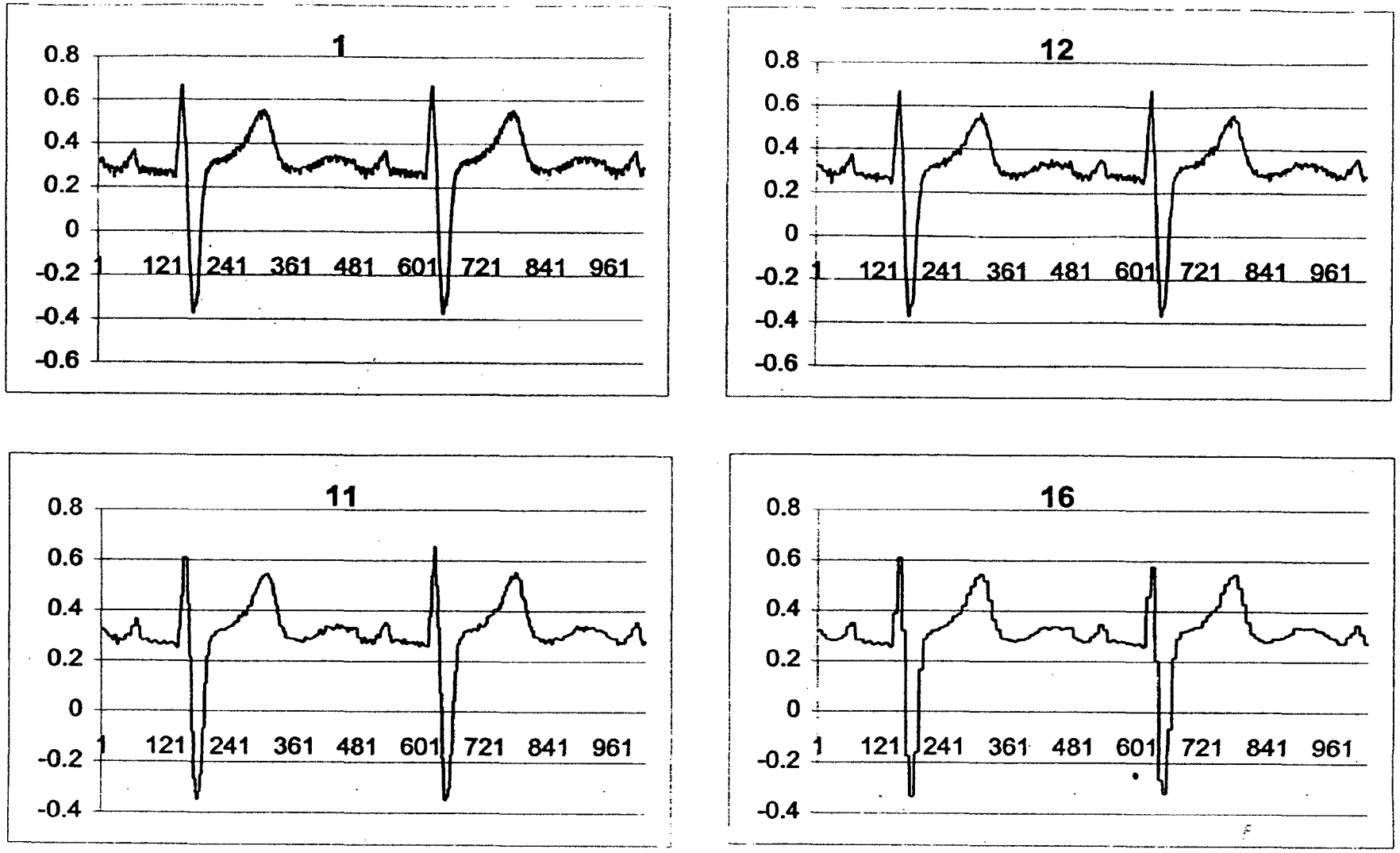
WALSH ANALYSIS OF 1AF ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

FIG. 4.78 WALSH ANALYSIS OF 1AF ECG DATA

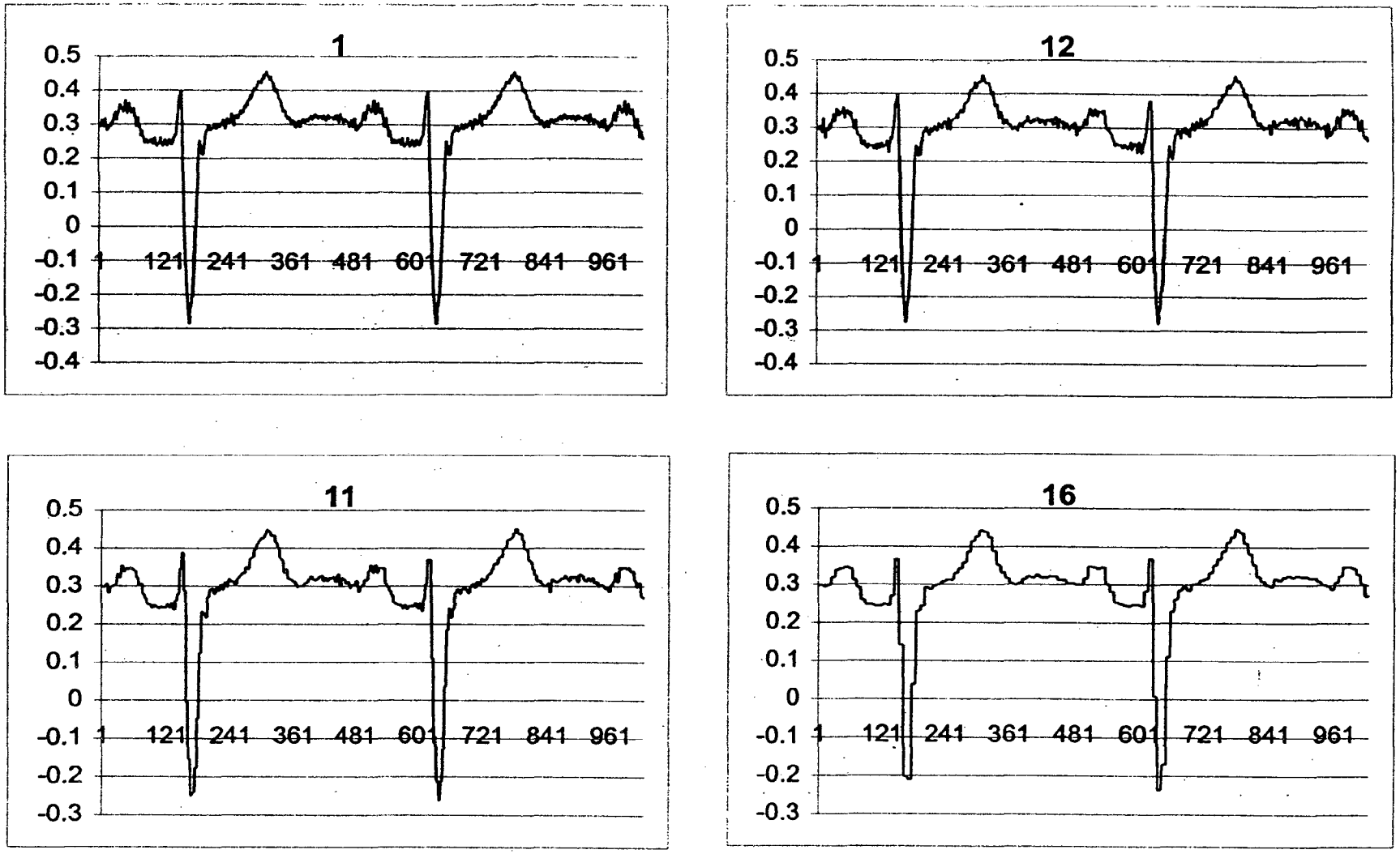
WALSH ANALYSIS OF 1X ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

FIG. 4.79 WALSH ANALYSIS OF 1X ECG DATA

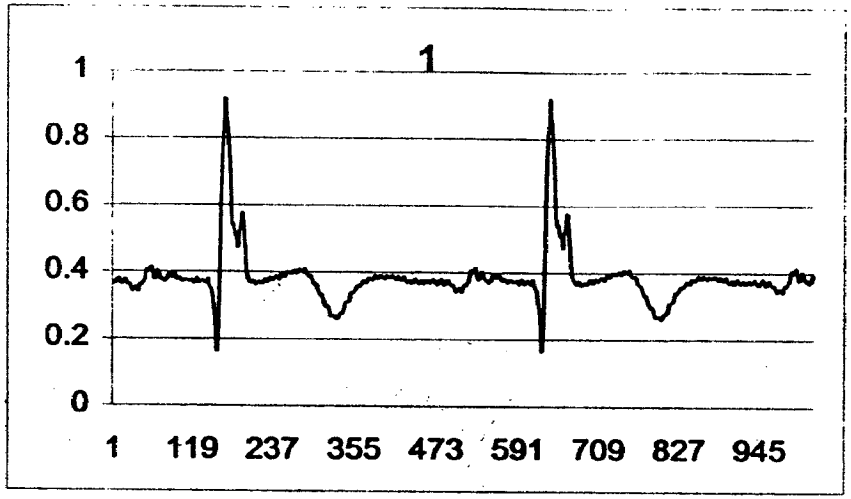
WALSH ANALYSIS OF 1Y ECG DATA

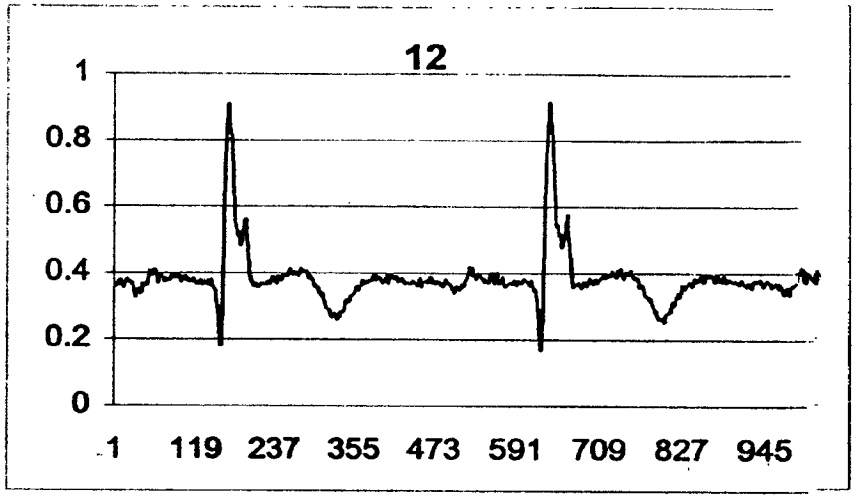


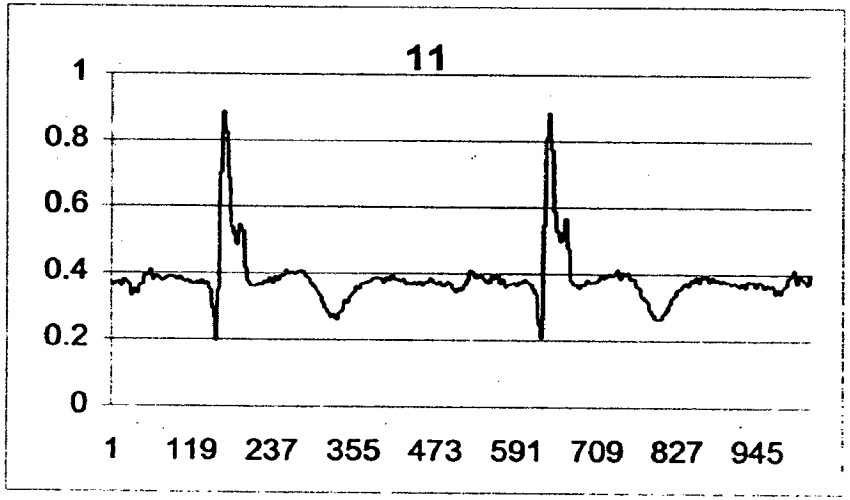
'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

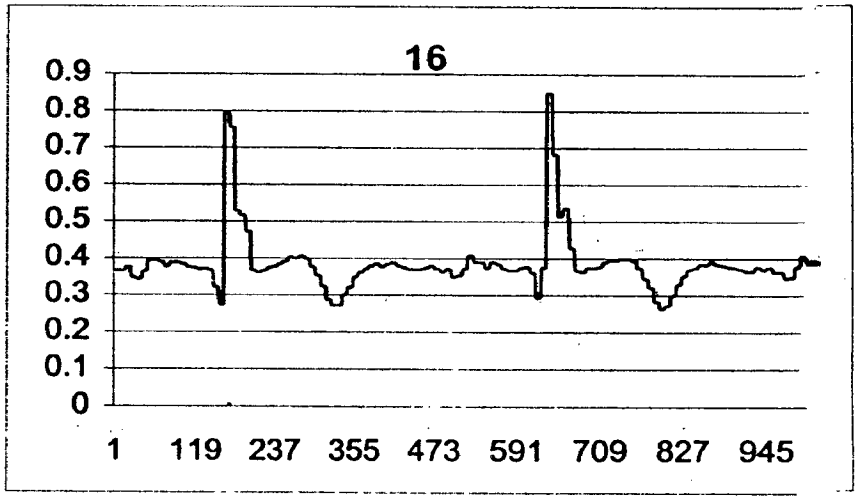
FIG. 4.80 WALSH ANALYSIS OF 1Y ECG DATA

WALSH ANALYSIS OF 1Z ECG DATA





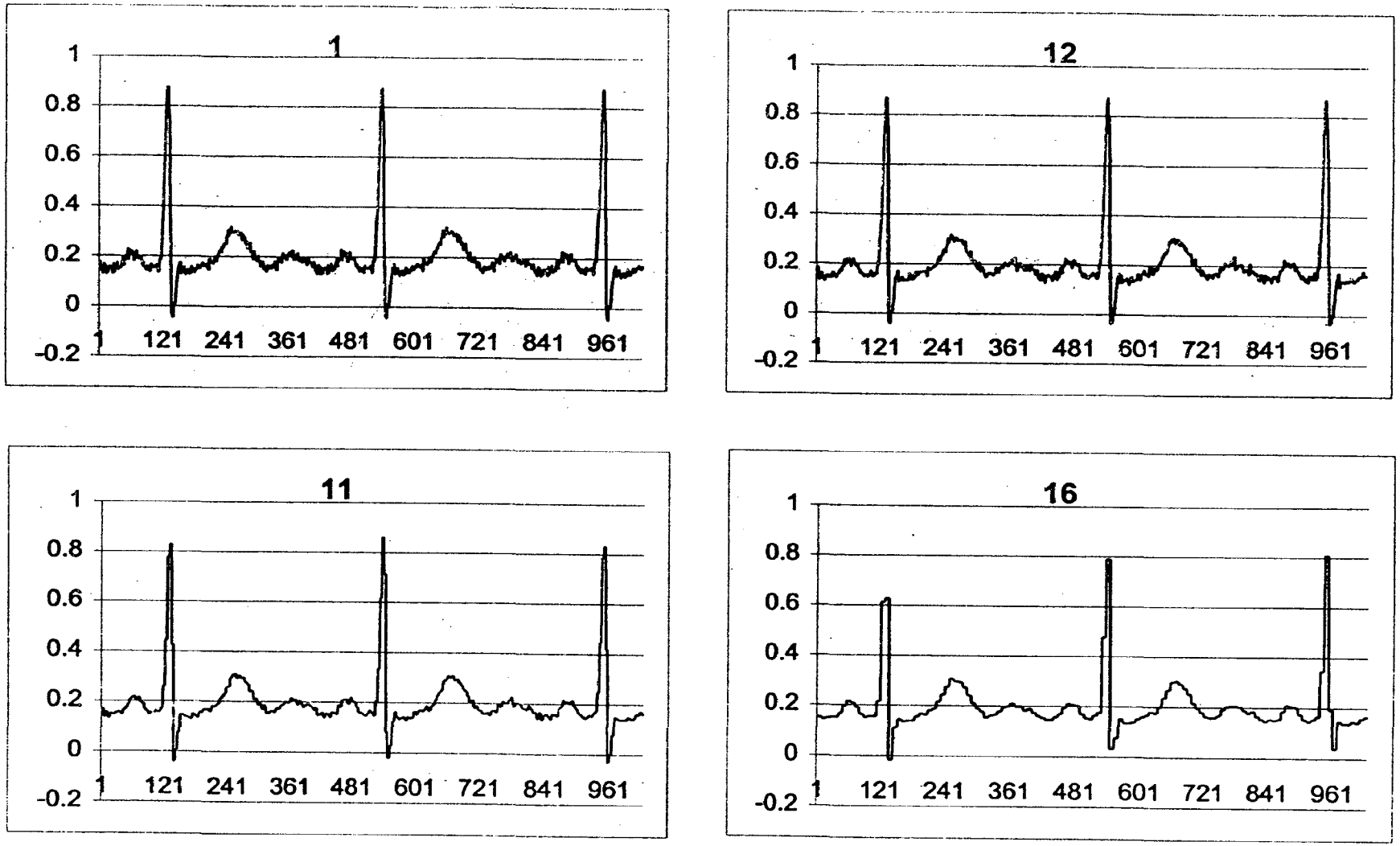




'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

FIG. 4.81 WALSH ANALYSIS OF 1Z ECG DATA

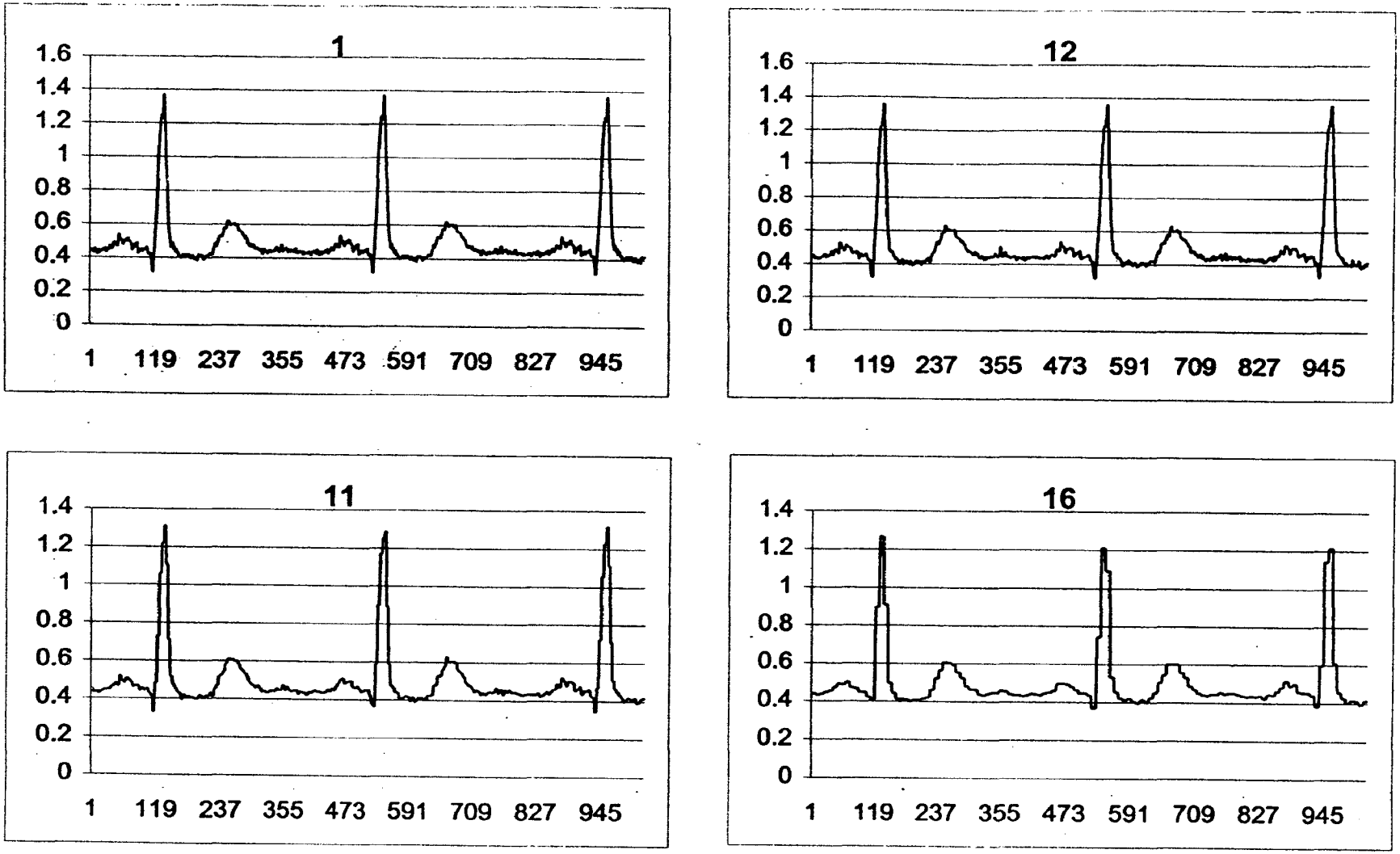
WALSH ANALYSIS OF 4L1 ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

FIG. 4.82 WALSH ANALYSIS OF 4L1 ECG DATA

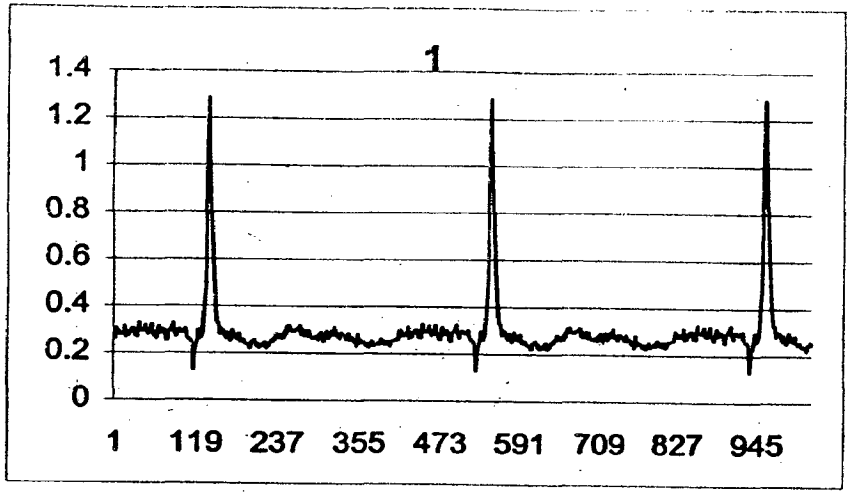
WALSH ANALYSIS OF 4L2 ECG DATA

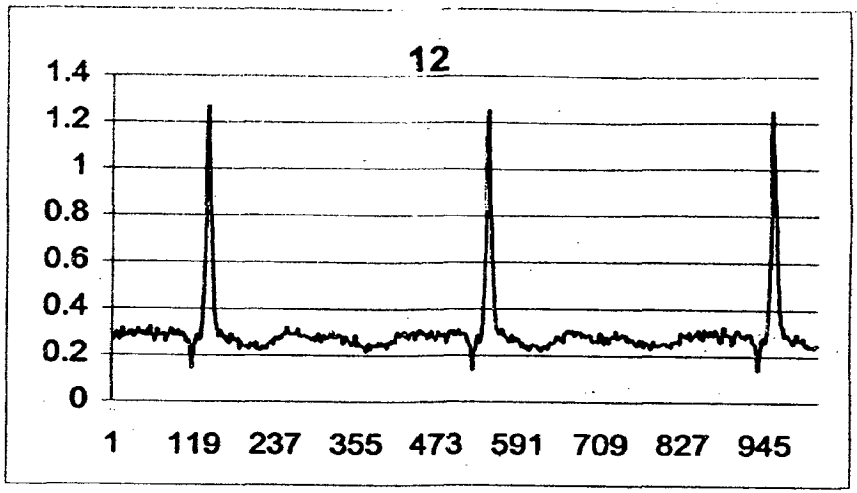


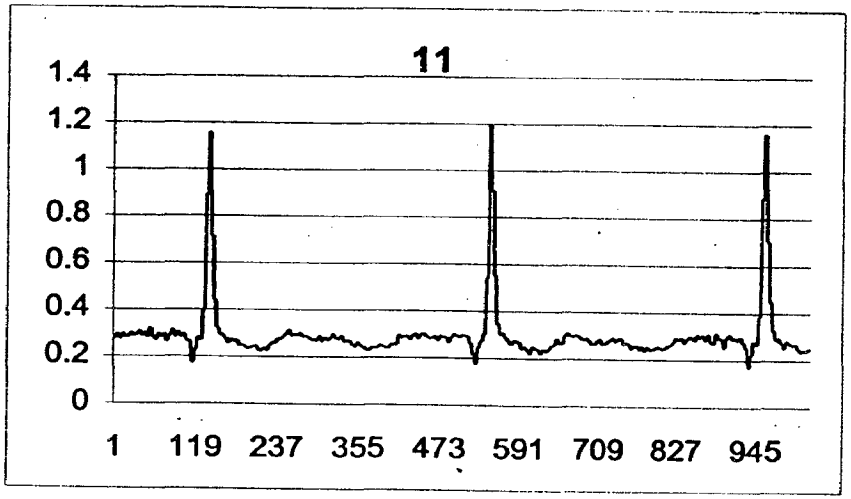
'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

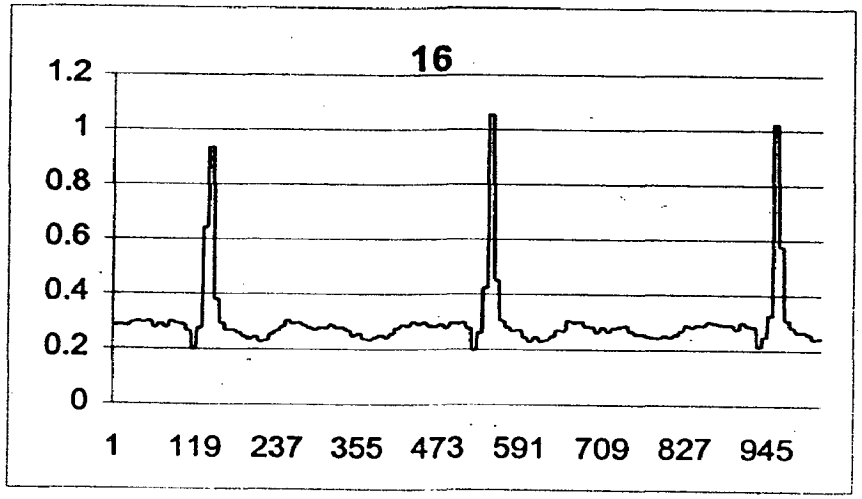
FIG. 4.83 WALSH ANALYSIS OF 4L2 ECG DATA

WALSH ANALYSIS OF 4L3 ECG DATA





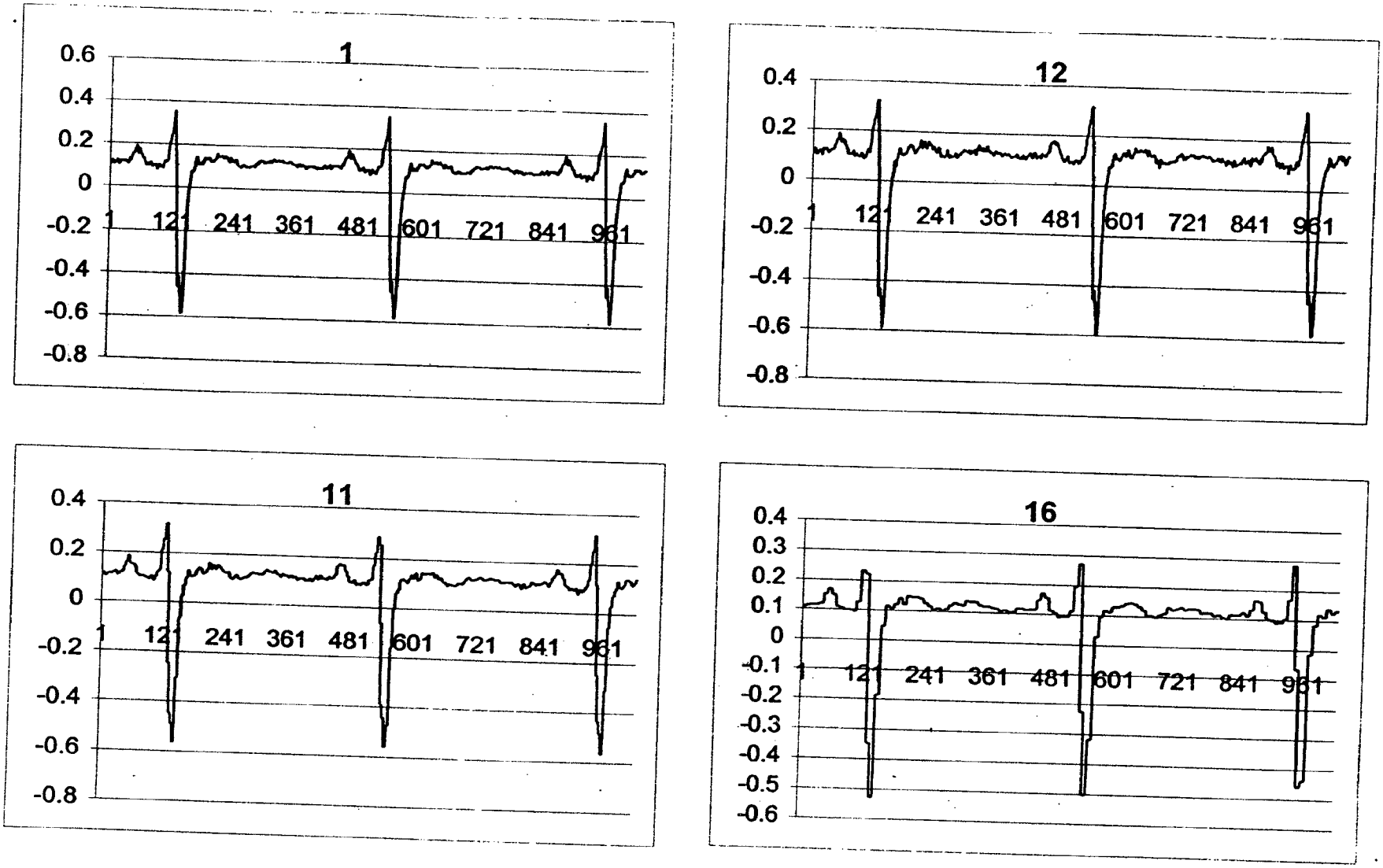




'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

FIG. 4.84 WALSH ANALYSIS OF 4L3 ECG DATA

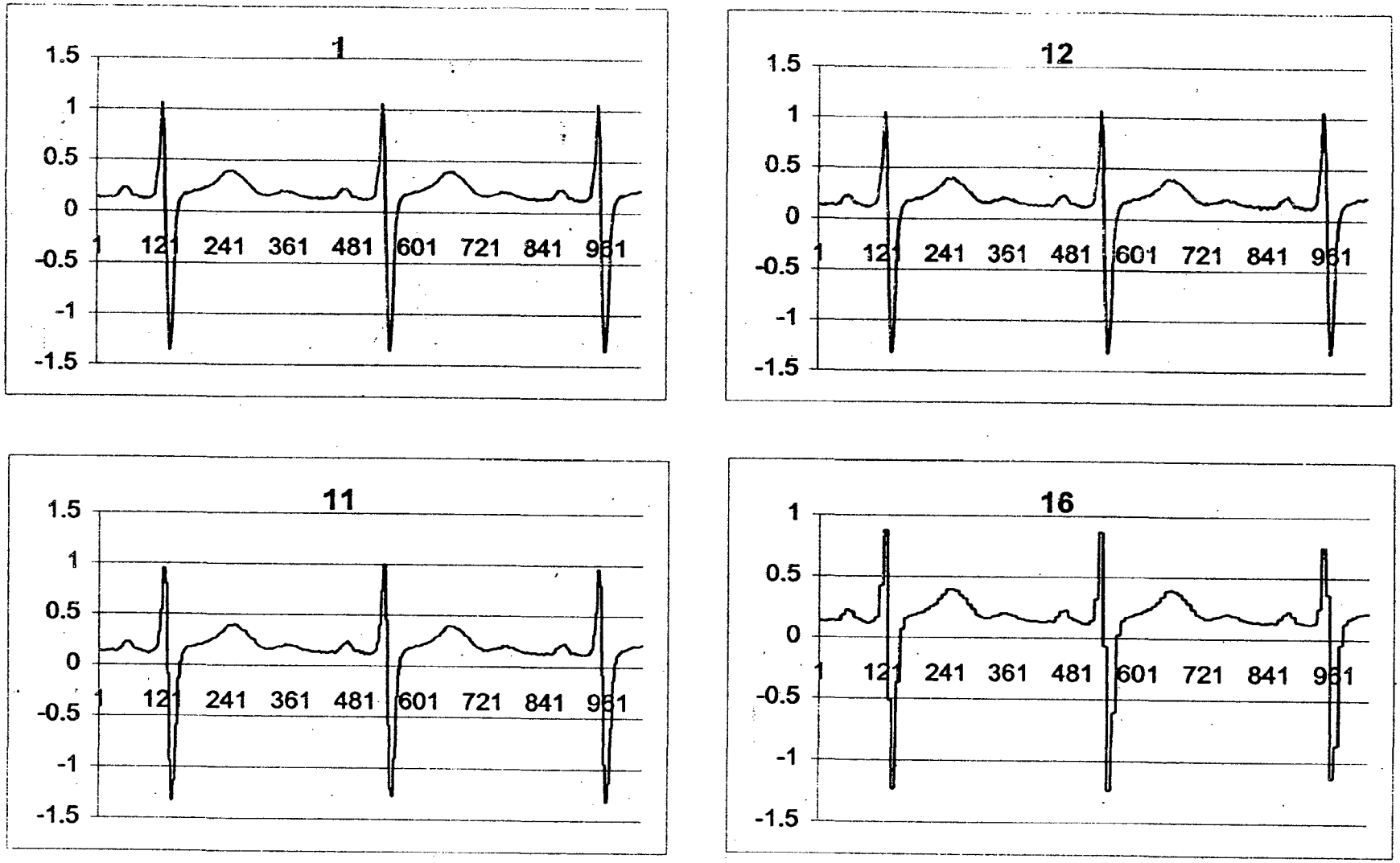
WALSH ANALYSIS OF 4V1 ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

FIG. 4.85 WALSH ANALYSIS OF 4V1 ECG DATA

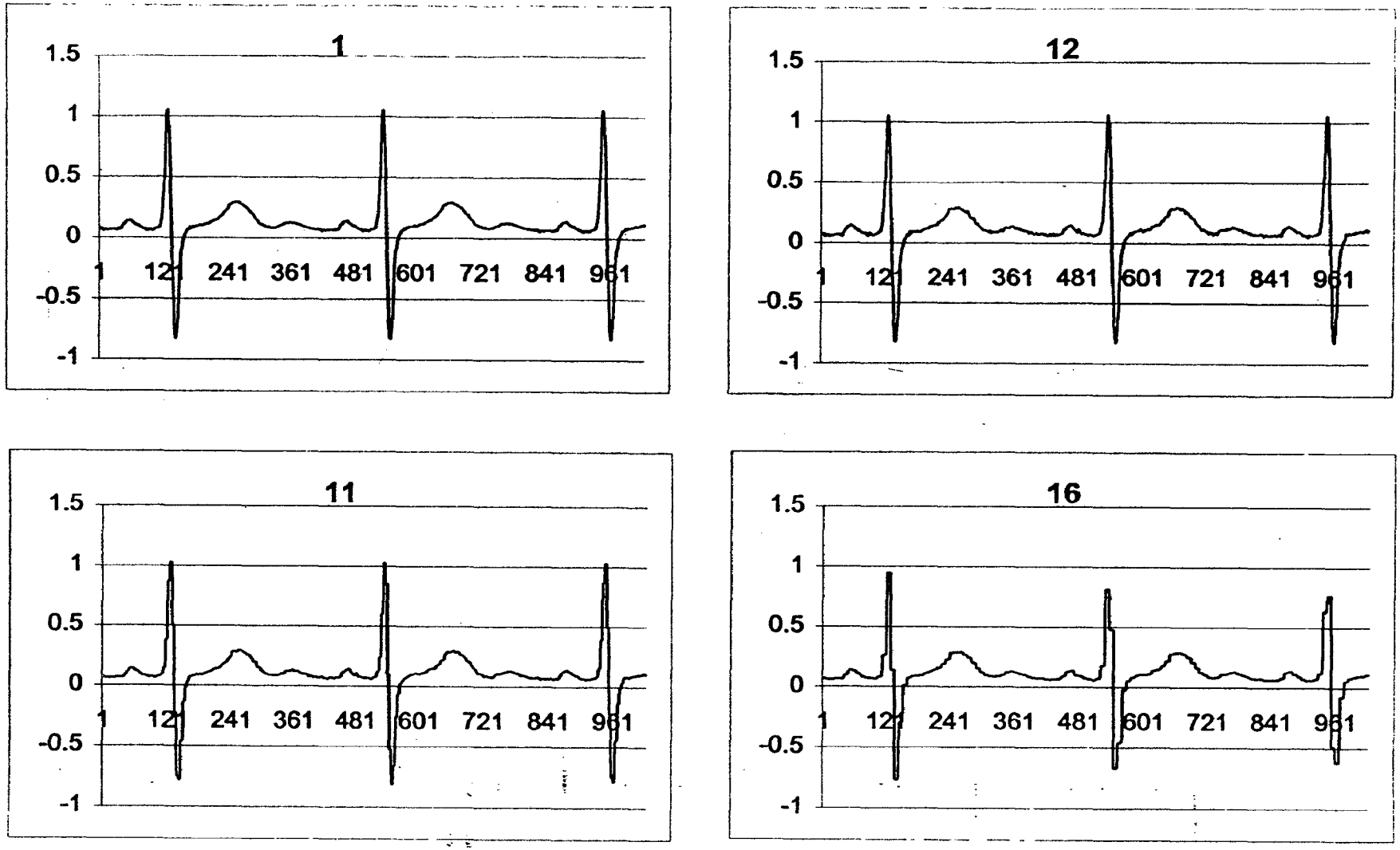
WALSH ANALYSIS OF 4V2 ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

FIG. 4.86 WALSH ANALYSIS OF 4V2 ECG DATA

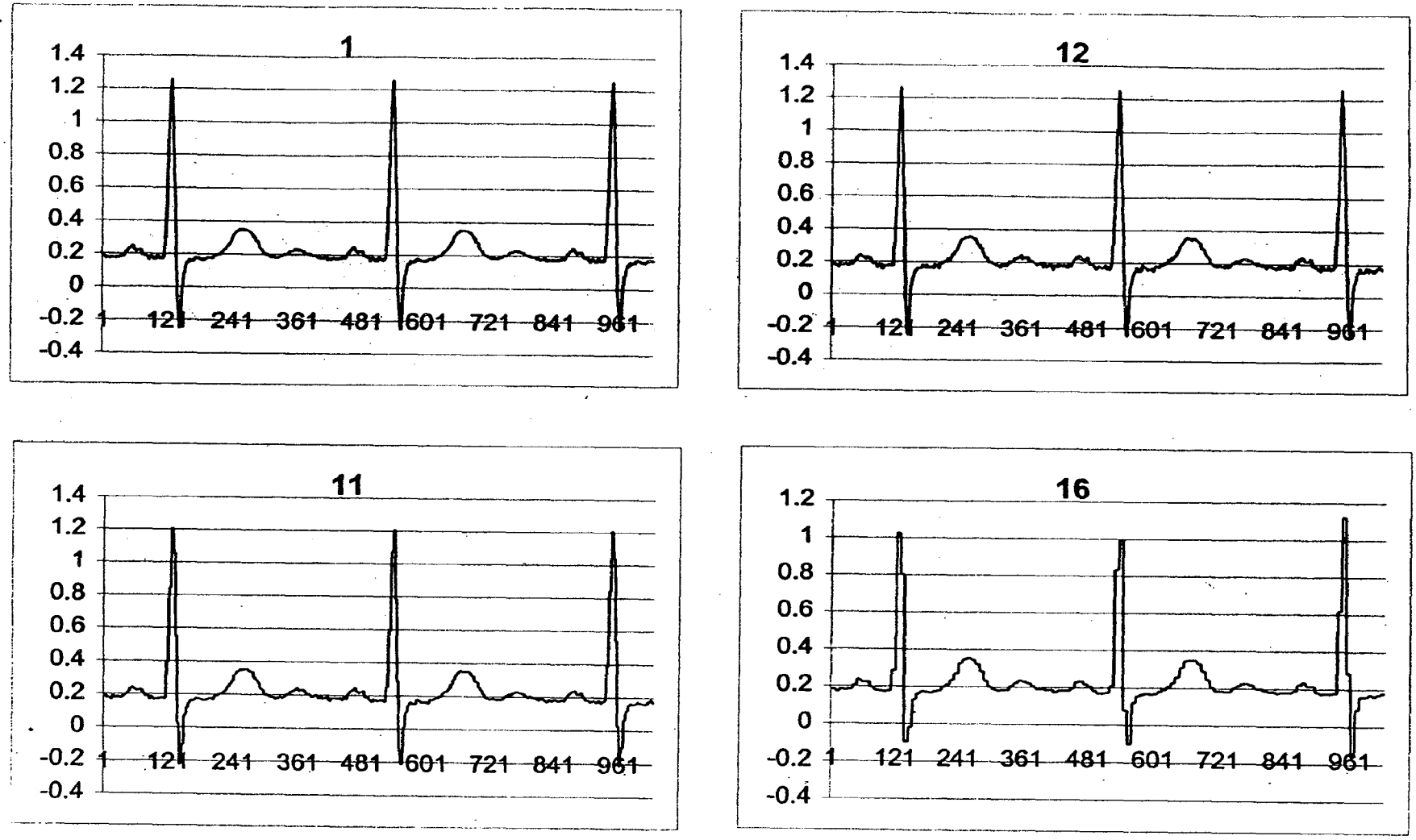
WALSH ANALYSIS OF 4V3 ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

FIG. 4.87 WALSH ANALYSIS OF 4V3 ECG DATA

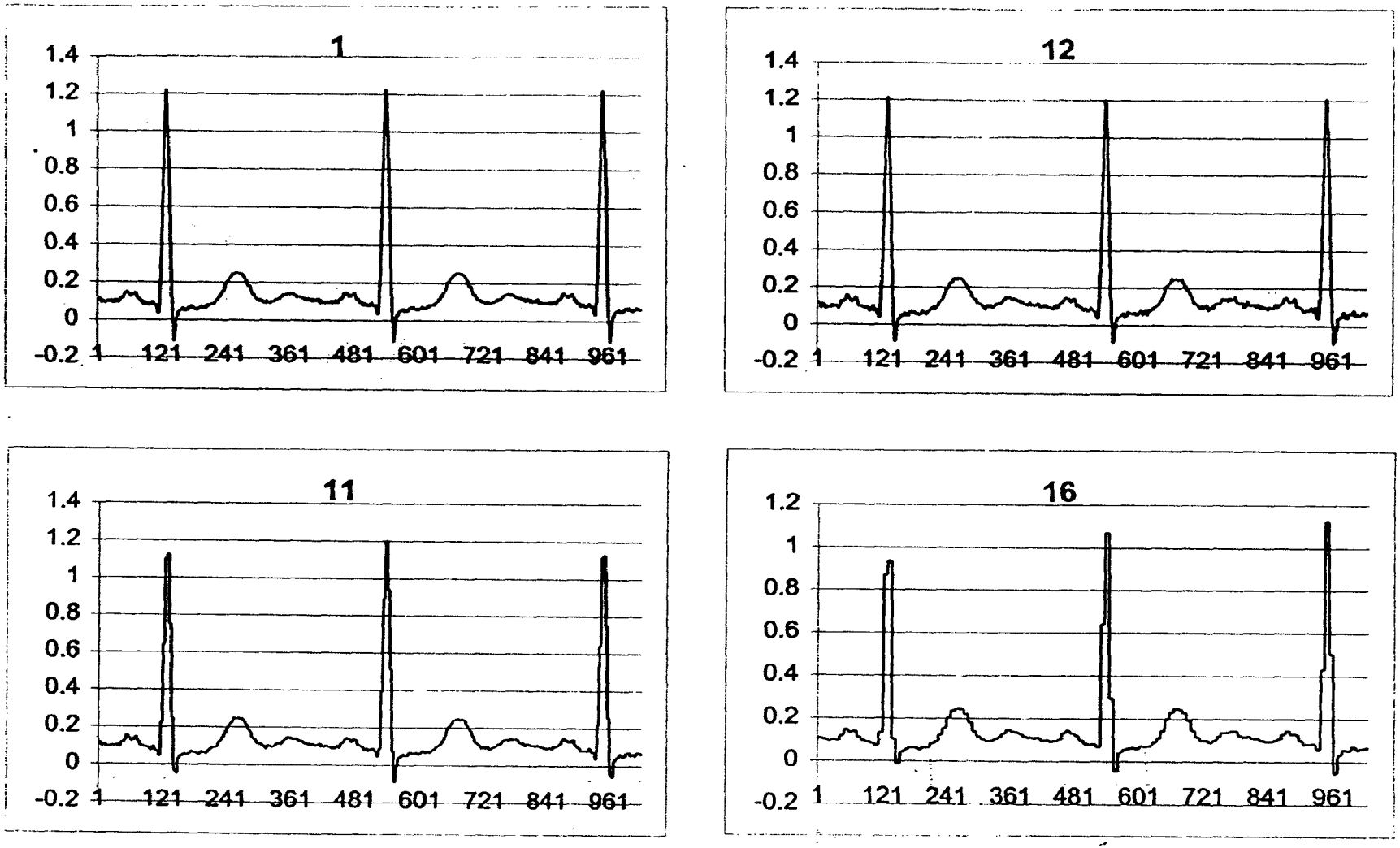
WALSH ANALYSIS OF 4V4 ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

FIG. 4.88 WALSH ANALYSIS OF 4V4 ECG DATA

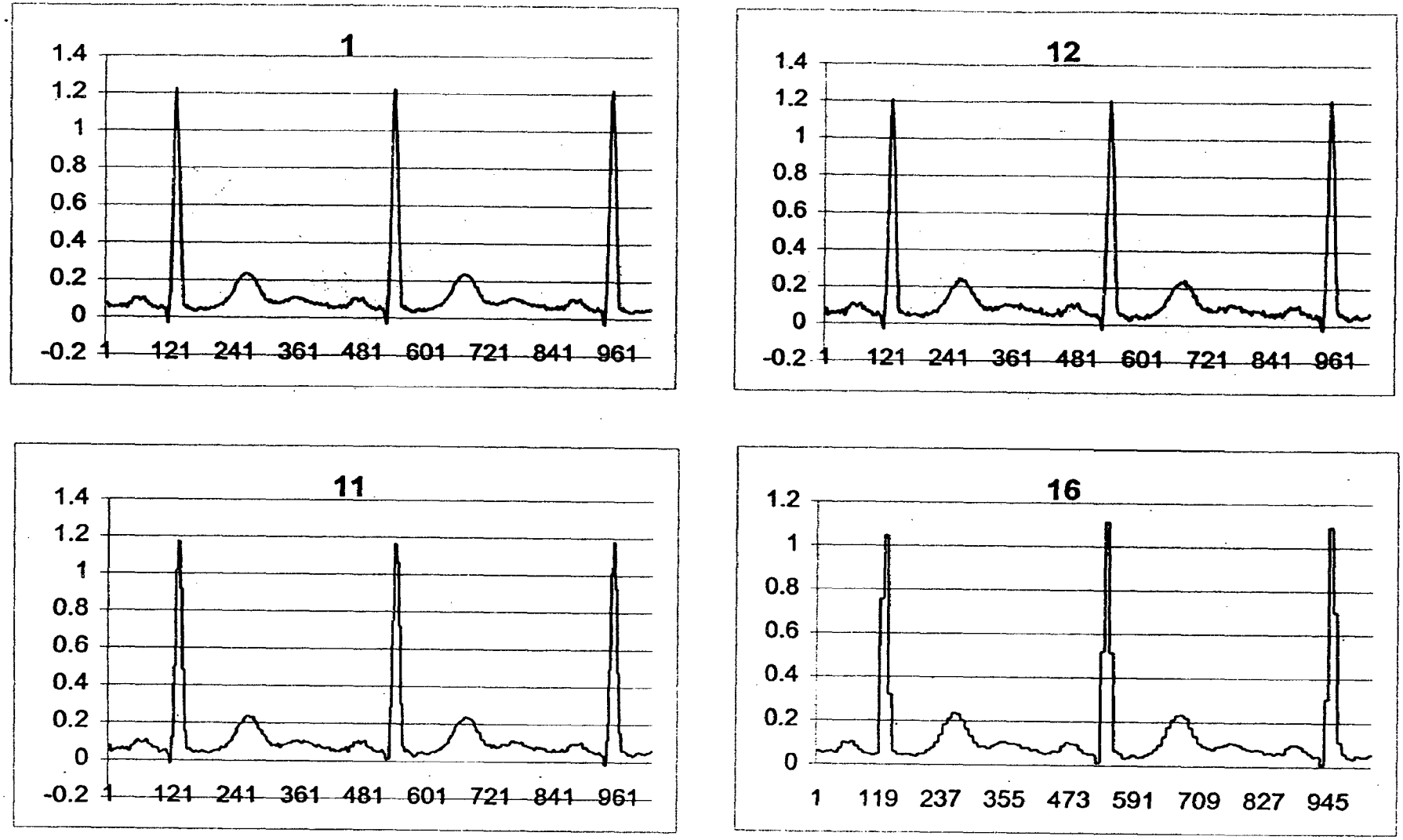
WALSH ANALYSIS OF 4V5 ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

FIG. 4.89 WALSH ANALYSIS OF 4V5 ECG DATA

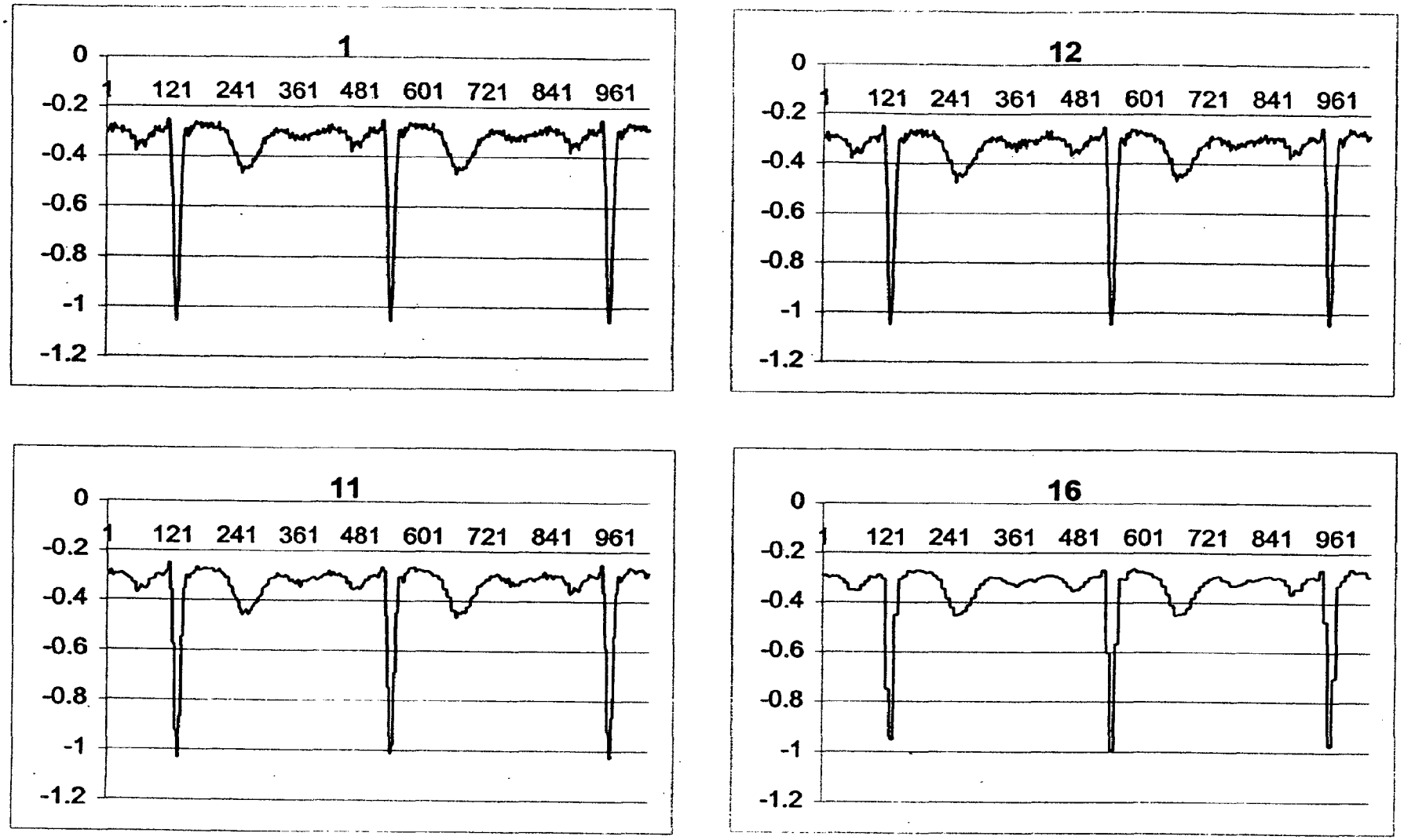
WALSH ANALYSIS OF 4V6 ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

FIG. 4.90 WALSH ANALYSIS OF 4V6 ECG DATA

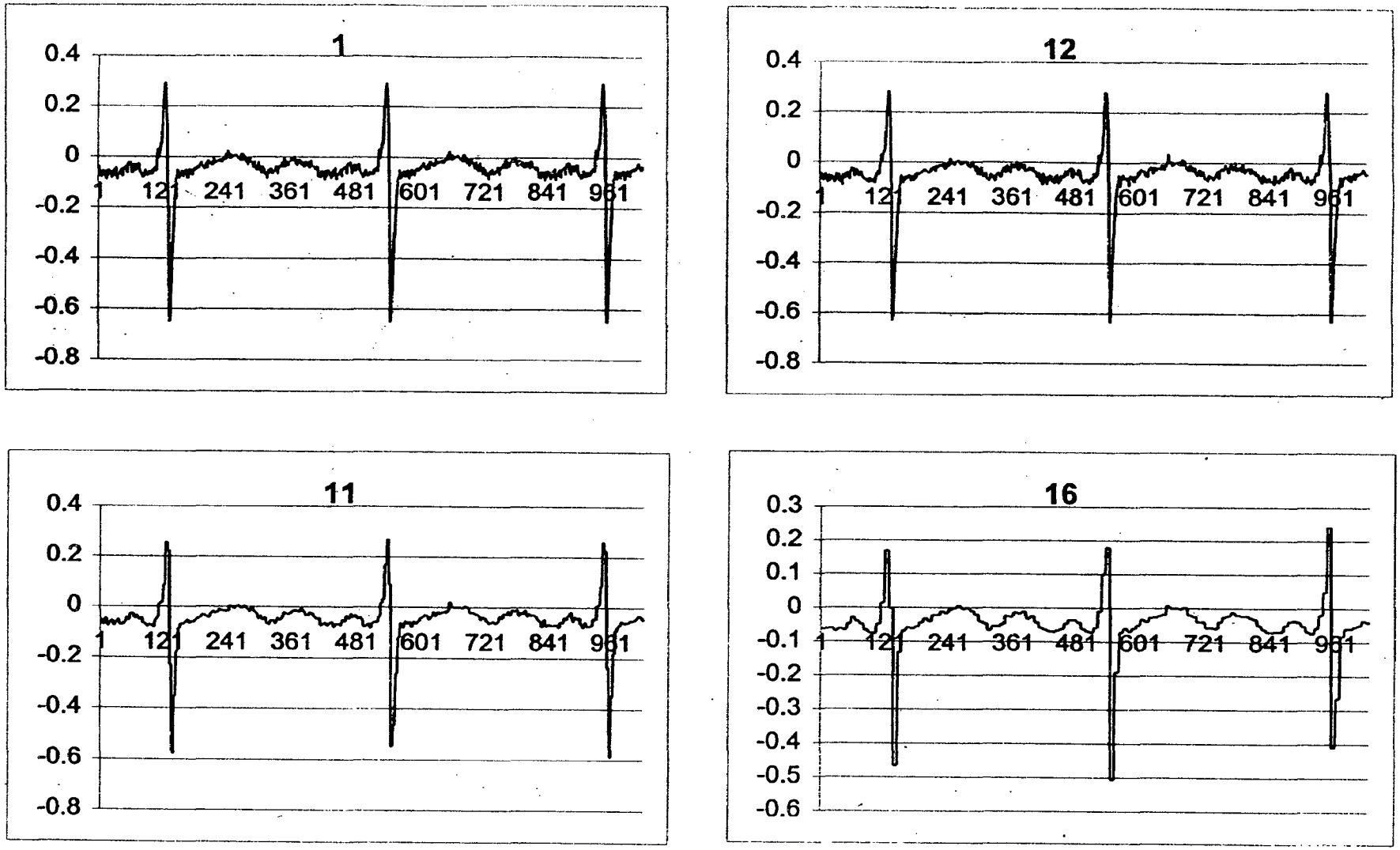
WALSH ANALYSIS OF 4AR ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

FIG. 4.91 WALSH ANALYSIS OF 4AR ECG DATA

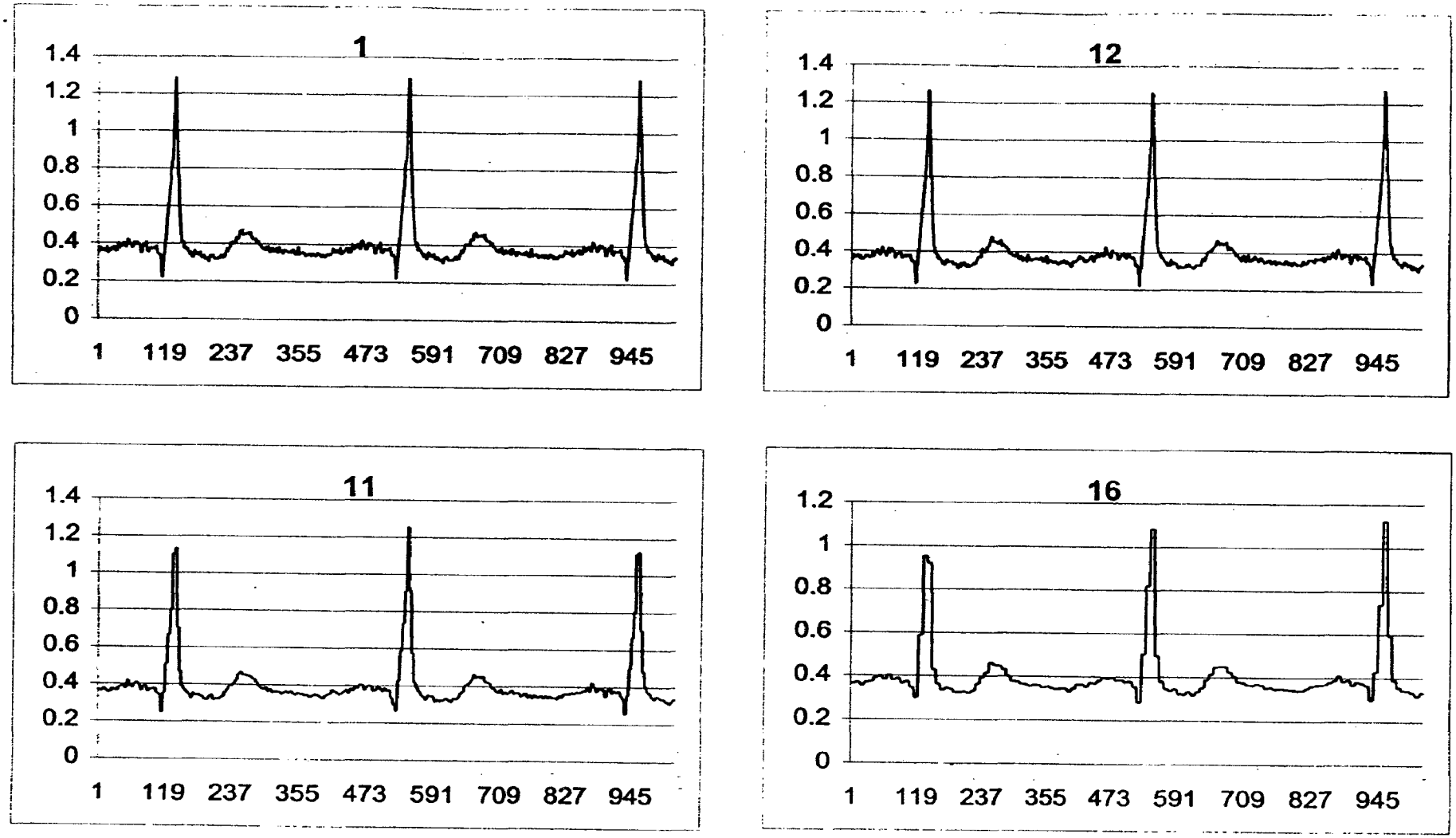
WALSH ANALYSIS OF 4AL ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

FIG. 4.92 WALSH ANALYSIS OF 4AL ECG DATA

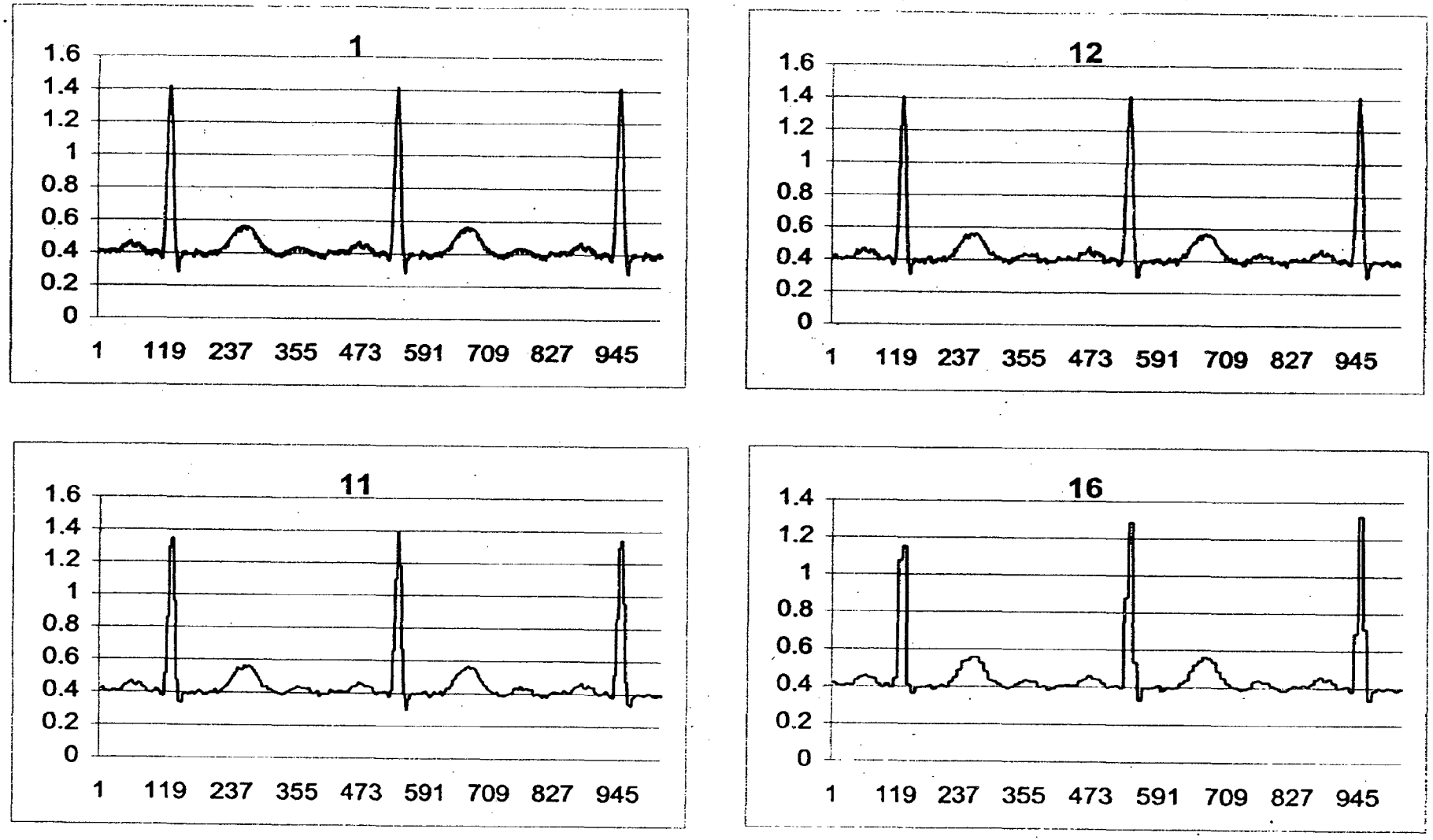
WALSH ANALYSIS OF 4AF ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

FIG. 4.93 WALSH ANALYSIS OF 4AF ECG DATA

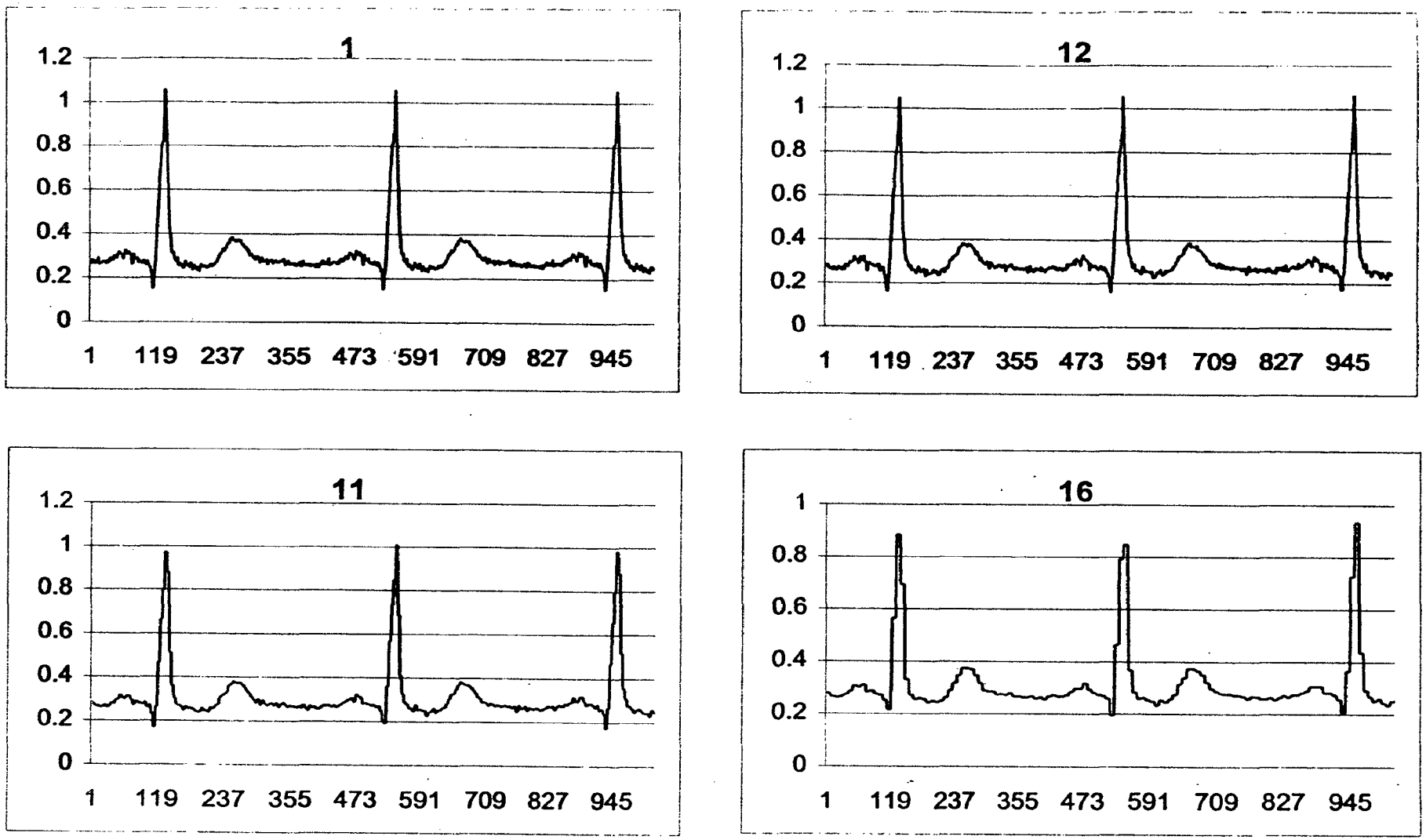
WALSH ANALYSIS OF 4X ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

FIG. 4.94 WALSH ANALYSIS OF 4X ECG DATA

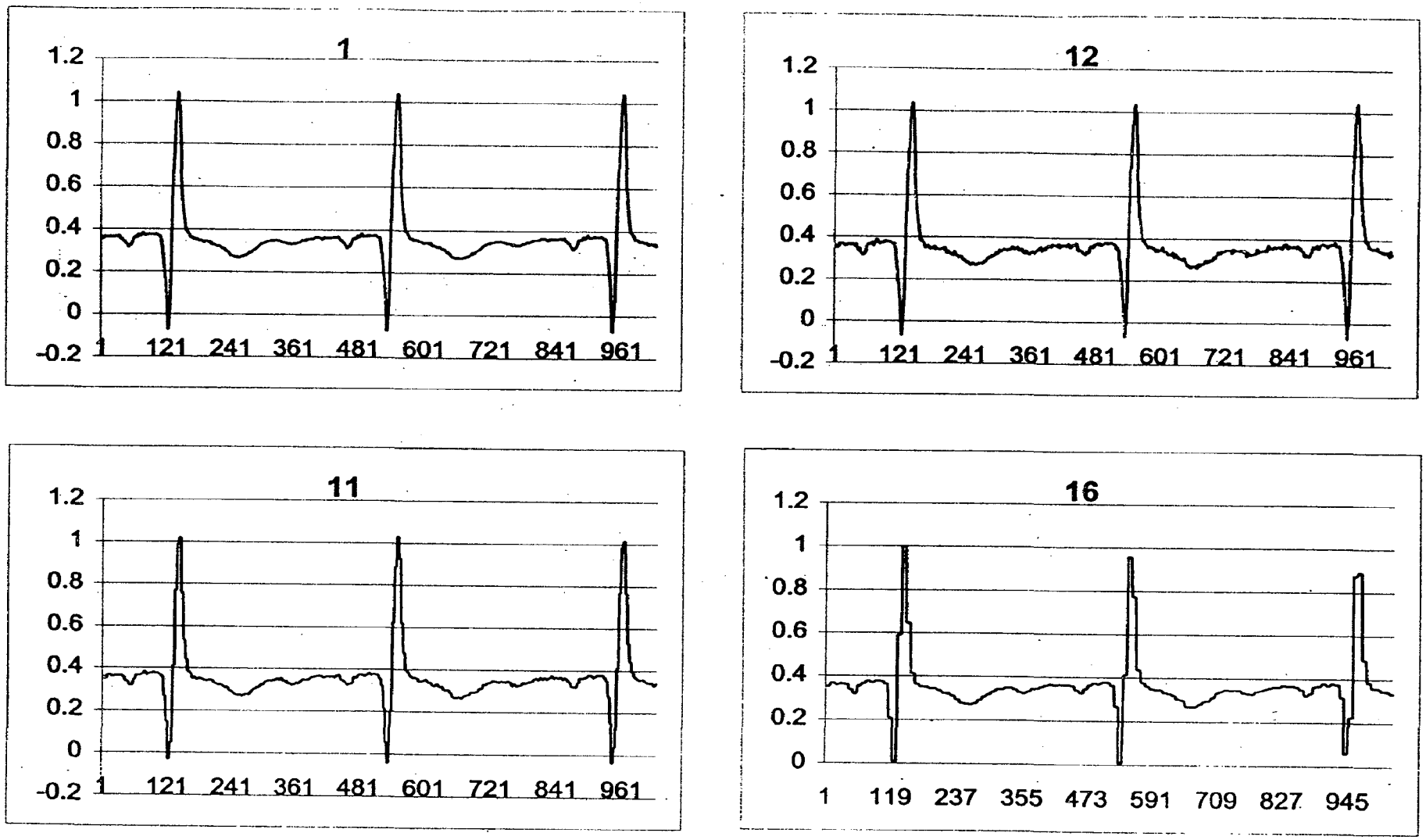
WALSH ANALYSIS OF 4Y ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

FIG. 4.95 WALSH ANALYSIS OF 4Y ECG DATA

WALSH ANALYSIS OF 4Z ECG DATA



'1' = ORIGINAL 1024 POINT DATA
'12' = 512 POINT DATA
'11' = 256 POINT DATA
'16' = 128 POINT DATA

FIG. 4.96 WALSH ANALYSIS OF 4Z ECG DATA

$$PRD = \frac{\sum_{n=1}^{N}\left[X_{org} - X_{rec}\right]^2}{\sum_{n=1}^{N} X_{org}} \times 100\ \%$$

The flow chart and program in "C" for PRD are given in Appendix I and II respectively. For different lead and for different compression ratios i.e. for 1:2, 1:4 and 1:8, the results are given in Tables 4.3 to 4.10 for FFT and in Table 4.11 to 4.18 for FWT.

4.4.2 VISUAL INSPECTION

The graphs in Figs. 4.3 to 4.62 and 4.67 to 4.96, for visual inspection of original and reconstructed signal are plotted by using MS Office97 Excell package on Pentium II computer. For different compression ratio values i.e. 1:2, 1:4 and 1:8 and results are analysed for all leads for both the transformative techniques. In the visual inspection, we compare the original and reconstructed ECG wave for different segments and for different peaks of ECG data signal for different compression ratio.

4.4.3 DETECTION OF P,Q,R,S AND T WAVE PEAKS

The flow chart and program in "C" for detection of P,Q,R,S and T wave peaks are given in Appendix I and II respectively for different compression values i.e. for 1:2, 1:4 and 1:8 and the results are analysed.

The ECG wave consists of P,Q,R,S and T segments. These segments play a pivotal role in diagnosis. Our aim is to detect them. The segments should be such that these should be within the permissible limits and should not loose their diagnostic value.

Fourier analysis of 1 CSE DATA

Compression ratio	1to 2	1 to 4	1to8
Number of points	512	256	128
Lead	PRD	PRD	PRD
L1	2.99	4.36	8.3
L2	1.24	2.11	4.37
L3	2.3	3.54	5.75
V1	0.64	1.35	2.85
V2	0.43	1.16	2.96
V3	1.25	2.53	6.9
V4	0.87	1.44	3.29
V5	1.76	2.49	4.39
V6	2.99	3.9	5.83
AR	1.32	2.17	4.85
AL	4.91	7.14	11.31
AF	1.56	2.53	4.61
X	2.69	3.4	4.13
Y	1.34	1.95	3.46
Z	0.55	1.19	3.15

TABLE 4.3

Walsh analysis of 1 CSE DATA

	1to 2	1 to 4	1to8
Number of points	512	256	128
	PRD	PRD	PRD
L1	6.79	9.62	15.02
L2	2.94	4.95	8.93
L3	4.64	7.93	14.51
V1	2.35	4.22	7.99
V2	1.88	3.67	6.12
V3	5.25	10.4	18.07
V4	3.05	5.78	10.49
V5	4.77	8.11	15.17
V6	6.87	9.76	17.35
AR	3.52	5.07	8.25
AL	10.43	15.87	27.2
AF	3.49	5.88	10.79
X	3.74	5.8	9.81
Y	3.03	4.58	8.1
Z	2.44	3.99	7.11

TABLE 4.11

Fourier analysis of 2CSE DATA

Compression ratio	1to 2	1 to 4	1to8
Number of points	512	256	128
Lead	PRD	PRD	PRD
L1	1.87	3.85	8.02
L2	2.41	7.65	25.03
L3	1.87	8.31	21.8
V1	1.2	2.12	10.67
V2	1.6	5.77	19.71
V3	2.02	7.96	22.54
V4	1.39	5.02	14.44
V5	1.4	3.75	11.43
V6	1.45	3.11	9.54
AR	2.15	4.15	14.14
AL	1.78	6.02	13.5
AF	2.25	9.56	28.03
X	1.84	3.07	5.74
Y	2.72	7.96	23.25
Z	1.09	1.93	6.35

TABLE 4.4

Walsh analysis of 2 CSE DATA

	1to 2	1 to 4	1to8
	512	256	128
	PRD	PRD	PRD
L1	6.14	11.65	20.92
L2	9.96	19.02	32.42
L3	8.97	16.94	27.87
V1	6.7	14.25	27.23
V2	8.33	15.24	24.79
V3	11.75	22.98	40.03
V4	9.54	19.76	36.71
V5	9.02	18.81	35.65
V6	8.43	17.39	33.19
AR	7.51	14.03	25.06
AL	7.14	13.27	22.55
AF	10.87	20.4	33.59
X	5.22	11.19	21.38
Y	8.7	16.73	27.25
Z	5.46	10.77	20.65

TABLE 4.12

Fourier analysis of 4 CSE DATA

Compression ratio	1to 2	1 to 4	1to8
Number of points	512	256	128
Lead	PRD	PRD	PRD
L1	3.15	5.41	10.58
L2	1.11	1.89	3.52
L3	2.29	4.8	11.6
V1	2.36	4.31	12.6
V2	1.2	2.71	6.2
V3	1.44	3.47	6.34
V4	1.23	2.62	4.18
V5	1.54	3.08	4.63
V6	1.6	2.98	4.34
AR	1.47	2.21	2.91
AL	6.4	12.54	28.91
AF	1.33	2.71	6.4
X	1.77	2.5	3.24
Y	1.44	2.75	5.96
Z	0.59	1.15	3.06

TABLE 4.5

Walsh analysis of 4 CSE DATA

	1to 2	1 to 4	1to8
Number of points	512	256	128
	PRD	PRD	PRD
L1	6.67	12.66	20.27
L2	2.69	4.83	8.78
L3	5.6	10.71	16.28
V1	7.73	13.36	25.11
V2	7.25	15.26	30.45
V3	8.16	16.99	33.96
V4	5.71	11.39	22.62
V5	6.79	13.18	25.8
V6	6.63	12.83	24.78
AR	3.3	5.76	10.34
AL	14.95	28.24	42.23
AF	3.45	6.33	10.36
X	3.45	6.23	11.27
Y	3.77	6.47	11.29
Z	3.26	6.04	11.97

TABLE 4.13

Fourier analysis of 5CSE DATA

Compression ratio	1to 2	1 to 4	1to8
Number of points	512	256	128
Lead	PRD	PRD	PRD
L1	3.97	6.96	12.14
L2	2.64	5.25	11.35
L3	3.25	5.91	10.47
V1	1.37	3.23	6.69
V2	1.65	4.89	20.56
V3	2.23	5.33	22.91
V4	5.67	13.07	32.24
V5	5.08	8.82	20.83
V6	5.28	9.52	20.53
AR	3.55	6.46	12.34
AL	3.82	6.68	11.38
AF	2.6	5.09	10.27
X	3.52	5.54	11.4
Y	1.22	2.1	3.95
Z	1.33	2.45	9.18

TABLE 4.6

Walsh analysis of 5 CSE DATA

	1to 2	1 to 4	1to8
	512	256	128
	PRD	PRD	PRD
L1	7.06	11.29	14.94
L2	7.89	15.82	26.33
L3	7.9	15.02	23.29
V1	5.06	10.43	19.19
V2	9.69	21.42	32.85
V3	10.03	23.45	34.67
V4	14.95	33.62	49.72
V5	11.17	23.77	35.55
V6	13.16	25.1	37.8
AR	7.64	12.84	19.34
AL	7.56	12.78	17.82
AF	8.51	16.75	27.47
X	6.51	11.98	16.78
Y	3.5	6.38	10.04
Z	5.84	12.06	20.04

TABLE 4.14

Fourier analysis of 8 CSE DATA

Compression ratio	1to 2	1 to 4	1to8
Number of points	512	256	128
Lead	PRD	PRD	PRD
L1	2.69	4.5	12.16
L2	1.74	2.65	8.31
L3	3.75	6.32	16.82
V1	1.72	2.77	9.58
V2	2.08	5.16	8.73
V3	2.44	6.58	10.47
V4	1.98	4.7	8.77
V5	1.58	3.12	9.31
V6	1.64	2.36	7.95
AR	1.92	3.04	8.88
AL	3.83	6.41	16.75
AF	2.45	3.99	11.41
X	0.88	1.67	5.4
Y	2.38	3.08	6.75
Z	0.68	1.27	2.78

TABLE 4.7

Walsh analysis of 8 CSE DATA

	1to 2	1 to 4	1to8
Number of points	512	256	128
	PRD	PRD	PRD
L1	8.73	18.11	35.89
L2	4.48	9.39	15.44
L3	8.14	15.47	30.98
V1	4.98	10.41	21.73
V2	6.83	14.11	24.99
V3	9.24	19.16	34.44
V4	8.26	17.3	32.17
V5	8.17	17.5	33.93
V6	7.16	15.4	30.39
AR	6.55	13.85	26.08
AL	10.35	20.76	42.1
AF	4.28	7.83	12.24
X	4.48	9.59	19.13
Y	3.86	7.29	11.48
Z	1.99	4.05	7.14

TABLE 4.15

Fourier analysis of 12 CSE DATA

Compression ratio	1to 2	1 to 4	1to8
Number of points	512	256	128
Lead	PRD	PRD	PRD
L1	5.76	7.45	13.45
L2	2.07	3.48	7.36
L3	6.83	8.97	19.77
V1	1.27	3.27	6.97
V2	2.14	5.31	12.63
V3	1.4	3.34	10.43
V4	1.46	2.94	8.15
V5	2.98	4.75	16.37
V6	0.89	1.55	3.97
AR	2.73	4.05	7.24
AL	20.66	25.72	51.69
AF	3.12	4.61	10.51
X	2.12	3.11	6.48
Y	4.95	6.56	13.39
Z	0.95	1.85	4.58

TABLE 4.8

Walsh analysis of 12 CSE DATA

	1to 2	1 to 4	1to8
	512	256	128
	PRD	PRD	PRD
L1	7.01	10.94	14.23
L2	3.28	6.99	11.89
L3	9.33	16.42	25.66
V1	2.86	5.65	9.78
V2	6.74	13.76	25.23
V3	5.7	12.4	23.4
V4	4.23	9.36	17.6
V5	6.69	13.51	23.02
V6	2.17	4.33	7.96
AR	4.71	9.34	15.65
AL	25.9	41.07	57.36
AF	3.6	6.7	10.23
X	3.04	5.36	8.29
Y	6.72	12.01	18.96
Z	2.77	6.02	11.71

TABLE 4.16

Fourier analysis of 20 CSE DATA

Compression ratio	1to 2	1 to 4	1to8
Number of points	512	256	128
Lead	PRD	PRD	PRD
L1	2.34	3.77	5.77
L2	1.86	3.22	8.58
L3	3.11	5.9	14.63
V1	5.04	8.07	14.53
V2	3.55	5.95	12.64
V3	3.96	6.94	16.03
V4	2.02	9.07	33.19
V5	5.1	9.3	24.75
V6	4.81	7.38	14.59
AR	1.76	2.84	6.87
AL	12.39	23.26	46.39
AF	2.18	3.97	10.53
X	3.26	5.32	10.39
Y	3.41	5.97	14.98
Z	2.31	3.8	7.81

TABLE 4.9

Walsh analysis of 20 CSE DATA

	1to 2	1 to 4	1to8
Number of points	512	256	128
	PRD	PRD	PRD
L1	3.01	4.87	9.42
L2	3.54	8.28	11.18
L3	5.93	13.6	15.6
V1	6.53	14.95	24.63
V2	5.54	12.27	18.52
V3	7.26	15.43	24.85
V4	12.81	24.03	51.14
V5	9.03	17.04	41.5
V6	6.2	12.52	25.28
AR	2.98	6.65	10.14
AL	19.96	40.29	42.84
AF	4.28	10.02	12.5
X	4.52	7.62	17.44
Y	6.05	14.39	18.69
Z	3.5	7.83	12.42

TABLE 4.17

Fourier analysis of 21CSE DATA

Compression ratio	1to 2	1 to 4	1to8
Number of points	512	256	128
Lead	PRD	PRD	PRD
L1	3.18	4.96	8.86
L2	1.43	2.33	6.18
L3	1.18	2.24	4.74
V1	0.91	1.59	3.36
V2	0.6	2	3.01
V3	0.81	2.6	3.31
V4	1.13	3.42	6.95
V5	1.09	2.77	6.03
V6	1.2	2.08	4.53
AR	2.21	3.28	8.31
AL	1.47	2.73	4.58
AF	1.21	2.16	5.29
X	1.33	2.98	6.02
Y	1.55	2.34	4.99
Z	1.06	2.15	2.96

TABLE 4.10.

Walsh analysis of 21 CSE DATA

	1to 2	1 to 4	1to8
	512	256	128
	PRD	PRD	PRD
L1	6.03	11.46	20.76
L2	2.77	5.91	11.34
L3	1.89	3.41	5.42
V1	2.28	5.17	9.09
V2	1.97	4.62	7.05
V3	2.15	4.77	7.65
V4	3.81	8.26	12.19
V5	4.22	9.4	15.41
V6	4.18	9.44	16.91
AR	4.49	9.48	18.33
AL	2.37	3.77	5.12
AF	2.13	4.35	7.97
X	5.26	11.23	19.4
Y	2.23	3.88	6.78
Z	1.95	3.72	5.69

TABLE 4.18

The step by step R wave detection is shown in Figs. 4.97 to 4.99 graphs, "Detection of R wave peak CSE Data", On the basis of algorithm developed by Anand and Kumar [63]. The P,Q,S and T wave peaks are calculated by taking R wave detection as reference and the scanning the ECG data wave for max and min values on right and left hand size of R peaks.

P Peak - maximum value at left side of Q wave

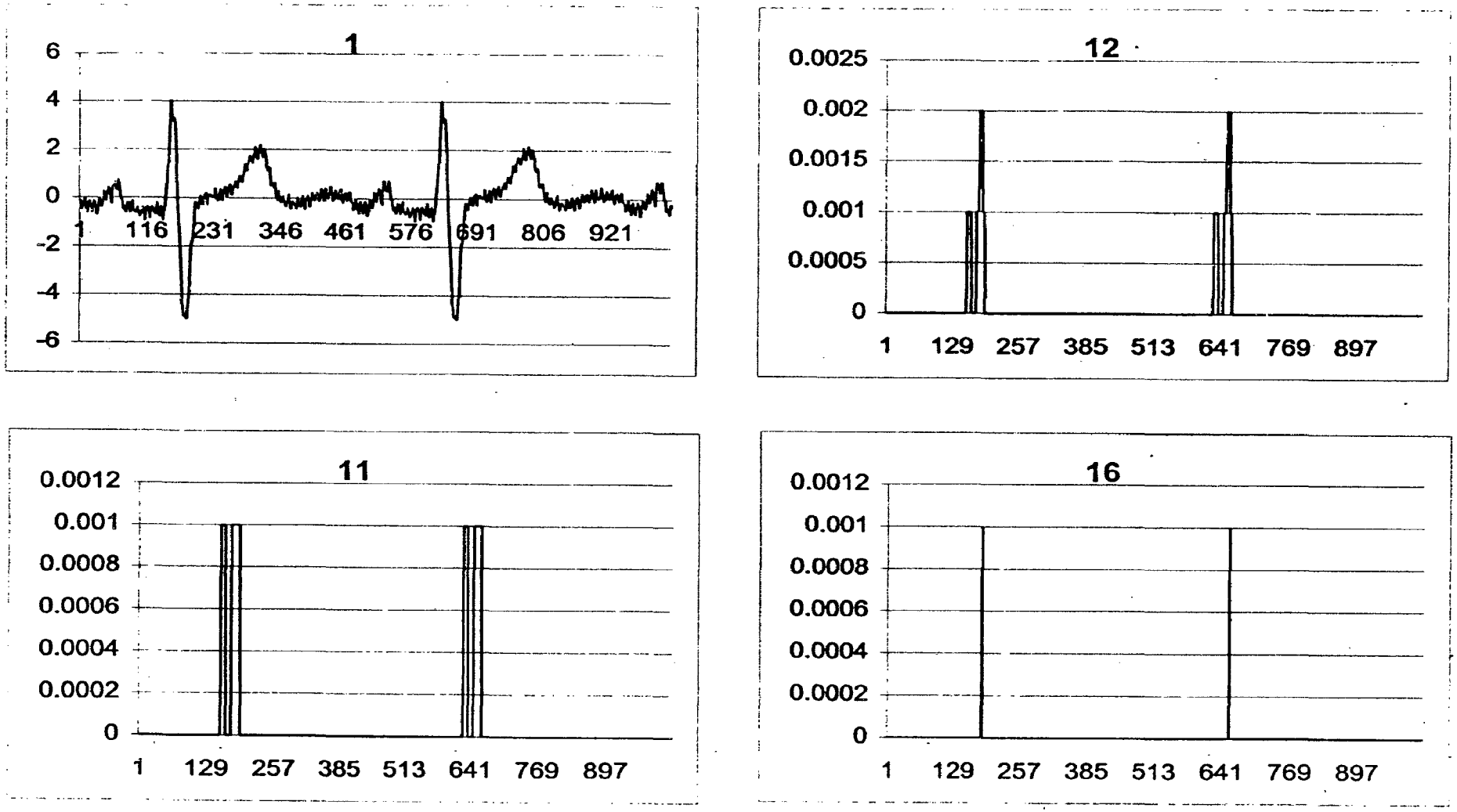
Q Peak - minimum value at left side of R wave

S Peak - minimum value at right side of R wave

T Peak - maximum value at right side of S wave

The detection of P,Q,R,S and T waves are given in Tables 4.19 to 4.24 for FFT and in Table 4.25 to 4.28 for FWT.

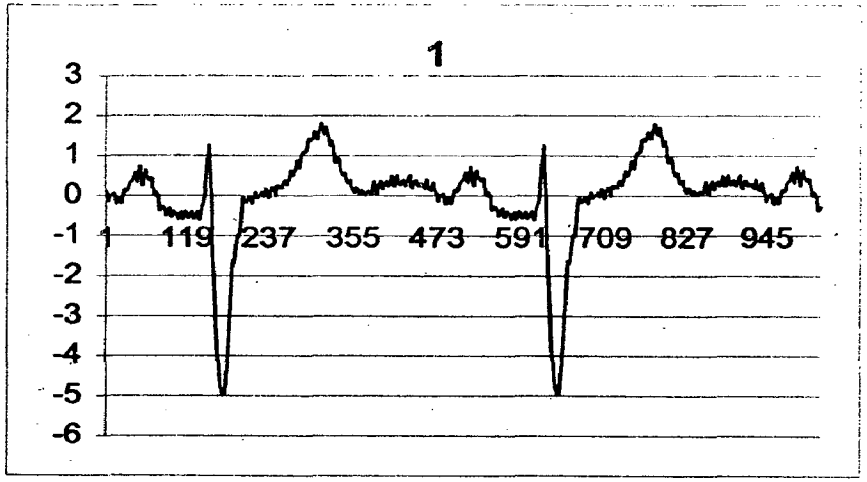
R DETECTION OF 1L1 ECG DATA

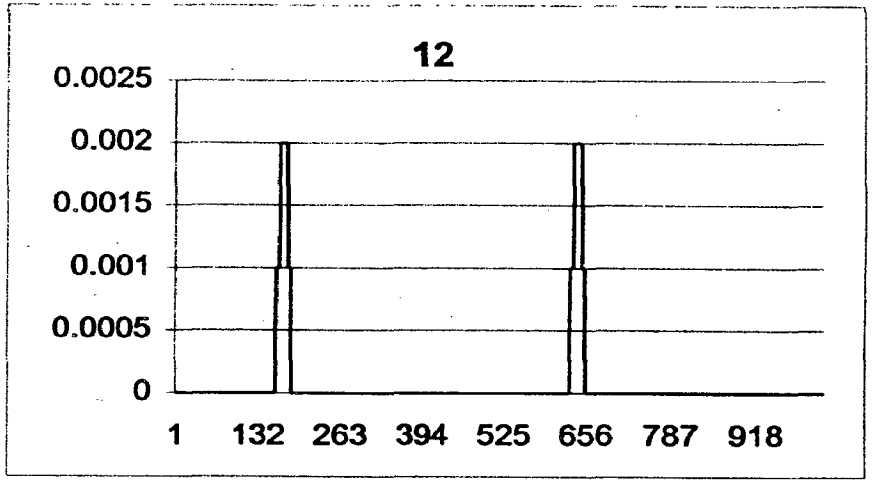


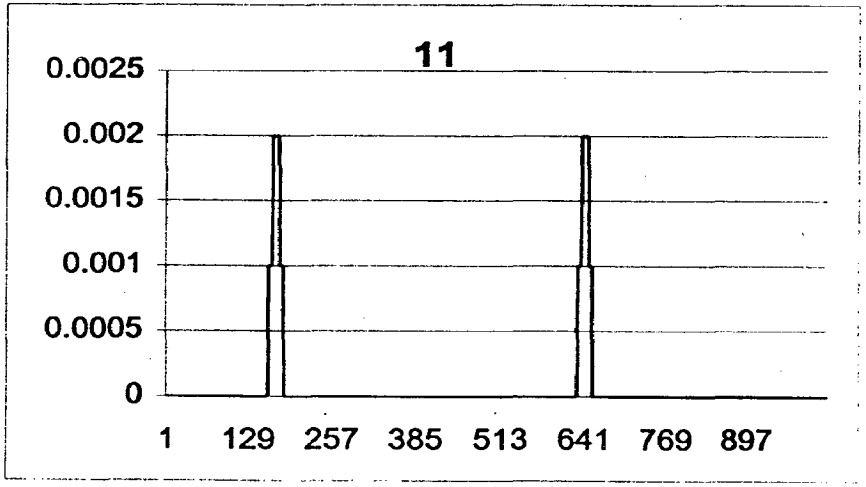
'1' =NORMALISE DATA FILE
'12' = SECOND OUTPUT DATA FILE
'11' =THIRD OUTPUT DATA FILE
'16' = FOURTH OUTPUT DATA FILE

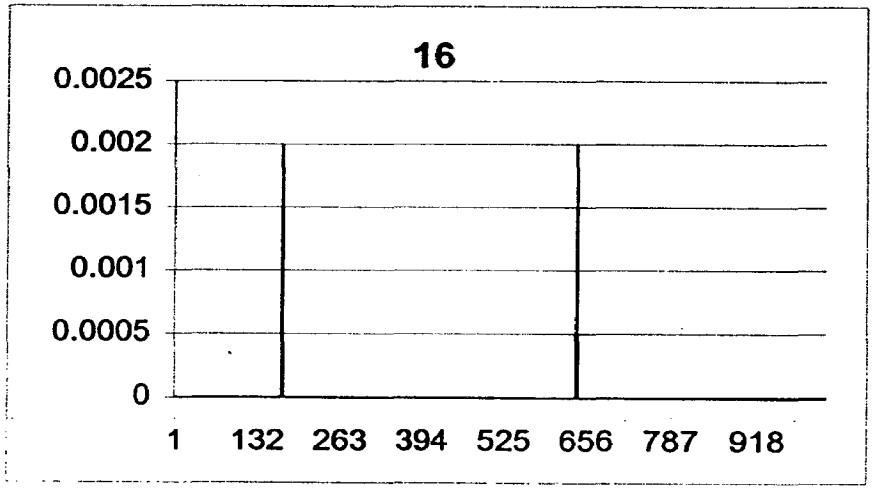
FIG. 4.97 R DETECTION OF 1L1 ECG DATA

R DETECTION OF 1L2 ECG DATA





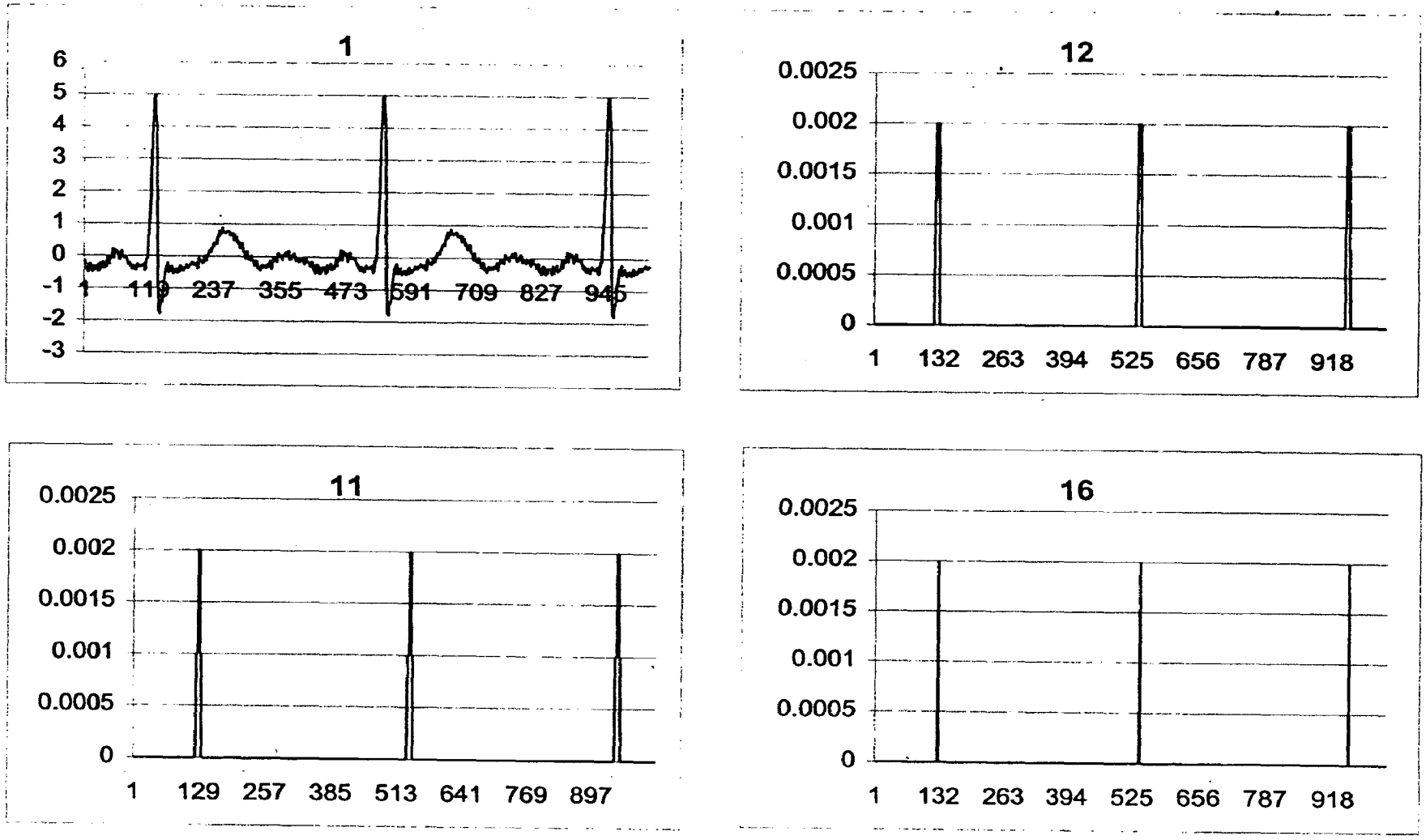




'1' = NORMALISE DATA FILE
'12' = FIRST OUTPUT DATA FILE
'11' =SECOND DATA FILE
'16' = THIRD DATA FILE

FIG. 4.98 R DETECTION OF 1L2 ECG DATA

R DETECTION OF 4L1 ECG DATA



'1' = NORMALISE DATA FILE
'12' = FIRST OUTPUT DATA FILE
'11' =SECOND DATA FILE
'16' = THIRD DATA FILE

FIG. 4.99 R DETECTION OF 4L1 ECG DATA

Input Lead Name	Compre Ratio	.peak no.	P	Q	R	S	T	
L1	1	WAVES	NOT	DETECTE	PROPERLY			
L1	2	WAVES	NOT	DETECTE	PROPERLY			
L1	4	1	59	145	160	185	309	
L1	8	WAVES	NOT	DETECTE	PROPERLY			
L2	1	1	151	248	307	364	519	
L2	2	WAVES	NOT	DETECTE	PROPERLY			
L2	4	1	151	254	308	363	519	
L2	8	WAVES	NOT	DETECTE	PROPERLY			
L3	1	1	10	22	59	105	151	
L3	1	WAVES	NOT	DETECTE	PROPERLY			
L3	2	WAVES	NOT	DETECTE	PROPERLY			
L3	4	WAVES	NOT	DETECTE	PROPERLY			
L3	8	programe	fail					
V1	1	1	31	164	178	192	329	
V1	2	WAVES	NOT	DETECTE	PROPERLY			
V1	4	1	29	164	178	192	330	
V1	8	WAVES	NOT	DETECTE	PROPERLY			
V2	1	1	62	146	156	170	318	
V2	2	WAVES	NOT	DETECTE	PROPERLY			
V2	4	1	63	143	156	171	318	
V2	8	WAVES	NOT	DETECTE	PROPERLY			
V3	1	1	62	144	156	172	313	
V3	2	WAVES	NOT	DETECTE	PROPERLY			
V3	4	1	63	143	156	172	316	
V3	8	WAVES	NOT	DETECTE	PROPERLY			
V4	1	1	62	114	156	173	313	
V4	2	WAVES	NOT	DETECTE	PROPERLY			
V4	4	1	62	141	156	173	310	
V4	8	WAVES	NOT	DETECTE	PROPERLY			
V5	1	1	63	114	154	175	307	
V5	2	WAVES	NOT	DETECTE	PROPERLY			
V5	4	1	61	141	154	175	309	
V5	8	WAVES	NOT	DETECTE	PROPERLY			
V6	1	1	7	75	78	114	152	
V6	1	WAVES	NOT	DETECTE	PROPERLY			
V6	2	WAVES	NOT	DETECTE	PROPERLY			
V6	4	WAVES	NOT	DETECTE	PROPERLY			
V6	8	1	6	78	79	123	153	
V6	8	WAVES	NOT	DETECTE	PROPERLY			
AR	1	WAVES	NOT	DETECTE	PROPERLY			
AR	1	2	177	400	401	433	582	
AR	2	1	112	152	176	233	387	
AR	4	WAVES	NOT	DETECTE	PROPERLY			
AR	8	1	108	151	176	233	370	
AL	1	WAVES	NOT	DETECTE	PROPERLY			
AL	2	1	68	148	164	185	250	

AL	4 WAVES	NOT	DETECTE PROPERLY				
AL	8	1	90	147	164	185	227
AF	1 WAVES	NOT	DETECTE PROPERLY				
AF	2	1	151	250	309	364	519
AF	4 WAVES	NOT	DETECTE PROPERLY				
AF	8	1	149	250	309	366	514
X	1 WAVES	NOT	DETECTE PROPERLY				
X	2	1	67	143	156	176	307
X	4 WAVES	NOT	DETECTE PROPERLY				
X	8	1	65	141	156	179	308
Y	1 WAVES	NOT	DETECTE PROPERLY				
Y	2	1	151	250	309	366	518
Y	4 WAVES	NOT	DETECTE PROPERLY				
Y	8	1	148	258	310	367	520
Z	1	1	10	38	83	128	166
Z	1	2	166	361	409	436	527
Z	2	1	57	154	167	211	275
Z	4	1	57	154	167	212	275
Z	8	1	16	39	83	119	167
Z	8	2	167	361	409	446	522

TABLE 4.19. The P,Q, R,S AND T WAVE PEAKS FOR 1 CSE DA TA FOURIER TRANSFO RM ANALYSIS

Input Name	Lead Compre Ratio	.peak no.	P	Q	R	S	T
L1	1	1	45	93	108	132	238
L1	1	2	106	359	374	398	504
L1	1	3	372	625	640	664	770
L1	2	1	53	111	129	137	251
L1	2	2	128	521	539	547	661
L1	4	1	59	94	128	138	252
L1	4	2	129	504	538	548	662
L1	8	WAVES NOT DETECTE PROPERLY					
L1	8	WAVES NOT DETECTE PROPERLY					
L2	1	programe fail					
L2	2	1	53	114	133	183	251
L2	2	2	134	524	543	593	661
L2	4	programe fail					
L2	8	programe fail					
L3	1	WAVES NOT DETECTE PROPERLY					
L3	1	WAVES NOT DETECTE PROPERLY					
L3	1	WAVES NOT DETECTE PROPERLY					
L3	2	1	54	114	137	183	252
L3	2	2	136	524	547	593	662
L3	4	programe fail					
L3	8	programe fail					
V1	1	WAVES NOT DETECTE PROPERLY					
V1	1	WAVES NOT DETECTE PROPERLY					
V1	1	WAVES NOT DETECTE PROPERLY					
V1	2	1	50	89	119	141	205
V1	2	2	122	499	529	551	615
V1	4	WAVES NOT DETECTE PROPERLY					
V1	4	WAVES NOT DETECTE PROPERLY					
V1	8	1	50	93	119	138	206
V1	8	2	119	503	529	548	615
V2	1	WAVES NOT DETECTE PROPERLY					
V2	1	WAVES NOT DETECTE PROPERLY					
V2	1	WAVES NOT DETECTE PROPERLY					
V2	2	1	49	88	123	138	256
V2	2	2	122	498	533	548	666
V2	4	WAVES NOT DETECTE PROPERLY					
V2	4	WAVES NOT DETECTE PROPERLY					
V2	8	1	55	96	123	138	255
V2	8	2	123	506	533	548	664
V3	1	WAVES NOT DETECTE PROPERLY					
V3	1	WAVES NOT DETECTE PROPERLY					
V3	1	WAVES NOT DETECTE PROPERLY					
V3	2	1	56	87	125	140	253
V3	2	2	125	497	535	550	663
V3	4	WAVES NOT DETECTE PROPERLY					
V3	4	WAVES NOT DETECTE PROPERLY					

V3	8	1	56	97	125	139	256
V3	8	2	125	507	535	549	666
V4	1	WAVES NOT DETECTE PROPERLY					
V4	1	WAVES NOT DETECTE PROPERLY					
V4	1	WAVES NOT DETECTE PROPERLY					
V4	2	1	56	87	127	142	258
V4	2	2	127	497	537	552	668
V4	4	1	54	97	127	142	256
V4	4	2	126	507	537	552	666
V4	8	1	56	112	127	142	260
V4	8	2	127	522	537	552	670
V5	1	WAVES NOT DETECTE PROPERLY					
V5	1	WAVES NOT DETECTE PROPERLY					
V5	1	WAVES NOT DETECTE PROPERLY					
V5	2	1	55	114	128	143	262
V5	2	2	128	524	538	553	672
V5	4	1	53	113	128	143	260
V5	4	2	128	523	538	553	670
V5	8	WAVES NOT DETECTE PROPERLY					
V5	8	WAVES NOT DETECTE PROPERLY					
V6	1	1	28	91	107	121	239
V6	1	2	106	357	373	387	505
V6	1	3	372	623	639	653	771
V6	2	programe fail					
V6	4	programe fail					
V6	8	programe fail					
AR	1	WAVES NOT DETECTE PROPERLY					
AR	1	WAVES NOT DETECTE PROPERLY					
AR	1	WAVES NOT DETECTE PROPERLY					
AR	2	programe fail					
AR	4	programe fail					
AR	8	programe fail					
AL	1	1	69	93	110	133	144
AL	1	2	112	359	376	399	410
AL	1	3	378	625	642	665	676
AL	2	WAVES NOT DETECTE PROPERLY					
AL	2	WAVES NOT DETECTE PROPERLY					
AL	4	1	59	85	128	137	245
AL	4	2	128	495	538	547	655
AL	8	WAVES NOT DETECTE PROPERLY					
AL	8	WAVES NOT DETECTE PROPERLY					
AF	1	1	126	2	36	65	94
AF	1	2	34	268	302	331	360
AF	1	3	300	534	568	597	626
AF	2	programe fail					
AF	4	programe fail					
AF	8	programe fail					
X	1	1	106	174	175	192	253
X	1	2	372	440	441	458	519

X	1	2	372	440	441	458	519
X	1	3	638	706	707	724	785
X	2	1	58	113	128	143	260
X	2	2	128	523	538	553	670
X	4	programe fail					
X	8	programe fail					
Y	1	WAVES NOT DETECTE PROPERLY					
Y	1	WAVES NOT DETECTE PROPERLY					
Y	1	WAVES NOT DETECTE PROPERLY					
Y	2	1	65	114	135	183	253
Y	2	2	135	524	545	593	663
Y	4	programe fail					
Y	8	programe fail					
Z	1	1	10	25	48	59	108
Z	1	WAVES NOT DETECTE PROPERLY					
Z	1	WAVES NOT DETECTE PROPERLY					
Z	1	WAVES NOT DETECTE PROPERLY					
Z	2	1	76	122	140	189	326
Z	2	2	141	532	550	600	736
Z	4	1	77	123	140	188	326
Z	4	2	140	533	550	599	736
Z	8	WAVES NOT DETECTE PROPERLY					
Z	8	WAVES NOT DETECTE PROPERLY					

TABLE 4.20 The P,Q, R,S AND T WAVE PEAKS FOR 2 CSE DA TA FOURIER TRANSFO RM ANALYSIS

Input Lead Name	Compre Ratio	.peak no.	P	Q	R	S	T	
L1	1	1	53	112	129	139	251	
L1	1	2	129	522	539	549	661	
L1	2	1	53	111	129	137	251	
L1	2	2	128	521	539	547	661	
L1	4	1	59	94	129	138	252	
L1	4	2	129	504	538	548	662	
L1	8	WAVES NOT DETECTE PROPERLY						
L1	8	WAVES NOT DETECTE PROPERLY						
L2	1	1	53	114	133	167	251	
L2	1	2	134	524	543	577	661	
L2	2	programe fail						
L2	4	programe fail						
L2	8	programe fail						
L3	1	programe fail						
L3	2	programe fail						
L3	4	programe fail						
L3	8	programe fail						
V1	1	WAVES NOT DETECTE PROPERLY						
V1	1	WAVES NOT DETECTE PROPERLY						
V1	2	1	50	89	120	141	205	
V1	2	2	122	499	530	551	615	
V1	4	WAVES NOT DETECTE PROPERLY						
V1	4	WAVES NOT DETECTE PROPERLY						
V1	8	1	50	93	119	138	206	
V1	8	2	119	503	529	548	615	
V2	1	WAVES NOT DETECTE PROPERLY						
V2	1	WAVES NOT DETECTE PROPERLY						
V2	2	1	49	88	123	138	256	
V2	2	2	122	498	533	548	666	
V2	4	WAVES NOT DETECTE PROPERLY						
V2	4	WAVES NOT DETECTE PROPERLY						
V2	8	1	55	96	123	138	255	
V2	8	2	123	506	533	548	664	
V3	1	WAVES NOT DETECTE PROPERLY						
V3	1	WAVES NOT DETECTE PROPERLY						
V3	2	1	56	87	125	140	253	
V3	2	2	125	497	535	550	663	
V3	4	WAVES NOT DETECTE PROPERLY						
V3	4	WAVES NOT DETECTE PROPERLY						
V3	8	1	56	97	125	139	256	
V3	8	2	125	507	535	549	666	
V4	1	WAVES NOT DETECTE PROPERLY						
V4	1	WAVES NOT DETECTE PROPERLY						
V4	2	1	56	87	127	142	258	
V4	2	2	127	497	537	552	668	
V4	4	WAVES NOT DETECTE PROPERLY						
V4	4	WAVES NOT DETECTE PROPERLY						

V4	8	1	56	112	127	142	260
V4	8	2	127	522	537	552	670
V5	1	WAVES NOT DETECTE PROPERLY					
V5	1	WAVES NOT DETECTE PROPERLY					
V5	2	1	55	114	128	143	262
V5	2	2	128	524	538	553	672
V5	4	WAVES NOT DETECTE PROPERLY					
V5	4	WAVES NOT DETECTE PROPERLY					
V5	8	1	57	113	128	144	261
V5	8	2	128	523	538	554	671
V6	1	programe fail					
V6	2	programe fail					
V6	4	programe fail					
V6	8	programe fail					
AR	1	WAVES NOT DETECTE PROPERLY					
AR	1	WAVES NOT DETECTE PROPERLY					
AR	2	programe fail					
AR	4	programe fail					
AR	8	programe fail					
AL	1	1	62	86	127	136	245
AL	1	2	127	496	537	546	655
AL	2	WAVES NOT DETECTE PROPERLY					
AL	2	WAVES NOT DETECTE PROPERLY					
AL	4	1	59	85	128	137	245
AL	4	2	128	495	538	547	655
AL	8	WAVES NOT DETECTE PROPERLY					
AL	8	WAVES NOT DETECTE PROPERLY					
AF	1	1	53	114	136	183	253
AF	1	2	135	524	546	593	663
AF	2	programe fail					
AF	4	programe fail					
AF	8	programe fail					
X	1	programe fail					
X	2	programe fail					
X	4	programe fail					
X	8	programe fail					
Y	1	programe fail					
Y	2	programe fail					
Y	4	programe fail					
Y	8	programe fail					
Z	1	WAVES NOT DETECTE PROPERLY					
Z	1	WAVES NOT DETECTE PROPERLY					
Z	2	1	76	122	140	189	326
Z	2	2	141	532	550	600	736
Z	4	WAVES NOT DETECTE PROPERLY					
Z	4	WAVES NOT DETECTE PROPERLY					
Z	8	1	79	123	140	188	325
Z	8	2	139	533	550	598	734

TABLE 4.21. The P,Q,R ,S AND T WAVE PEAKS FOR 4 CSE DA TA FOURIER TRANSFO RM ANALYSIS

Input Lead Name	Compre Ratio	.peak no.	P	Q	R	S	T	
L1	1	WAVES	NOT	DETECTE PROPERLY				
L1	1	WAVES	NOT	DETECTE PROPERLY				
L1	2	WAVES	NOT	DETECTE PROPERLY				
L1	2	WAVES	NOT	DETECTE PROPERLY				
L1	4	1	36	36	56	74	74	
L1	4	2	53	200	220	238	238	
L1	4	3	53	356	376	394	394	
L1	4	4	53	520	540	558	558	
L1	4	5	53	676	696	714	714	
L1	4	6	53	840	860	878	878	
L1	8	WAVES	NOT	DETECTE PROPERLY				
L1	8	WAVES	NOT	DETECTE PROPERLY				
L2	1	programe	fail					
L2	2	programe	fail					
L2	4	programe	fail					
L2	8	programe	fail					
L3	1	programe	fail					
L3	2	programe	fail					
L3	4	programe	fail					
L3	8	programe	fail					
V1	1	WAVES	NOT	DETECTE PROPERLY				
V1	1	WAVES	NOT	DETECTE PROPERLY				
V1	2	WAVES	NOT	DETECTE PROPERLY				
V1	2	WAVES	NOT	DETECTE PROPERLY				
V1	4	1	47	70	106	120	253	
V1	4	2	106	390	426	440	573	
V1	8	WAVES	NOT	DETECTE PROPERLY				
V1	8	WAVES	NOT	DETECTE PROPERLY				
V2	1	1	49	69	106	116	208	
V2	1	2	106	389	426	436	528	
V2	2	WAVES	NOT	DETECTE PROPERLY				
V2	2	WAVES	NOT	DETECTE PROPERLY				
V2	4	1	49	87	106	115	207	
V2	4	2	106	390	426	435	527	
V2	8	WAVES	NOT	DETECTE PROPERLY				
V2	8	WAVES	NOT	DETECTE PROPERLY				
V3	1	1	49	98	108	117	208	
V3	1	2	109	418	428	437	528	
V3	2	WAVES	NOT	DETECTE PROPERLY				
V3	2	WAVES	NOT	DETECTE PROPERLY				
V3	4	1	46	98	108	117	207	
V3	4	2	108	418	428	437	527	
V3	8	WAVES	NOT	DETECTE PROPERLY				
V3	8	WAVES	NOT	DETECTE PROPERLY				
V4	1	1	46	96	109	117	229	

V4	1		2	109	416	429	437	549
V4	2	WAVES	NOT	DETECTE	PROPERLY			
V4	2	WAVES	NOT	DETECTE	PROPERLY			
V4	4		1	51	98	109	117	229
V4	4		2	109	418	429	437	549
V4	8	WAVES	NOT	DETECTE	PROPERLY			
V4	8	WAVES	NOT	DETECTE	PROPERLY			
V5	1		1	52	98	109	117	229
V5	1		2	109	418	429	437	549
V5	2	WAVES	NOT	DETECTE	PROPERLY			
V5	2	WAVES	NOT	DETECTE	PROPERLY			
V5	4		1	53	97	109	118	227
V5	4		2	109	417	429	438	547
V5	8	WAVES	NOT	DETECTE	PROPERLY			
V5	8	WAVES	NOT	DETECTE	PROPERLY			
V6	1		1	46	94	108	119	229
V6	1		2	109	414	428	439	549
V6	2	WAVES	NOT	DETECTE	PROPERLY			
V6	2	WAVES	NOT	DETECTE	PROPERLY			
V6	4		1	46	89	108	118	220
V6	4		2	108	409	428	438	540
V6	8	WAVES	NOT	DETECTE	PROPERLY			
V6	8	WAVES	NOT	DETECTE	PROPERLY			
AR	1		1	25	108	121	154	177
AR	1		2	117	428	441	474	497
AR	2	WAVES	NOT	DETECTE	PROPERLY			
AR	2	WAVES	NOT	DETECTE	PROPERLY			
AR	4		1	24	107	121	154	176
AR	4		2	117	427	441	474	496
AR	8	WAVES	NOT	DETECTE	PROPERLY			
AR	8	WAVES	NOT	DETECTE	PROPERLY			
AR	8	WAVES	NOT	DETECTE	PROPERLY			
AL	1	programe	fail					
AL	2	programe	fail					
AL	4	programe	fail					
AL	8	programe	fail					
AF	1		1	45	69	109	124	239
AF	1		2	108	389	429	444	559
AF	2		1	45	69	109	124	243
AF	2		2	108	389	429	444	563
AF	4	WAVES	NOT	DETECTE	PROPERLY			
AF	4	WAVES	NOT	DETECTE	PROPERLY			
AF	8		1	44	69	109	126	239
AF	8		2	109	389	429	445	559
X	1	WAVES	NOT	DETECTE	PROPERLY			
X	1	WAVES	NOT	DETECTE	PROPERLY			
X	2		1	56	97	108	117	217
X	2		2	109	417	428	437	537
X	4	WAVES	NOT	DETECTE	PROPERLY			

X	8	1	55	96	108	119	219	
X	8	2	107	416	428	439	540	
Y	1	WAVES	NOT	DETECTE	PROPERLY			
Y	1	WAVES	NOT	DETECTE	PROPERLY			
Y	2	1	41	69	108	123	239	
Y	2	2	107	389	428	443	559	
Y	4	WAVES	NOT	DETECTE	PROPERLY			
Y	4	WAVES	NOT	DETECTE	PROPERLY			
Y	8	1	43	68	109	124	239	
Y	8	2	108	388	429	444	559	
Z	1	WAVES	NOT	DETECTE	PROPERLY			
Z	1	WAVES	NOT	DETECTE	PROPERLY			
Z	2	1	70	105	118	146	218	
Z	2	2	117	425	438	466	538	
Z	4	WAVES	NOT	DETECTE	PROPERLY			
Z	4	WAVES	NOT	DETECTE	PROPERLY			
Z	8	1	64	104	119	144	218	
Z	8	2	118	424	439	464	538	

TABLE 4.22. The P,Q,R ,S AND T WAVE PEAKS FOR 5 CSE DATA FOURIER TRANSFORM ANALYSIS

Input Lead Name	Compre Ratio	peak no.	P	Q	R	S	T	
L1	1	1		72	129	144	176	290
L1	2	1		72	128	144	177	281
L1	4	1		73	128	144	155	289
L1	8	1		76	130	144	157	289
L2	1	1		73	128	143	194	282
L2	2	1		6	52	67	110	141
L2	2	2		141	377	410	444	575
L2	4	1		73	127	143	202	281
L2	8	1		73	128	144	201	286
L3	1	WAVES	NOT	DETECTE	PROPERLY			
L3	2	1		60	144	155	207	357
L3	4	WAVES	NOT	DETECTE	PROPERLY			
L3	8	1		68	144	157	165	332
V1	1	WAVES	NOT	DETECTE	PROPERLY			
V1	2	1		68	91	133	146	344
V1	4	WAVES	NOT	DETECTE	PROPERLY			
V1	8	1		67	123	132	146	363
V2	1	WAVES	NOT	DETECTE	PROPERLY			
V2	2	1		68	106	140	153	288
V2	4	WAVES	NOT	DETECTE	PROPERLY			
V2	8	1		74	112	140	153	287
V3	1	WAVES	NOT	DETECTE	PROPERLY			
V3	2	1		68	101	142	153	285
V3	4	WAVES	NOT	DETECTE	PROPERLY			
V3	8	1		74	114	142	155	289
V4	1	WAVES	NOT	DETECTE	PROPERLY			
V4	2	1		68	127	142	153	286
V4	4	1		68	127	142	155	285
V4	8	1		74	128	142	156	290
V5	1	1		70	127	143	157	276
V5	2	1		69	128	143	157	281
V5	4	1		68	128	143	155	283
V5	8	1		74	129	143	156	290
V6	1	1		73	129	144	193	281
V6	2	1		73	129	144	194	281
V6	4	1		73	128	144	194	290
V6	8	1		75	129	144	172	290
AR	1	WAVES	NOT	DETECTE	PROPERLY			
AR	2	WAVES	NOT	DETECTE	PROPERLY			
AR	2	WAVES	NOT	DETECTE	PROPERLY			
AR	4	WAVES	NOT	DETECTE	PROPERLY			
AR	8	WAVES	NOT	DETECTE	PROPERLY			
AL	1	1		80	132	145	153	280
AL	2	1		5	60	68	109	144
AL	2	2		144	369	409	451	582

AL	8	1	77	131	144	157	291
AF	1	1	7	52	69	111	140
AF	1	2	140	377	408	435	575
AF	2	programe fail					
AF	4	programe fail					
AF	8	programe fail					
X	1	1	80	128	144	184	280
X	2	1	80	128	144	193	287
X	4	1	80	128	144	193	289
X	8	1	75	129	143	157	290
Y	1	1	71	128	143	201	280
Y	2	1	72	128	143	201	281
Y	4	1	6	27	68	102	141
Y	4	2	141	352	401	418	574
Y	8	1	70	128	144	197	285
Z	1	1	53	139	154	213	219
Z	2	WAVES NOT DETECTE PROPERLY					
Z	4	1	52	139	154	214	218
Z	8	WAVES NOT DETECTE PROPERLY					

TABLE 4.23. The P,Q,R ,S AND T WAVE PEAKS FOR 8 CSE DA TA FOURIER TRANSFO RM ANALYSIS

Input Lead Name	Compre Ratio	.peak no.	P	Q	R	S	T	
L1	1		1	33	189	202	215	358
L1	2		1	117	188	201	215	358
L1	4	WAVES	NOT	DETECTE	PROPERLY			
L1	8		1	17	92	120	170	201
L1	8		2	201	455	462	510	621
L2	1	WAVES	NOT	DETECTE	PROPERLY			
L2	2	WAVES	NOT	DETECTE	PROPERLY			
L2	4		1	118	163	200	221	367
L2	8	WAVES	NOT	DETECTE	PROPERLY			
L3	1		1	118	163	199	222	376
L3	2	WAVES	NOT	DETECTE	PROPERLY			
L3	4		1	118	163	199	222	376
L3	8	WAVES	NOT	DETECTE	PROPERLY			
V1	1	WAVES	NOT	DETECTE	PROPERLY			
V1	1		2	215	409	450	493	520
V1	2	WAVES	NOT	DETECTE	PROPERLY			
V1	4	WAVES	NOT	DETECTE	PROPERLY			
V1	8		1	5	86	125	134	191
V1	8		2	191	412	459	493	520
V2	1	WAVES	NOT	DETECTE	PROPERLY			
V2	2	WAVES	NOT	DETECTE	PROPERLY			
V2	4	WAVES	NOT	DETECTE	PROPERLY			
V2	8	programe	fail					
V3	1	WAVES	NOT	DETECTE	PROPERLY			
V3	2		1	195	277	357	434	437
V3	4	WAVES	NOT	DETECTE	PROPERLY			
V3	8		1	196	277	357	435	435
V4	1	WAVES	NOT	DETECTE	PROPERLY			
V4	2		1	119	174	203	212	358
V4	4	WAVES	NOT	DETECTE	PROPERLY			
V4	8		1	119	175	202	215	361
V5	1	WAVES	NOT	DETECTE	PROPERLY			
V5	2		1	122	188	203	215	361
V5	4	WAVES	NOT	DETECTE	PROPERLY			
V5	8		1	120	188	203	216	362
V6	1	programe	fail					
V6	2	programe	fail					
V6	4	programe	fail					
V6	8	WAVES	NOT	DETECTE	PROPERLY			
V6	8	WAVES	NOT	DETECTE	PROPERLY			
AR	1		1	14	200	219	293	512
AR	2		1	5	200	218	293	513
AR	4	WAVES	NOT	DETECTE	PROPERLY			
AR	8		1	8	200	217	295	530
AL	1	programe	fail					

AL	2	1	8	110	111	120	201
AL	2	2	201	509	528	542	597
AL	4	programe fail					
AL	8	programe fail					
AF	1	1	51	80	119	163	198
AF	1	2	198	513	516	563	626
AF	2	WAVES NOT DETECTE PROPERLY					
AF	4	1	118	163	200	221	367
AF	8	WAVES NOT DETECTE PROPERLY					
X	1	1	8	188	203	215	358
X	2	WAVES NOT DETECTE PROPERLY					
X	4	1	8	188	202	214	359
X	8	WAVES NOT DETECTE PROPERLY					
Y	1	1	119	172	200	220	365
Y	2	WAVES NOT DETECTE PROPERLY					
Y	4	1	117	170	200	221	367
Y	8	WAVES NOT DETECTE PROPERLY					
Z	1	1	9	195	213	283	520
Z	2	WAVES NOT DETECTE PROPERLY					
Z	4	1	11	196	215	292	529
Z	8	WAVES NOT DETECTE PROPERLY					

TABLE 4.24 The P,Q,R ,S AND T WAVE PEAKS FOR 21 CSE DA TA

Input Lead Name	Compre Ratio	.peak no.	P	Q	R	S	T	
L1	1	WAVES	NOT	DETECTE	PROPERLY			
L1	2	WAVES	NOT	DETECTE	PROPERLY			
L1	4	1	67	115	160	180	308	
L1	8	WAVES	NOT	DETECTE	PROPERLY			
L2	1	1	151	248	307	364	519	
L2	2	WAVES	NOT	DETECTE	PROPERLY			
L2	4	1	151	255	309	352	516	
L2	8	WAVES	NOT	DETECTE	PROPERLY			
L3	1	1	10	22	59	105	151	
L3	1	WAVES	NOT	DETECTE	PROPERLY			
L3	2	WAVES	NOT	DETECTE	PROPERLY			
L3	4	WAVES	NOT	DETECTE	PROPERLY			
L4	8	program	fail					
V1	1	1	31	164	178	192	329	
V1	2	1	33	73	94	102	176	
V1	2	2	177	256	257	274	336	
V1	2	3	177	419	427	434	500	
V1	2	4	177	557	578	586	646	
V1	2	5	647	726	727	746	802	
V1	4	1	31	167	178	192	328	
V1	8	1	39	167	179	192	328	
V2	1	1	62	146	156	170	318	
V2	2	1	63	147	156	170	316	
V2	4	WAVES	NOT	DETECTE	PROPERLY			
V3	1	WAVES	NOT	DETECTE	PROPERLY			
V3	2	WAVES	NOT	DETECTE	PROPERLY			
V3	4	1	63	143	156	172	316	
V3	8	WAVES	NOT	DETECTE	PROPERLY			
V4	1	1	62	114	156	173	313	
V4	2	WAVES	NOT	DETECTE	PROPERLY			
V4	4	1	63	115	156	172	312	
V4	8	WAVES	NOT	DETECTE	PROPERLY			
V5	1	1	63	114	154	175	307	
V5	2	WAVES	NOT	DETECTE	PROPERLY			
V5	4	1	63	123	155	172	308	
V5	8	WAVES	NOT	DETECTE	PROPERLY			
V6	1	1	7	75	78	114	152	
V6	1	WAVES	NOT	DETECTE	PROPERLY			
V6	2	WAVES	NOT	DETECTE	PROPERLY			
V6	4	WAVES	NOT	DETECTE	PROPERLY			
V6	8	1	159	247	304	360	528	
AR	1	WAVES	NOT	DETECTE	PROPERLY			
AR	2	1	113	153	176	232	370	
AR	4	WAVES	NOT	DETECTE	PROPERLY			
AR	8	1	111	151	176	232	368	

AL	2		1	69	149	164	184	250
AL	4	WAVES	NOT	DETECTE	PROPERLY			
AL	8		1	71	151	164	184	248
AF	1	WAVES	NOT	DETECTE	PROPERLY			
AF	2		1	151	251	309	362	518
AF	4	WAVES	NOT	DETECTE	PROPERLY			
AF	8		1	151	255	311	368	512
X	1	WAVES	NOT	DETECTE	PROPERLY			
X	2		1	67	143	156	176	308
X	4	WAVES	NOT	DETECTE	PROPERLY			
X	8		1	71	143	156	176	304
Y	1	WAVES	NOT	DETECTE	PROPERLY			
Y	2		1	151	251	310	366	508
Y	4	WAVES	NOT	DETECTE	PROPERLY			
Y	8		1	151	255	311	368	504
Z	1		1	10	38	83	128	166
Z	1		2	166	361	409	436	527
Z	2		1	59	155	168	210	258
Z	4		1	59	155	168	204	256
Z	8		1	23	39	82	120	160
Z	8		2	167	359	407	432	520

TABLE 4.25 The P,Q, R,S AND T WAVE PEAKS FOR 1 CSE DATA WALSH TRANSFORM ANALYSIS

Input Lead Name	Compre Ratio	.peak no.	P	Q	R	S	T
L1	1	1	53	112	129	139	251
L1	1	2	129	522	539	549	661
L1	2	1	61	97	129	138	250
L1	2	2	129	521	538	546	664
L1	4	1	63	91	128	136	252
L1	4	2	131	507	539	548	660
L1	8	WAVES NOT DETECTE PROPERLY					
L1	8	WAVES NOT DETECTE PROPERLY					
L2	1	1	53	114	133	167	251
L2	1	2	134	524	543	577	661
L2	2	programe fail					
L2	4	programe fail					
L2	8	programe fail					
L3	1	programe fail					
L3	2	programe fail					
L3	4	1	139	223	279	300	436
L3	8	programe fail					
V1	1	WAVES NOT DETECTE PROPERLY					
V1	1	WAVES NOT DETECTE PROPERLY					
V1	2	1	49	97	120	140	206
V1	2	2	123	499	530	550	614
V1	4	WAVES NOT DETECTE PROPERLY					
V1	4	WAVES NOT DETECTE PROPERLY					
V1	8	1	55	103	119	136	200
V1	8	2	119	495	531	544	624
V2	1	WAVES NOT DETECTE PROPERLY					
V2	1	WAVES NOT DETECTE PROPERLY					
V2	2	1	55	95	123	138	256
V2	2	2	123	505	533	546	666
V2	4	WAVES NOT DETECTE PROPERLY					
V2	4	WAVES NOT DETECTE PROPERLY					
V2	8	programe fail					
V3	1	1	57	86	125	140	253
V3	1	2	125	496	535	550	663
V3	2	WAVES NOT DETECTE PROPERLY					
V3	2	WAVES NOT DETECTE PROPERLY					
V3	4	1	55	99	125	140	252
V3	4	2	127	511	535	548	664
V3	8	WAVES NOT DETECTE PROPERLY					
V3	8	WAVES NOT DETECTE PROPERLY					
V4	1	1	55	86	127	143	253
V4	1	2	127	496	537	553	663
V4	2	WAVES NOT DETECTE PROPERLY					
V4	2	WAVES NOT DETECTE PROPERLY					
V4	4	1	55	95	127	140	260
V4	4	2	127	499	537	552	660
V4	8	WAVES NOT DETECTE PROPERLY					
V4	8	WAVES NOT DETECTE PROPERLY					

V5	1	1	55	114	128	143	258	
V5	1	2	128	524	538	553	668	
V5	2	WAVES NOT DETECTE PROPERLY						
V5	2	WAVES NOT DETECTE PROPERLY						
V5	4	1	55	115	128	144	252	
V5	4	2	131	523	538	552	668	
V5	8	WAVES NOT DETECTE PROPERLY						
V5	8	WAVES NOT DETECTE PROPERLY						
V6	1	1	70	114	129	163	258	
V6	1	2	129	524	539	573	668	
V6	2	programe fail						
V6	4	programe fail						
V6	8	programe fail						
AR	1	WAVES NOT DETECTE PROPERLY						
AR	1	WAVES NOT DETECTE PROPERLY						
AR	2	1	114	53	61	118	184	
AR	2	WAVES NOT DETECTE PROPERLY						
AR	4	WAVES NOT DETECTE PROPERLY						
AR	4	WAVES NOT DETECTE PROPERLY						
AR	8	WAVES NOT DETECTE PROPERLY						
AL	1	1	62	86	128	136	245	
AL	1	2	127	496	538	546	655	
AL	2	WAVES NOT DETECTE PROPERLY						
AL	2	WAVES NOT DETECTE PROPERLY						
AL	4	1	63	103	127	136	260	
AL	4	2	127	507	537	544	652	
AL	8	WAVES NOT DETECTE PROPERLY						
AL	8	WAVES NOT DETECTE PROPERLY						
AF	1	1	53	114	136	183	253	
AF	1	2	135	524	546	593	663	
AF	2	programe fail						
AF	4	programe fail						
AF	8	1	135	431	474	520	544	
X	1	WAVES NOT DETECTE PROPERLY						
X	1	WAVES NOT DETECTE PROPERLY						
X	2	1	59	111	128	142	266	
X	2	2	129	521	538	552	670	
X	4	programe fail						
X	8	programe fail						
Y	1	programe fail						
Y	2	programe fail						
Y	4	programe fail						
Y	8	1	129	23	62	112	128	
Y	8	2	135	527	552	600	656	
Z	1	1	76	122	140	189	318	
Z	1	2	140	532	550	599	728	
Z	2	WAVES NOT DETECTE PROPERLY						
Z	2	WAVES NOT DETECTE PROPERLY						
Z	4	1	79	123	140	188	324	
Z	4	2	143	535	550	588	602	
Z	8	WAVES NOT DETECTE PROPERLY						
Z	8	WAVES NOT DETECTE PROPERLY						

TABLE 4.26. The P,Q, R,S AND T WAVE PEAKS FOR 4 CSE DATA WALSH TRANSFORM ANALYSIS

Input Lead Name	Compre Ratio	.peak no.	P	Q	R	S	T	
L1	1	1	33	189	202	215	358	
L1	2	1	117	189	202	214	358	
L1	4	WAVES NOT DETECTE PROPERLY						
L1	8	1	56	95	120	160	360	
L2	1	WAVES NOT DETECTE PROPERLY						
L2	2	1	119	171	201	220	368	
L2	4	WAVES NOT DETECTE PROPERLY						
L2	8	1	207	287	364	432	444	
L3	1	WAVES NOT DETECTE PROPERLY						
L3	2	1	118	163	199	222	376	
L3	4	WAVES NOT DETECTE PROPERLY						
L3	8	programe fail						
V1	1	programe fail						
V1	2	programe fail						
V1	4	1	107	135	192	200	272	
V1	8	WAVES NOT DETECTE PROPERLY						
V2	1	programe fail						
V2	2	1	195	277	354	424	438	
V2	4	WAVES NOT DETECTE PROPERLY						
V2	8	1	199	279	354	408	434	
V3	1	WAVES NOT DETECTE PROPERLY						
V3	2	1	195	277	357	432	437	
V3	4	WAVES NOT DETECTE PROPERLY						
V3	8	1	199	279	358	432	435	
V4	1	WAVES NOT DETECTE PROPERLY						
V4	2	1	119	175	203	212	358	
V4	4	WAVES NOT DETECTE PROPERLY						
V4	8	1	123	175	204	216	352	
V5	1	WAVES NOT DETECTE PROPERLY						
V5	2	1	122	189	203	216	358	
V5	4	WAVES NOT DETECTE PROPERLY						
V5	8	1	123	191	204	216	360	
V6	1	programe fail						
V6	2	programe fail						
V6	4	WAVES NOT DETECTE PROPERLY						
V6	4	2	203	567	570	596	768	
V6	8	1	123	191	204	216	360	
AR	1	1	14	200	218	293	512	
AR	2	WAVES NOT DETECTE PROPERLY						
AR	4	1	23	203	218	296	512	
AR	8	WAVES NOT DETECTE PROPERLY						
AL	1	1	8	51	99	120	202	
AL	1	2	202	476	513	542	640	
AL	2	programe fail						
AL	4	programe fail						

AL	8	programe	fail					
AF	1	WAVES	NOT	DETECTE	PROPERLY			
AF	2	WAVES	NOT	DETECTE	PROPERLY			
AF	4	1	119	163	200	220	364	
AF	8	WAVES	NOT	DETECTE	PROPERLY			
X	1	1	8	188	203	215	358	
X	2	WAVES	NOT	DETECTE	PROPERLY			
X	4	1	27	187	203	212	360	
X	8	WAVES	NOT	DETECTE	PROPERLY			
Y	1	1	119	172	200	220	365	
Y	2	WAVES	NOT	DETECTE	PROPERLY			
Y	4	programe	fail					
Y	8	1	207	295	368	416	448	
Z	1	programe	fail					
Z	2	WAVES	NOT	DETECTE	PROPERLY			
Z	4	1	27	195	215	292	524	
Z	8	WAVES	NOT	DETECTE	PROPERLY			

TABLE 4.27. The P,Q, R,S AND T WAVE PEAKS FOR 21 CSE. DA TA WALSH TRANSFO RM ANALYSIS

CHAPTER - 5

RESULTS AND DISCUSSION

In the present work, the FFT and FWT transformative techniques have been used for the analysis of ECG data signal. In both the methods, L_1, L_2, L_3, V_1, V_2, V_3, ... V_6, AR, AL, AF, X, Y and Z leads of different CSE data have been used for analysis. In both the techniques, the comparison of original and reconstructed signal has been done in the following three ways for different compression ratios (i.e. 1:2, 1:4 and 1:8) :

(1) PRD calculation

(2) Visual inspection

(3) Detection of P,Q,R,S and T wave peaks

Although the results are presented in table form (Tables 4.3 to 4.18) for 8 sets of ECG signals for FFT and for 8 sets for FWT, in graphical form (Figs. 4.3 to 4.62, 4.67 to 4.96) for 4 sets of ECG signals and for 2 sets for FWT and in table form of peak detection (Tables 4.19 to 4.27) for 6 sets of CSE for FFT and 3 sets for FWT, but as such, the testing in graphical form for 4 sets of ECG signals for FFT and for 2 sets for FWT, but as such, the testing has been carried out on large number of sets taken from CSE data base.

For FFT with compression ratio (CR) of 1:2 by viewing the reconstructed L1 leads, we found that the shapes of different segments in reconstructed signal remains the same as that of the original signal. Similar conclusion is applicable to remaining leads L2 to Z.

For FFT with CR of 1:4, by viewing the L1 leads of different reconstructed sample, we found that the reconstructed signal has some distortion. The high frequency component is suppressed to some extent and the wave segments are clearly visible. Similar pattern is found for other leads also.

For FFT with CR of 1:8 by viewing the L1 leads of different reconstructed samples, we found the elimination of high frequency component (i.e. electrosurgical and power line noise). The wave segments are slightly distorted and may result in detection with some error.

For FWT CR of 1:2 by viewing the reconstructed leads the shapes of different segments in reconstructed signal remain the same as that of the original signal. Similar conclusion is applicable to leads L2 to Z.

For FWT with CR of 1:4 by viewing the L1 leads of different reconstructed sample we saw some discontinuity in the signal and noise is suppressed. Detection of wave segment on and off is difficult but is possible with some degree of error. Similar pattern is found for other leads also.

For FWT with compression ratio of 1:8 by viewing L1 leads of different samples of reconstructed signal we observe large signal distortion compared to original. Similar conclusion is applicable to remaining rates L2 to Z.

Finally we have compared the performance results of both the transformative techniques. The results obtained here are comparable to the

results reported by Kulkarni [62]. Now by considering the above three cases, one by one, we obtained the following results :

⇒ **PRD CALCULATION**

In both the transformative techniques, the table for original and reconstructed signal are shown in tables. 4.3 to 4.10 for Fourier Analysis of CSE ECG data and in tables 4.11 to 4.18 for Walsh Analysis of CSE ECG data for all the leads.

The PRD values are calculated and given in table are obtained, by comparing the PRD values for different samples of CSE data for all the leads for compression ratio of 1:2,1:4 and 1:8, we found that :

* The PRD values for FWT are higher than the FFT for same compression ratios.
* The PRD for different samples are different.
* The PRD values are within the reasonable limit for most of the data.
* The PRD value may be different because of presence of noise, base line wander, electrosurgical noise and 50 Hz interference signal.

⇒ **VISUAL INSPECTION**

In both the transformative techniques, the graph for original and reconstructed signal are shown in Figs. 4.3 to 4.62 for Fourier Analysis of CSE ECG data and in Figs. 4.67 to 4.96 for Walsh Analysis of CSE ECG data for all the leads.

By visual inspection of all the graphs, the following observation are made for different compression ratios.

* By viewing original graph of different CSE data, the presence of base line wander and noise is clearly visible and is different for different CSE data.

* For 1:2 compression ratio for FFT, there is very little distortion in shape by comparing with original signal.

* For 1:2 compression ratio for FWT, there is very little distortion in shape but higher than FFT, when compared with original signal.

* For 1:2 compression ratio for both the technique, all the segments are clearly visible.

* For 1:4 compression ratio for FFT, the segments are clearly visible.

* For 1:4 compression ratio for FWT in the reconstructed graph, we get same amplitude for different sample number.

* For 1:8 compression ratio, the shape is distorted and is of less diagnostic value for FWT.

* By visual inspection of both analysis, we found that with the increase in compression ratio there is a tendency to reduce the superimposed noise.

* By visual inspection of FWT analysis, we found that with the increase in compression ratio, the R peak becomes more wider.

* We can very well use the compression upto 1:8 for heart rate in both the techniques.
* For 1:8 compression ratio, the shape is some distorted but segments are clearly visible and can be used for diagnostic value for FFT.

⇒ **DETECTION OF P,Q,R,S AND T WAVE PEAKS**

The detection of P,Q,R,S and T wave peaks have been done and following are the observation are found in Figs. 4.97 to 4.99 and tables 4.19 to 4.30.

* For original signal, the P,Q,R,S and T are calculated.
* For 1:2 compression ratio, the P,Q,R,S and T wave peaks are calculated and found very little distortion by both the methods.
* For 1:4 compression ratio, the P,Q,R,S and T wave peaks are calculated and have slight distortion.
* For some CSE Data, our program fails to detect all peak because of software limitation of our program.

COMMENTS

FFT based on successive doubling principle is described in this chapter. These are tested for all leads for different CSE data.

A comparison ratio of 8 has been achieved while preserving the clinical significant information. Visual comparison reveals that the reconstructed signal contains negligible amount of noise, because of

inherent smoothing by the algorithm. This will also reduce electromyographic (EMG noise).

ECG data compression using FWT based on successive doubling method is presented. A compression ratio of 4 has been obtained while retaining the clinically important information. Reconstruction of the signal requires smoothing in order to remove discontinuities, high frequency noise and to make the signal suitable for visual examination by the cardiologists.

We see that in FWT, only real value is calculated, so computation time is fast and less computation is done, compared to FFT for the same compression ratio.

CHAPTER 6
CONCLUSION AND FUTURE SCOPE OF WORK

In the present work we have done the ECG data compression using FFT and FWT methods. The results of and both the method are compared. The above methods of ECG data compression can be very well used in transmitting large amount of ECG data through a band-limited communication channel or to store the ECG data in limited memory. The compressed ECG can be very well transmitted over the telephone network for knowing the medical experts opinion.

It has been observed that using FFT based data compression better compression ratio can be achieved with respect to Walsh based data compression, with comparatively less loss of information although FFT based data compression takes little longer time as compared to Walsh based data compression.

The algorithm for detection of P,Q,R,S and T wave peaks needs further modification in order to use it for all types of ECG CSE data. This algorithm can further be improved to detect different segments and intervals. Which will enable to have a more comparison of data compression techniques, in terms of loss in diagnostic value. The ECG data can be filtered before using it for the detection of peaks. Further work can be carried out to see the comparative features of different orthogonal transform techniques, for ECG data compression and their usage for compression of ECG signals.

REFERENCE

1. Abenstein J.P. and Tompkins W.J., "New data-reduction algorithm for real-time ECG analysis", IEEE Trans. Biomed. Eng., Vol.BME-29, pp. 43-48, Jan.1982.
2. Ahmed, N., Milne P.J., and Harris, S.G., 1975, "Electrocardiographic data compression via orthogonal transforms", IEEE Transactions on Biomedical Engineer, 22, 484-487.
3. Andrews C.A., Davies J.M., and Schwarz G.R., "Adaptive data compression", Proc. IEEE, Vol.55, pp. 267-277, Mar. 1967.
4. Benelli G., Cappellini V., and Lotti F., "Data compression techniques and applications", Radio Electron. Eng. Vol. 50, no. 1/2, pp.29-53, 1980.
5. Berti E., Chiaraluce F., Evans N.E. and Mekee J.J. "Double logarithmic quantisation of the Walsh spectrum: application to real ECGs", Electronics Letters 28th Aug. 1997, Vol. 33, No. 18 Pg. 1513-1515.
6. Blerkom R.Van, Schwarz G.R., and Ward R.J., "An adaptive compression,"An adaptive composite data compressiuon algorithm with reduced computation reuuirements", in Proc. 1968 Nat. Telemetry Conf. 1968, pp. 90-95.
7. Cady L.D., Woodbury M.A., Tick L. J., and Gertler M.M., 1961, "A method for electrocardiogram wave-pattern estimation", Circulation Research, 9, 1078-1082.
8. Cox J.R., Nolle F.M., Fozzard H.A., and Oliver G.C., "AZTEC, a preprocessing program for real-time ECG rhythm analysis", IEEE Trans. Biomed Eng., Vol.BME - 15, pp. 128-129, Apr.1968.
9. Cox J.R., Fozzard H.A., Nolle F.M., and Oliver G.C., "Some data transformations useful in electrocardiography," in Computers and Biomedical Research Vol.III, R.W. Stacy and B.D. Waxman,Eds. Vol.III.New York: Academic, 1974,pp.181-206.
10. Cox J.R., Nolle F.M., and Arthur R.M., "Digital analysis of the electroencephalogram, the blood pressure wave, and the ECG", Proc. IEEE, Vol. 60, pp. 1137-1164, Oct. 1972.

11. Cox J.R. and Tipley K.L., "Compact digital coding of electrocardiographic data", in Proc. VI Int. Conf. Syst. Sci., Jan.1973. pp. 333-336.
12. Cromwell. L., Weibell. F.J. and Pfieffer. E.A., "Biomedical Instrumentation and Measurments", Second Edition, Prentice Hall,1995.
13. ---,"Data compression using straight line interpolation," IEEE Trans, Inform. Theory, Vol.IT-14, pp. 390-394, May 1968.
14. ---, "Data compression of Holter ECG's", M.S. thesis, Univ. Tulsa, Tulsa, OK, 1987.
15. ---,"Date compression and the quality of the reconstructed ECG", in Optimitation of computer ECG Processing, H.K.Wolf, and P.W. MacFarlane, Eds. New Yourk: North-Holland, 1980, pp.77-85.
16. Davisson L.D., "The theroretical analysis of data compression systems," Proc.IEEE, Vol.56, pp. 176-186, Feb. 1968.
17. Davisson L.D., "An approximation of prediction for data compression," IEEE Trans, Inform. Theory, Vol.IT-13, pp. 274-278, Apr. 1967.
18. De Jager.F., "Delta Modulation, a method of pcm transmission using the I-unit code", Philips Res. 7, pp. 442-66, December 1952.
19. Elias P., "Predictive coding - Part I and Part II," IRE Trans. Inform. Theory, Vol.IT - I,pp.16-33, Mar.1955.
20. Ehrman L., "Analysis of some redundancy removal bandwidth compression techniques", Proc.IEEE, Vol.55, pp. 278-287, Mar. 1967.
21. Freidmen, H. "Diagnostic electrocardiography and vector cardiography". McGraw Hill, 1971.
22. Gardenhire L.W., "Redundancy reduction the key to adaptive telemetry", in Proc. 1964 Nat. Telemetry Conf. 1964, pp. 1-16.
23. Gonzalez R.C., Woods R.E. "Digital Image Processing", Page No. 119-136.
24. Hambley, A.R., Moruzzi, R.L., and Feldman, C.L., 1974, "The use of intrinsic components in an ECG filter", IEEE Transaction on Biomedical Engineering, 21, 469-479.
25. Holter N.J., "New method for heart studies", Science, Vol. 134, pp. 1214-1220, 1961.
26. Huffman D.A., "A method for the construction of minimum-redundancy codes", Proc. IRE, Vol.40, pp.1098-1101, Sept.1952.

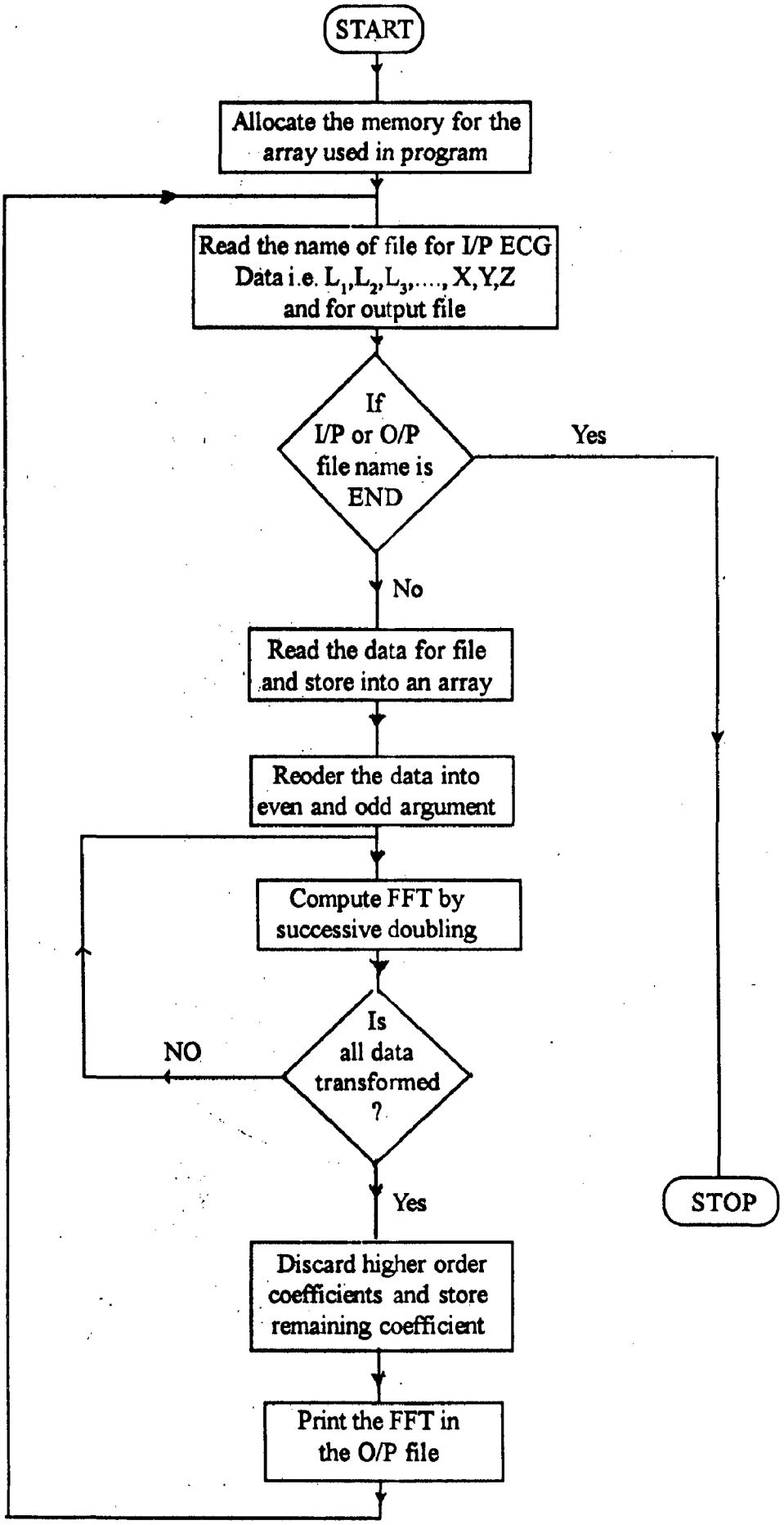
27. Imai H., Kimura N., and Yoshida Y., "An efficient encoding method for electorcardiography using Spline functions", Syst. Comput. Japan, Vol. 16, no.3, pp. 85-94, 1985.
28. Ishijima M., Shin S.B., Hostetter G.H., and Sklansky J., "Scanalong polygon approximation for data compression of electrocardigrams", IEEE Trans. Biomed. Eng., vol. BME-30, pp. 723-729, Nov. 1983.
29. Jain A.K., "Image data compression", Proc. IEEE, Vol. 69,pp. 349-389, Mar.1981.
30. Jalaleddine S.M.S., Hutchens C.G., Coberly W.A., and Strattan R.D., "Compression on Holter ECG data", Biomed. Sci. Instrument., Vol. 24, pp. 35-45, Apr. 1988.
31. S.M.S. Jalaleddine, Hutchens G.C., Strattan R.D. and Coberly W.A., "ECG data compression techniques - A unified approach," IEEE Trans. Biomed. Eng., vol. BME-37, pp. 329-343, Apr. 1990.
32. Kao T., Beverly C.Yu and Chung-yin Chen, "ECG data compression for digital transmission on telephone lines", Journal of Clinical Engineering, Vol.14, No. 5, Sept. Oct. 1989, Pg. 431-438.
33. Kortman C.M., "Redundancy reduction-a practical method of data compression", Proc. IEEE, Vol. 55, pp.253-263, Mar.1967.
34. Krishnakumar A.S., Karpowicz J.L., Belic N, Singer D.H., and Jenkins J.M., "Microprocessor-based data compression scheme for enhanced digital transmission of Holter recordings", Computers Cardiol. Long Beach, CA, PP. 435-437, 1980.
35. Kuklinski W.S., "Fast Walsh transform data-compression algorithm; ECG applications", Med. Biol. Eng. Comput., Vol. 21, pp. 465-472, July 1983.
36. Kulkarni P.K., Kumar V. and Verma H.K., "Diagnostic acceptability of FFT-based ECG data compression", Journal of Medical Engineering and Technology, Vol. 21, No. 5, (Sept. - Oct. 1997), pages 185-189.
37. Lowenberg E.C., 1960, "Signal theory applied to the analysis of electrocardiograms",IRE Transactions on Medical Electronics, 7, 7-12.
38. Makhoul J., "Linear prediction: A tutorial review", Proc.IEEE, Vol.63, pp. 561-580, April 1975.
39. Medlin J.E., "Sampled-data prediction for telemetry bandwidth compression", IEEE Trans. Space Electron. Telem., Vol. SET - 11, pp. 29-36, Mar.1965.

40. Mueller W.C., "Arrhythmia detection program for an ambulatory ECG monitor", Biomed. Sci. Instrument., Vol. 14. pp. 81-85,1978.
41. Oliver B.M., "Effcient coding", Bell Syst. Tech. J., Vol.31, pp.724-750, July 1952.
42. Pahlm G. and Sornomo, L. "Software QRS detection in ambulatory monitory, Medical, Biological Engineering and Computing", vol. 22, pp. 289-297, 1994.
43. Pan J. and Tompkins, W.J. "A real-time QRS detection algorithm", IEEE Transactions on BME, vol. 32, pp. 230-236, 1985.
44. Philips W. and Jonghe G.D. "Data Compression of ECG's by High-Degree Polynomial Approximation", IEEE Transactions on Biomedical Engineering, Vol. 39, No.4, April 1992.
45. Rao K.D., "DWT Based detection of R-peaks and data compression of ECG signals", IETE Journal of Research Vol. 43, No. 5, Sept. - Oct. 1997, pp. 345-349.
46. Reddy B.R.S. and Munthy I.S.N., "ECG data compression using Fourier descriptors", IEEE Trans. Biomed. Eng., Vol. BME-33, pp. 428-434, Apr. 1986.
47. Relations Between Haar and Walsh/Hadamard Transforms, Proceeding Letters, Proceedings of the IEEE, May 1972, Page No. 647-648.
48. Ruttimann U.E. and Pipberger H.V., "Compression of the ECG by prediction or interpolation and entropy encoding", IEEE Trans. Biomed Eng., Vol.BME - 26, pp. 613-623, Nov.1979.
49. Saxena S.C., Sharma A. and Chaudhary S.C., "Data compression and feature extraction of ECG signals", International Journal of Systems Science, 1977, vol. 28, no. 5, pages 483-498.
50. Schafer R.W., and Rabiner L.R.", Proc.IEEE, Vol.61, pp. 692-702, June 1973.
51. Shanks J.L., "Computation of the Fast Walsh-Fourier Transform", IEEE Transactions on Computers, May 1969, Page 457-459.
52. Some Notes on the Walsh Functions, IEEE Transactions on Electronic Computer, 1964, Page No. 50-52.
53. Special issue on redundancy reduction, Proc.IEEE, Vol.55, Mar. 1967.
54. Steele.R., "Delta Modulation Systems", (Pentech Press, London,1975).
55. Stewart D., Dower G.E., and Suranyi O, "An ECG compression code", J. Electocardiol., Vol. 6. no.2, pp. 175-176. 1973.

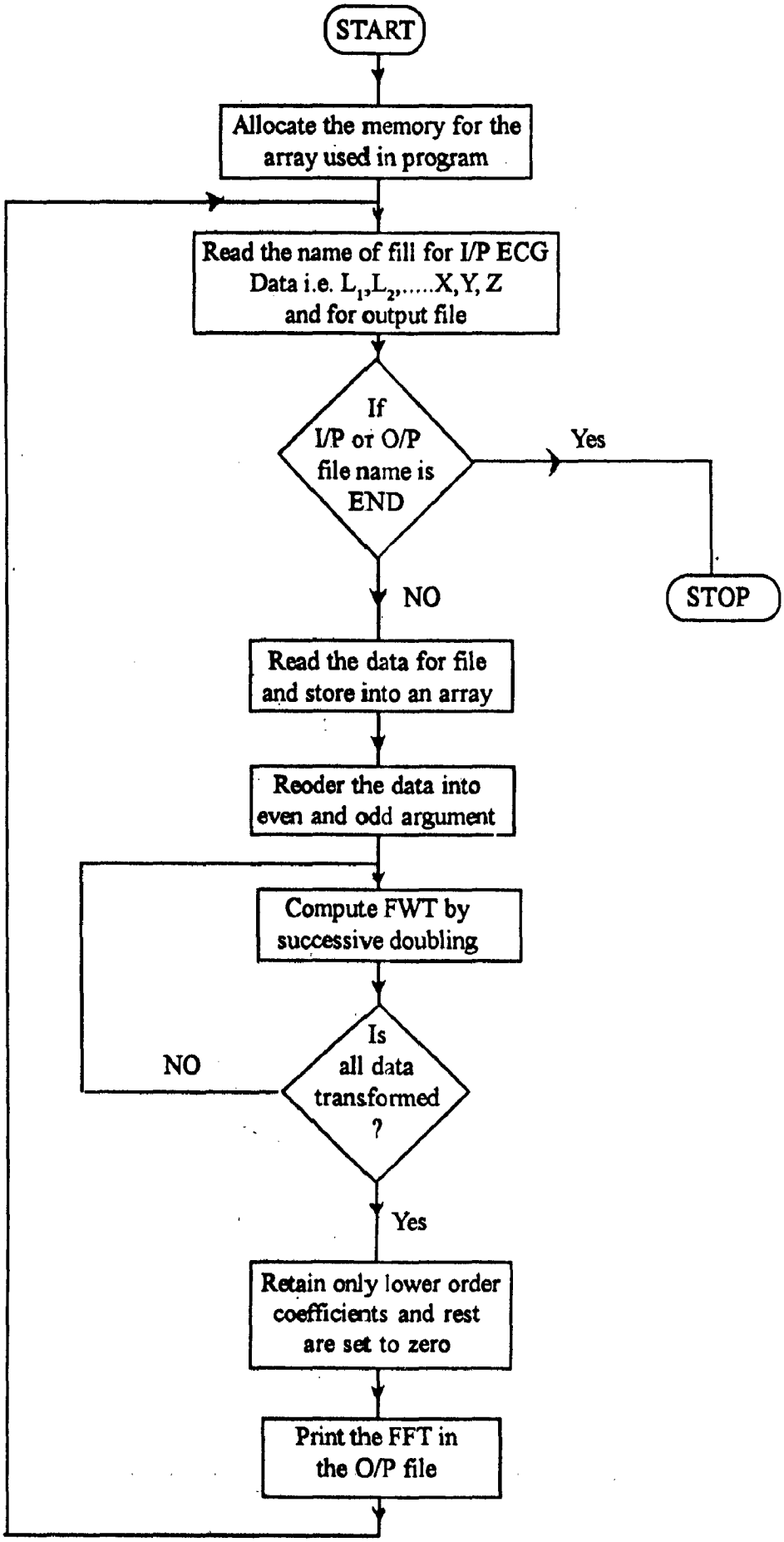
56. The CSE Working Party "Recommendations for ambulatory subjects", Biomedical Science Instramentation, Vol. 14, pp.67-72, 1978.
57. Trahanians P. and Skordalakis E. "Syntactic pattern recognition of the ECG, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 12, pp. 349-357, 1990.
58. Weber D.R. and Wynhoff F.J., "The concept of self-adaptive data compression", in Proc. IRENat. Symp. Space Electron. Telemetry, IEEE LG-SET REC., Sect. 4.1, 1962, pp. 1-10.
59. Willems, "Common Standards of quantitative electrocardiography", J. Med. Eng. Techn., Vol.9,m pp 209-217, 1985.
60. Womble M.E., Halliday J.S., Mitter S.K., Lancaster M.C., and Triebwasser J.H., "Data compression for storing and transmitting ECGs/VCGs", Proc. IEEE, Vol. 65, pp. 702-706, May 1977.
61. Young, T.Y., and Huggings, W.H., 1963, "On the representation of electrocardiograms", IEEE Transactions on Biomedical Engineering, 10, 86-95.
62. Kukkarni P.K. "Ambulatary monitoring and analysis of ECG Signals", Ph.D. Thesis, September, 1997.
63. Anand R.S. and Kumar V., "Efficient and reliable detection of QRS segment in ECG signals," Proceedings First Regional Conference of IEEE Engg. in Medicine and Biology Society New Delhi, Feb. 1995, pp. 2.56-2.57.

APPENDIX-I

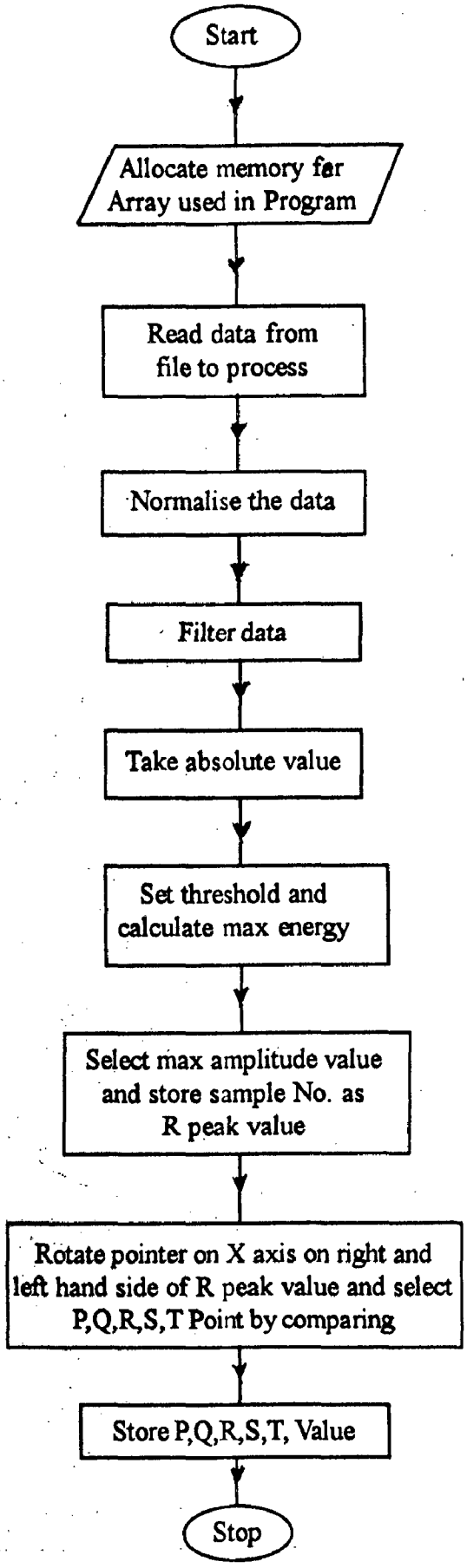
FLOW CHART



FLOW CHART FOR FFT BASED ECG DATA COMPRESSION ALGORITHM



FLOW CHART FOR FWT BASED ECG DATA COMPRESSION ALGORITHM



Flow chart for detecting P,Q,R,S and T Peak value of ECG signal

APPENDIX-II

LISTING OF PROGRAM

```
/* program for fft determination */
/* file = f1.c */
#include <stdio.h>
#include <math.h>
#include <dos.h>
#include  <graphics.h>
#include <conio.h>
#include <alloc.h>
#include <io.h>
FILE *fp1,*fp2;
float max,min;
int nr,x1,x2,y1,y2;
float xmax,ymax,xmin,ymin;
char str1[40],str2[40];
int ntype, log2n;

main()
{
      FILE *fp10,*fp20;
      void fft(float *xr, float *xi);
      void graph(float *xr, float *ar);
      void amax(float *ar);
      int i,nr_start,nr_end,lr,strt,cmp;
      char input[40],output[40],ch,*temp1,*temp2;
      float *ar,*xr,*xi,*yr,*ti;
      float a,b,ts,tsamp,fsamp,tr;
      clrscr();
      lr = 1100 ;
      yr = (float *) malloc(lr * sizeof(float));
      if(yr == NULL )
      {
      printf("Insufficient memory\n");
      exit(1);
      }
      ti = (float *) malloc(lr * sizeof(float));
      if(ti == NULL )
      {
      printf("Insufficient memory\n");
      exit(1);
      }
      xr = (float *) malloc(lr * sizeof(float));
      if(xr == NULL )
      {
      printf("Insufficient memory\n");
      exit(1);
      }
      ar = (float *) malloc(lr * sizeof(float));
      if(ar == NULL )
      {
      printf("Insufficient memory\n");
      exit(1);
      }
      xi = (float *) malloc(lr * sizeof(float));
      if(xi == NULL )
      {
      printf("Insufficient memory\n");
      exit(1);
```

```
      }
      fp10  = fopen("1f.dat","r");
        fp20  = fopen("2f.dat","r");
      again :
      temp1 = (char*)malloc(20 * sizeof(char));
      fscanf(fp10,"%s",temp1);
      cmp = strcmp(temp1,"END");
      if (cmp == 0)
      goto end1;
      strcpy(input,temp1);
      fp1 = fopen(input,"r");
      temp2 = (char*)malloc(20 * sizeof(char));
      fscanf(fp20,"%s",temp2);
      cmp = strcmp(temp2,"END");
      if (cmp == 0)
      goto end1;
      if (temp2 == "null")
      goto end1;
      strcpy(output,temp2);
      fp2 = fopen(output,"w");
      fscanf(fp1,"%d\n",&lr);
/*    printf(" Number of points in file are %d\n",lr);
*/    printf("%s    %s\n",input,output);
      for(i=0; i <= lr-1; ++i)
      {
      fscanf(fp1,"%f %f\n",&ts,&tr);
      ti[i] = ts;
      yr[i] = tr;
      }
      fclose(fp1);
      tsamp = ti[1]-ti[0];

      do {
/*    printf("\n");
      printf("Input the start point : ");
      scanf("%d",&nr_start);
      printf("Input the End point (diff <= 4096) : ");
      scanf("%d",&nr_end);
*/      nr_start = 0;
      nr_end = 1024;
      nr = nr_end - nr_start ;
/*    printf("The number of points specified are %d\n",nr);
*/    strt=1;
      while(strt < nr)
                  strt=strt*2;
      if(strt > nr)
        {
          printf("This number %d is not in power of 2 \n",nr);
          printf(" Please modify it to powers of 2 and feed the number\n");
          printf(" Specified number is: ");
          scanf("%d",&nr);
        }
      fsamp =(1.0/(ti[1]-ti[0]))/nr;
      if(nr > (lr-nr_start))
       {
          for(i=lr; i<nr+nr_start; i++)
            {
```

```
            yr[i] = 0.0;
            ti[i] = i*tsamp;
           }
       }
/* fft program part starts from here */

      log2n = apws(nr); ntype = 1;
      for(i=0; i<= nr-1; ++i)
      {
            xr[i+1] = yr[i+nr_start];
            xi[i+1] = 0.;
       }

       fft(xr,xi);

/*    nr = (nr/2)+1;  */
      fprintf(fp2,"%d\n",nr);
      for(i=1; i<=nr; ++i)
      {
            a = xr[i];
            b = xi[i];
            /*xr[i] = sqrt(a*a + b*b);*/
            ts = (i-1)*fsamp;
            ar[i-1] = ts;
            /*xi[i] = 0.; */
            fprintf(fp2,"%4.1f %2.3f %2.3f\n",ts,xr[i],xi[i]);
      }
      rewind(fp2);
      for (i=1; i <= nr; ++i)
      ar[i-1] = xr[i];
      for(i=0; i<= nr-1; ++i)
      xr[i] = i*fsamp;
      } while (ch == 'y' || ch == 'y');
      fclose(fp2);
      goto again;
      end1:
      fclose(fp10);
      getch();
      fclose(fp20);
}
void fft(float *xr, float *xi)
{
      float sign,ain;
      double ur,ui,tr,ti,wr,wi,pi;
      int n,nv2,nm1,j,i,k,l,le,le1,ip;
      sign = -1.0;
      if(ntype < 0 ) sign = 1.0;
      n = pws(2,log2n);
      nv2 = n/2;
      nm1 = n-1;
      j = 1;
      for(i=1; i<=nm1; i++)
      {
      if(i < j)
           {
           tr = xr[j];
           ti = xi[j];
```

```
            xr[j] = xr[i];
            xi[j] = xi[i];
            xr[i] = tr;
            xi[i] = ti;
            }
             k = nv2;
          while ( k < j)
              {
               j = j-k;
               k =k/2;
              }
           j = j+k;
           }

           pi = 4.*atan(1.);

        for(l=1; l<=log2n; l++)
         {
          le = pws(2,l);
          le1 = le/2;
          ur = 1.;
          ui = 0.;
          wr = cos(pi/le1);
          wi = sign*sin(pi/le1);

          for(j=1; j <= le1; j++)
            {
              for(i=j; i<=n; i=i+le)
               {
               ip = i + le1;
               tr = xr[ip]*ur - xi[ip]*ui;
               ti = xr[ip]*ui + xi[ip]*ur;
               xr[ip] = xr[i] - tr;
               xi[ip] = xi[i] - ti;
               xr[i] = xr[i] + tr;
               xi[i] = xi[i] + ti;
               }

            tr = ur*wr - ui*wi;
            ti = ur*wi + ui*wr;
            ur = tr;
            ui = ti;
            }
          }
          if(ntype <= 0)
          {
         ain = 1./n;

             for(i=1; i<=n; i++)
              {
               xr[i] = xr[i]*ain;
               xi[i] = xi[i]*ain;
              }
          }
}
pws(x,y)
{
```

```
        int z,i;
        z = 1;
        for (i=0; i<=y-1; ++i)
        z = x*z;
        return z;
}
apws(nr)
{
        int i;
        i = 0;
        while ( nr >1 )
        {
                nr = nr/2;
                i=i+1;
        }

        return i;
}

void amax(float *ar)
{
        int i;

        max = ar[0];
        min = ar[0];

        for(i=1; i <= nr-1; ++i)

        {
        if( max < ar[i] )
                max = ar[i];
        else
                max = max;
        }

        for(i=1; i <= nr-1; ++i)
        {
        if(min > ar[i] )
                min = ar[i];
        else
                min = min;
        }

}
```

```
/* program for fft inverse determination */
/* file = 1i.c */
#include <stdio.h>
#include <math.h>
#include <dos.h>
#include  <graphics.h>
#include <conio.h>
#include <alloc.h>
FILE *fp1,*fp2;
float max,min;
int nr,x1,x2,y1,y2;
float xmax,ymax,xmin,ymin;
char str1[40],str2[40];
int ntype, log2n;

main()
{
      FILE *fp10,*fp20;
      void fft(float *xr, float *xi);
      void graph(float *xr, float *ar);
      void amax(float *ar);
      int i,nr_start,nr_end,lr,strt,cmp;
      char input[40],output[40],ch,*temp1,*temp2;
      float *ar,*xr,*xi,*yr,*ti;
      float a,b,ts,tsamp,fsamp,tr,t2;
      clrscr();
      lr = 1100;
        yr = (float *) malloc((lr+1) * sizeof(float));
      if(yr == NULL )
      {
      printf("Insufficient memory\n");
      exit(1);
      }
      ti = (float *) malloc((lr+1) * sizeof(float));
      if(ti == NULL )
      {
      printf("Insufficient memory\n");
      exit(1);
      }
      xr = (float *) malloc((lr+1) * sizeof(float));
      if(xr == NULL )
      {
      printf("Insufficient memory\n");
      exit(1);
      }
      ar = (float *) malloc((lr+1) * sizeof(float));
      if(ar == NULL )
      {
      printf("Insufficient memory\n");
      exit(1);
      }
      xi = (float *) malloc((lr+1) * sizeof(float));
      if(xi == NULL )
      {
      printf("Insufficient memory\n");
      exit(1);
```

```
        }
        fp10  = fopen("2fo.dat","r");
          fp20  = fopen("3f.dat","r");
        again :
        temp1 = (char*)malloc(20 * sizeof(char));
        fscanf(fp10,"%s",temp1);
        cmp = strcmp(temp1,"END");
        if (cmp == 0)
        goto end1;
        strcpy(input,temp1);
        temp2 = (char*)malloc(20 * sizeof(char));
        fscanf(fp20,"%s",temp2);
        if (temp2 == 5)
        goto end1;
        if (temp2 == "null")
        goto end1;
        strcpy(output,temp2);
        printf("%s       %s\n",input,output);
        fp1 = fopen(input,"r");
        fp2 = fopen(output,"w");
        fscanf(fp1,"%d\n",&lr);
/*      printf(" Number of points in file are %d\n",lr);
*/      for(i=0; i <= lr-1; ++i)
        {
        fscanf(fp1,"%f %f %f\n",&ts,&tr,&t2);
        ti[i] = tr;
        yr[i] = t2;
        }
        fclose(fp1);
        fsamp = ti[1]-ti[0];

        do {
/*      printf("\n");
        printf("Input the start point : ");
        scanf("%d",&nr_start);
        printf("Input the End point (diff <= 4096) : ");
        scanf("%d",&nr_end);
*/        nr_start = 1;
        nr_end = 1025;
        nr = nr_end - nr_start ;
/*      printf("The number of points specified are %d\n",nr);
*/      strt=1;
        while(strt < nr)
                    strt=strt*2;
        if(strt > nr)
          {
            printf("This number %d is not in power of 2 \n",nr);
            printf(" Please modify it to powers of 2 and feed the number\n");
            printf(" Specified number is: ");
            scanf("%d",&nr);
          }
        tsamp =(1.0/(ti[1]-ti[0]))/nr;
        if(nr > (lr-nr_start))
         {
            for(i=lr; i<nr+nr_start; i++)
              {
               xr[i] = ti[i];
```

```
            xi[i] = yr[i];
           }
       }
/* fft program part starts from here */

      log2n = apws(nr); ntype = -1;
      for(i=0; i<= nr-1; ++i)
      {
            xr[i+1] = ti[i+nr_start-1];
            xi[i+1] = yr[i+nr_start-1];
       }

       fft(xr,xi);

/*    nr = (nr/2)+1;*/
      fprintf(fp2,"%d\n",nr);
      for(i=1; i<=nr; ++i)
      {
            a = xr[i];
            fprintf(fp2,"%d %1.3f\n",i,xr[i]);
      }
      rewind(fp2);
      for (i=1; i <= nr; ++i)
     ar[i-1] = xr[i];
      for(i=0; i<= nr-1; ++i)
      xr[i] = i*fsamp;
      } while (ch == 'y' || ch == 'y');
      fclose(fp2);
      goto again;
      end1 :
      fclose(fp10);
      getch();
      fclose(fp20);
}
void fft(float *xr, float *xi)
{
      float sign,ain;
      double ur,ui,tr,ti,wr,wi,pi;
      int n,nv2,nm1,j,i,k,l,le,le1,ip;
      sign = -1.0;
      if(ntype < 0 ) sign = 1.0;
      n = pws(2,log2n);
      nv2 = n/2;
      nm1 = n-1;
      j = 1;
      for(i=1; i<=nm1; i++)
      {
      if(i < j)
            {
            tr = xr[j];
            ti = xi[j];
            xr[j] = xr[i];
            xi[j] = xi[i];
            xr[i] = tr;
            xi[i] = ti;
            }
             k = nv2;
```

```
        while ( k < j)
            {
             j = j-k;
             k =k/2;
            }
         j = j+k;
         }

         pi = 4.*atan(1.);

      for(l=1; l<=log2n; l++)
       {
        le = pws(2,l);
        le1 = le/2;
        ur = 1.;
        ui = 0.;
        wr = cos(pi/le1);
        wi = sign*sin(pi/le1);

        for(j=1; j <= le1; j++)
          {
           for(i=j; i<=n; i=i+le)
            {
            ip = i + le1;
            tr = xr[ip]*ur - xi[ip]*ui;
            ti = xr[ip]*ui + xi[ip]*ur;
            xr[ip] = xr[i] - tr;
            xi[ip] = xi[i] - ti;
            xr[i] = xr[i] + tr;
            xi[i] = xi[i] + ti;
            }

          tr = ur*wr - ui*wi;
          ti = ur*wi + ui*wr;
          ur = tr;
          ui = ti;
          }
        }
        if(ntype <= 0)
        {
       ain = 1./n;

           for(i=1; i<=n; i++)
            {
             xr[i] = xr[i]*ain;
             xi[i] = xi[i]*ain;
            }
         }
}
pws(x,y)
{
      int z,i;
      z = 1;
      for (i=0; i<=y-1; ++i)
      z = x*z;
      return z;
}
```

```
apws(nr)
{
        int i;
        i = 0;
        while ( nr >1 )
        {
              nr = nr/2;
              i=i+1;
        }

        return i;
}

void amax(float *ar)
{
        int i;

        max = ar[0];
        min = ar[0];

        for(i=1; i <= nr-1; ++i)

        {
        if( max < ar[i] )
               max = ar[i];
        else
              max = max;
        }

        for(i=1; i <= nr-1; ++i)
        {
        if(min > ar[i] )
               min = ar[i];
        else
              min = min;
        }

}
```

```
/*prog. to modify two line data for Walsh Trasformation for t=1024,
and value of data upto (a-1)  is zero*/
#include <stdio.h>
#include <math.h>
#include <dos.h>
#include <conio.h>
#include <alloc.h>
main()
{
      int i=0,n,t=1024,a,c,b,q;
      int cmp,j;
      FILE *fp1,*fp2,*fp10,*fp20;
      float *x,*y,*z;
      float u,v,w;
      char *temp1,*temp2,input[40],output[40];
      clrscr();
        y = (float *) malloc(t * sizeof(float));
      if(y == NULL )
      {
      printf("Insufficient memory\n");
      exit(1);
      }
      x = (float *) malloc(t * sizeof(float));
      if(x == NULL )
      {
      printf("Insufficient memory\n");
      exit(1);
      }
      z = (float *) malloc(t * sizeof(float));
      if(z == NULL )
      {
      printf("Insufficient memory\n");
      exit(1);
      }
      fp10  = fopen("wo.dat","r");
            fp20  = fopen("2wo.dat","r");
      again :
      n=0;
       temp1 = (char*)malloc(20 * sizeof(char));
      fscanf(fp10,"%s",temp1);
      cmp = strcmp(temp1,"END");
      if (cmp == 0)
      goto end1;
      strcpy(input,temp1);
      fp1 = fopen(input,"r");
        fscanf(fp1,"%d",&q);
        for(i=0;i<=t;i++,n++)
       {
       fscanf( fp1," %f  %f   \n",&u,&v);
       x[n]=u;
       y[n]=v;
       }

      for (j=0;j<=2;++j)
      {
      temp2 = (char*)malloc(20 * sizeof(char));
      fscanf(fp20,"%s",temp2);
```

```
        cmp = strcmp(temp2,"END");
        if (cmp == 0)
        goto end1;
        if (temp2 == "null")
        goto end1;
        strcpy(output,temp2);
         fp2=fopen(output,"w");
         if (j==0)
         a=512;
         if (j==1)
         a=256;
         if (j==2)
         a=128;
         fprintf(fp2,"%d\n",q);
         for(n=0;n<a;++n)
         fprintf(fp2,"%f   %f   \n",x[n],y[n]);
         for(n=a;n<=t;++n)
         {
         y[n]=0.0;
         fprintf(fp2,"%f   %f   \n",x[n],y[n]);
         }
         printf("%s\t %s\n",input,output);
         fclose(fp2);
        }
        fclose(fp1);
        goto again;
        end1:
        fclose(fp10);
        getch();
        fclose(fp20);
}
```

```
/*prog. to calculate PRD data  */
#include <stdio.h>
#include <math.h>
#include <dos.h>
#include  <graphics.h>
#include <conio.h>
#include <alloc.h>
main()
{
      int i=0,n,t=1024,a,b,c,d;
      FILE *fp1,*fp2,*fp10,*fp20,*fp30;
      int cmp,j,k=0;
      char *temp1,*temp2,input[40],output[40];
      float *x,*y,*z,*xa,*ya,*za,p[50];
      float q,s,u,v,w,prd=0.0,tot=0.0,sum=0.0,sum1=0.0,tot1=0.0;
      x = (float *) malloc(t * sizeof(float));
      if(x == NULL )
      {
      printf("Insufficient memory\n");
      exit(1);
      }
      y = (float *) malloc(t * sizeof(float));
      if(y == NULL )
      {
      printf("Insufficient memory\n");
      exit(1);
      }
      xa = (float *) malloc(t * sizeof(float));
      if(xa == NULL )
      {
      printf("Insufficient memory\n");
      exit(1);
      }
      ya = (float *) malloc(t * sizeof(float));
      if(ya == NULL )
      {
      printf("Insufficient memory\n");
      exit(1);
      }
      clrscr();
      fp30 = fopen("15f.dat","w");
      fprintf(fp30,"\n\t 512     256     128\n");
      fp10  = fopen("1f.dat","r");
      fp20  = fopen("3f.dat","r");
      again :
      n=1;
        temp1 = (char*)malloc(20 * sizeof(char));
      fscanf(fp10,"%s",temp1);
      cmp = strcmp(temp1,"END");
      if (cmp == 0)
      goto end1;
      strcpy(input,temp1);
      fp1 = fopen(input,"r");
      fscanf(fp1,"%d",&c);
        for(i=0;i<=t;i++,n++)
       {
       fscanf( fp1," %f  %f   \n",&u,&v);
```

```
        x[n]=u;
        y[n]=v;
        }
       for (j=0;j<=2;++j)
       {
       temp2 = (char*)malloc(20 * sizeof(char));
       fscanf(fp20,"%s",temp2);
       cmp = strcmp(temp2,"END");
       if (cmp == 0)
       goto end1;
       if (temp2 == "null")
       goto end1;
       strcpy(output,temp2);
       fp2=fopen(output,"r");
       n = 1 ;
       fscanf( fp2," %d \n",&d);
       for(i=0;i<=t;i++,n++)
       {
       fscanf( fp2," %f %f \n",&q,&s);
       xa[n]=q;
       ya[n]=s;
       }
      n=1;
       prd = 0.0;
       sum = 0.0;
       sum1 = 0.0;
       tot = 0.0;
       tot1 = 0.0;
        for(n=1;n<=t;n++)
       {
        sum1 = (y[n]-ya[n]) ;
        sum1 = sum1 * sum1 ;
       tot1 = y[n] * y[n];
       sum+=sum1;
       tot+=tot1;
       }
        prd = sum/tot;
        prd = sqrt(prd);
/*       printf("prd=%f\n",prd);
*/       prd = prd * 100.0 ;
/*       printf("prd=%3.2f\n",prd);
*/       p[j] = prd ;
/*       printf("\n%s\t %s\n",input,output);
        printf(" p[%d] = %f\t\n",j,p[j]);
        getch();
 */       fclose(fp2);
        }
       k=k+1;
       if(k==1)
       fprintf(fp30,"L1\t");
       if(k==2)
       fprintf(fp30,"L2\t");
       if(k==3)
       fprintf(fp30,"L3\t");
       if(k==4)
       fprintf(fp30,"V1\t");
       if(k==5)
```

```
        fprintf(fp30,"V2\t");
        if (k==6)
        fprintf(fp30,"V3\t");
        if (k==7)
        fprintf(fp30,"V4\t");
        if (k==8)
        fprintf(fp30,"V5\t");
        if (k==9)
        fprintf(fp30,"V6\t");
        if (k==10)
        fprintf(fp30,"AR\t");
        if (k==11)
        fprintf(fp30,"AL\t");
        if (k==12)
        fprintf(fp30,"AF\t");
        if (k==13)
        fprintf(fp30,"X\t");
        if (k==14)
        fprintf(fp30,"Y\t");
        if (k==15)
        fprintf(fp30,"Z\t");
        fprintf(fp30,"%3.2f\t%3.2f\t%3.2f\n",p[0],p[1],p[2] );
        fclose(fp1);
        printf(" continue");
        goto again;
        end1:
        fclose(fp10);
        fclose(fp20);
        fclose(fp30);
}
```

```
/* program for forward and inverse walsh  Transform determination */
/* file = 1w.c */
#include <stdio.h>
#include <math.h>
#include <dos.h>
#include  <graphics.h>
#include <conio.h>
#include <alloc.h>
FILE *fp1,*fp2;
float max,min;
int nr,x1,x2,y1,y2;
int ntype, log2n;

main()
{
      FILE *fp10,*fp20;
      void w(float *xr);
      void graph(float *xr, float *ar);
      void amax(float *ar);
      int i,nr_start,nr_end,lr,strt,cmp,walsh,nr;
      char input[40],output[40],ch,*temp1,*temp2;
      float *ar,*xr,*xi,*yr,*ti;
      float a,b,ts,tsamp,fsamp,tr;
      clrscr();
      nr = 1050;
      yr = (float *) malloc(nr * sizeof(float));
      if(yr == NULL )
      {
      printf("Insufficient memory\n");
      exit(1);
      }
      ti = (float *) malloc(nr * sizeof(float));
      if(ti == NULL )
      {
      printf("Insufficient memory\n");
      exit(1);
      }
      xr = (float *) malloc(nr * sizeof(float));
      if(xr == NULL )
      {
      printf("Insufficient memory\n");
      exit(1);
      }
      ar = (float *) malloc(nr * sizeof(float));
      if(ar == NULL )
      {
      printf("Insufficient memory\n");
      exit(1);
      }
      printf("Enter the value of walsh=1 if walsh transform\n");
      printf("Enter the value of walsh=-1 if inverse walsh transform\n");
      printf("walsh= ");
      scanf("%d",&walsh);
      printf("walsh=%d\n ",walsh);
      if (walsh ==1)
      {fp10  = fopen("1w.dat","r");
      fp20  = fopen("2w.dat","r");}
```

```
        log2n = apws(nr); ntype = 1;
        for(i=0; i<= nr-1; ++i)
        {
              xr[i+1] = yr[i+nr_start];
         }

         w(xr);

/*      nr = (nr/2)+1;     */
        fprintf(fp2,"%d\n",nr);
        for(i=1; i<=nr; ++i)
        {
              a = xr[i];
              ts =(i-1) *fsamp;
              if (walsh == 1)
              {xr[i] = xr[i]/nr ;
              fprintf(fp2,"%f %2.3f\n",ts,xr[i]);}
              if (walsh == -1)
              fprintf(fp2,"%f\t %f\n",ts,xr[i]);

        }
        rewind(fp2);
        for (i=1; i <= nr; ++i)
        ar[i-1] = xr[i];
        for(i=0; i<= nr-1; ++i)
        xr[i] = i*fsamp;
        } while (ch == 'y' || ch == 'y');
        fclose(fp2);
        goto again;
        end1:
        fclose(fp10);
        getch();
        fclose(fp20);
}
void w(float *xr)
{
        float sign,ain;
        double ur,ui,tr,ti,wr,wi,pi;
        int n,nv2,nm1,j,i,k,l,le,le1,ip;
        sign = -1.0;
        if(ntype < 0 ) sign = 1.0;
        n = pws(2,log2n);
        nv2 = n/2;
        nm1 = n-1;
        j = 1;
        for(i=1; i<=nm1; i++)
        {
        if(i < j)
              {
              tr = xr[j];
              xr[j] = xr[i];
              xr[i] = tr;
              }
               k = nv2;
          while ( k < j)
               {
```

```
            j = j-k;
            k =k/2;
           }
        j = j+k;
        }

        pi = 4.*atan(1.);

     for(l=1; l<=log2n; l++)
      {
       le = pws(2,l);
       le1 = le/2;
       for(j=1; j <= le1; j++)
         {
           for(i=j; i<=n; i=i+le)
            {
            ip = i + le1;
            tr = xr[ip];
            xr[ip] = xr[i] - tr;
            xr[i] = xr[i] + tr;
            }

         }
       }
       if(ntype <= 0)
       {
      ain = 1./n;

          for(i=1; i<=n; i++)
           {
            xr[i] = xr[i]*ain;
           }
        }
}
pws(x,y)
{
      int z,i;
      z = 1;
      for (i=0; i<=y-1; ++i)
      z = x*z;
      return z;
}
apws(nr)
{
      int i;
      i = 0;
      while ( nr >1 )
      {
            nr = nr/2;
            i=i+1;
      }

      return i;
}
```

```
/* FILE C:\SP\QR.C*/
/*
This program is used for P,Q,R,S and T wave peaks determiation in ECG Signlas.
The Technique involves :
1. mean subtraction
2. 3-sample area estimation
3. step averaging
4. thresholding and peak detection
*/
#include <stdio.h>
#include <math.h>
#include <conio.h>
#include <alloc.h>

#define ARRAY1_SIZE 2500
#define ARRAY2_SIZE 300
FILE *fp1,*fp2,*fp3,*fp4,*fp5,*fp6,*fp7,*fp10,*fp20;
float max,min,dmax,davr,thrs_fctr,amp;
int ech_arr[100];
int av_range,fit,l=10,maxval,ps=15;
int maxnum();
main()
{
      void amax(float *ar,int nr);
      void maxavr(float *ar,int nr);
      void av_val(float *yr,float *xr,int nr);
      int eng_set(float *ar, float *yr, int nr);
        int min1(float *xa,int q);
      int
prev_cnt=0,ech_cnt,s,q,cnt,k,p=0,p1=0,p2=0,t=0,t1=0,t2=0,q1=0,s1=0,cmp=0;
      int nr,i,j,mr,ntype,imid,range_start,range_end,fil_range,q2=0,o,m=0;
      float *xr,*yr,*ar,*xi,*hm,*bm,*xa;
      float amp_level,av_amp=0.0,thrs_step;
      float ts,tr,diff,maxdmin,insnr,outsnr,snrimp,scl_mul;
      float xr_mean,sum,mod_amp,mod_fctr,snr,av_sum;
      char infile[20],outfile1[20],outfile2[20],outfile3[20],outfile4[20],
           outfile5[20],*temp1,*temp2,outfile[20];
      clrscr();

      xr = (float *) malloc(ARRAY1_SIZE * sizeof(float));
      if(xr == NULL )
      {
      printf("Insufficient memory for XR\n");
      exit(1);
      }

      hm = (float *) malloc(ARRAY1_SIZE * sizeof(float));
      if(hm == NULL )
      {
      printf("Insufficient memory for HM\n");
      exit(1);
      }

      bm = (float *) malloc(ARRAY1_SIZE * sizeof(float));
      if(bm == NULL )
      {
```

```
      printf("Insufficient memory for BM\n");
      exit(1);
      }

      yr = (float *) malloc(ARRAY1_SIZE * sizeof(float));
      if(yr == NULL )
      {
      printf("Insufficient memory for YR\n");
      exit(1);
      }
      ar = (float *) malloc(ARRAY1_SIZE * sizeof(float));
      if(ar == NULL )
      {
      printf("Insufficient memory for AR\n");
      exit(1);
      }
      xa = (float *) malloc(ARRAY2_SIZE * sizeof(float));
      if(xa == NULL )
      {
      printf("Insufficient memory for XA\n");
      exit(1);
      }
      fp10  = fopen("in.dat","r");
      fp20  = fopen("out.dat","r");
      temp2 = (char*)malloc(20 * sizeof(char));
      fscanf(fp20,"%s",temp2);
      cmp = strcmp(temp2,"END");
      if (cmp == 0)
      goto end1;
      if (temp2 == "null")
      goto end1;
      strcpy(outfile,temp2);
        fp7 = fopen(outfile,"w");
      printf("\nInput File Name       peak no.\t\tP\tQ\tR\tS\tT\n\n\n");
      fprintf(fp7,"\nInput File Name       peak no.\t\tP\tQ\tR\tS\tT\n\n\n");
/*    printf(" outnput file name to p,q,r,s,t :  %s \n",outfile);
*/
      out2 :
      o=0;
      again :
      temp1 = (char*)malloc(20 * sizeof(char));
      fscanf(fp10,"%s",temp1);
      cmp = strcmp(temp1,"END");
      if (cmp == 0)
      goto end1;
      strcpy(infile,temp1);
/*    printf(" Input the file name to process :    %s \n",infile);
      printf(" outnput file name to p,q,r,s,t :  %s \n",outfile);
*/    fp1 = fopen(infile,"r");
      fp5 = fopen("75.dat","w");
      fp2 = fopen("72.dat","w");
      fp3 = fopen("73.dat","w");
      fp4 = fopen("74.dat","w");
      fp6 = fopen("76.dat","w");
      fscanf(fp1,"%d\n",&mr);
/*    printf("Total number of points are %d \n",mr);
*/    nr =1024;
```

```
      amp_level = 5;
      av_range =3;
/* Read data from specified file */

          for(i=0;i<=nr-1;++i)
            {
            fscanf(fp1,"%f %f\n",&ts,&tr);
            xr[i] = tr;
            av_amp += xr[i]/(float)nr;
            }
            fclose(fp1);
          for(i=0;i<=nr-1;++i)
            xr[i] = xr[i]-av_amp;
          amax(xr,nr);
          maxdmin = max - min;
          if(max < fabs(min) )
                  max = fabs(min);
          for(i=0;i<=nr-1;++i)
            xr[i] = amp_level*(xr[i]/max);
          fprintf(fp5,"%d\n",nr);
          for(i=0; i<nr; i++)
            fprintf(fp5,"%d %f\n",i,xr[i]);
          for(i=0;i<=nr-1;++i)
            ar[i] = xr[nr-1-i];
          fclose(fp5);
/* Step 1 forward estimation */
      amax(ar,nr);
      maxdmin = max - min;
      scl_mul = maxdmin * (float)nr;

/* program for filtering */

       fil_range = av_range;
       av_range = 1;

/* step 1 forward step averaging */
       av_sum =
       0.0;
      for(i=0;i<=nr-1;++i)
      {
      range_start = i - av_range;
      range_end = i + av_range;

      if ( range_start < 0)
      {
      av_sum = av_sum + xr[i+av_range];
      range_start = 0;
      }

      else if( range_end > nr-1 )
      {
      av_sum = av_sum - xr[i-av_range];
      range_end = nr-1;
      }

      else
            av_sum = av_sum + xr[i+av_range] - xr[i-av_range];
```

```
      hm[i] = av_sum/(float)(range_end - range_start + 1);
      hm[i] = hm[i]/scl_mul;
      }
/* step 1 ended */

/* step 2 backward step averaging */
      av_sum = 0.0;
      for(i=0;i<=nr-1;++i)
      {
      range_start = i - av_range;
      range_end = i + av_range;

      if ( range_start < 0)
      {
      av_sum = av_sum + ar[i+av_range];
      range_start = 0;
      }

      else if( range_end > nr-1 )
      {
      av_sum = av_sum - ar[i-av_range];
      range_end = nr-1;
      }

      else
            av_sum = av_sum + ar[i+av_range] - ar[i-av_range];

      bm[i] = av_sum/(float)(range_end - range_start + 1);
      bm[i] = bm[i]/scl_mul;
      }
      for(i=0; i<nr; i++)
            ar[i] = bm[nr-i-1];
/* step 2 ended */

      for(i=0; i<= nr-1; i++)
            if( fabs(hm[i]) > fabs(ar[i]) ) hm[i] = ar[i];

      for(j=0; j<=nr-1; ++j)
      yr[j] = hm[j]*xr[j];

      fprintf(fp2,"%d\n",nr);
      for(i=0; i<=nr-1;++i)
      fprintf(fp2,"%f %f\n",(float)i,yr[i]);
      fclose(fp2);

      for(i=0; i<=nr-1; ++i)
      ar[i] = xr[i];
      maxavr(ar,nr);
      snr = dmax/davr;
      insnr = snr;

/*    printf("The dmax = %f davr = %f and snr = %f\n",dmax,davr,snr);
*/    for(i=0; i<=nr-1; ++i)
      ar[i] = yr[i];
      maxavr(ar,nr);
      snr = dmax/davr;
```

```
        outsnr = snr;
        snrimp = 20.0*log10(outsnr/insnr);
        for(i=0; i<=nr-1; ++i)
        yr[i] = fabs(yr[i]);
        av_range = fil_range;
        av_val(yr,xr,nr);
        amax(xr,nr);
        for(i=0; i<=nr-1; ++i)
        xr[i] = xr[i]  - min;
        fprintf(fp3,"%d\n",nr);
        for(i=0;i<=nr-1;++i)
        fprintf(fp3,"%f %f\n",(float)i,xr[i]);
        fclose(fp3);
        thrs_fctr = 1.10;
        /*printf("\nfit  =   %d\n", fit);*/
        thrs_step = 0.5;
        do {
            for(i=0;i<=nr-1; ++i)
                    ar[i] = xr[i];
         ech_cnt  = eng_set(ar,yr,nr);
if(thrs_fctr>=15.0)
        {
          printf ("\t %s\t   programe fail\n ",infile);
          fprintf(fp7,"%s \t\tPROGRAME FAIL\n",infile);
          goto again;
        }
        if(prev_cnt == 0.0)
             prev_cnt =  ech_cnt;
        if(prev_cnt < ech_cnt)
            {
             thrs_fctr -= thrs_step;
             thrs_step = thrs_step/2.0;
             prev_cnt = ech_cnt;
            }
        thrs_fctr += thrs_step;
        } while( fit == 0);

        fprintf(fp4,"%d\n",nr);
        for(i=0; i<=nr-1;++i)
        fprintf(fp4,"%f %f\n",(float)i,yr[i]);
        fclose(fp4);
        maxval =maxval/2;
        for(k=0; k<=ech_cnt-2; ++k)
            {  cnt=ech_arr[k];
               fp5 = fopen("75.dat","r");
               fscanf(fp5,"%d",nr);
               for(i=0;i<=nr-1;++i)
               { fscanf(fp5,"%f %f\n",&ts,&tr);
                 xr[i] = tr;
                 av_amp += xr[i]/(float)nr;
                }
               fclose(fp5);
/*             printf ("%d",maxval);
*/             for(i=cnt;i<=cnt+maxval/4;++i)
                {
                  if (xr[i]<xr[i+1])
                     xr[i+1]=xr[i];
```

```
    if (xr[i]<=xr[i+1])
       xr[i+1]=xr[i];
      amp=xr[i];
  }

        for(i=cnt;i<=cnt+maxval/4;++i)
     { if (amp==xr[i])
        { s=i;
          break;
        }
     }
  for(i=cnt+maxval/4;i<=cnt+maxval;++i)
     {
       if  (xr[i]>xr[i+1])
      xr[i+1]=xr[i];
       if (xr[i]>=xr[i+1])
      xr[i+1]=xr[i];
      amp=xr[i];
    }

         for(i=cnt+maxval/4;i<=cnt+maxval;++i)
     {if (amp==xr[i])
        { t=i;
          break;
          }
     }
for(i=cnt;i>=cnt-maxval/4;i=i-1)
      {
 if (xr[i]>xr[i+1])
       xr[i]=xr[i+1];
  if (xr[i]>=xr[i+1])
     xr[i]=xr[i+1];
     amp=xr[i];
}
      for(i=cnt;i>=cnt-maxval/4;i=i-1)
        {if (amp==xr[i])
       {q=i;
         break;
         }
      }
    for(i=cnt-maxval/4;i>=5;i=i-1)
{
     if (xr[i]<xr[i+1])
      xr[i]=xr[i+1];
    if (xr[i]<=xr[i+1])
      xr[i]=xr[i+1];
     amp=xr[i];
}
     for(i=cnt-maxval/4;i>=5;i=i-1)
     {  if (amp==xr[i])
          { p=i;
            break;
          }
     }
    printf("%s\t%d\t\t%d\t%d\t%d\t%d\t%d\n",infile,k+1,p,q,
    ech_arr[k]+1,s,t);
    fprintf(fp7,"%s\t%d\t\t%d\t%d\t%d\t%d\t%d\n",infile,k+1,p,q,
```

```
                ech_arr[k]+1,s,t);
        }
        goto again;
        end1:
        fclose(fp10);
        printf("\n\nThe P,Q,R,S and T wave peaks for CSE DATA\n ");
        fprintf(fp7,"\n\nThe P,Q,R,S and T wave peaks for CSE DATA\n ");
       fclose(fp7);

}
/*function  amax() */
void amax(float *ar,int nr)
{
      int i,j;
      max = ar[0];
      min = ar[0];
      for(i=1;i<=nr-1;++i)
      {
      if(max < ar[i] ) max = ar[i];
      if(min > ar[i] ) min = ar[i];
      }
}
/* function to find maximum and mean average */
void maxavr(float *ar,int nr)
{

      int i;

      dmax = fabs(ar[0]);
      for(i=1; i<=nr-1; ++i)
            if(dmax < fabs(ar[i]) ) dmax = fabs(ar[i]);
      davr = 0.0;
      for(i=0; i<=nr-1; ++i)
      davr = davr + fabs(ar[i])/(float)nr;
}

void av_val(float *yr,float *xr,int nr)
{
      int i,j,range_start,range_end;
      float av_sum;

      av_sum = 0.0;
      for(j=0; j<=av_range-1; ++j)
      av_sum = av_sum + yr[j];

      for(i=0;i<=nr-1;++i)
      {
      range_start = i - av_range;
      range_end = i + av_range;

      if ( range_start < 0)
      {
      av_sum = av_sum + yr[i+av_range];
      range_start = 0;
      }

      else if( range_end > nr-1 )
```

```
        {
        av_sum = av_sum - yr[i-av_range];
        range_end = nr-1;
        }

        else
              av_sum = av_sum + yr[i+av_range] - yr[i-av_range];

        xr[i] = av_sum/(float)(range_end - range_start + 1);
        }

}

int eng_set(float *ar, float *yr, int nr)
{
        int i,j,imid,intrvl[100],ech_count;
        float av_sum,half_sum,thrs_value;
        int start_point,end_point;

        amax(ar,nr);
         thrs_value = max/thrs_fctr;

        for(i=0;i<=nr-1;++i)
        if(ar[i] < thrs_value ) ar[i] = 0.0;

        for(i=0;i<=nr-1;++i)
              yr[i] = 0.0;
         i = 0;
         ech_count = 0;

       do {
/*loop A*/ if(ar[i] > 0.0)
           {
           start_point = i;
           end_point = start_point;
              av_sum = 0.0;
              while(ar[i] > 0)
                 {
                 av_sum = av_sum + ar[i]*ar[i];
                 ++i;
                 }
                 end_point = i-1;

                       half_sum = av_sum/2.0;
                       av_sum = 0.0;
                          for(j=start_point; j<=end_point; ++j)
                          {
                          av_sum = av_sum + ar[j] * ar[j];
                          imid = j;
                          if( av_sum >= half_sum ) break;
                          }
                          yr[imid] = ar[imid];
                          ech_arr[ech_count] = imid;
                          ++ech_count;

            } /*loop A */
            else
```

```
            ++i;
/* end Loop A */


      } while( i<= nr-1);
/*    printf("Number of echoes detected are = %d\n",ech_count);
*/    for(i=0;i<ech_count; ++i)
/*    printf("The location of echo %d is %d\n",i+1,ech_arr[i]);
*/
      for(i=0; i<ech_count; i++)
           if(i<ech_count-1)
                intrvl[i] = ech_arr[i+1]-ech_arr[i];
      maxval = maxnum(intrvl,ech_count-1);
      for(i=0; i<ech_count-1; i++)
/*         printf("Interval for peak %d and %d is %d\n",i+1,i+2,intrvl[i]);
*/    if((ech_arr[0] > maxval) || ( nr-ech_arr[ech_count-1] > maxval))
             {
               fit = 0;
               return ech_count;
             }
           else
             {
              start_point = maxval - maxval/5;
              end_point = maxval + maxval/5;
              for(i=0; i<ech_count-1; i++)
              if((start_point <= intrvl[i]) && (end_point >= intrvl[i]))
                 {
                 fit = 1;
/*               printf("interval %d is ok %d\n",i+1,intrvl[i]);
*/               }
              else
                 {
/*               printf("interval %d is not ok %d\n",i+1,intrvl[i]);
*/               fit = 0;
                 return ech_count;
                  }
             }
           return ech_count;
}
int maxnum(intrvl,nr)
int intrvl[100],nr;
{
     int i,max;
     float avrval=0.0;
/*   max = intrvl[0];
     for(i=1; i<nr; i++)
          if(max < intrvl[i]) max = intrvl[i];
*/
     for(i=0; i<nr; i++)
          avrval += (float)intrvl[i]/(float)nr;
     max = (int)avrval;
     return max;
}
```