NASA-CR-192835

Determination of Design and Operation Parameters for Upper Atmospheric Research Instrumentation to Yield Optimum Resolution with Deconvolution

P. 79

NASA Grant NAG 1-804

FINAL REPORT

APPENDIX 3

(NASA-CR-192835) DETERMINATION OF DESIGN AND OPERATION PARAMETERS FOR UPPER ATMOSPHERIC RESEARCH INSTRUMENTATION TO YIELD OPTIMUM RESOLUTION WITH DECONVOLUTION, APPENDIX 3 (New Orleans Univ.) 79 p

N93-32231

Unclas

G3/46 0176608

Dr. George E. Ioup, Principal Investigator
Dr. Juliette W. Ioup, Principal Investigator
Department of Physics
University of New Orleans
New Orleans, LA 70148

Determination of Design and Operation Parameters for Upper Atmospheric Research Instrumentation to Yield Optimum Resolution with Deconvolution

NASA Grant NAG 1-804

FINAL REPORT

APPENDIX 3

Dr. George E. Ioup, Principal Investigator
Dr. Juliette W. Ioup, Principal Investigator
Department of Physics
University of New Orleans
New Orleans, LA 70148

OPTIMIZATION OF SINGLE FILTER APPLICATION OF ALWAYS-CONVERGENT ITERATIVE DECONVOLUTION

A Thesis

Presented to

the Faculty of the Graduate School

University of New Orleans

In Partial Fulfillment

of the Requirements for the Degree of

Master of Science in Physics

by

Haihong Ni

July 1989

ACKNOWLEDGEMENTS

I am greatly indebted to Dr. George Ioup and Dr. Juliette Ioup for their guidance, availability and assistance in completion of this work. I would also like to thank Mr. Abolfazl Amini for his helpful suggestions and assistance.

This thesis is dedicated to my parents Yongfu and Zhiqing for their support and help.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

In this thesis, the Always-Convergent Iterative Noise Removal and Deconvolution Method of Ioup is applied as a single-filter in the transform domain to deconvolution with both narrow and wide Gaussian impulse reponse functions. The wraparound error for both cases is also studied. A method is developed by which one can find the optimum iteration number for single-filter iterative deconvolution of sampled data. The method employs the mean square error (MSE), the square of the difference between the deconvolved result and the input, for optimization. The MSE decreases as the deconvolution iterations proceed, but at the optimum iteration number, the MSE starts to increase. This procedure is repeated for signal-to-noise ratio of 10 to 150. The optimum iteration number and the MSE are plotted vs SNR. By knowing the SNR for a particular experiment, one can find the optimum iteration number and MSE.

The result shows that the narrow impulse response has smaller optimum iteration numbers and MSE than the wide Gaussian, which means the deconvolution of the narrow Gaussian has better results than the wide Gaussian. The optimum iteration numbers for the narrow Gaussian range from 2.8 to 19.2 and for the wide Gaussian from 3.0 to 53.0.

This thesis also shows that even when noise is present, zeros can be added to the original function to reduce the wraparound error. An optimum number of zeros has to be chosen to save computer time.

CHAPTER I

THEORETICAL BACKGROUND

1.1 INTRODUCTION

For much signal processing work, the main task is to recover the input signal from the output signal and the impulse response function. For many linear systems, the relation between the output and input may be given by a convolution operation. In other words, the convolution of the input with the impulse response function of the system gives the output. Thus, in order to recover the input signal, one has to deconvolve the output signal with the impulse response function.

There are many deconvolution methods. The iterative deconvolution methods are among those that are widely used.

In this thesis, an input signal with three peaks is chosen to be convolved with a narrow and a wide Gaussian impulse response function. Gaussian type noise is added to the corresponding output. The Always-Convergent Iterative Deconvolution Method of Ioup (Ioup, 1981) is used to deconvolve the output. The mean square error (MSE) between the deconvolved result and the input is calculated for the optimization. The MSE decreases as the deconvolution iterations proceed, but at the optimum iteration number, the MSE starts to increase. The procedure is repeated for signal-to-noise ratios of 10 to 150. By knowing the optimum iteration numbers and mean square errors for the narrow and wide Gaussian cases, one is able to evaluate the deconvolution for these two cases.

The deconvolutions are also performed at different data lengths for the same signal-to-noise ratios to check the wraparound error. Conclusions about whether one need to add zeros in the time domain (extending the period) to reduce the wraparound error are made after the deconvolutions of different data lengths are done.

This thesis contains four chapters. Chapter One gives a brief mathematical background, including Fourier Transforms, convolution and deconvolution, and especially iterative deconvolution methods; Chapter Two introduces the Always-Convergent Deconvolution Method of Ioup, with discussion and conditions for using the method; Chapter Three describes Gaussian type noise and shows how to generate the noise; In Chapter Four, the Always-Convergent Method is applied to deconvolve synthetic data, and conclusions about the results and the conditions for the deconvolution are made.

1.2 THE FOURIER TRANSFORM

The mathematical expressions of the Fourier Transform and its inverse are:

$$F(s) = \int_{-\infty}^{+\infty} f(x) \exp(-i2\Pi xs)\, dx \qquad (1.1)$$

and

$$f(x) = \int_{-\infty}^{\infty} F(s) \exp(+i2\Pi xs)\, ds, \qquad (1.2)$$

where F(s) is the forward Fourier Transform of f(x) and f(x) is the inverse Fourier Transform of F(s) (Bracewell, 1984). This process is possible if f(x) satisfies the following conditions:

1) The integral of from minus infinity to plus infinity of |f(x)| exists.
2) Any discontinuties in f(x) are finite.

For discrete data, the Fourier Transforms become the following summations:

$$F(n) = \sum_{m=-\infty}^{\infty} f(m) \exp(-i2\Pi nm) \qquad (1.3)$$

and

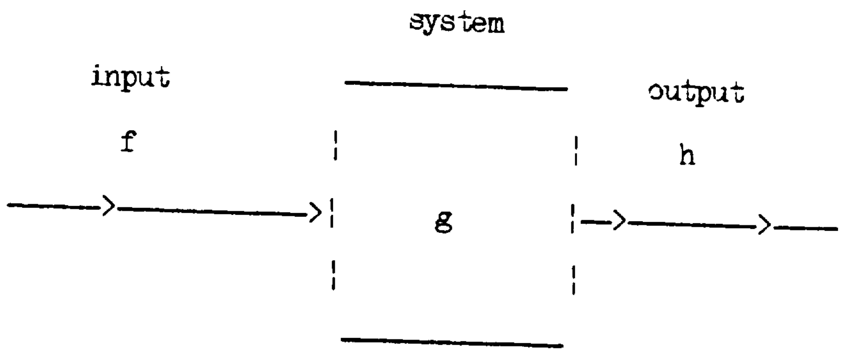
$$f(m) = \sum_{n=-\infty}^{\infty} F(n) \exp(+i2\Pi nm), \qquad (1.4)$$

where m and n are discrete variables.

1.3 CONVOLUTION AND THE CONVOLUTION THEOREM

1.3.1 CONVOLUTION

For a linear system the input convolved with the impulse response function of the system gives the output (Robinson, 1980). The following diagram shows a convolution model of a linear shift-invariant system:



The mathematical expression for the convolution is:

$$h(x) = \int_{-\infty}^{\infty} f(u)\ g(x-u)\ du = f(x)*g(x) \qquad (1.5)$$

where $h(x)$ is the output signal, $f(x)$ is the input signal and $g(x)$ is the impulse response function of the instrument.

For the discrete situation, the convolution may be given by the following equation:

$$h_n = \sum_{m=-\infty}^{\infty} f_m\, g_{n-m}, \qquad (1.6)$$

where h_n, f_m and g_{n-m} are discrete functions and n and m are discrete variables.

1.3.2 CONVOLUTION THEOREM

Convolution is a comparatively complicated operation because integration is involved. By using the Convolution Theorem we are able to simplify the calculation of convolution greatly.

The Convolution Theorem states that the convolution operation in one domain corresponds to a complex multiplication operation in the other domain (Bracewell, 1984).

If there is a convolution operation in the function domain as given in equation (1.5) above, then the Fourier Transforms F(s), G(s) and H(s) have the following multiplicative relation:

$$F(s)G(s) = H(s). \tag{1.7}$$

To use the Convolution Theorem to calculate the convolution between f(x) and g(x):

$$h(x) = f(x)*g(x), \tag{1.8}$$

first we take the transforms of f(x) and g(x), which are F(s) and H(s), then mutiply the transforms F(s) and G(s):

$$H(s) = F(s)G(s). \tag{1.9}$$

The convolution result is obtained by simply taking the inverse transform of the product of F(s) and G(s). Thus the operation in the transform domain can be faster to obtain on the computer.

1.3.3 DECONVOLUTION

The relationship between the output signal of a linear shift-invariant instrument and its input signal has been established.

The input convolved with the impulse response function of the instrument gives the output signal. But it is often the situation that we know the output signal and the impulse response function of the instrument. What we need to find is the input signal. In other words we need to recover the actual input signal f in terms of the response function g and the output signal h. This process is called deconvolution.

There are many deconvolution methods. One of the most popular methods is inverse filtering. The following diagram shows inverse filtering or deconvolution in the function domain.

$$\begin{tikzcd}
{} \arrow[r, "h"] & \boxed{g^{-1}} \arrow[r, "f"] & {}
\end{tikzcd}$$

$$h = f*g, \tag{1.10}$$

$$f = h*g^{-1} = f*g*g^{-1}, \tag{1.11}$$

By the Convolution Theorem the above operations in the transfom domain become multiplication and division:

$$F(s)G(s) = H(s) \tag{1.12}$$

and

$$F(s) = H(s)/G(s). \tag{1.13}$$

This result is undefined where $G(s) = 0$. The principal solution suggested by Bracewell and Roberts (1954) is:

$$F_p(s) = \begin{cases} H(s)/G(s) & [\ s: G(s)\neq 0\ ] \\ 0 & [\ s: G(s)=0\ ]. \end{cases} \tag{1.14}$$

Figure 1.1 and 1.2 show the magnitudes of the transforms. Because of symmetry, only values at positive frequencies are presented. Figure 1.1 shows $|F|$, $|G|$, and $|H|$, while Figure 1.2 shows $|F_p|$, $|G|$ and $|H|$.

In the transform domain, at regions where $G(s)$ is zero, $F_p(s)$ is set to zero. This is sometimes equivalent to mutiplying $F_p(s)$ by a truncating function (rectangular window):

Figure 1.1

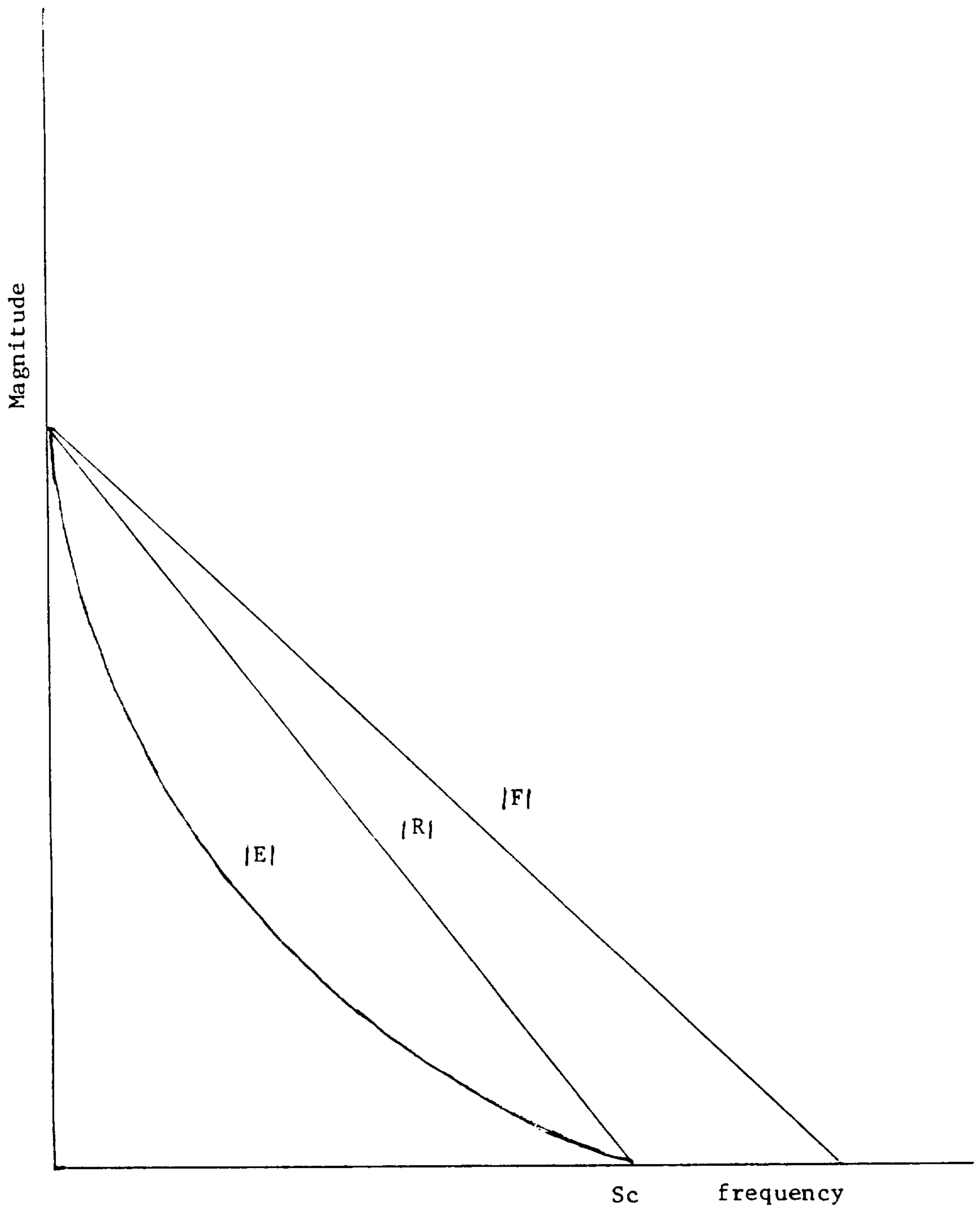
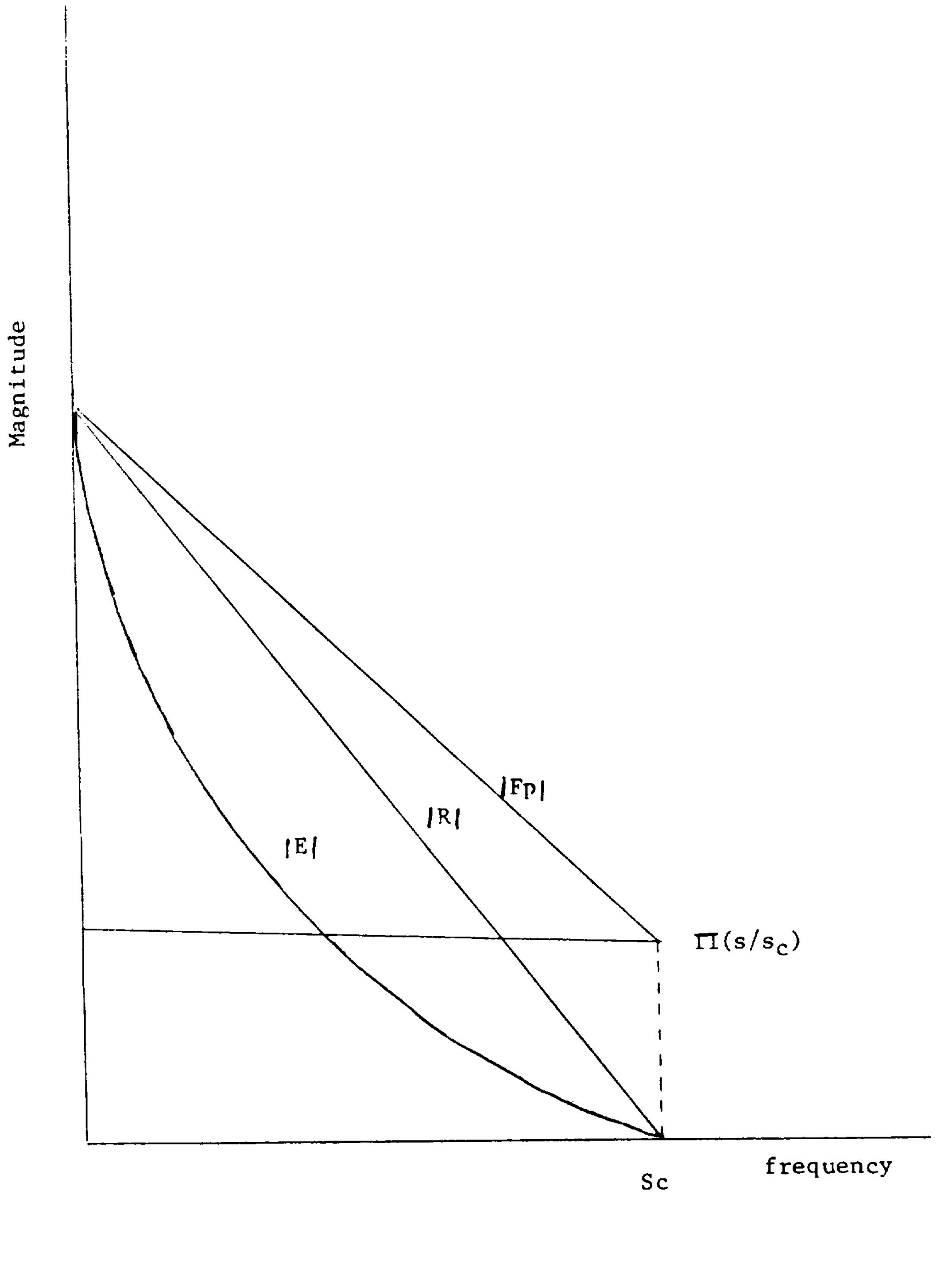


Figure 1.2



$$F_p(s) = F(s)\Pi(s/2s_c), \qquad (1.15)$$

where $\Pi(s/s_c)$ is the rectangular window with the cutoff frequency s_c.

Since $\mathrm{sinc}(2s_c x)$ is the inverse Fourier Transform of $\Pi(s/2s_c)$, by the Convolution Theorem the principal solution for this case is:

$$f_p(x) = f(x) * s_c \mathrm{sinc}\, 2s_c x, \qquad (1.16)$$

in the function domain. The superposition of these effects (convolution with the sinc inverse transform) is the approximate solution $f_p(x)$. Proper tapering of the rectangular window reduces the oscillations introduced by the sinc.

Notice that $f_p(x)$ is not a unique solution to the deconvolution problem. Any data having a transform which is zero at regions where $G(s)$ is non-zero and is non-zero at regions where $G(s)$ equals zero can be added to $f(x)$ to give a solution to the same deconvolution problem (Bracewell and Roberts, 1954; Ioup and Ioup, 1983).

If additive noise $n(x)$ is present in the data, the output becomes:

$$d(x) = h(x) + n(x). \qquad (1.17)$$

The resulting experimental data d(x) are:

$$d(x) = f(x)*g(x) + n(x). \qquad (1.18)$$

If no noise removal is applied, the principal solution becomes:

$$F_p(s) = H(s)/G(s) + N(s)/G(s) \qquad (1.19)$$

Beyond s_c, the cutoff frequency of G(s), H(s) is zero. However, the noise N(s) is not necessarily zero in this region. For this region, we can simply remove the noise entirely first because H(s) is expected to be zero in this region.

1.4 ITERATIVE DECONVOLUTION METHODS

For application to real data, iterative deconvolution methods are widely used. The ones involved in this research are:

1. Van Cittert's Method (1931), and

2. Always-Convergent Deconvolution Method of

Ioup (1981), which will be discussed in the next chapter.

1.4.1 Van Cittert's Deconvolution Method

In 1931 Van Cittert created an iterative deconvolution method. The iterations in the function domain can be described by the following equations:

$$\begin{aligned}
f_0(x) &= h(x) \\
f_1(x) &= f_0(x) + [\ h(x) - f_0(x)*g(x)\ ] \\
f_2(x) &= f_1(x) + [\ h(x) - f_1(x)*g(x)\ ] \\
\cdot \quad & \quad \cdot \qquad\qquad\qquad \cdot \\
\cdot \quad & \quad \cdot \qquad\qquad\qquad \cdot \\
\cdot \quad & \quad \cdot \qquad\qquad\qquad \cdot \\
f_{n-1}(x) &= f_{n-2}(x) + [h(x) - f_{n-2}(x)*g(x)\ ] \\
f_n(x) &= f_{n-1}(x) + [\ h(x) - f_{n-1}(x)*g(x)\ ] \qquad (1.20)
\end{aligned}$$

Van Cittert recognized that the image data h(x) could be considered as a first approximation, $f_0(x)$, to the desired f(x). Then he took the difference $h(x) - f_0(x)*g(x)$ to be the correction for the second iteration. A similar correction is obtained for each iteration. In other words, for each iteration, the approximation result $f_n(x)$ is the sum of the previous approximation $f_{n-1}(x)$ and the correction factor which is the difference between the image data $h(x)$ and the convolution of the previous approximation $f_{n-1}(x)$ with the response function $g(x)$.

In the transform domain the iterations become:

$$F_0(s) = H(s)$$

$$F_1(s) = F_0(s) + [\ H(s) - F_0(s)G(s)\ ]$$

$$F_2(s) = F_1(s) + [\ H(s) - F_1(s)G(s)\ ]$$

$$\begin{matrix} \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{matrix}$$

$$F_{n-1}(s) = F_{n-2}(s) + [\ H(s) - F_{n-2}(s)G(s)\ ]$$

$$F_n(s) = F_{n-1}(s) + [\ H(s) - F(s)_{n-1}G(s)\ ] \qquad (1.21)$$

The last equation may also be written as:

$$F_n = [\ 1 + (1-G)^1 + (1-G)^2 + \ldots\ldots + (1-G)^n\ ]\ H$$

$$= \{ [1 - (1-G)^{n+1}] / G \} H \qquad (1.22)$$

This result was obtained by successive substitutions. From equation (1.22), we can see that in order for the iterations to be convergent, the following conditions have to be satisfied:

$$| 1-G | < 1 \quad \text{for regions where } G(s) \text{ is non-zero,} \qquad (1.23)$$

$$H(s) = 0 \quad \text{for other regions.} \qquad (1.24)$$

These are the Convergence Criteria (Ioup and Ioup, 1983). In the absence of noise, the convergence depends heavily on the impulse response function g(x). Bracewell and Roberts (1954) gave those conditions originally. Hill (1973) and Hill and Ioup (1976) used these conditions to find necessary restrictions on the shape and location of the impulse response function g(x).

In satisfying condition (1.23), the origin of the impulse response function g(x) is of importance (Hill, 1973; Hill and Ioup, 1976) ! Sometimes we can adjust the origin of the impulse response function g(x) to such a position that G(s) satisfies condition (1.23).

CHAPTER II

ALWAYS-CONVERGENT DECONVOLUTION METHOD OF IOUP

Van Cittert's Iterative Deconvolution Method is a very effective method and has been used since the early thirties. But in order to use this method, the impulse response function has to satisfy Convergence Criteria (1.23 and 1.24), which means that the transform of the impulse response function has to be inside the unit circle centered on 1+i0 in the complex plane. Many impulse response functions do not satisfy Convergence Criteria (1.23 and 1.24), and often there is not too much we could do to make them satisfy (1.23 and 1.24). In 1981, Ioup created a new method to overcome the convergence problem, which is the so-called Always-Convergent Iterative Deconvolution Method (Ioup,1981, Whitehorn 1980, Ioup and Whitehorn, 1981, Amini 1987).

The Always-Convergent Method of Ioup is a modified version of van Cittert's method. Two modifications are made. First, the transform of the impulse response function G(s) is replaced by

$$G_m(s) = |G(s)| \;/\; |G(s)|_{max}, \qquad (2.1)$$

and secondly, in the iteration equations, $H(s)$ is replaced by the product of the principal solution and $G_m(s)$.

From the definition of $G_m(s)$, we can see that $G_m(s)$ is always real and between 0 and 1, which guarantees that $G_m(s)$ satisfies Convergence Criterion (1.23), thus the iterations are always convergent.

Applying the above modifications to the van Cittert iterations in the transform domain, the Always-Convergent Method in the transform domain can be described by the following equations:

$$F_0 = H$$

$$F_1 = F_0 + [\ F_p - F_0\ ]\ G_m$$

$$\begin{array}{ccc} \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{array}$$

$$F_n = F_{n-1} + [\ F_p - F_{n-1}\ ]\ G_m. \qquad (2.2)$$

With successive substitutions, the above equations can be written in the general form:

$$F_n = [\ 1 - (1-G)(1-G_m)^n\ ]\ H/G. \qquad (2.3)$$

For regions where G(s) is zero, then:

$$F_n = 0. \tag{2.4}$$

The Always-Convergent Deconvolution Method can be also used in the time domain. The iterations are given by the following equations:

$$f_0 = h_0$$

$$f_1 = f_0 + [\ f_p - f_0\ ] * g_m$$

$$\begin{matrix} \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{matrix}$$

$$f_n = f_{n-1} + [\ f_p - f_{n-1}\ ] * g_m, \tag{2.5}$$

where g_m and f_p are the inverse Fourier Transforms of G_m and F_p.

With the Always-Convergent Method in the time domain, since the convolution operation is involved, it is harder to do the iterations and takes more computer time, but one usually has more information about the original signal than the transform. Thus during the deconvolution process, one may be able to check the result and make corrections to the result to make the deconvolution more precise. For example, for a positive signal, one knows that the result should be non-negative, thus the non-negative constraint may be applied to the deconvolved signal to improve the deconvolution.

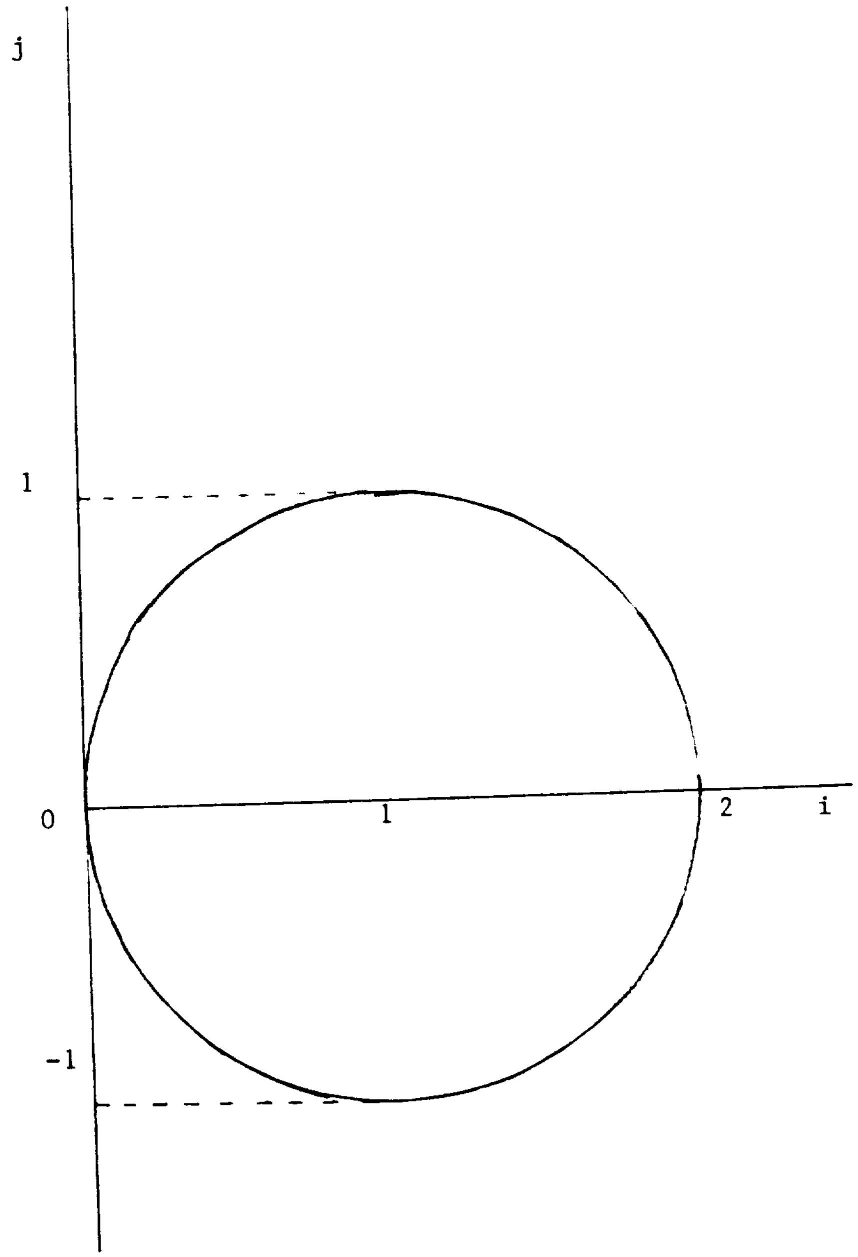
For the Always-Convergent Method in the transform domain, only complex multiplication is involved, so it is easier and much faster to deconvolve on the computer. The disadvantage is that one usually has less information about the transform than about the original signal, thus it is harder to make adjustments during the deconvolution process to get a more accurate deconvolved result.

At regions where G(s) is zero, the deconvolution equation (2.3) becomes:

$$[F_n(s)][0] = H(s). \qquad (2.6)$$

Obviously any arbitrary function will satisfy (2.6), and we are not be able to recover the input. For all of the above deconvolution methods, the solution F(s) is defined to be zero in this region. This does not really solve the problem, as we still have no idea what the real input is. Howerer, the application of function domain constraints can extend the solution into this region !

Figure 2.1



CHAPTER III

NOISE

In the real world there will be always noise present. Data processing systems are usually required to handle a large assortment of signals in the presence of noise. There are many types of noise: noise could be the wrong signal, or the right signal out of place, and what is noise to one observer might be signal to another. In any application, care must be given to the classification of which signals are considered to be useful and which ones are considered undesirable, or noise (Ronbinson, 1980).

The noise studied in this research is Gaussian distributed noise. Gaussian noise has a Gaussian or bell curve distribution about any given data point. If enough noise cases are generated, a plot of occurrence versus the magnitude of that amplitude approaches a Gaussian shape (Fig.3.1).

There are two types of Gaussian distributed noise: constant and ordinate-dependent noise. For constant Gaussian noise, the width of the bell curve is fixed; it is Gaussian noise with the same standard deviation at each point. However, ordinate-dependent Gaussian noise is that for which the width of the bell curve depends on the ordinate value of the data point.

To add Gaussian noise to the data, one can use the well-known result that the sum of comparatively few random numbers from a uniform distribution gives a very good approximation to a normal Gaussian distribution (Hamming, 1962). Utilization of the Central-Limit Theorem leads to the above conclusion (Hamming, 1962). On the computer, in the case of a decimal machine (expressed as a base 10 number) it is customary to use 12 random numbers to get a variance of 1 (Hamming, 1962). Since the mean in this case is not zero (the mean is half of the number of points), a necessary amount (in this case 6) must be subtracted from the sum of the numbers to make the mean zero. For the constant noise case the expression is:

$$h_{nc}(l) = (A_i - 6)\, SF^{1/2} + h(l) \qquad (3.1)$$

where the A_i are the sum of random numbers, SF is the scale factor, and h is the noise free data. With the help of the SF one can generate a noisy data set which has approximately the signal to noise ratio (SNR) of interest (Leclere, 1984).

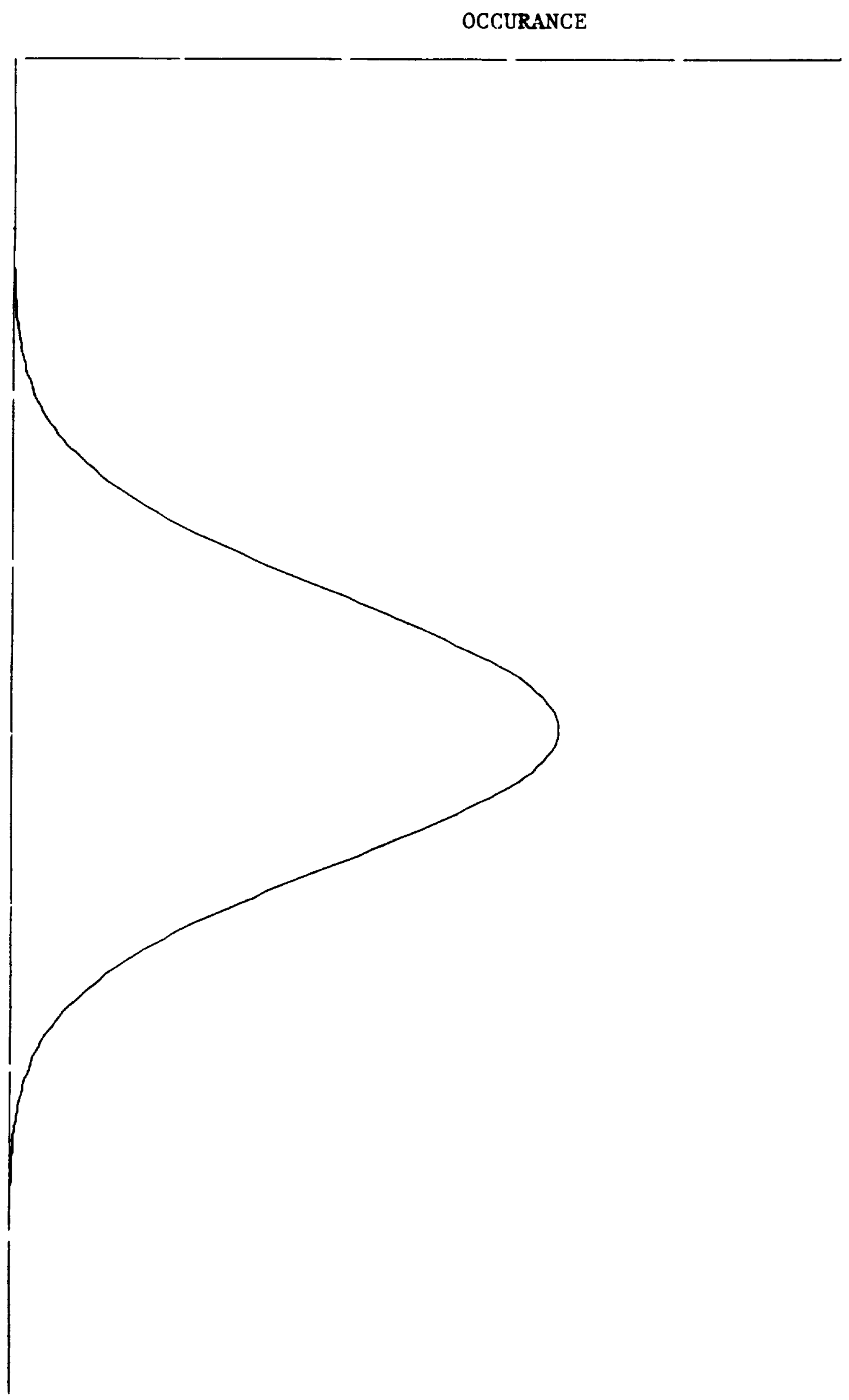


FIGURE 3.1
GAUSSIAN NOISE

CHAPTER IV

SINGLE-FILTER APPLICATION OF THE ALWAYS-CONVERGENT DECONVOLUTION METHOD TO PHYSICAL DATA

In this chapter, the use of the Always-Convergent Deconvolution Method to deconvolve synthetic data is discussed. The input used here consists of three narrow Gaussian peaks which are shown in Fig.4.1. The second and third peaks are located close enough together so that after convolution there is some overlap. To get the output data, the input is convolved with both narrow and wide Gaussian impulse response functions. The narrow Gaussian impulse response function contains 9 points as shown in Fig.4.2, while the wide Gaussian contains 21 points as shown in Fig.4.3. Gaussian-type noise with various signal-to-noise ratios is added to the output data. The deconvolved result is compared to the input, and the mean square error (MSE) between the deconvolved result and the input is calculated as a quality measure. The minimization of the mean square error between the deconvolved result and the known synthetic input is used to determine the optimum iteration number.

The mean square error of the deconvolved result F' and the input F in the transform domain is given by:

$$MSE = \left[\sum_{i=1}^{\infty} (F'_i - F_i)^2 \right]^{1/2}, \qquad (4.1)$$

which is the square root of the sum over all frequencies of the squares of the difference between the transform of the deconvolved result and the transform of the input.

Two Gaussian impulse response functions were used since many instruments have Gaussian impulse response functions. We use both narrow and wide Gaussian impulse response functions, expecting to cover approximately the range of typical instruments.

Discrete Fourier Transforms of the sampled impulse response function g(x) and the output h(x) (with or without noise) are taken for both narrow and wide Gaussian impulse response functions, using an FFT3D subroutine (IMSL). Equation (2.3) is used to perform the deconvolution iterations in the transform domain.

The mean square error between the known synthetic input and the deconvolved result is calculated for each iteration and compared to that of the previous iteration. Previous research results (Amini, 1986) show that the curve of mean square error versus iteration number under certain signal-to-noise ratios has the shape of Fig.4.7. This means that the mean square error decreases as the iteration number increases until the optimum iteration is reached. Beyond the optimum iteration number, the mean square error increases as the iteration number increases. During the process of deconvolution, once the MSE is found to increase compared to the previous iteration, we simply terminate the deconvolution and set that iteration number to be the optimum iteration number.

Table 4.1 shows the optimum iteration number for the narrow Gaussian case at a signal to noise ratio of 10 to 150. The optimum iteration number is 3 at a SNR of 10, and increases to about 19 at a SNR of 150; the optimum iteration number is larger at lower SNR regions and smaller at higher SNR regions.

Table 4.2 shows similar values for the wide Gaussian case. The optimum iteration number increases from 3 at a SNR of 10 to 53 at a SNR of 150. This has the same behavior as that of the narrow Gaussian case, except the magnitudes are larger.

At the regions of low signal to noise ratio, the noise is large, and the large noise will terminate the iterations quickly. That is why the optimum iteration numbers are smaller in this region. As the signal to noise ratio increases, the noise becomes smaller and more signal can be restored. It will take more iterations to reach the optimum level of restoration, thus the optimum iteration number becomes larger in this region.

If we compare the optimum iteration numbers for these two cases, we see that the narrow Gaussian case has a lower value than the wide Gaussian. This is a reasonable result, because the narrow Gaussian impulse response function has a wider transform than that of the wide Gaussian. Thus at most frequencies the G_m of a narrow Gaussian is closer to one (or $|1-G_m|$ is closer to 0) than that for the wide Gaussian. From equation(2.3), the closer $|1-G_m|$ is to zero, the faster the convergence speed is. Thus the narrow Gaussian reaches the optimum iteration faster, or has a smaller optimum iteration number.

Table 4.3 shows the mean square error for the narrow Gaussian case in the frequency domain at a SNR range from 10 to 150. To save computer time, the mean square error is calculated in the transform domain. The mean square error decreases rapidly from 247.7 at a SNR of 10 to 34.9 at a SNR of 150. The data show clearly that the mean square error is very large at regions of low signal to noise ratio and drops rapidly as the signal to noise ratio increases. This is obviously a reasonable result. Under the same input signal, the mean square error depends on two factors: noise and the impulse response function. At regions of low signal to noise ratio, the noise is large, and we simply get a large mean square error due to large noise. At regions of high signal to noise ratio, the noise is small and the impulse response function becomes a more important source of mean square error due to lack of restoration.

The mean square error depends heavily on the shape of the impulse response function. Figure 4.4 shows the transform of a narrow Gaussian impulse response function. We see that the transform has only one almost zero point and increases rapidly to some considerably large amplitudes on both sides. In other words, at almost all frequencies the transform G(s) of the impulse response function has relatively large amplitudes. If G(s) is non zero, we do not have to assume H(s) to be zero. If the amplitude of G(s) is not very small, a little noise added to H(s) will not affect the result of H(s)/G(s) significantly. Thus the mean square error is very small at high signal-to-noise ratios for the narrow Gaussian case.

Table 4.4 presents similar MSE results in the transform domain for the wide Gaussian case. The mean square error decreases slowly from 559.0 at a SNR of 10 to 449.2 at a SNR of 150. The mean square error is very large in the low SNR region (high noise region) and is only slightly smaller in the high SNR region (low noise region). This is obviously a different situation from the narrow Gaussian case. At regions of low signal to noise ratio (high noise region), it is natural that due to the large noise, the mean square error is large. But in the high SNR region (low noise region), the mean square error is still very large, only slightly smaller than that of the high noise region. Consider the impulse response function. Figure 4.5 is the transform of the wide Gaussian impulse response function. In a very wide frequency region, the transform appears to be zero. Actually the transform is not zero but is very small (on the order of 10^{-7} or 10^{-6}). Thus when we do the division H(s)/G(s), a small amount of noise will cause significantly large mean square errors.

From the above results, we see that the narrow Gaussian case has a smaller mean square error and thus a better deconvolution result; the wide Gaussian has a larger error and worse deconvolution results.

Wraparound error can also be a significant error source during the deconvolution process. If the inverse of the impulse response function and the output signal are not narrow enough compared to the whole time interval, errors will arise at the edges due to overlapping effects. This is called the wraparound error. One way to reduce or get rid of the wraparound error is to add more zero points to the original function to increase the length. But the question is: will this affect the deconvolution result ? BenSueid (1987) showed that when no noise is present, adding zeros to the original function will not affect the deconvolution result if the means square error is calculated in the time domain. Later we will show that even when noise is present, we will have the same result in the time domain. This means that in order to reduce the wraparound error one can add as many zeros as he wants to the original function. But if zeros are added, it will take more computer time. So one needs to choose an optimum length (adding the proper number of zeros) which will minimize the wraparound error but do slow down the computer unnecessarily.

Table 4.5 shows the optimum iteration numbers and mean square errors (in the time domain) for different lengths for the narrow Gaussian case at a SNR of 10 to 150. The optimum iteration numbers and mean square errors change significantly when the length increases from 32 points to 64 points; after that, they do not change at all while the length keeps increasing from 64 to 128 and from 128 to 256 and so on. From the fact that the optimum iteration numbers and mean square errors do not change when the length takes a value longer than 64 points, we can conclude that adding zeros beyond 64 points to the original function even when noise is present does not affect the deconvolution result. This conclusion allows one to reduce the wraparound error for the noise situation.

The mean square errors and the optimum iteration numbers change significantly when the length increases from 32 points to 64 points, and remain constant when the length is over 64 points for this case. Therefore, the wraparound error is significant when the length is 32 points and vanishes when the length is over 64 points. Thus we can simply choose 64 to be the optimum length for this case to reduce wraparound error and save computer time.

Table 4.6 shows similar quantities for the wide Gaussian case. The results have the same behavior as the narrow Gaussian case. Thus the same conclusion can be made as for the narrow Gaussian case.

One important thing the above results show is that the narrow Gaussian case has better deconvolution results than the wide Gaussian case. But does a narrow impulse response function always have a better deconvolution result than a wide impulse response ? If so, why ? What can we do to get a better deconvolution result ? Analysis in the transform domain has been made on whether a deconvolution is successful ! How about analysis in the time domain ? Curiosity makes me look for the answers to these questions !

Figure 4.12 illustrates some general impulse response functions (sometimes called windows). Curve g_c is an given impulse response function. As curve g_c becomes wider yet, it takes the shape of curve g_b. As it gets wider, it becomes a flat "curve" (curve g_a), which is a horizontal straight line. On the other hand, if curve g_c gets narrower, it becomes a delta function (curve g_d). Therefore this graph shows windows of a range of widths, from the narrowest delta function (curve g_d) to the widest horizontal line (curve g_a).

Figure 4.13 is an input signal f with two peaks. To see what happens in the time domain during the deconvolution process, this input f is convolved with each of the above windows individually. In figure 4.14, h_a, h_b, h_c and h_d are the convolutions of f with g_a, g_b, g_c and g_d.

Recall that the convolution can be defined by the following:

$$h(x) = \int_{-\infty}^{\infty} f(u)\ g(x-u)\ du \qquad (4.2)$$

To get the convolution value at a certain point x, the window is reversed and centered on x, then multiplied by the input and integrated over the period (to avoid wraparound errors, the period is selected much larger than the width of the windows). This is actually an "average" calculation of the input with a "weight" function $g(x-u)$. If the window $g(x)$ is narrow, then we are "averaging" the input in a narrow region for each convolution value. Thus we will loose less detail about the input during the convolution process and the convolution will have a shape more like the input. The deconvolution will be easier in this situation. On the other hand, if we have a wide window, then we are "averaging" the input over a wide region for each convolution value. More details of the input get lost during convolution and it will be harder to recover the signal. This can be observed in Figure 4.14.

In Figure 4.12, g_d is a delta function or the narrowest window. h_d is the convolution of g_d with the input, which is the same as the input because the delta function is the identity for convolution. When the input is convolved with a delta function, it does not change at all. It is easy to do such a deconvolution when the output is identical to the input.

Now consider the windows g_c and g_b. Window g_c is narrower than window g_b. The outputs h_c and h_b are the convolution of g_c and g_b with the input f. We see that h_c lies between h_d (the same as the input) and h_b, which means h_c has values closer to the input or carries more information about the input than h_b. Thus it is easier to recover the input from h_c (the result of convolution with the narrower window) than from h_b (the result of convolution with the wider window).

If the window keeps getting wider, finally it will become the horizontal line g_a. The output h_a (convolution of g_a with the input f) is a constant; indeed, the convolution of this window with any kind of input will be a constant. This is not a surprising result, because we are actually "averaging" the entire input. Any input convolved with this window becomes a constant output. Obviously the solution to this deconvolution problem can be any function, or in other words, we are not able to get the solution we wanted.

We can also make a similar analysis in the transform domain and reach the same conclusion.

Figure 4.15 illustrates the transforms of the windows shown in Figure 4.12. G_a, G_b, G_c and G_d are the transforms of g_a, g_b, g_c and g_d. Note that the transform of a delta function is a constant and the transform of a constant is a delta function. Also we see that the narrower window h_c has a wider transform than that of the wider window h_b.

For the convolution with a delta function, deconvolving in the transform domain is simply the division H_d/G_d. Since G_d is a constant, this is almost as simple as the parallel deconvolution in the time domain.

Comparing the transforms G_c and G_b, we see that the transform G_c of the narrower window (g_c) is wider than the transform G_b of the wider window (g_b). This means G_b has more zero points or more small magnitude points than G_c. To do the deconvolution, one has to perform the division H/G. If in a certain region G is zero, then the division is undefined. In other words, the information in the input in this region gets lost. If G is not zero but has very small magnitude, then as discussed before, large errors will be produced in this region. Therefore from the transform domain the narrower window will retain more details about the input and produce less error.

The transform of the constant function g_a is G_a, which is a delta function. H_a is required to be zero in the entire transform domain except for one point. In other words, all the information about the input has been lost in the transform domain except the value at the origin. Any input F will give a solution to the deconvolution problem. This the same conclusion as was reached in the time domain.

CONCLUSION

In this thesis, the Always-Convergent Iterative Noise Removal and Deconvolution Method of Ioup (Ioup,1981) is applied as a single filter to deconvolution with both narrow and wide Gaussian impulse response functions. The wraparound error for both cases is also studied. The results show that the narrow Gaussian impulse response converges faster and has smaller mean square error than the wide Gaussian impulse response. In general, deconvolution using a narrow window usually produces better results (converging faster and having smaller mean square error) than a wide window. In our particular case, the deconvolution result of the narrow Gaussian case matches the original input very well, while the result using the wide Gaussian does not match the input so well.

In our case, the narrow Gaussian impulse response has smaller optimum iteration numbers and smaller mean square error than the wide Gaussian, which matches the above conclusion.

This thesis also shows that for a deconvolution problem, even when noise is present one can still add zeros to the original function (increasing the length) to reduce the wraparound error. This result enables one to minimize the wraparound error when noise is present. In order to save computer time, a optimum length should be chosen. To obtain the optimum length, one has to minimize both the wraparound error and the computer time. In the narrow Gaussian case, the optimum iteration numbers and the mean square errors change significantly when the length increases from the original 32 points to 64 points and remain unchanged when the length is over 64 points. This means the wraparound error is only significant at the length of 32 points and vanishes after the length is longer than 64 points. Thus the optimum length is obviously 64 points for this case. In the same way, the optimum length for the wide Gaussian case is chosen to be 88 points.

Fig.4.1

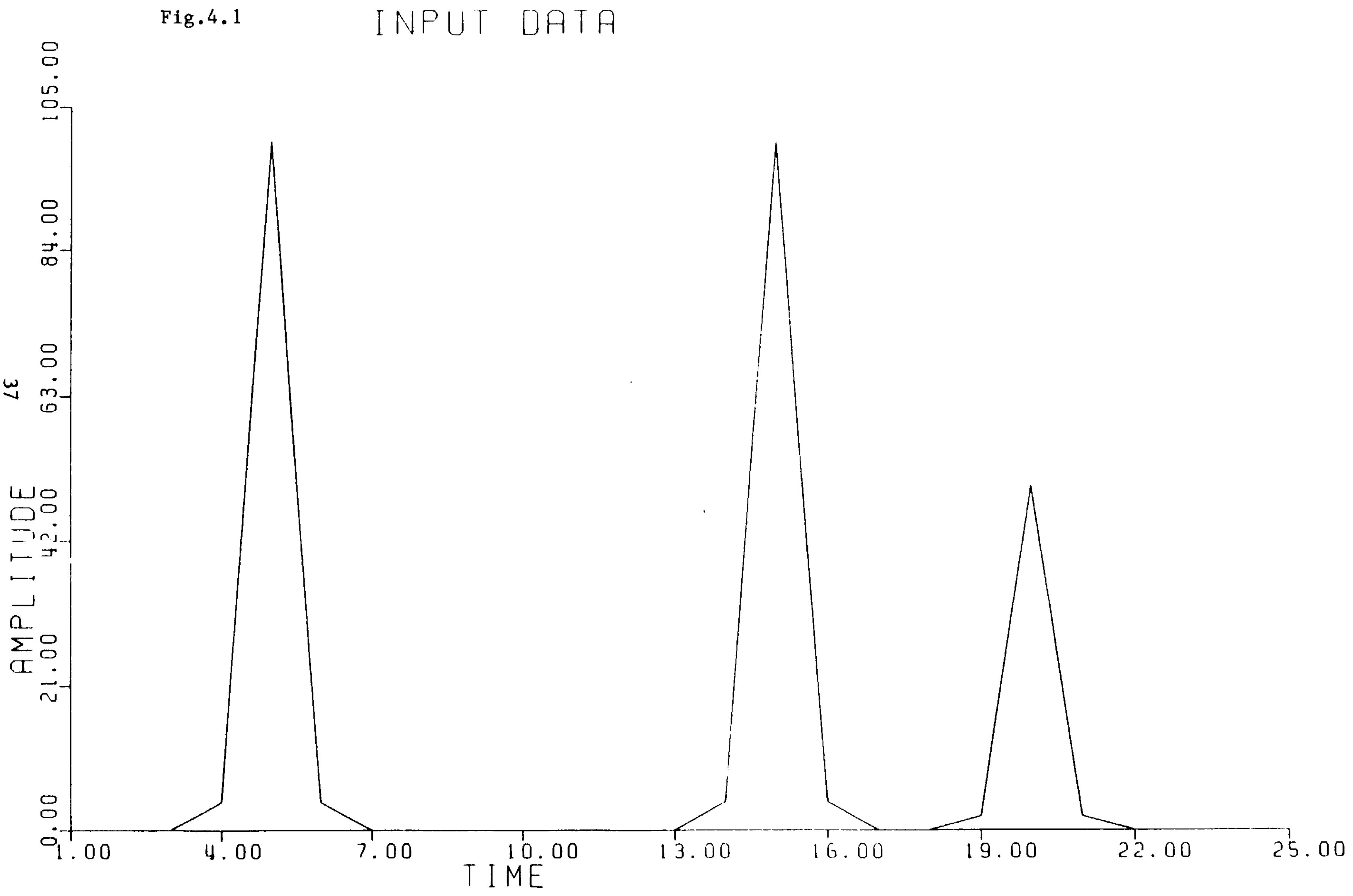

INPUT DATA
AMPLITUDE
105.00
84.00
63.00
42.00
21.00
0.00
TIME
1.00
4.00
7.00
10.00
13.00
16.00
19.00
22.00
25.00

fig.4.2 NARROW GUSSIAN

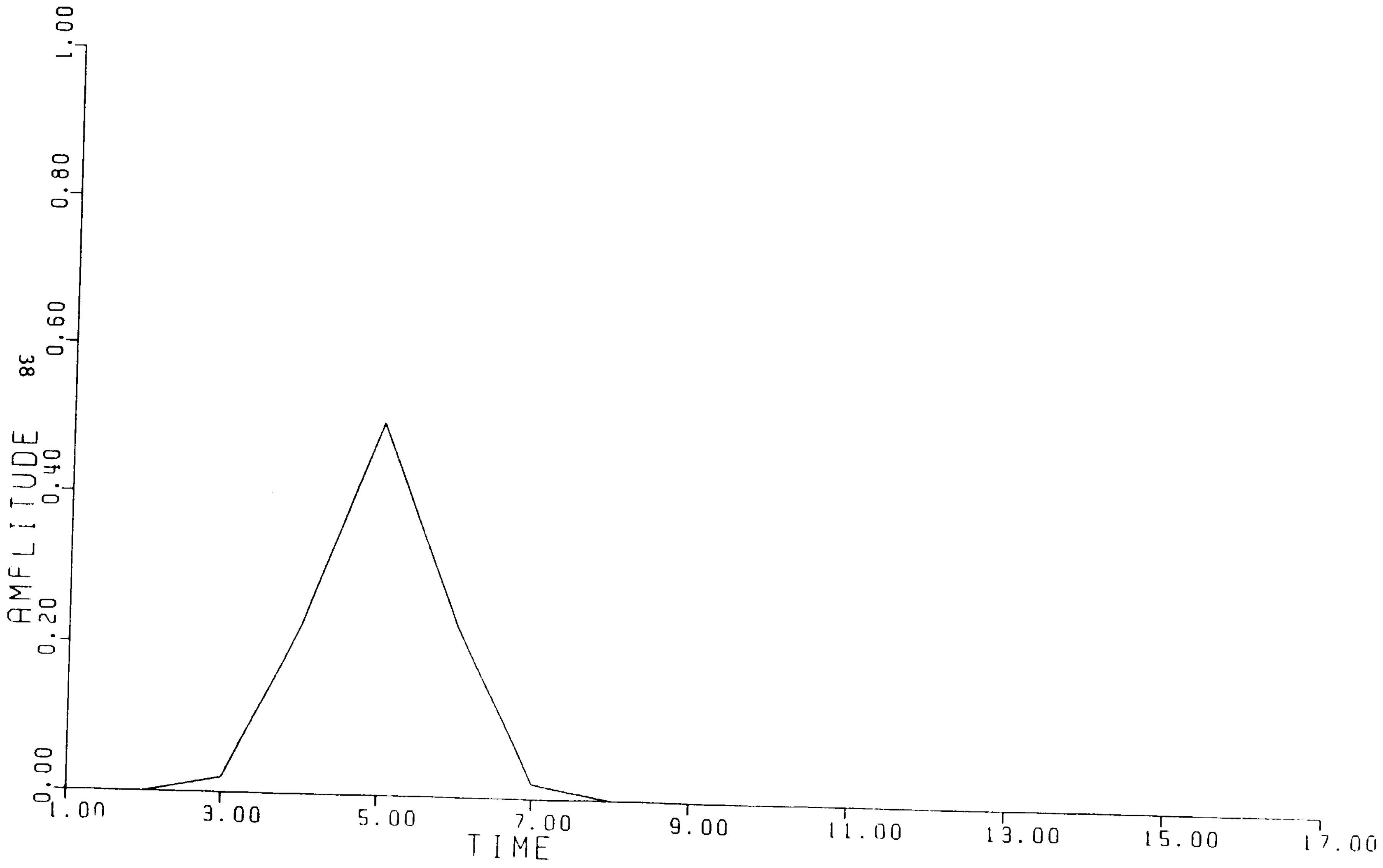


Fig.4.3

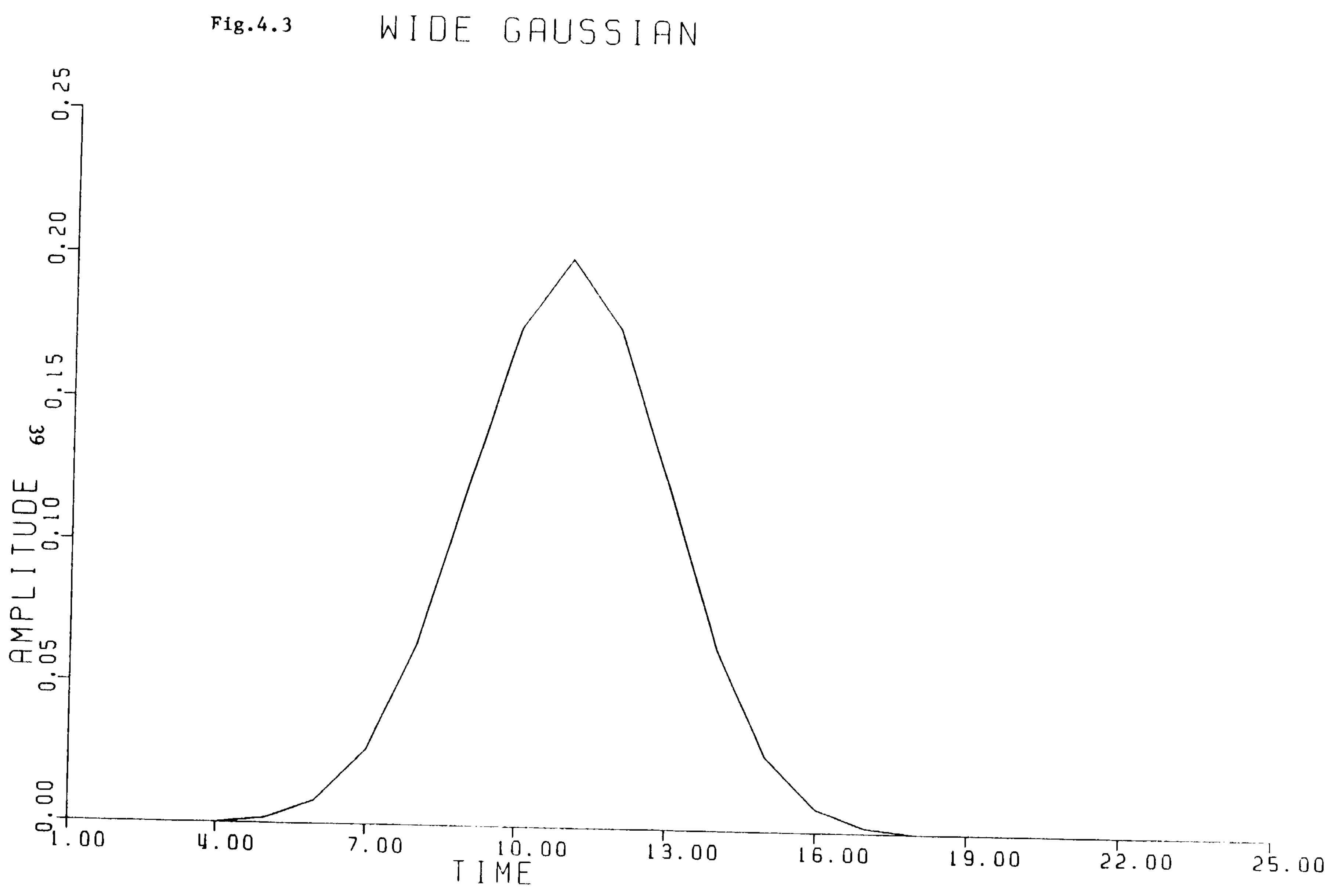

WIDE GAUSSIAN
AMPLITUDE
0.00
0.05
0.10
0.15
0.20
0.25
TIME
1.00
4.00
7.00
10.00
13.00
16.00
19.00
22.00
25.00

Fig.4.4

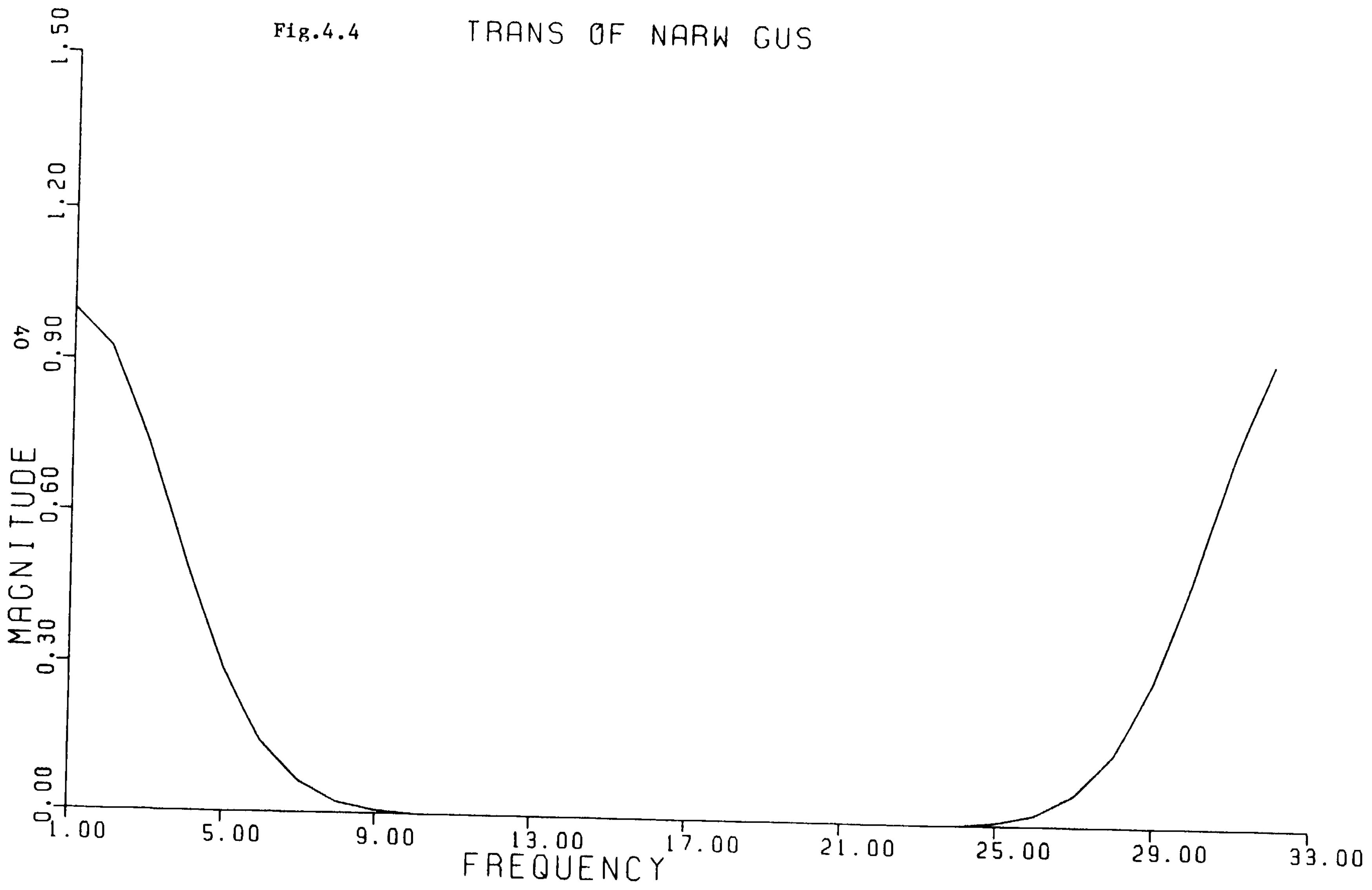

TRANS ØF NARW GUS
MAGNITUDE
0.00
0.30
0.60
0.90
1.20
1.50
FREQUENCY
1.00
5.00
9.00
13.00
17.00
21.00
25.00
29.00
33.00

Fig.4.5 TRANS OF WIDE GUS

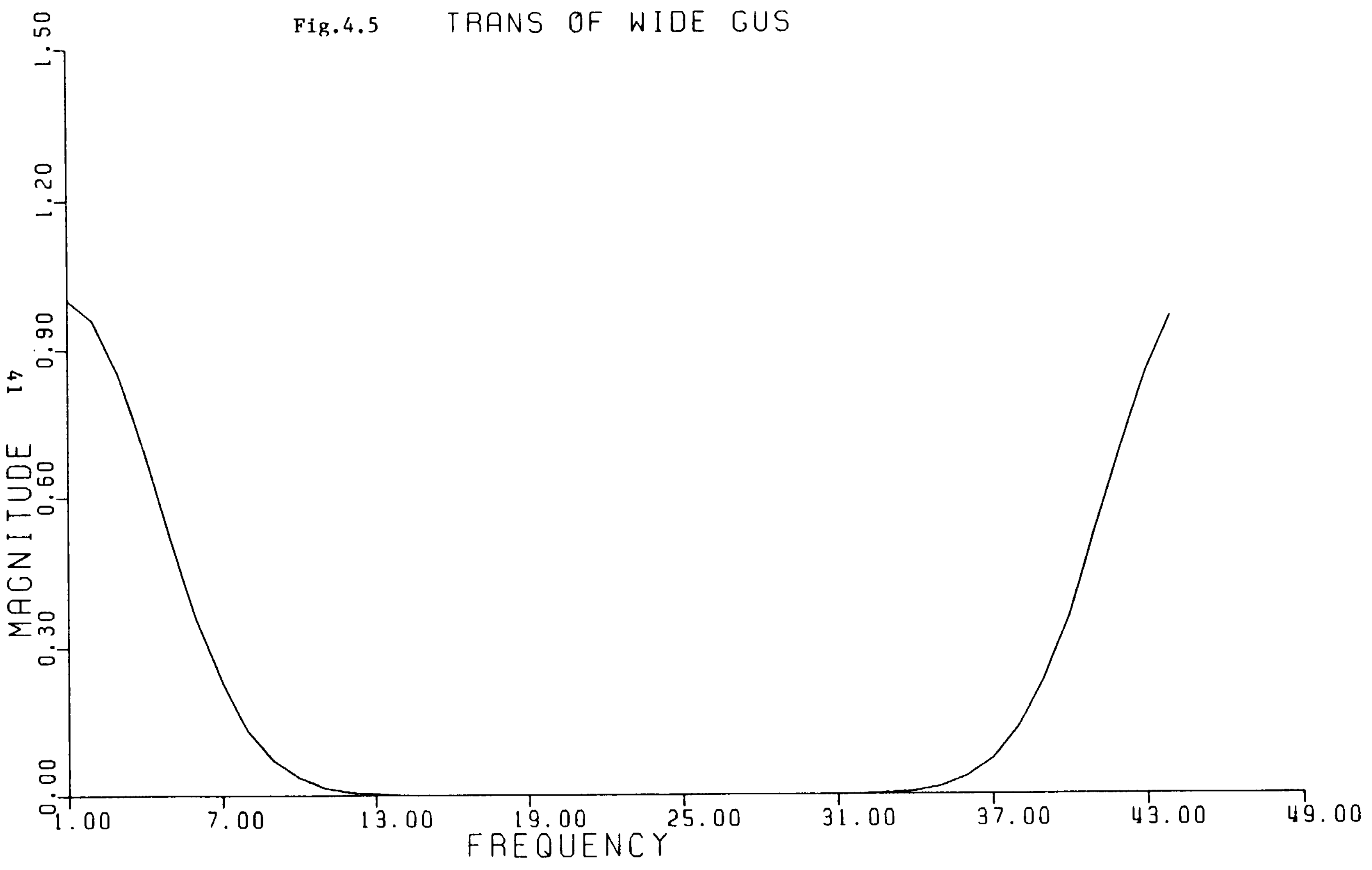


Fig.4.6 TRANS ØF INPUT

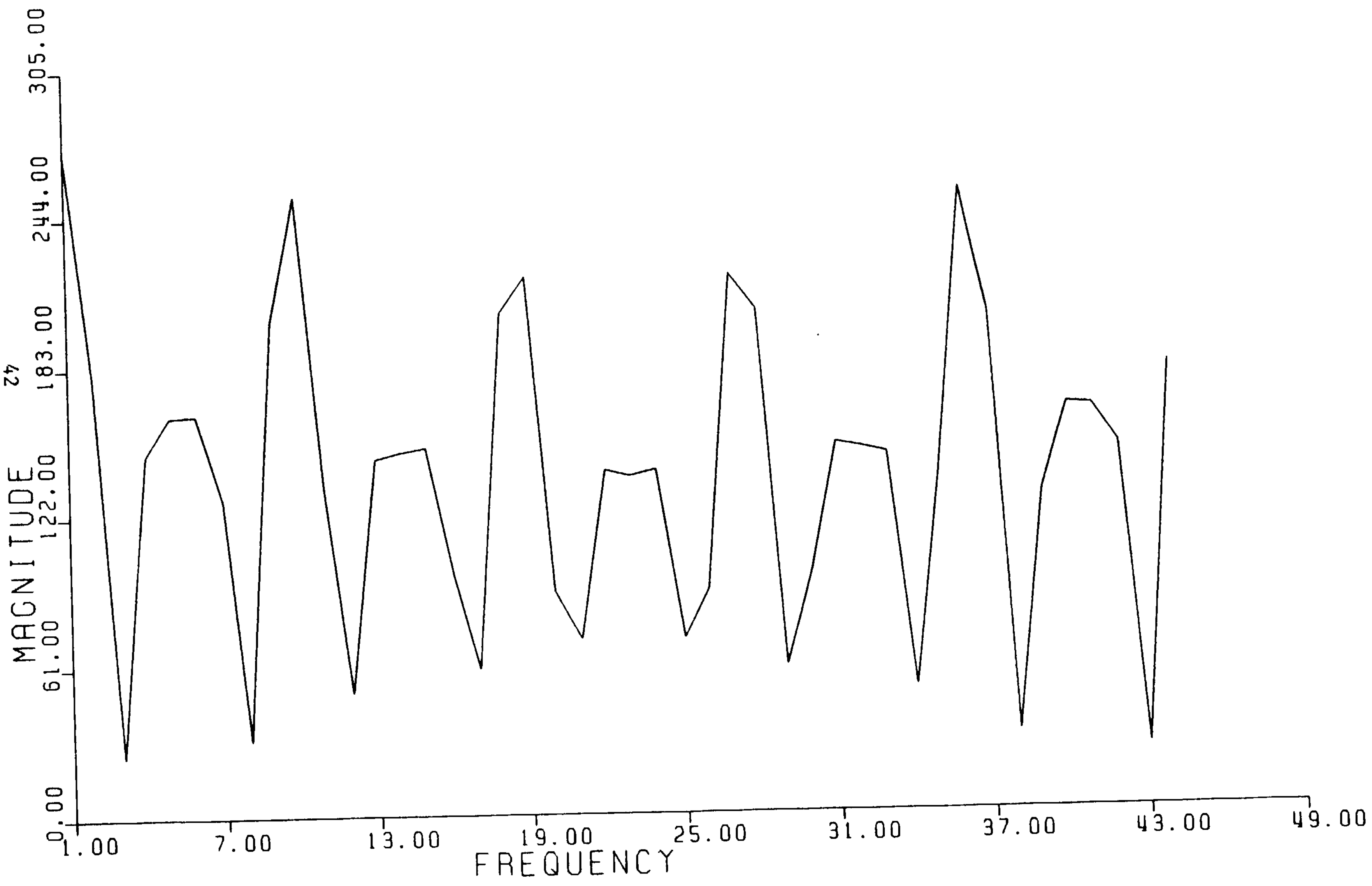
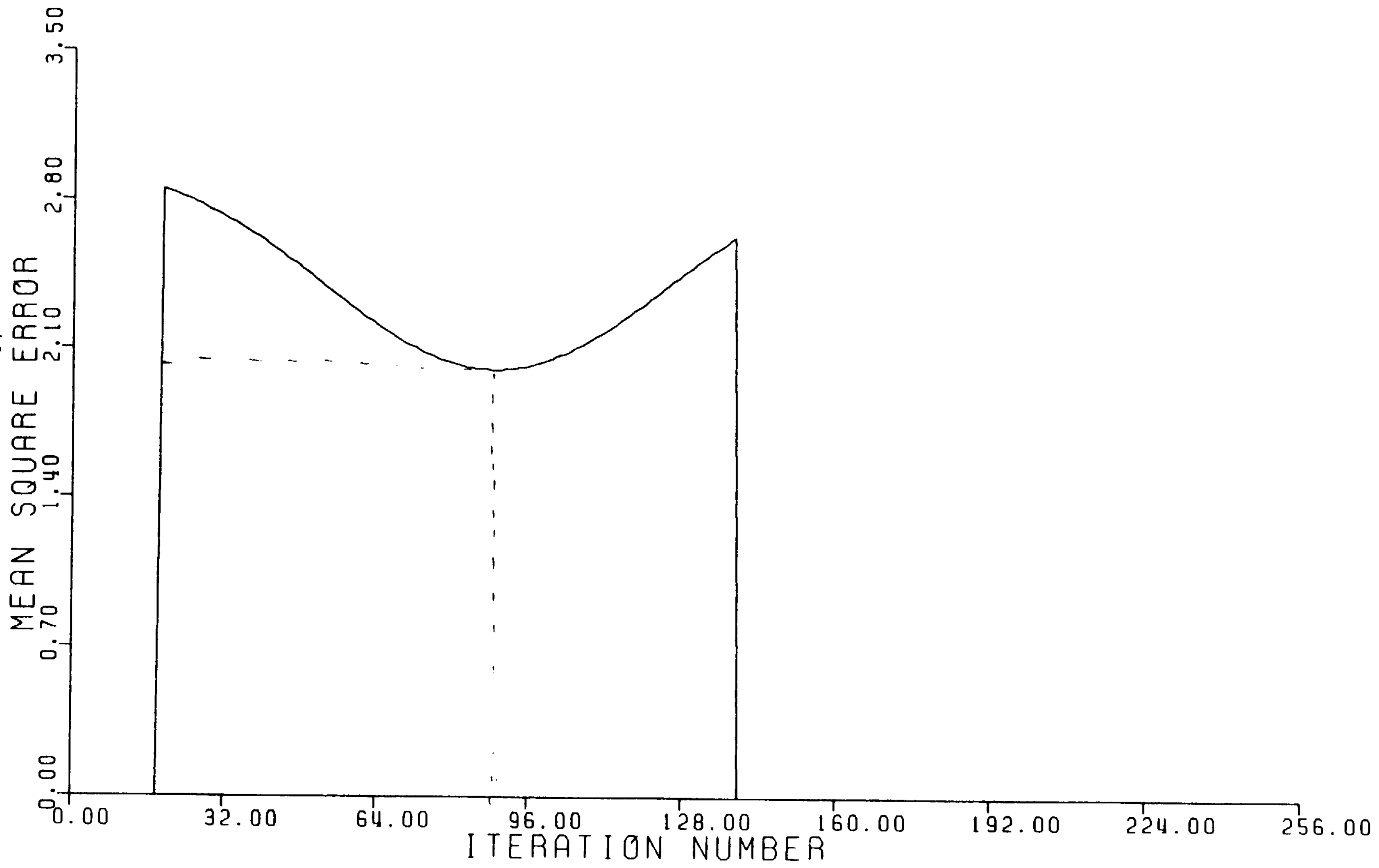


FIG.4.7



3.50
2.80
2.10
1.40
0.70
0.00
MEAN SQUARE ERROR
0.00
32.00
64.00
96.00
128.00
160.00
192.00
224.00
256.00
ITERATION NUMBER

Fig.4.8 DECON RESULT

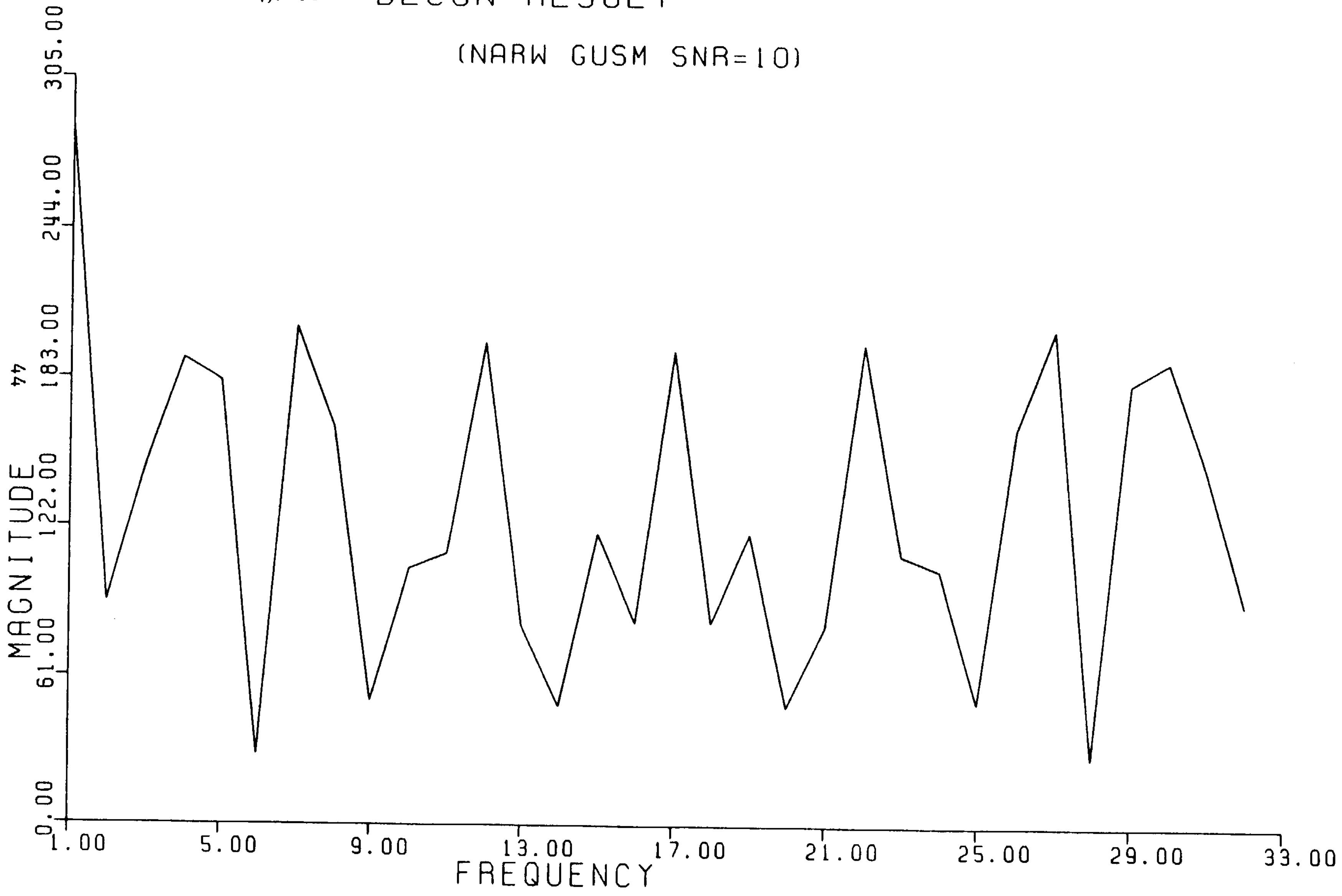


fig.4.9 DECON RESULT

(NARW GUS, SNR=150)

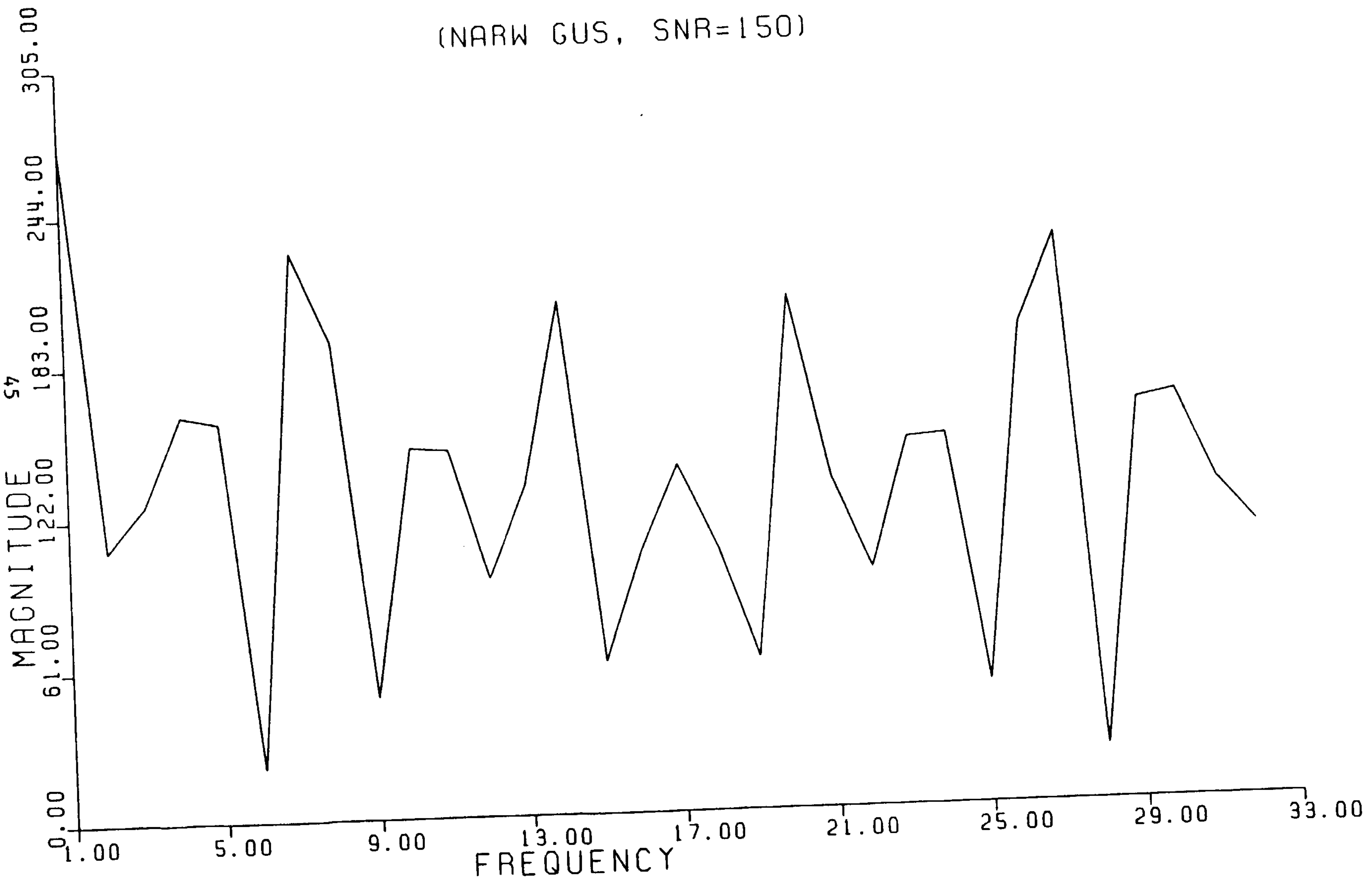


fig.4.10 DECON RESULT

(WIDE GUS, SNR=10)

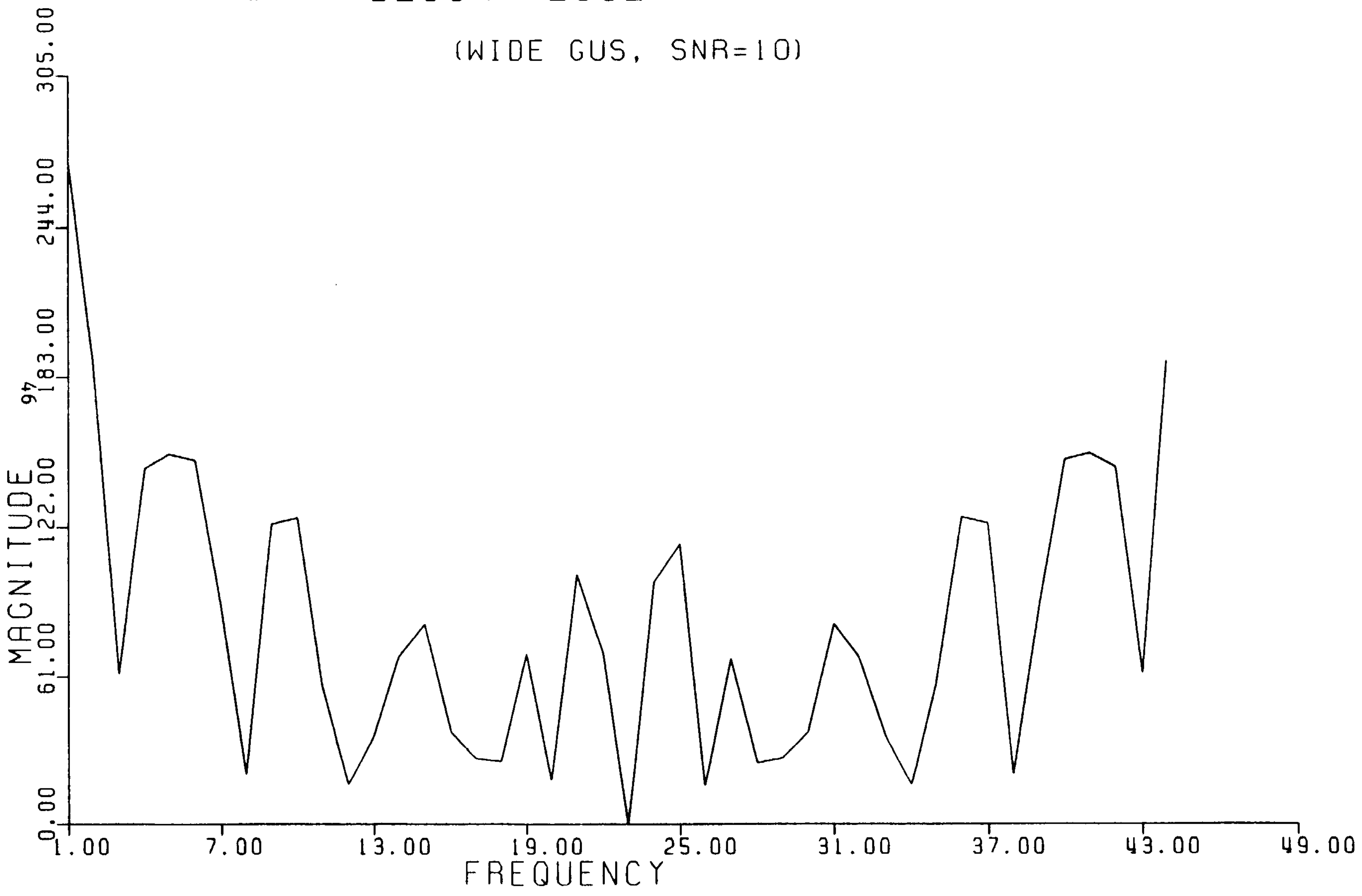


Fig.4.11 DECON RESULT

(WIDE GUS, SNR=150)

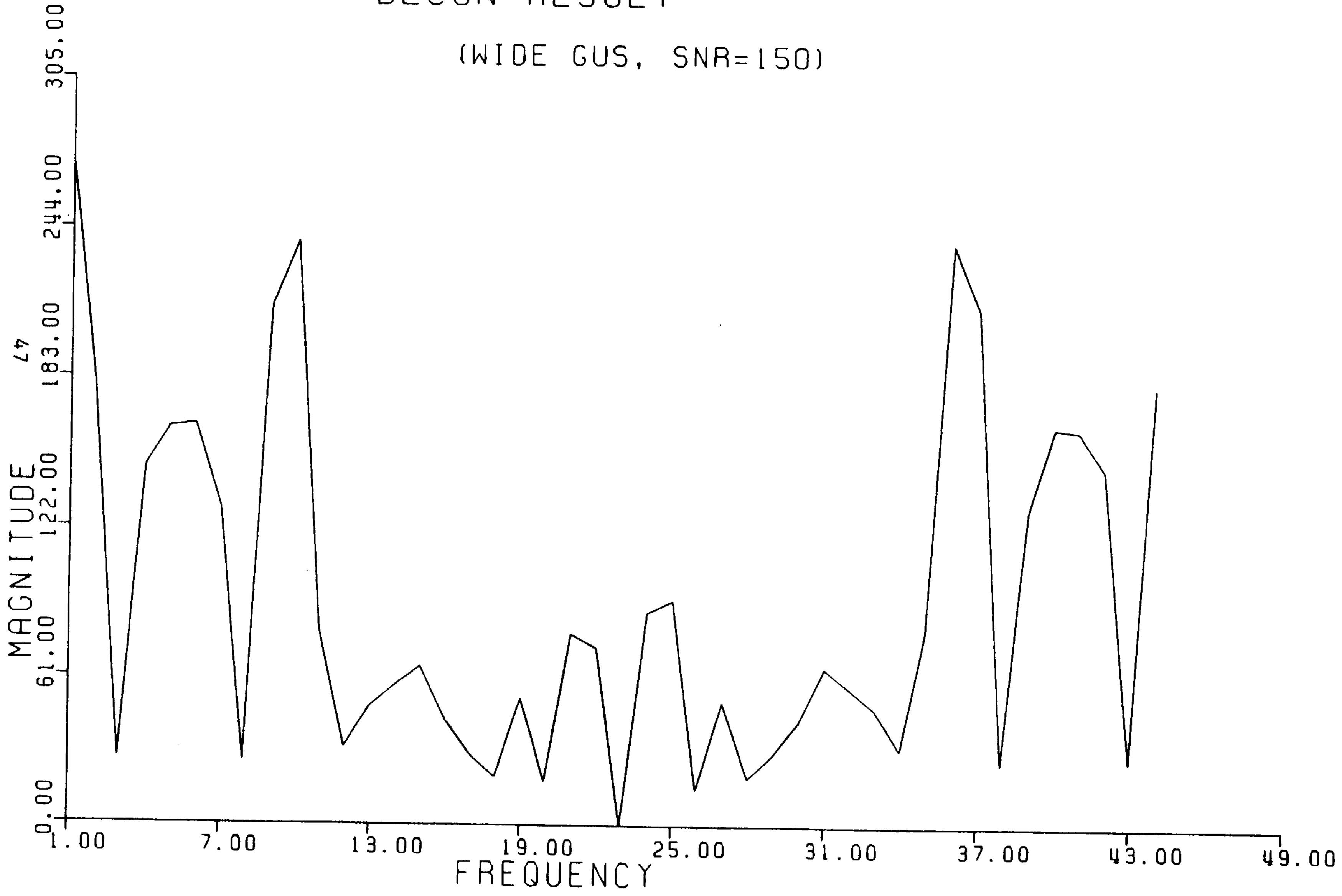


FIGURE 4.12

DIFFERENT WINDOWS

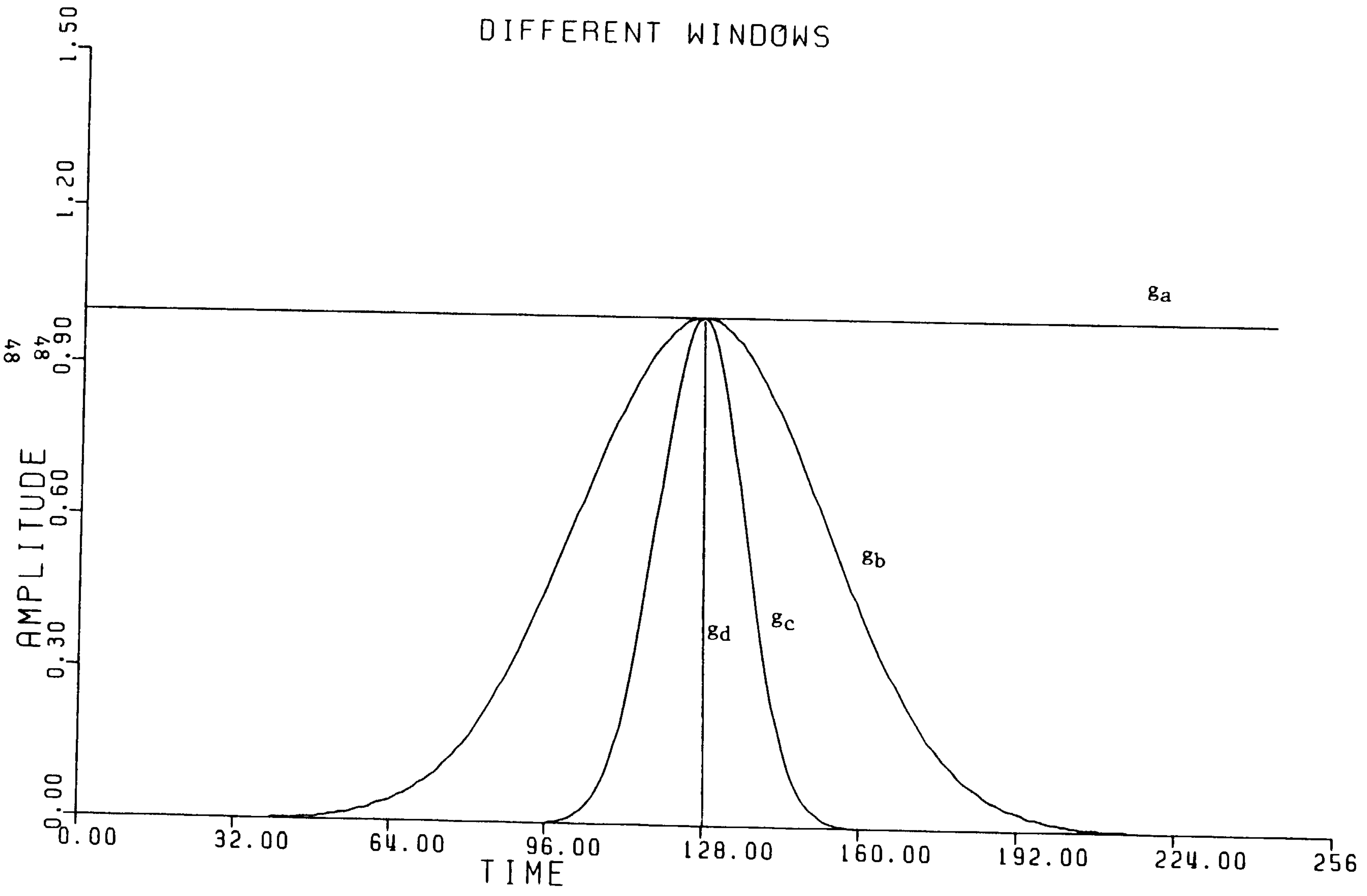


FIGURE 4.13

INPUT

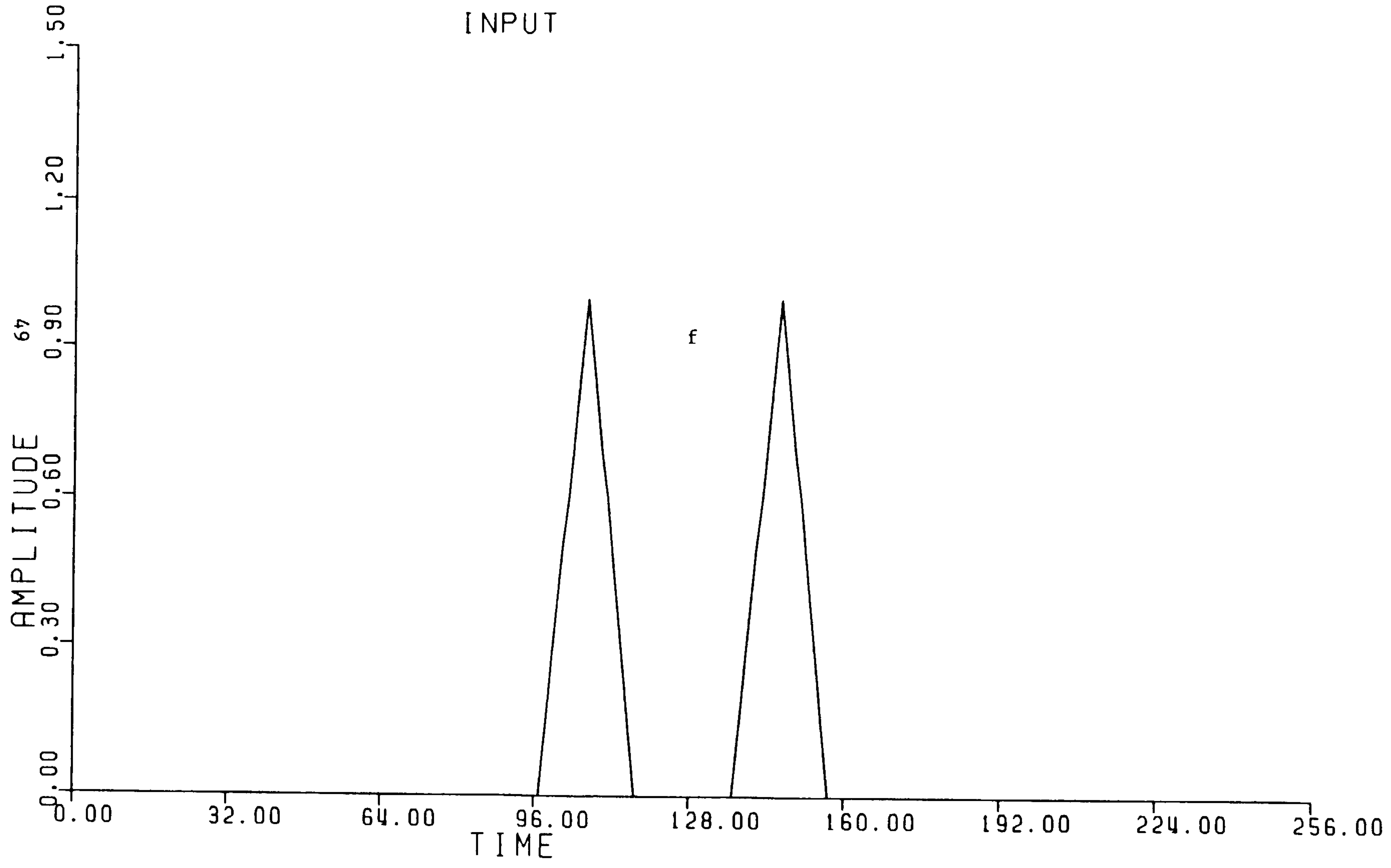


Fig.4.14 Outputs

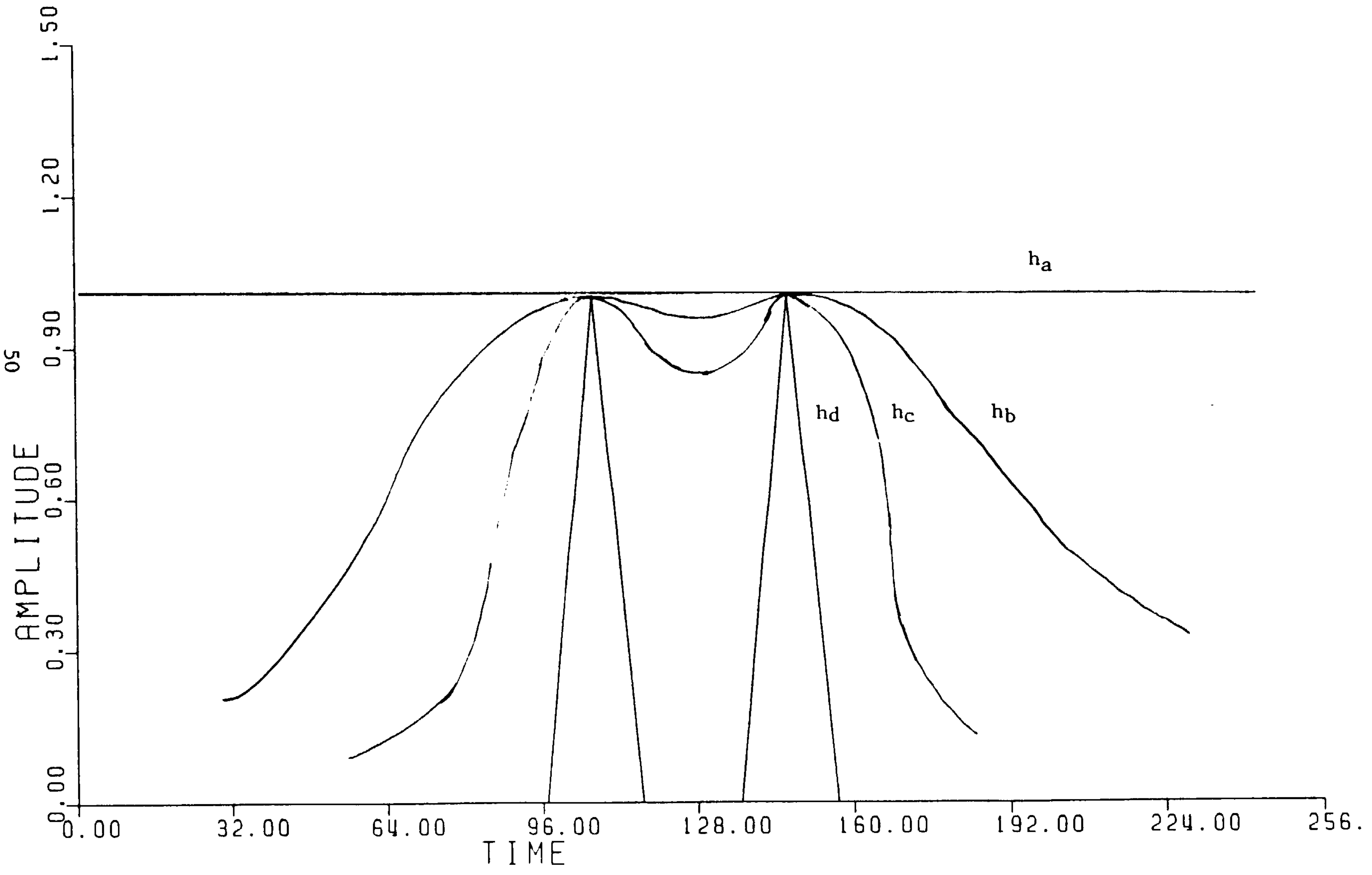
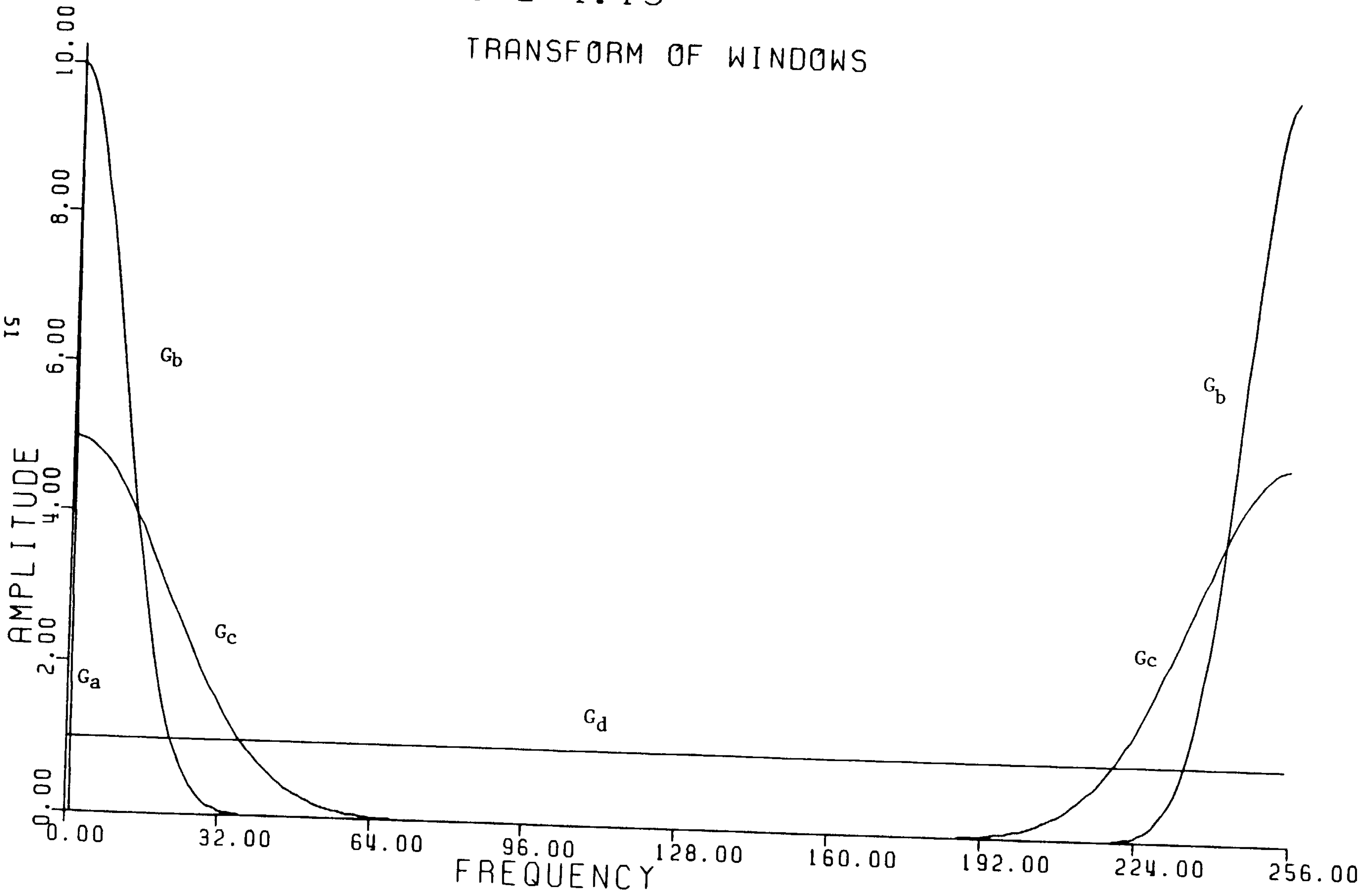


FIGURE 4.15

TRANSFORM OF WINDOWS



TRANS ØF INPUT

FIG.4.16

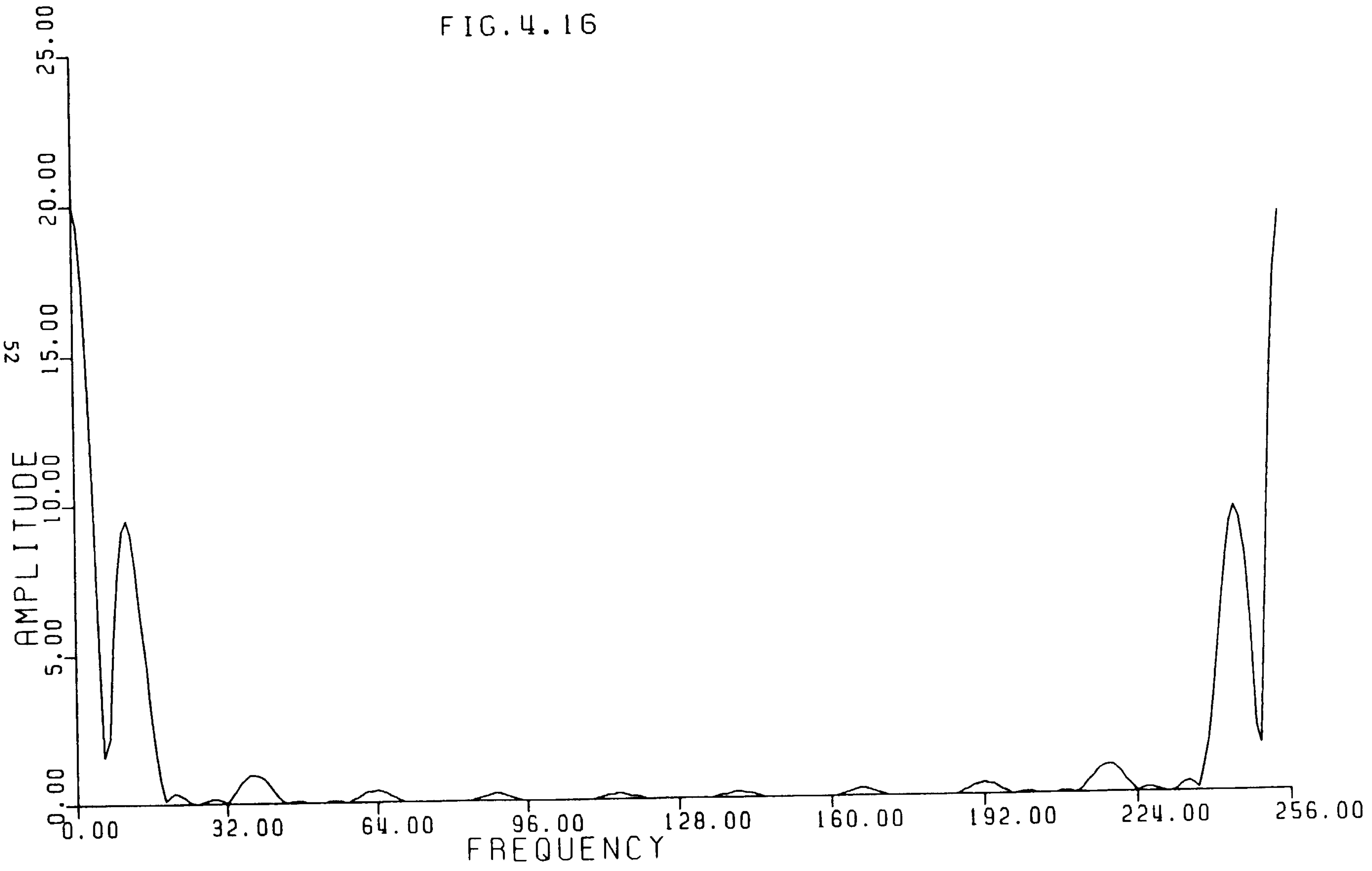


Fig.4.17

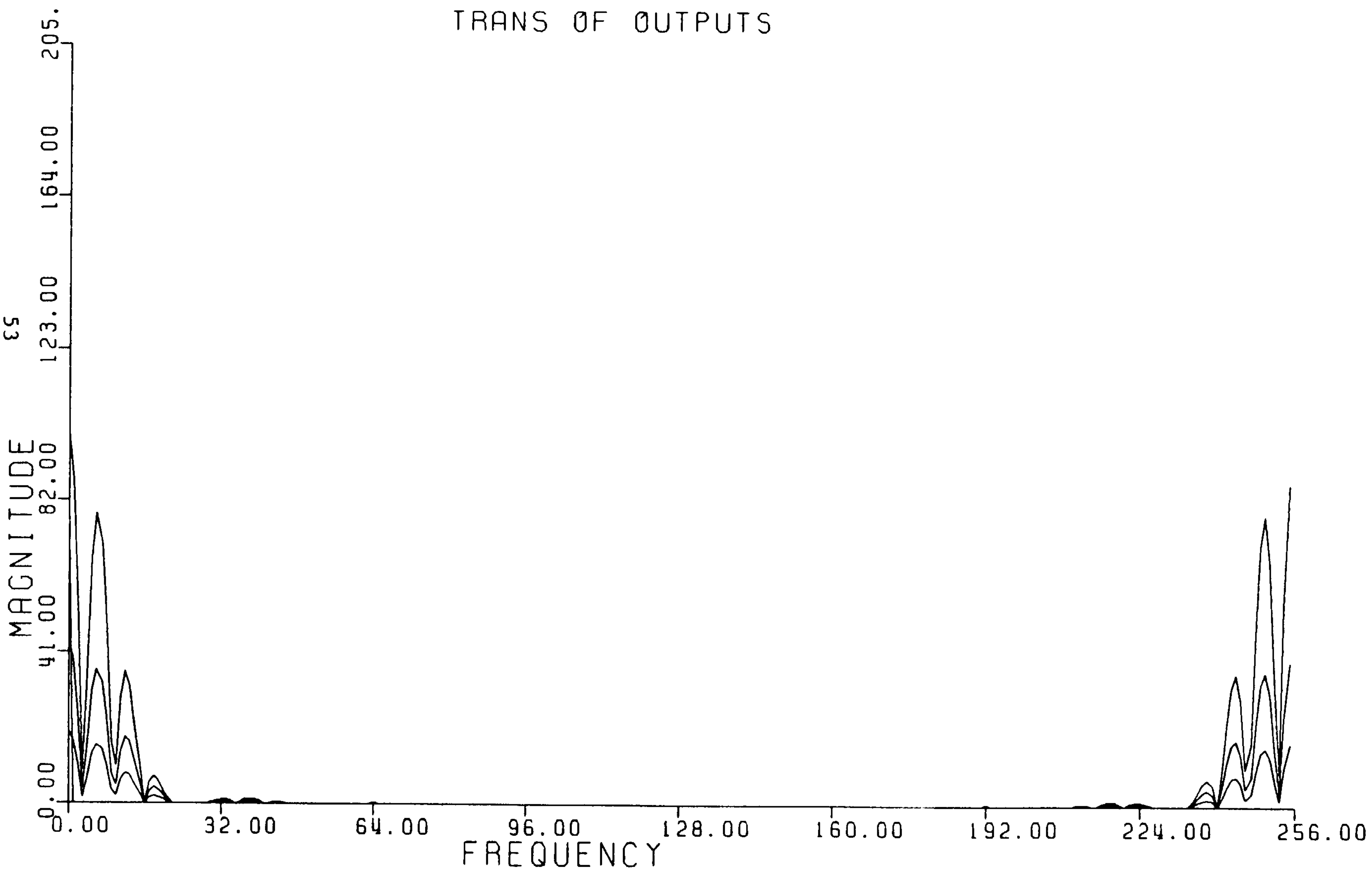

TRANS OF OUTPUTS
MAGNITUDE
0.00
41.00
82.00
123.00
164.00
205.00
FREQUENCY
0.00
32.00
64.00
96.00
128.00
160.00
192.00
224.00
256.00

Fig.4.18 OPTIMUM ITERATION

(NARROW GAUSSIAN)

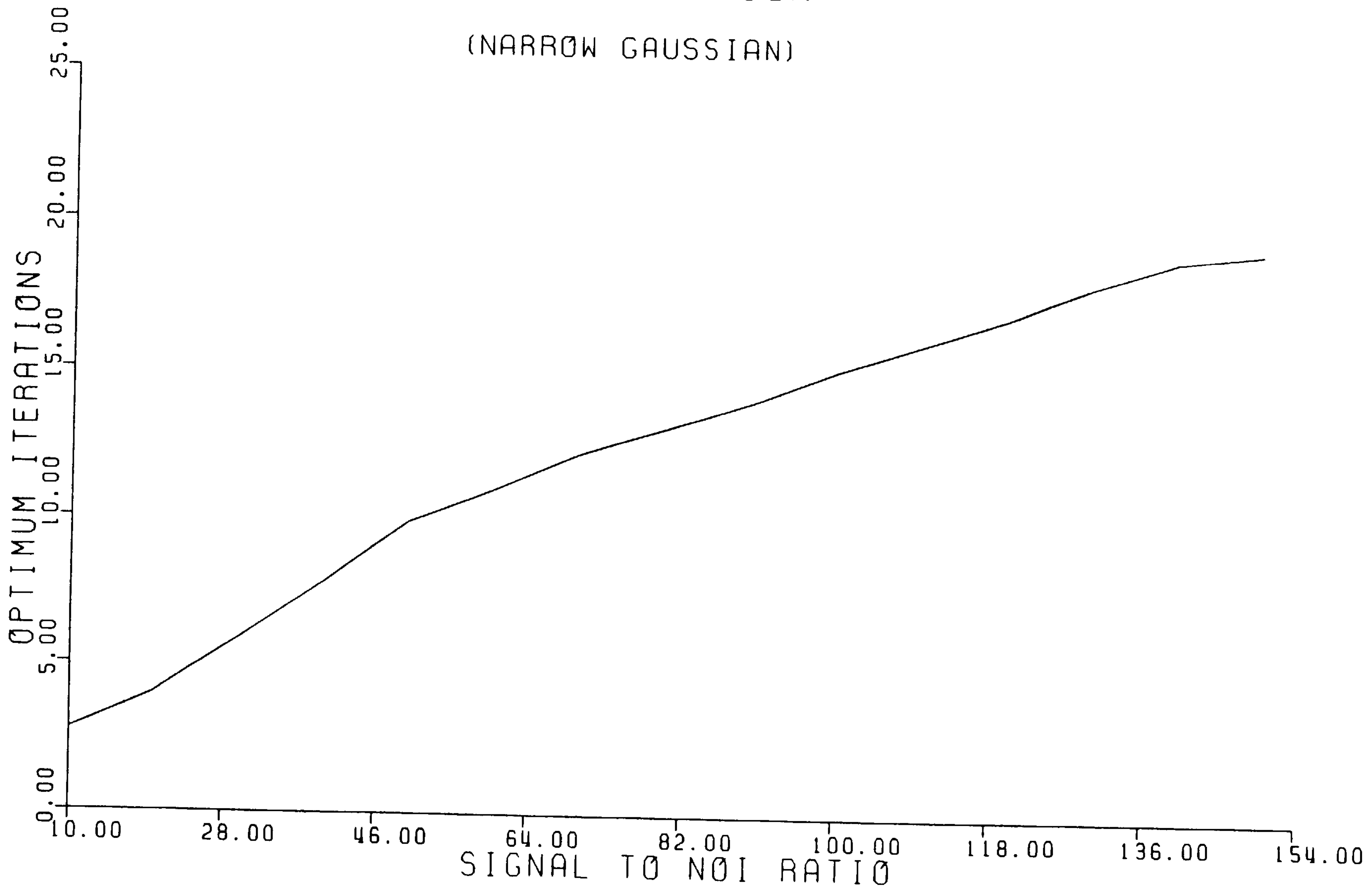


Fig.4.19 OPTUMUM ITERATIONS

(WIDE GAUSSIAN)

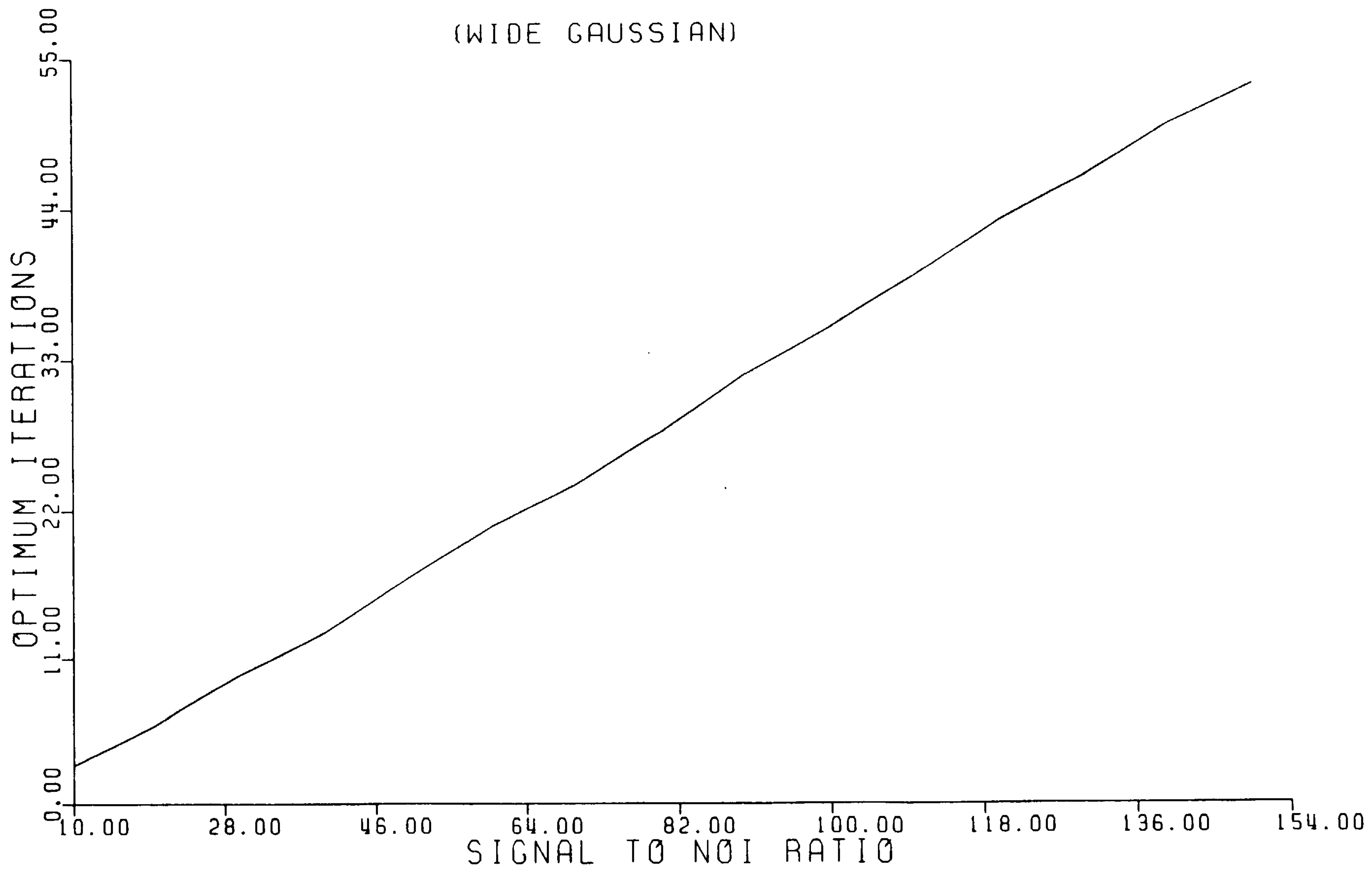


TABLE 4.2

OPTIMUM ITERATION NUMBERS OF THE WIDE GAUSSIAN CASE FROM

SIGNAL TO NOISE RATIO OF 10 TO 150
(50 CASES)

SIGNAL TO NOISE RATIO	OPTIMUM ITERATIONS
10	3.0
20	6.1
30	9.8
40	12.9
50	17.0
60	20.8
70	23.9
80	27.7
90	31.8
100	35.2
110	39.0
120	43.1
130	46.2
140	50.1
150	53.0

TABLE 4.1

OPTIMUM ITERATION NUMBERS OF THE NARROW GAUSSIAN CASE FROM

SIGNAL TO NOISE RATIO OF 10 TO 150
(50 CASES)

SIGNAL TO NOISE RATIO	OPTIMUM ITERATIONS
10	2.8
20	4.0
30	5.9
40	7.8
50	9.9
60	11.0
70	12.2
80	13.1
90	14.0
100	15.1
110	16.0
120	16.9
130	18.0
140	18.9
150	19.2

TABLE 4.4

THE MEAN SQUARE ERROR OF THE WIDE GAUSSIAN CASE FROM

SIGNAL TO NOISE RATIO OF 10 TO 150
(50 CASES)

SIGNAL TO NOISE RATIO	MEAN SQUARE ERROR
10	559.0
20	531.1
30	515.1
40	502.9
50	493.4
60	485.5
70	478.8
80	473.2
90	468.3
100	464.1
110	460.4
120	457.1
130	454.2
140	451.5
150	449.2

TABLE 4.3

THE MEAN SQUARE ERROR OF THE NARROW GAUSSIAN CASE FROM

SIGNAL TO NOISE RATIO OF 10 TO 150
(50 CASES)

SIGNAL TO NOISE RATIO	MEAN SQUARE ERROR
10	247.7
20	159.5
30	122.5
40	100.9
50	86.3
60	75.3
70	66.8
80	60.0
90	54.5
100	49.8
110	45.9
120	42.6
130	39.7
140	37.2
150	34.9

Fig.4.20

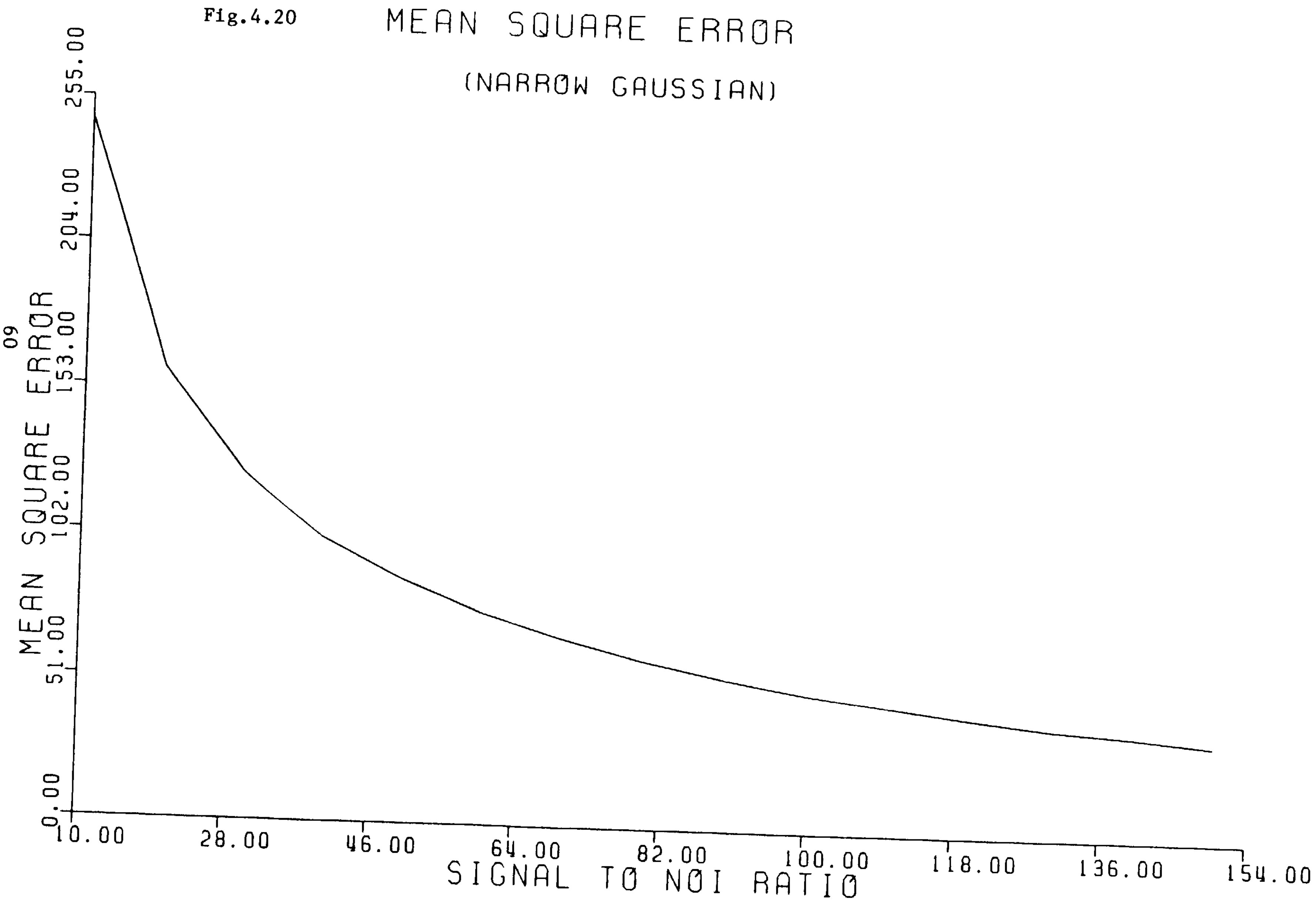

MEAN SQUARE ERROR
(NARROW GAUSSIAN)
MEAN SQUARE ERROR
0.00
51.00
102.00
153.00
204.00
255.00
10.00
28.00
46.00
64.00
82.00
100.00
118.00
136.00
154.00
SIGNAL TO NOI RATIO

Fig.4.21

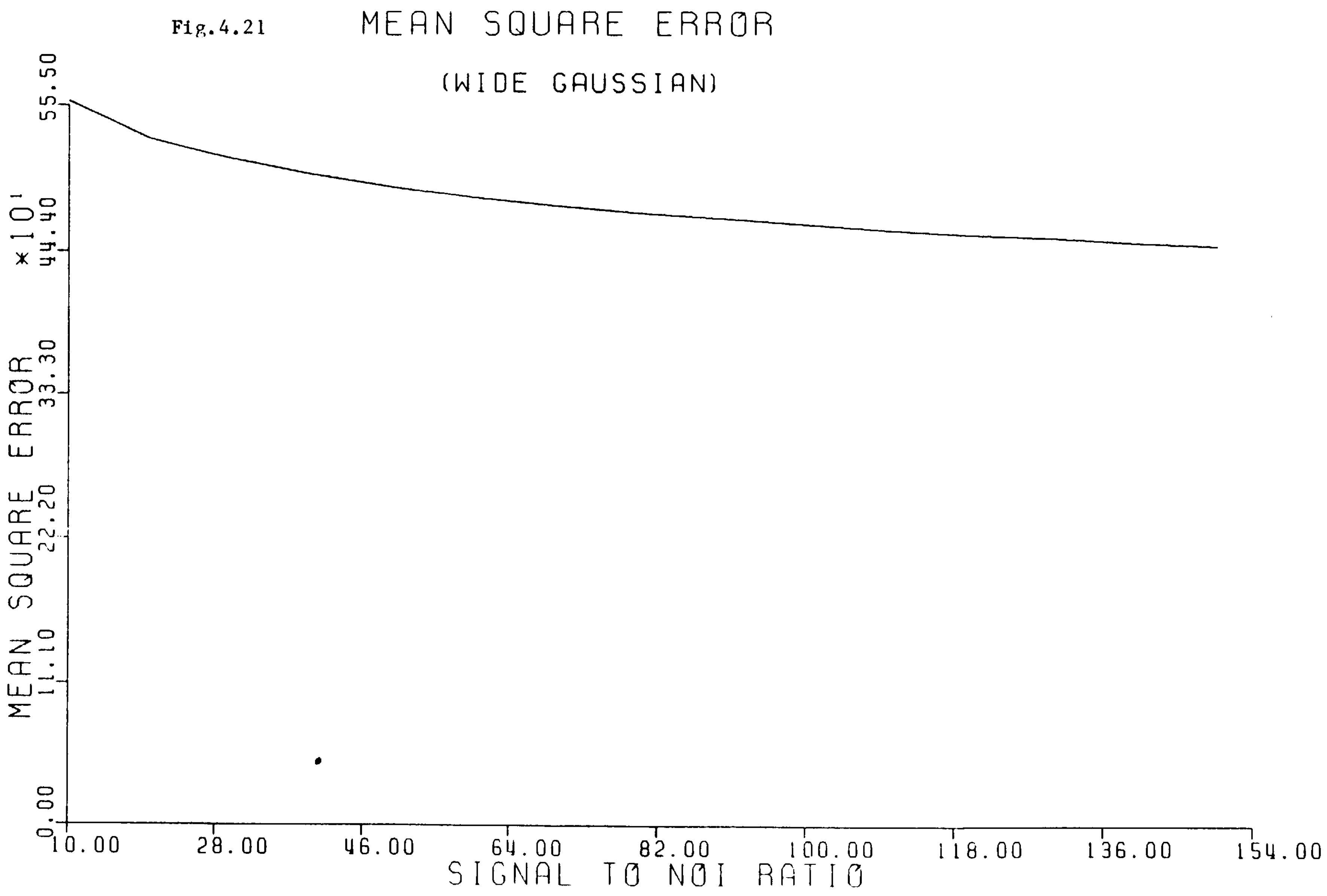

MEAN SQUARE ERROR
(WIDE GAUSSIAN)
MEAN SQUARE ERROR
*10¹
55.50
44.40
33.30
22.20
11.10
0.00
10.00
28.00
46.00
64.00
82.00
100.00
118.00
136.00
154.00
SIGNAL TO NOI RATIO

fig.4.22

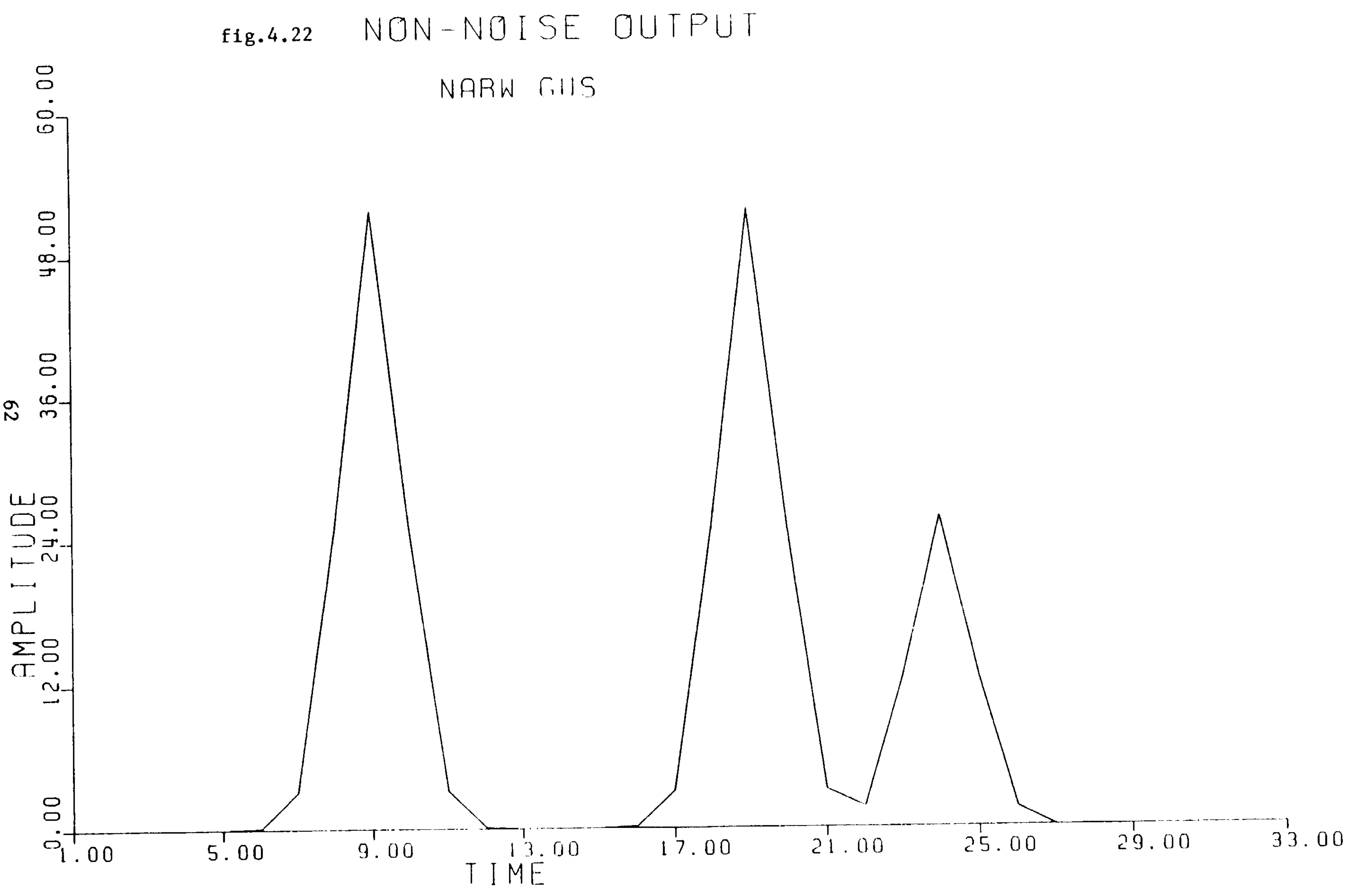

NON-NOISE OUTPUT
NARW GUS
AMPLITUDE
0.00
12.00
24.00
36.00
48.00
60.00
TIME
1.00
5.00
9.00
13.00
17.00
21.00
25.00
29.00
33.00

Fig.4.23

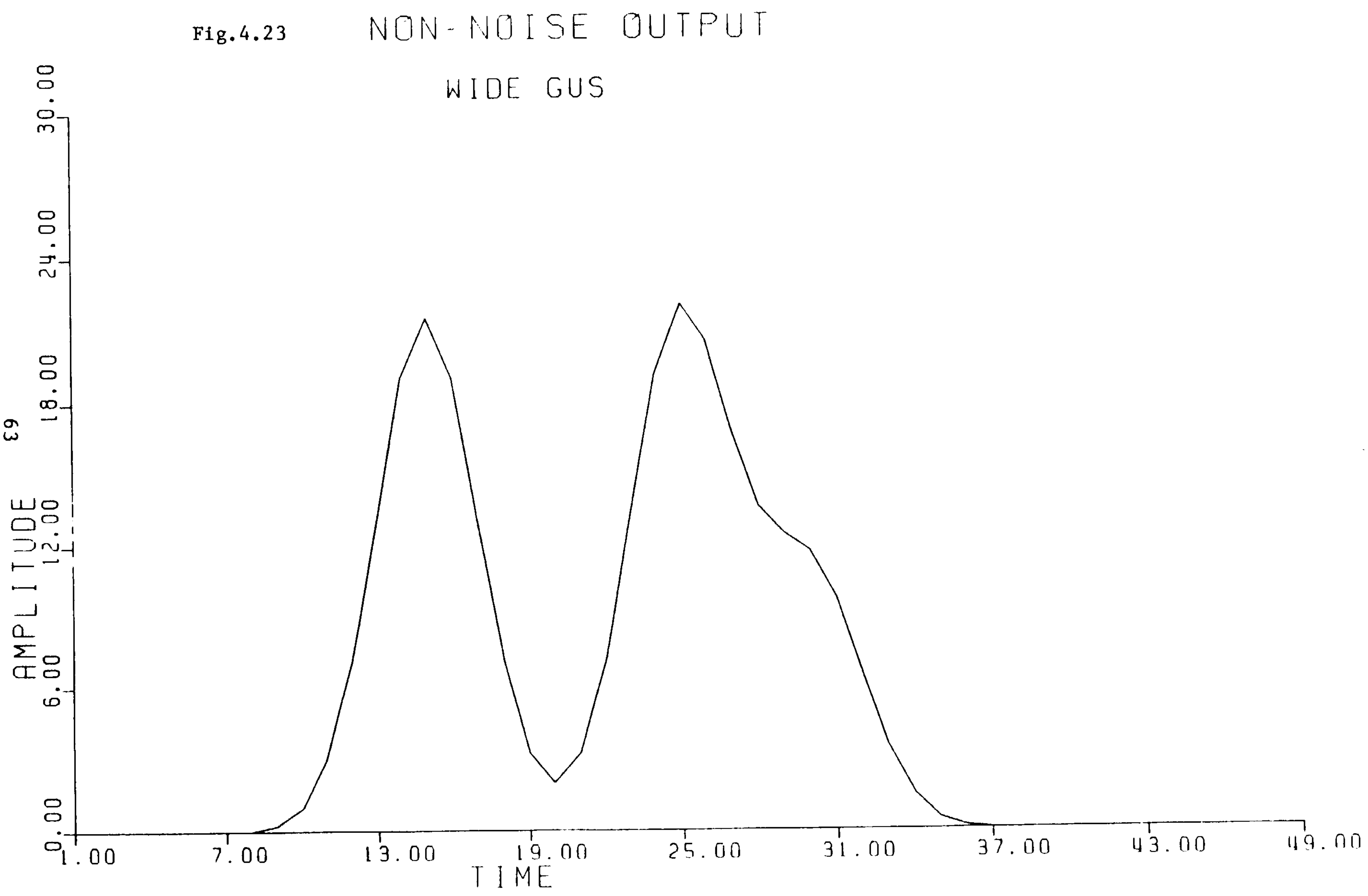

NON-NOISE OUTPUT
WIDE GUS
AMPLITUDE
0.00
6.00
12.00
18.00
24.00
30.00
TIME
1.00
7.00
13.00
19.00
25.00
31.00
37.00
43.00
49.00

fig.4.24

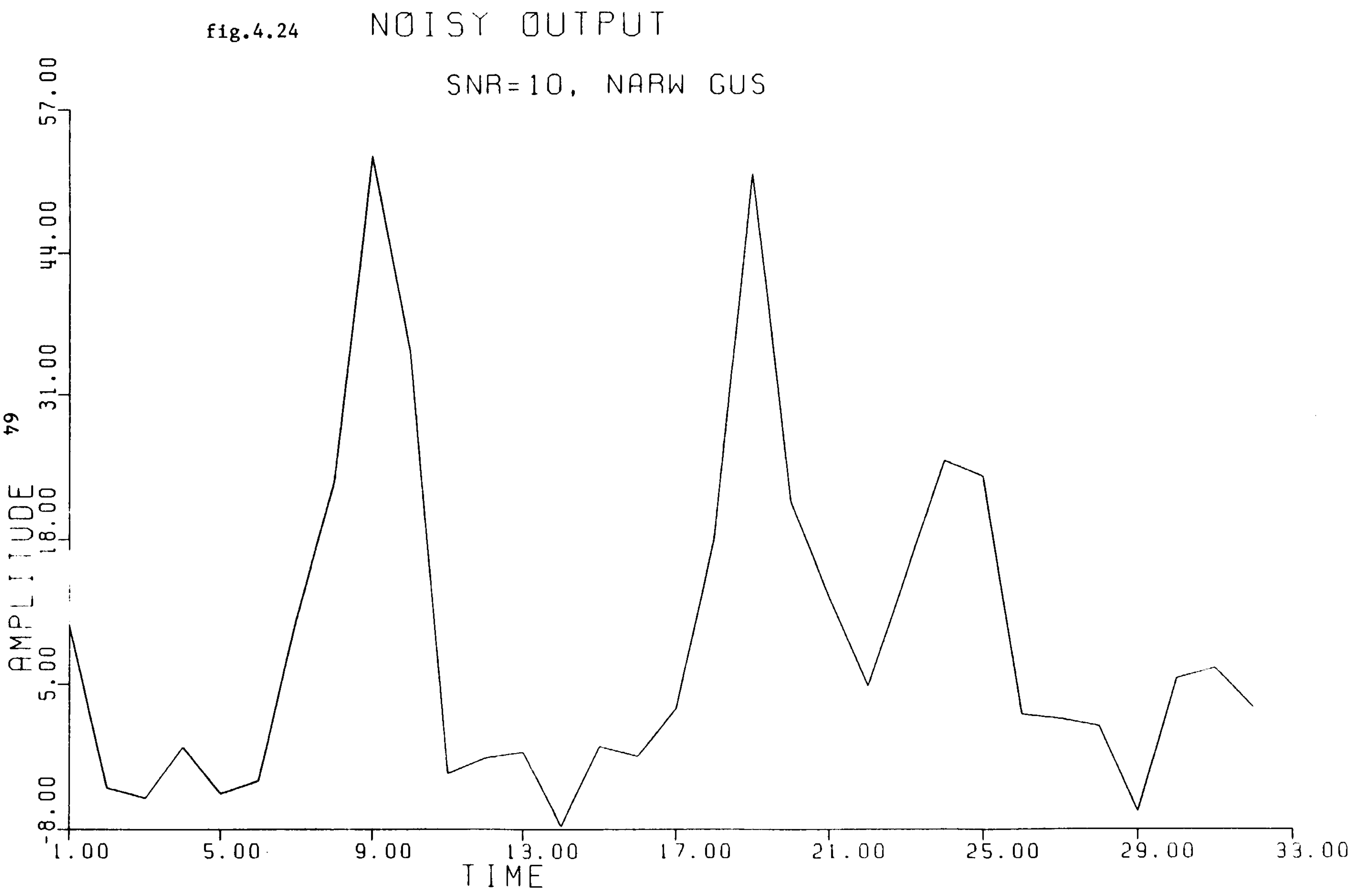

NOISY OUTPUT
SNR=10, NARW GUS
AMPLITUDE
57.00
44.00
31.00
18.00
5.00
-8.00
1.00
5.00
9.00
13.00
17.00
21.00
25.00
29.00
33.00
TIME

Fig.4.25

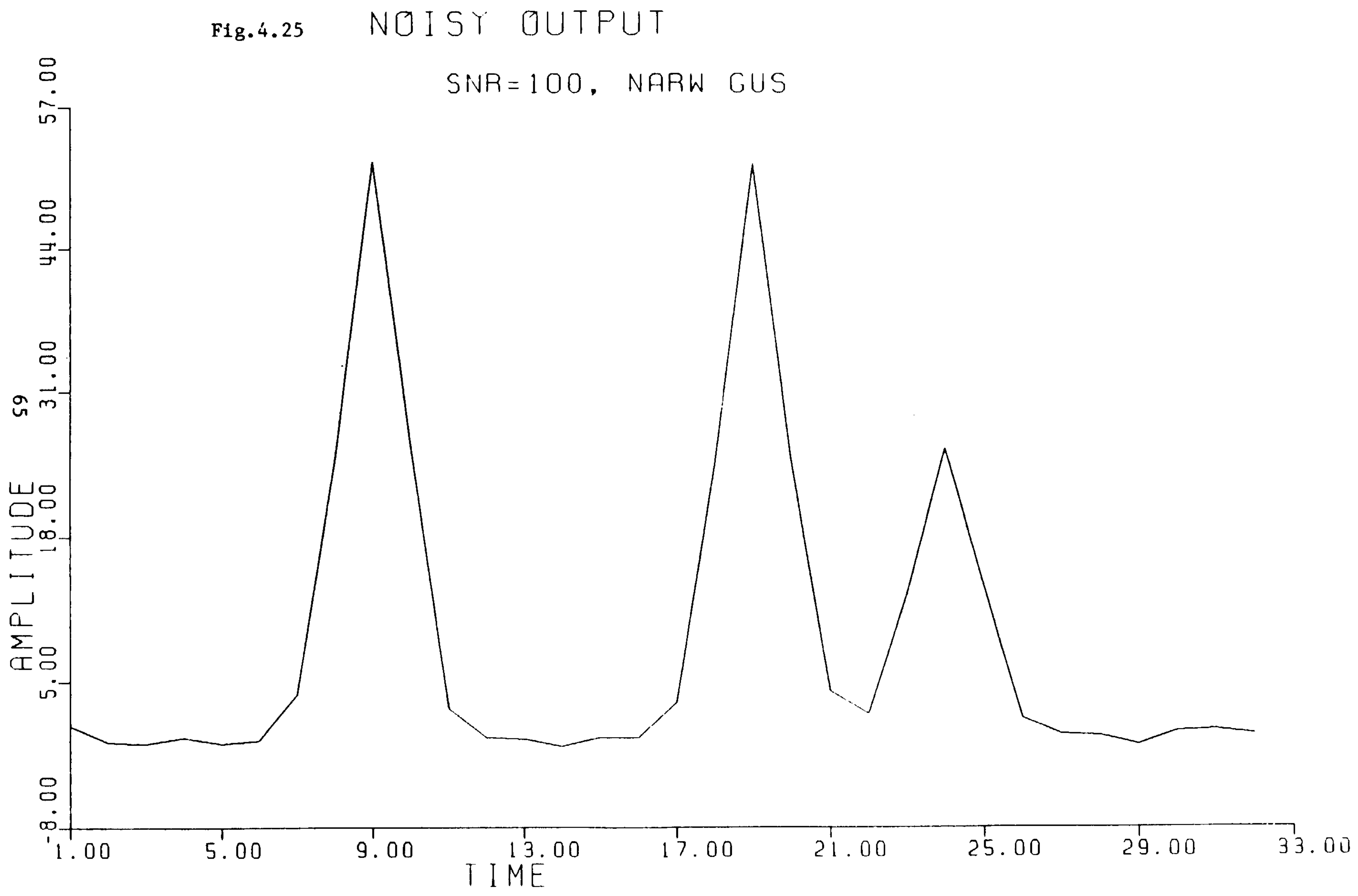

NOISY OUTPUT
SNR=100, NARW GUS
AMPLITUDE
57.00
44.00
31.00
18.00
5.00
-8.00
TIME
1.00
5.00
9.00
13.00
17.00
21.00
25.00
29.00
33.00

Fig.4.26

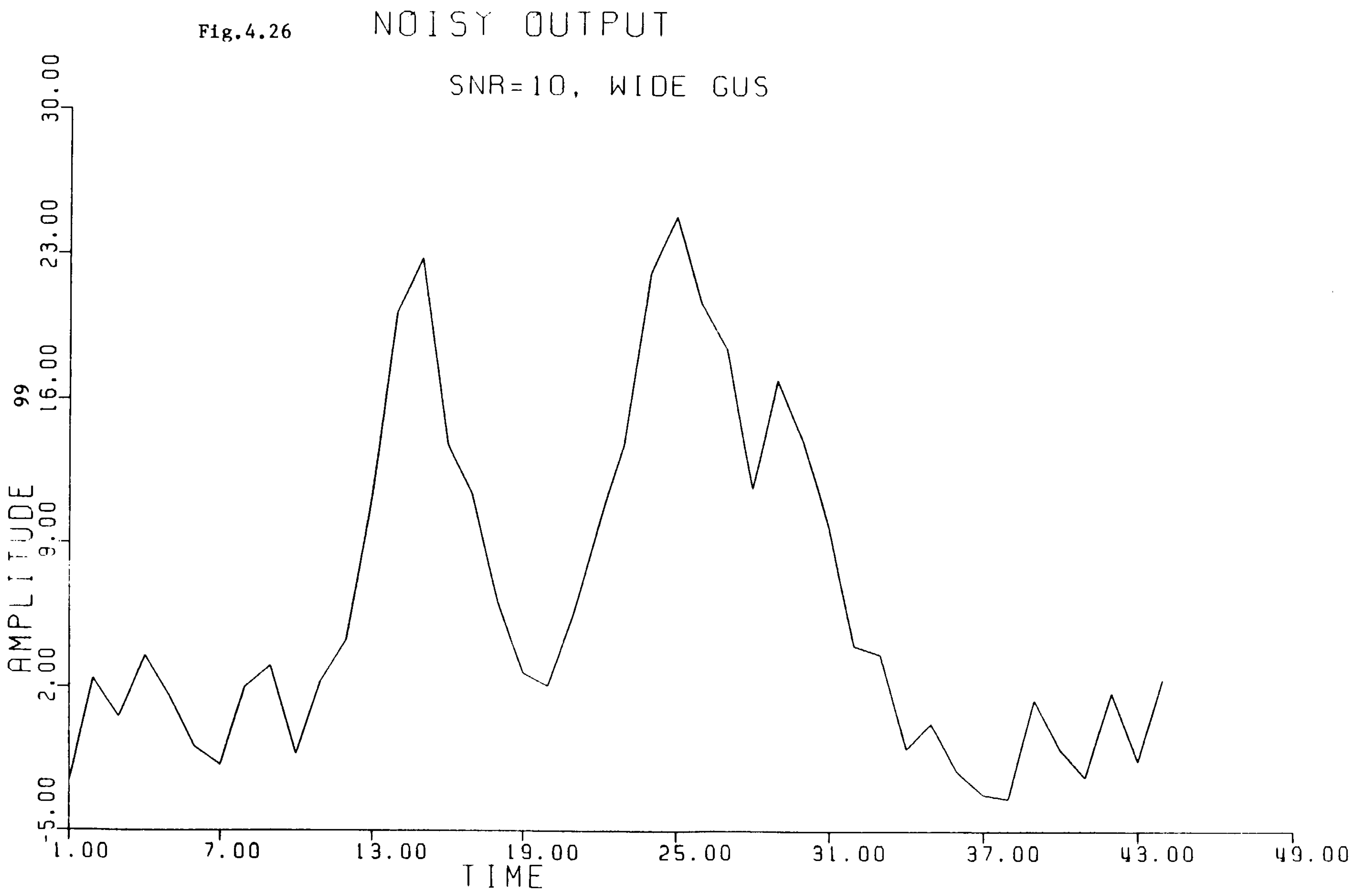

NOISY OUTPUT
SNR=10, WIDE GUS
AMPLITUDE
30.00
23.00
16.00
9.00
2.00
-5.00
TIME
1.00
7.00
13.00
19.00
25.00
31.00
37.00
43.00
49.00

Fig.4.27

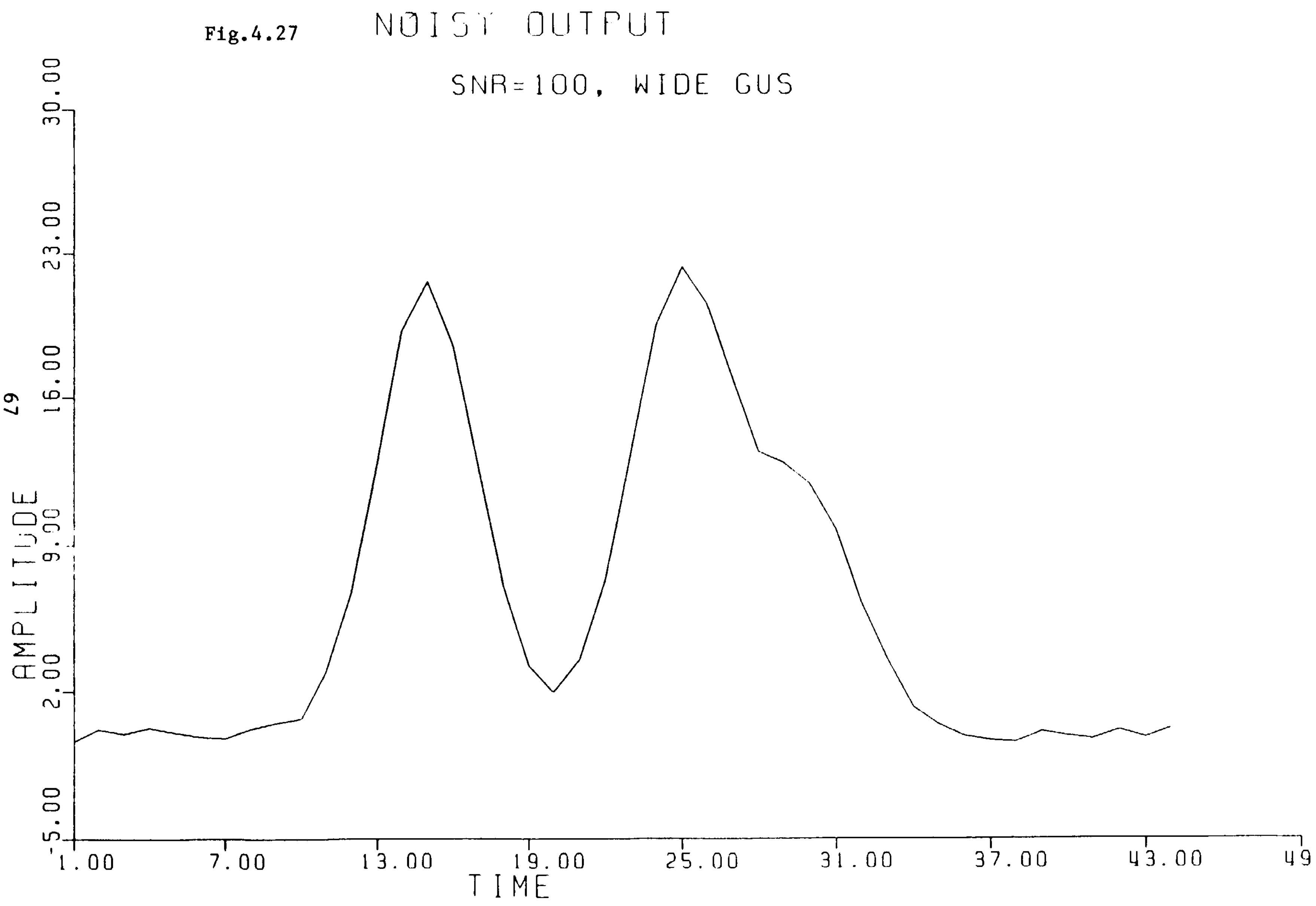

NOISY OUTPUT
SNR=100, WIDE GUS
AMPLITUDE
30.00
23.00
16.00
9.00
2.00
-5.00
TIME
1.00
7.00
13.00
19.00
25.00
31.00
37.00
43.00
49

TABLE 4.5

MSE RESULTS AT DIFFERENT LENGTHS, NARROW GAUSSIAN

SNR	LENGTH			
	32	64	128	256
	MSE			
10	82.7	80.3	80.3	80.3
20	56.6	53.6	53.6	53.6
30	41.7	38.8	38.8	38.8
40	32.3	30.1	30.1	30.1
50	26.5	24.7	24.7	24.7
60	22.4	21.0	21.0	21.0
70	19.4	18.2	18.2	18.2
80	17.1	16.0	16.0	16.0
90	15.2	14.3	14.3	14.3
100	13.7	12.9	12.9	12.9
110	12.5	11.8	11.8	11.8
120	11.5	10.8	10.8	10.8
130	10.6	10.0	10.0	10.0
140	9.9	9.3	9.3	9.3
150	9.2	8.7	8.7	8.7

REFERENCES

Ioup, G. E., 1981, Always-Convergent Iterative Noise Removal and Deconvolution: Bull.Am. Phys. Soc.,v.26,p.1213.

Ioup, G. E., and Ioup, J. W., 1983, Iterative deconvolution: Geophysics, v.48, p.1287-1290.

Amini, A. M., 1986, Optimization of Convergent Iterative Noise Removal and Deconvolution and an Evaluation of Phase-Shift Migration: M.S. thesis, University of New Orleans.

LaCoste, L. J. B., 1982, Deconvolution by Successive Approximations: Geophysics, v.47, p.1724-1730.

Leclere, J. H., 1984, Optimum use of Morrison's Iterative Method of Noise Removal for Deconvolution: M.S. thesis, University of New Orleans.

Powe, R., 1985, A Comparison of Convergent Iterative Deconvolution Methods with the Least Square Technique for Synthetic Seismic Data: M.S. thesis, University of New Orleans.

Robinson, E. A., 1980, Physical Application of Stationary Time-Series: New York, Macmillan.

Wright, K. A. R., 1980, A Study of Morrison's Iterative Noise Removal Method: M.S. thesis, University of New Orleans.

Bracewell, R. N., 1978, The Fourier Transform and Its Applications: New York, Mcgraw-Hill.