STUDY OF ONE- AND TWO-DIMENSIONAL FILTERING AND DECONVOLUTION ALGORITHMS FOR A STREAMING ARRAY COMPUTER

NASA GRANT NO. NSG-1648

FINAL REPORT

Appendix 1

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

A STUDY OF MORRISON'S ITERATIVE NOISE REMOVAL METHOD

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 Applied Physics

by

Karin Ann Ringer Wright

July 1980

ACKNOWLEDGEMENTS

I would like to thank Dr. George Ioup for his thoughtful guidance during the course of this study. I am indebted to him for his unfailing availability and his constant good humor. I would like to thank Mark and Kathy Whitehorn for their timely assistance. I would also like to thank Dr. Joseph Murphy and Dr. John Higgins.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

This thesis studies Morrison's iterative noise removal method, by characterizing its effect upon systems of differing noise level and response function.

The nature of data acquired from a linear shift invariant instrument is discussed, so as to define the relationship between the input signal, the instrument response function and the output signal.

Fourier analysis is introduced, along with several pertinent theorems, as a tool to more thorough understanding of the nature of and difficulties with deconvolution. In relation to such difficulties the necessity of a noise removal process is discussed. Morrison's iterative noise removal method and the restrictions upon its application are developed. The nature of permissible response functions is discussed, as is the choice of the response functions used in this study.

Ordinate dependendent gaussian distributed noise and constant gaussian distributed noise are discussed, along with the method used for their generation. Several parameters for the characterization of added noise are developed. Also developed are several parameters for characterization of error in the data and convergence in the method. The choice of experimental parameters is outlined.

The experimental data are presented and interpretted. Several figures containing the thrust of this work are discussed. The optimum number of iterations for noise removal is established under a variety of conditions, as is the degree of noise removal by Morrison's method. The way in which this work may be used for specific noise removal applications is discussed, along with possible extensions to the study. In the appendicies there are proofs of theorems, and a discussion and listings of various programs utilized in this study.

INTRODUCTION

Experimental observation of the physical universe is the foundation of modern science. When the phenomenon of interest is not directly observable, instruments of great delicacy and cunning have often been devised through which to observe the phenomenon. However, no instrument, regardless of the skill of its maker, is capable of receiving a signal and translating that signal into a usable form without in some way distorting the original information.

There is an entire branch of learning devoted to the techniques of enhancing experimental data and removing from it the effects of the instrumentation, regardless of the source of that data, or the nature of the instruments involved. This thesis investigates one such technique, Morrison's iterative noise removal method. It is the aim of this study to establish the degree of noise removal accomplished under various circumstances, and the optimum utilization of this method.

One method for removing the effects of instrumentation is deconvolution. Unfortunately, deconvolution fails rapidly as noise is added to the signal. For this reason, a noise removal technique, such as the one investigated in this thesis, is often a valuable precursor to deconvolution.

Chapter 1 establishes the foundations upon which this study rests, which are convolution and deconvolution, and the Fourier transform.

Chapter 2 outlines the body of the experiment. The choice of initial parameters is discussed, along with a justification for those parameters ultimately selected. Of chief interest is the selection of appropriate instrument response functions, and of a realistic (but practical) algorithm for the addition of noise.

Chapter 3 presents the results of this study. The data are categorized by their behavior, and that behavior is explained in terms of the initial parameters selected. Conclusions are drawn regarding the most efficient use of Morrison's method.

Appendix 1 contains proofs of theorems which are related to this study. Appendix 2 contains a brief discussion about and listings of the programs used in this study.

CHAPTER 1

When an instrument measures data it inevitably subjects those data to some distortion. It may lose those mathematically defined Fourier frequencies that it is unable to register and it may broaden the signal. If the instrument is linear and shift invariant the relationship between the input signal and the output is as follows:

$$h(x) = \int_{-\infty}^{\infty} f(y)\, g(x-y)\, dy$$
$$= f(x) * g(x)$$

That is, h, the output signal, is equal to the convolution of the input signal, f, with the instrument's response function, g. The convolution integral is an integral of the product of two functions , one of which is reversed and shifted across the domain of the other as the domain of the output is varied. In this manner the result is a weighted average of one function by the other (1). In the ideal case the response function of an instrument would be a delta function and

$$h(x) = f(x) * \delta(x)$$
$$= f(x)$$

so that the output signal would equal the input signal,

since $\delta(x)$ is the identity operator under convolution. In numerical work functions are often approximated as sequences, and integrals as summations. Indeed experimental data is only known discretely althougn analog data may require a large number of samples. The discrete analog of convolution is the serial product, discrete convolution, or convolution sum (2) denoted the same way as convolution, i.e. , $h=f*g$, but with f, g, and h as sequences.

Having thus been able to characterize the relationship between the input and output signals (providing that g is known, which is not always the case) it would be desirable to undo the convolution integral and to recover the actual input signal f in terms of g and h. This is known as deconvolution. Prior to further consideration of deconvolution it would be advantageous to develop the concept of the Fourier transform, as it affords several simplifications of, and greater insight into, the problem of deconvolution.

One commonly used definition of the Fourier transform is

$$F(s) = \int_{-\infty}^{\infty} f(x) e^{-i2\pi xs} dx$$

with the corresponding inverse transform (3)

$$f(x) = \int_{-\infty}^{\infty} F(x) e^{i2\pi xs} ds$$

where f(x) and F(s) are said to be a transform pair. (Note:

In this paper small letters will be used to denote functions and sequences, while capital letters will be used to denote the corresponding transforms.) Perhaps the most common example of a transform pair is a time domain waveform and its corresponding frequency spectrum. For this reason the function (t or x) domain is often referred to as the time or space domain, while the transform (s,f, or k) domain is often called the frequency or spatial frequency domain. For this work the frequency will be mathematically defined as the reciprocal of the function domain variable and will not necessarily be a physical frequency. While the information content of a signal in either domain is equivalent, in the appropriate domain it may manifest that information in a more usable aspect. Furthermore, several theorems relate operations in one domain to the corresponding operations in the other. Chief among these theorems is the convolution theorem: (Note: "$\supset$" means "has transform")

$$f(x) \supset F(s),\ g(x) \supset G(s) \Rightarrow f(x) * g(x) \supset F(s)\,G(s)$$

That is, the transform of a convolution is the product of the transforms.

The proof of the theorem is as follows (4):

$$f(x) * g(x) \supset F(s)\,G(s) \Rightarrow$$

$$\int_{-\infty}^{\infty} \{ f(x) * g(x) \}\, e^{-i2\pi xs}\, dx = F(s)G(s)$$

$$\int_{-\infty}^{\infty} \{ \int_{-\infty}^{\infty} f(y)\, g(x-y)\, dy \}\, e^{-i2\pi xs}\, dx$$

$$= \int_{-\infty}^{\infty} f(y) \left\{ \int_{-\infty}^{\infty} g(x-y) e^{-i2\pi xs} dx \right\} dy$$

$$= \int_{-\infty}^{\infty} f(y) e^{-i2\pi ys} dy \left\{ \int_{-\infty}^{\infty} g(x-y) e^{-i2\pi(x-y)s} d(x-y) \right\}$$

$$= \int_{-\infty}^{\infty} f(y) e^{-i2\pi ys} dy \; G(s)$$

$$= F(s) G(s)$$

It is frequently advantageous to consider the transform of a convolution since a product of functions is often easier to visualize and manipulate than is a convolution.

Returning to he problem of deconvolution, one may apply the convolution theorem as follows

$$h(x) = f(x) * g(x) \Rightarrow H(s) = F(s) G(s)$$

and it becomes apparent that it is much easier to solve for F(s) than for f(x). Indeed

$$F(s) = \frac{H(s)}{G(s)}$$

<u>when G(s) is not equal to zero.</u> This is an important restriction, and in fact, deconvolution develops difficulties in the regions where G is small, before G actually goes to zero. Consider Figure 1 (5,6) where the functions have been renormalized so that F(0)=G(0)=H(0)=1. The Definate Integral/Central Ordinate theorem states (see Appendix 1 for proof)

$$F(0) = \int_{-\infty}^{\infty} f(x) dx$$

If the original f was nonnegative, for this case, then

$$|F(s)| \leq |F(0)| = 1$$

as follows (5):

$$|F(s)| = \left| \int_{-\infty}^{\infty} f(x) e^{-i2\pi xs} dx \right|$$

$$\leq \int_{-\infty}^{\infty} \left| f(x) e^{-i2\pi xs} \right| dx$$

which, when f is real and nonnegative, becomes

$$= \int_{-\infty}^{\infty} f(x) \left| e^{-i2\pi xs} \right| dx$$

$$= \int_{-\infty}^{\infty} f(x) dx$$

$$= F(0)$$

by the above mentioned theorem, thus

$$|F(s)| \leq |F(0)|$$

Since ideally H(s) is a product of F(s) and G(s) where both are less than or equal to one, H(s) should be less than or equal to either, and should be zero in whatever domain either F(s) or G(s) is zero. However, the presence of noise in H(s) can cause it to be nonzero beyond the cutoff frequency of G(s) or at imbedded zeros. This noise is termed "incompatible". Noise contributions to H(s) below G(s)'s cutoff (or more specifically, where G(s)=0) are

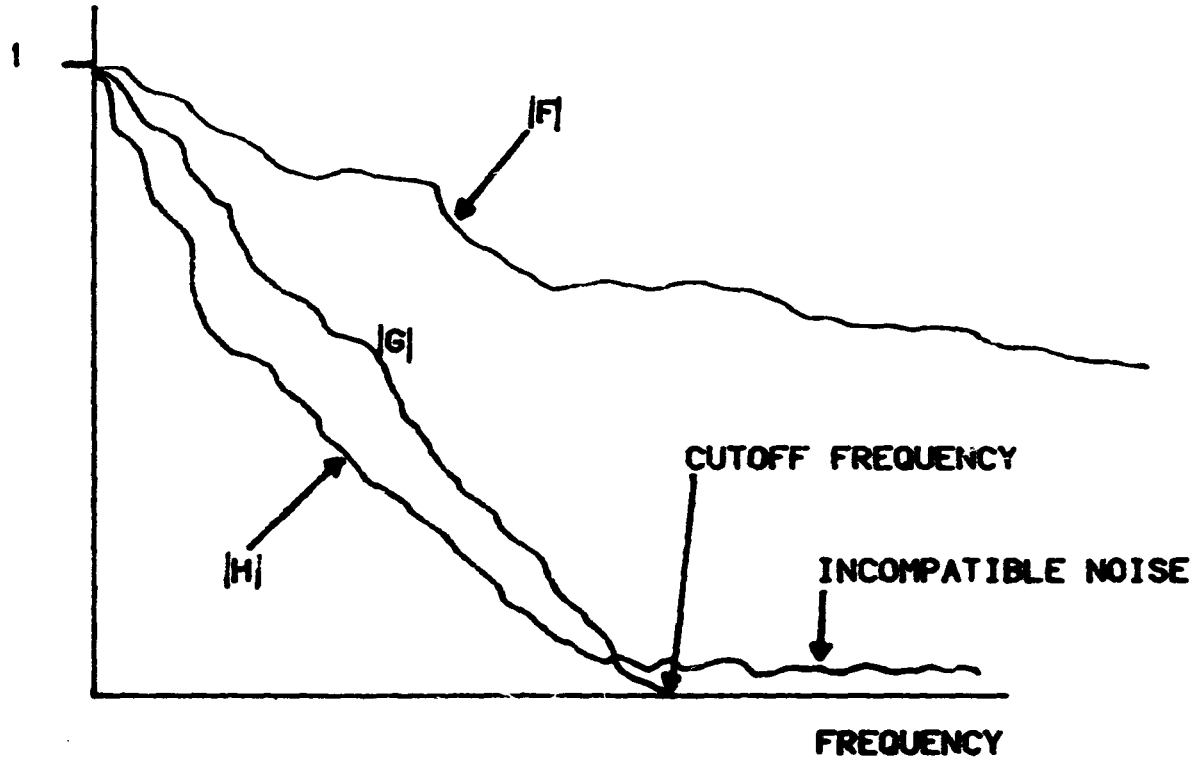


FIGURE 1

termed "compatible" noise. See Figure 1.

Deconvolution techniques are accutely sensitive in regions of small G(s) (and consequently small H(s)). F(s) is the ratio of two very small numbers, and slight fluctuations in the values of H(s) or G(s) can cause large variations in the value of F(s). For this reason it is often advantageous to subject data to some initial conditioning prior to attempting deconvolution. This thesis examines one conditioning technique, Morrison Smoothing, and attempts to characterize the effects of this technique according to the nature of the instrument response function, g, involved and the nature and amount of noise present in h, the output data.

CHAPTER 2

Morrison Smoothing is an iterative technique that should perhaps be termed Morrison Smoothing and Restoration The first iteration smooths the signal h, by convolving h with g, and each subsequent iteration proceeds to restore h back to h as follows (6):

$$h_1 = h * g$$

$$h_n = h_{n-1} + [h - h_{n-1}] * g$$

It is important to note that the first iteration, wherein h is convolved with g, results in a function (or sequence) h_1 that has no frequency component higher than those found in g, the instrument response function. (Indeed, it has no component at any frequency for which G(s) is zero.) Also, as h_n is restored back to h, it is restored by iterative convolution with g, so no higher frequencies can be introduced. In this way Morrison Smoothing eliminates incompatible noise from h.

To examine the convergence criteria for Morrison Smoothing it is useful to consider the equivalent operations in the transform domain.

$$H_1 = HG$$

$$H_2 = HG + [HG - HG^2]$$

$$= H[2G - G^2]$$

$$H_3 = 2HG - HG^2 + [HG - 2HG^2 + HG^3]$$
$$= H[3G - 3G^2 + G^3]$$

so

$$H_n(s) = [1-(1-G(s))^n]\, H(s)$$

For convergence $H_n(s)=H(s)$ which will occur if $|1-G(s)| < 1$ (8) or by the identity $0=0$ when $G(s)=0$. Therefore, as was previously stated, compatible noise is restored, weighted by the G function, with each iteration (9). It is therefore possible to cease restoration short of convergence and so to trade off resolution for noise reduction.

The convergence requirement on g is $|1-G(s)| < 1$ (10,11,12,13). For reasons to be discussed later the g functions chosen were all symmetric and singly peaked. Such functions have real transforms. Fourier analysis of several such g functions showed that the degree of negativity of a given G(s) correlated with the rate of divergence found for Morrison Smoothing of an arbitrary but noise free h. See Table 1. Since each g was normalized to have area 1, the maximum value of G, G(0), was 1 for each function, by the Definate Integral /Central Ordinate theorem. Thus the minima may be compared directly. The measure of convergence used was the variance between the original and the restored h's. Convergence was assured for functions with $G(s)>0$, for

all s, because G(s)<G(0), as discussed previously for nonnegative f(x).

Table 1

g	g's minimum	variance	
		20 iterations	100 iterations
$e^{-\pi\|x\|}$	.45	3×10^{-13}	0
$e^{-\pi x^2}$	.09	2×10^{-9}	2×10^{-12}
$sinc^2(x)$	-.005	.022	.019
sinc(x)	-.058	.156	24.4
cos(x)	-.082	.523	1033

When two functions are convolved together the result is a function broader than either. Considering the discrete analog of convolution, the serial product, if a sequence of m elements is convolved with a sequence of n elements the result is a sequence that has m+n-1 elements. If the origin for each sequence is located at the first element, the peak of the resulting sequence will be shifted, and this migration will occur at each iteration of Morrison Smoothing. The way to overcome this inconvenience is by the appropriate choice of origin for the g function. There are advantages to having the origin at the sequence maximum, or

at the center of gravity. As the functions expand under convolution it becomes difficult to locate an appropriate origin. If the origin is at the centroid it can be easily located after convolution because abscissas of centroids add under convolution (14) (See Appendix 2). However it has been found (13,14,15) that Morrison's method is more likely to converge if the origin of g is located at the maximum. For this study g functions were restricted to singly peaked, symmetric sequences (thus having the peak coincide with the centroid) with an odd number of elements. These constraints ensured a well behaved g function without being unduly confining.

Returning briefly to the transform of the nth iteration of Morrison Smoothing, recall that

$$H_n(s) = H(s)\left[1-(1-G(s))^n\right]$$

$H_n(s)$ will be most like H(s) when the term $[1-(1-G(s))^n]\sim 1$. For a constant n this will occur when G(s) is close to 1 (slightly less than or equal to 1). It is thus apparent that g functions which will cause Morrison Smoothing to converge rapidly are those possessing a broad transform.

Considering the above and the results of Table 1, three g functions were selected. First, a narrow gaussian (which has a broad transform) was chosen to represent functions which will converge rapidly. Second, a broad gaussian (whose transform is thus narrow) represents functions which

converge slowly. Third , a sinc squared function (whose transform is slightly negative) was chosen to represent functions which are slightly divergent. The last was included to test whether a slowly diverging function can be used for a few iterations before divergence becomes too accute.

Some care was required for an appropriate choice of a sinc squared function. Since the data are discrete it is implicit that the sampling interval between points is equal. This is important because a function with few points is then narrow by definition, and vice versa, if all functions are sampled at the same interval, as was the case for this study. The width of a function determines (inversely) its breadth in the transform domain, and so its rate of convergence, as discussed above. It was necessary to sample the main lobe of the sinc squared function coarsely enough that its breadth in the frequency domain was similar to that of the narrow gaussian (so that they might be compared) yet sufficiently finely so that the essential characteristics of the sinc squared function were retained. The Fourier transform was used to analyze each prospective g function. See Figure 2 for graphs of the transforms of the g functions used. The maximum negative value in the sinc squared transform (which determined the rate of divergence) was -0.0079, compared to the peak value of 1.0. For comparison, the transform of the broad gaussian is given in Figure 3.

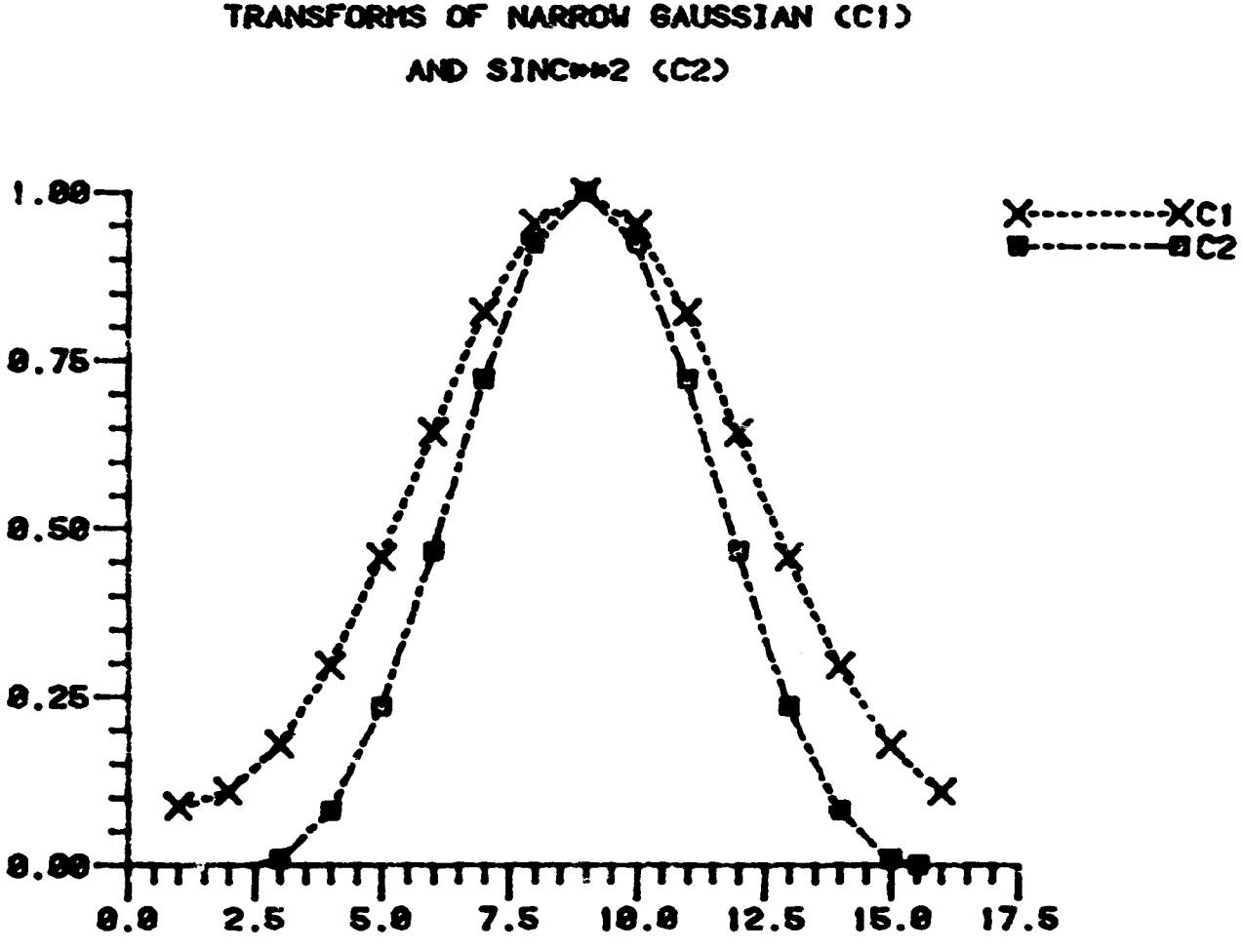

TRANSFORMS OF NARROW GAUSSIAN (C1)
AND SINC**2 (C2)
1.00
0.75
0.50
0.25
0.00
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
C1
C2

FIGURE 2

TRANSFORM OF BROAD GAUSSIAN

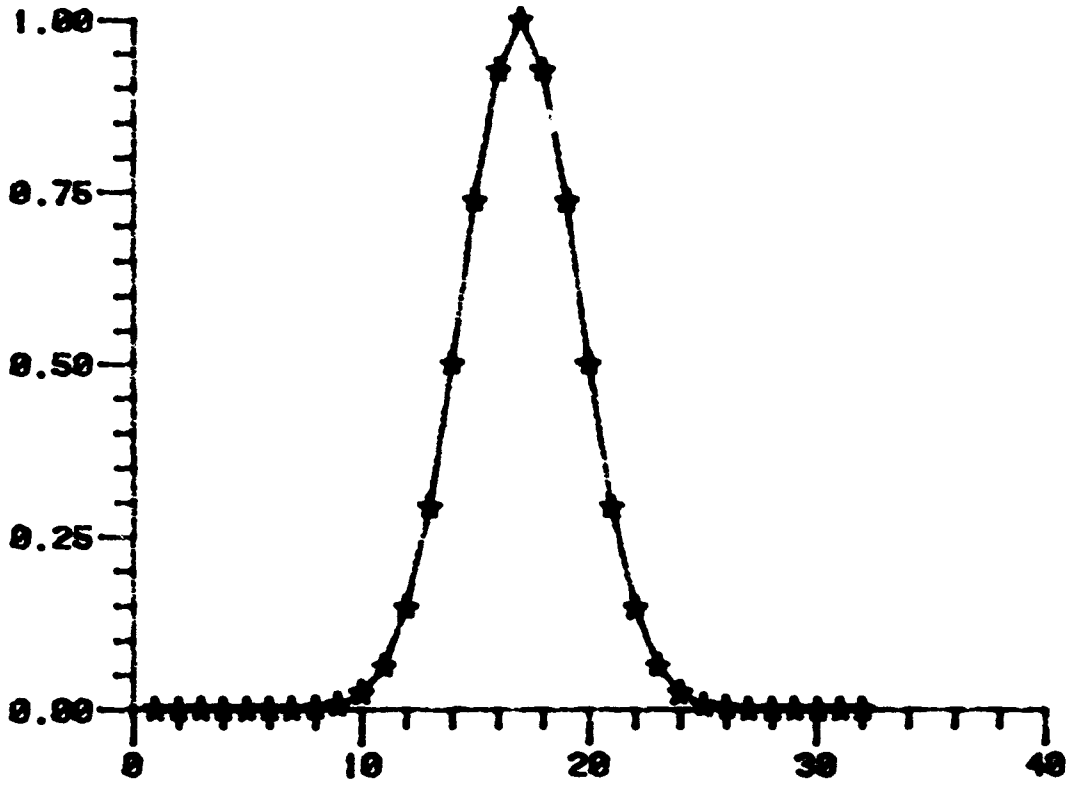


FIGURE 3

As was discussed initially, data, h, from a linear shift-invariant instrument is the convolution of the input signal, f, with the instrument response function, g. Morrison Smoothing concerns itself only with g and h, but for real data these two functions are related (since h=f*g). For this study h functions were constructed as follows. An arbitrary (but realistic) f sequence consisting of three narrow gaussians was convolved with each g function in turn, to produce the three basic h functions. See Figures 4-10. (note the difference in scale size.)

Having thus obtained the necessary g and h functions it was necessary to add noise to the h function to demonstrate the ability of Morrison Smoothing to remove incompatible noise. One goal of this study was to find if incomplete restoration (i.e. not iterating to convergence) would be useful. Since the noise builds proportionally to G with each restoration it was anticipated that the function might over some range of iterations be restored faster than the noise. Were this the case, it would be beneficial to terminate iterations at the end of that range, and thus achieve some noise reduction.

The types of noise considered were two, constant gaussian noise and ordinate dependent gaussian noise. Gaussian noise is that which has the bell curve ($e^{-\pi x^2}$) distribution about any given data point. For constant gaussian noise the width of that bell curve is fixed. For

STANDARD F FUNCTION

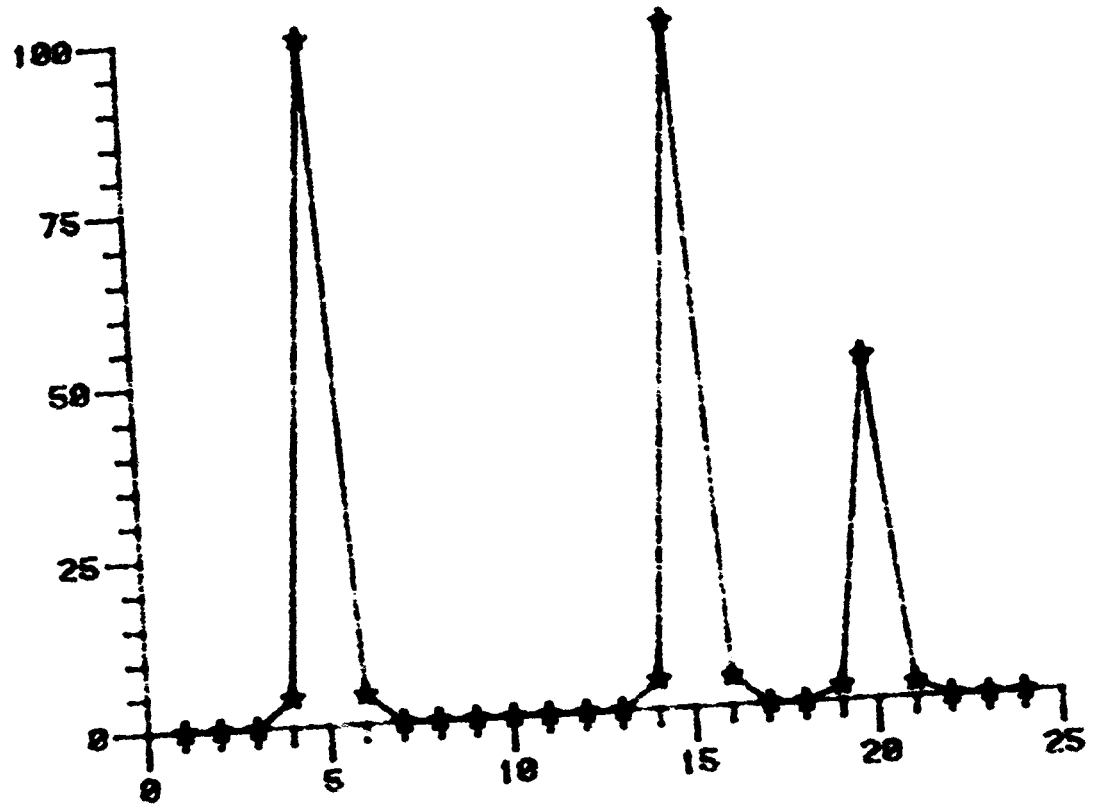


FIGURE 4

G FUNCTION FOR NARROW GAUSSIAN

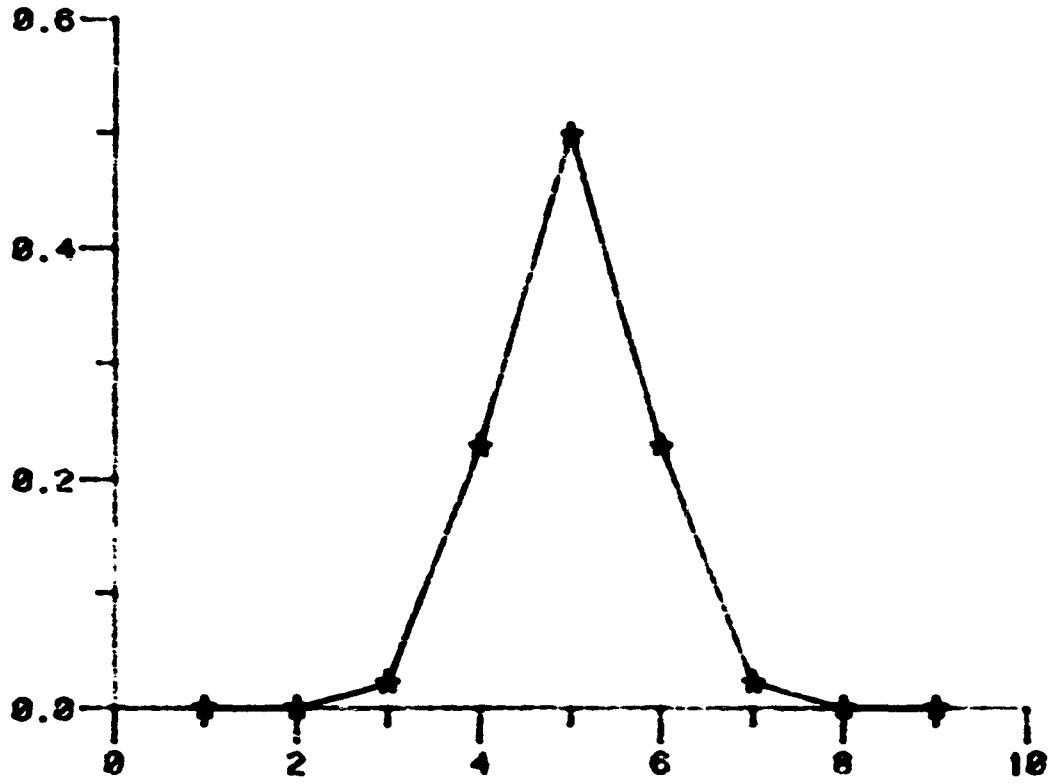


FIGURE 5

1 FOR NARROW GAUSSIAN

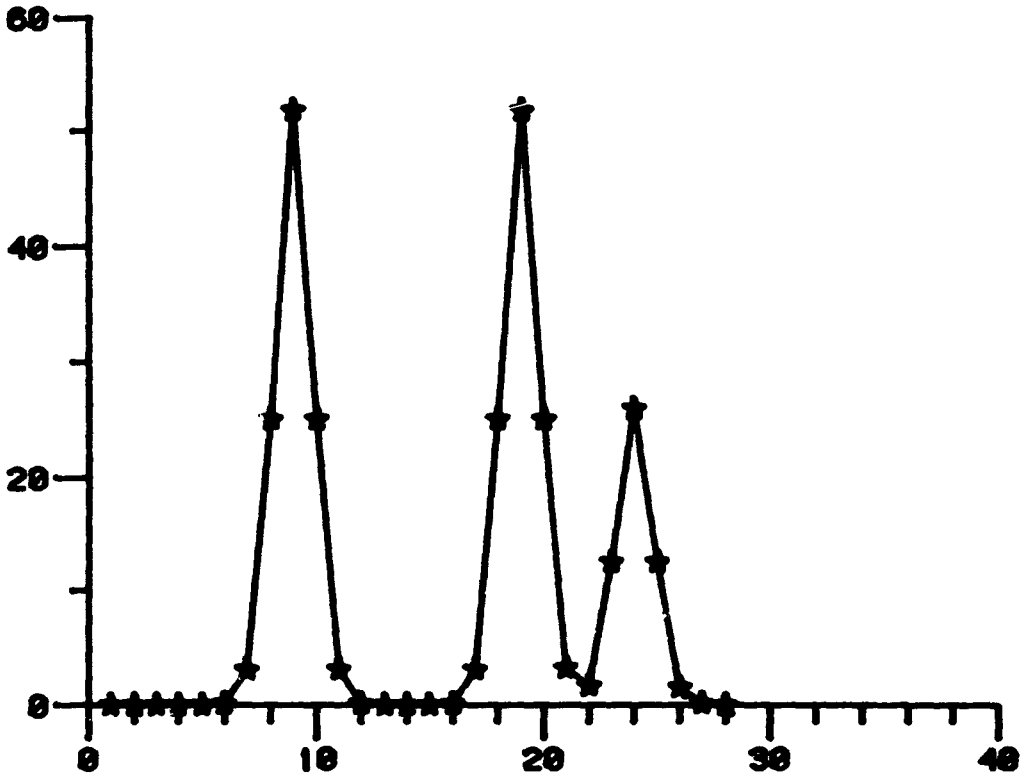


FIGURE 6

G FUNCTION FOR BROAD GAUSSIAN

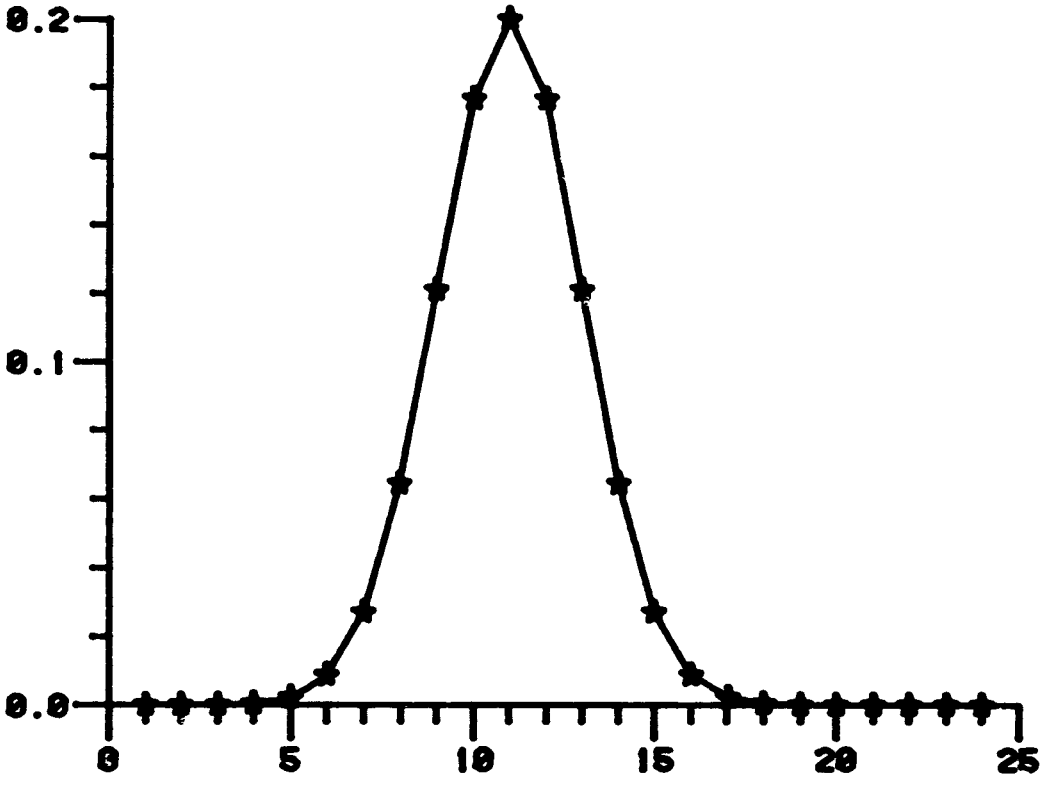


FIGURE 7

H FUNCTION FOR BROAD GAUSSIAN

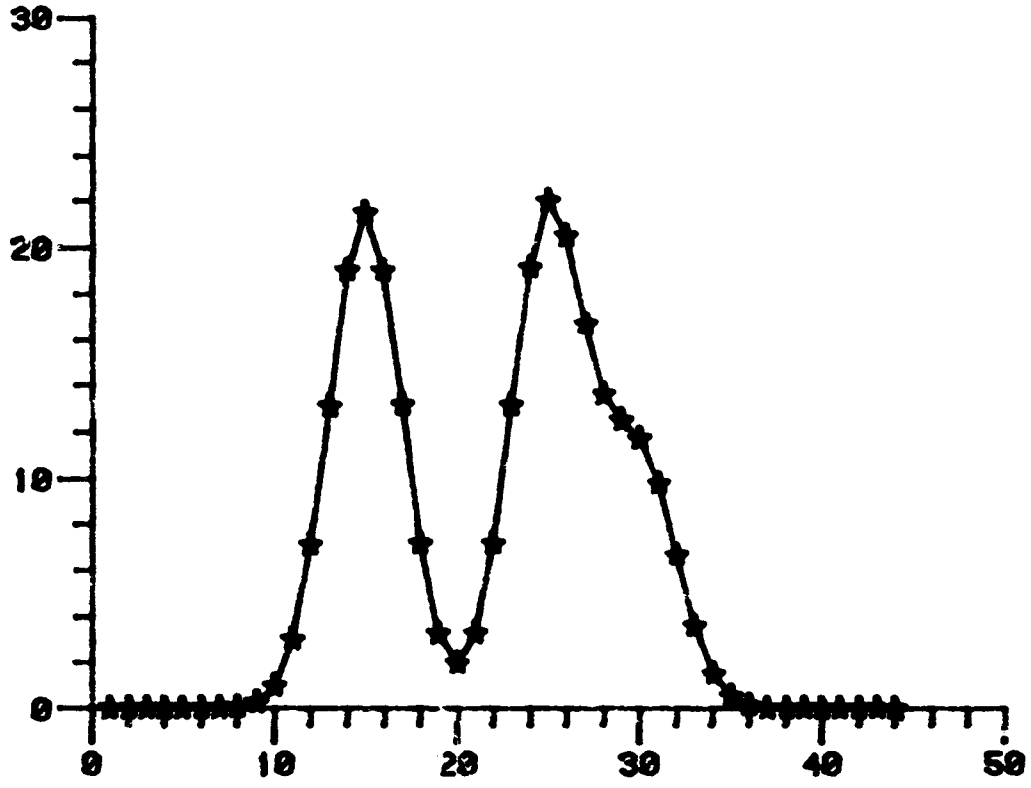


FIGURE 8

DIVERGING G FUNCTION

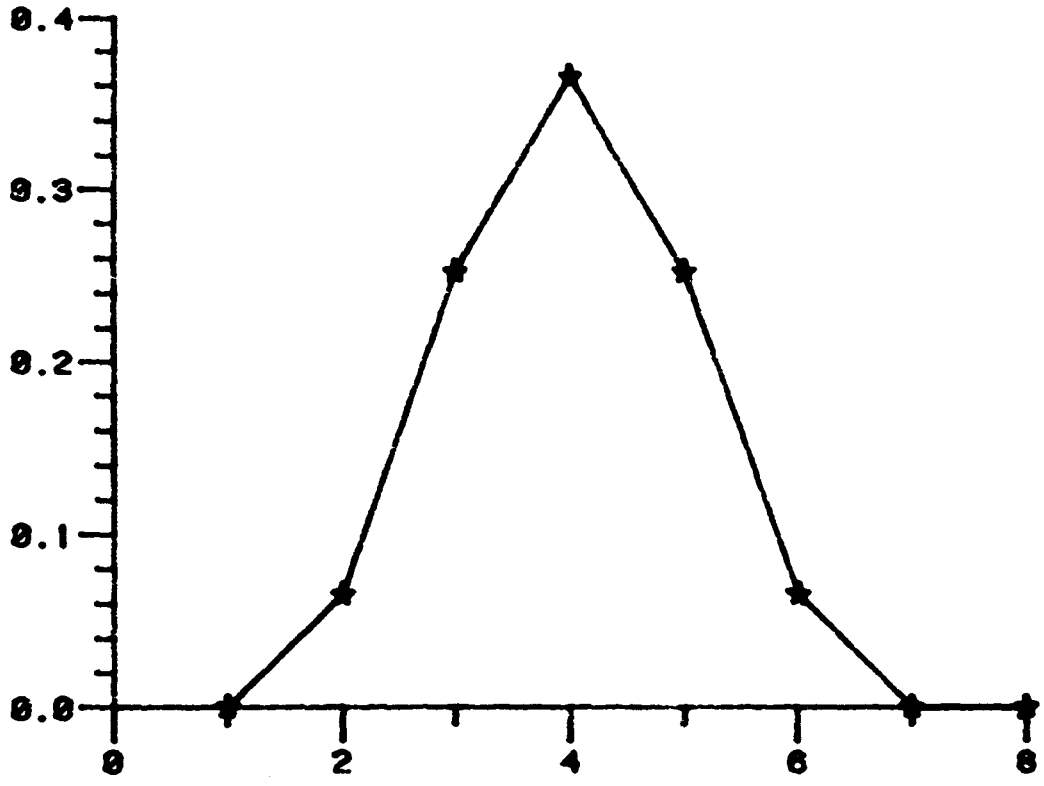


FIGURE 9

H FUNCTION FOR DIVERGING G FUNCTION

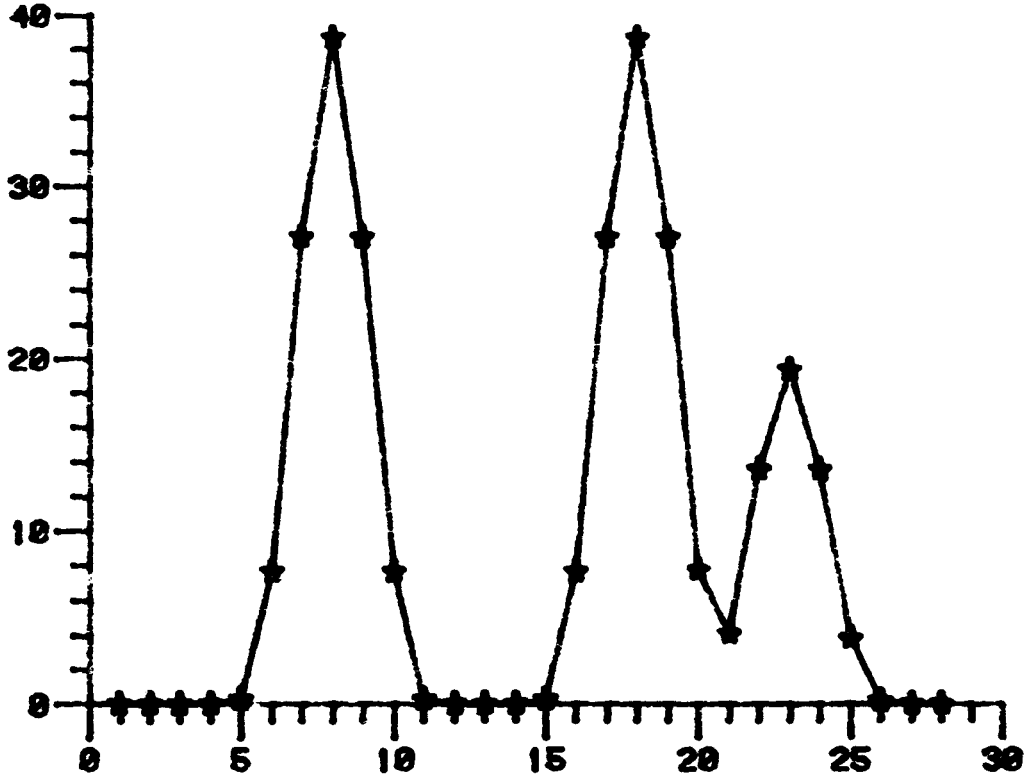


FIGURE 10

ordinate dependent gaussian noise the width of the bell depends upon the ordinate size of the data point in question, and varies as the square root of the ordinate.

To add gaussian noise to the data, the gaussian distribution must be sampled randomly. The technique used was as follows. If $y=e^{\frac{-x^2}{2\sigma^2}}$ then y can only assume values between 0 and 1. By generating a random value for y in this interval, one could solve for the corresponding x, which is then gaussian distributed noise. The full expression for ordinate dependent gaussian noise is

$$n=\sqrt{2\times SF\times h(x)\times(-\log(y))}$$

where n is the noise, SF a scale factor, h(x) the ordinate to which n will be added, and y a random number (necessarily less than 1, therefore log(y) is negative, preserving the overall positive character of the square root). Constant gaussian noise differs only by omission of the h(x) factor. After the magnitude, n, of the noise has been calculated, a separate (random) decision was made whether to add or subtract the noise from the datum. Sometimes (especially for data with small values) this would result in an overall negative value. In all such cases, desiring to keep the data positive, the sign of the difference was changed. When the data were essentially zero, small purely random noise was added, in imitation of "white" background noise.

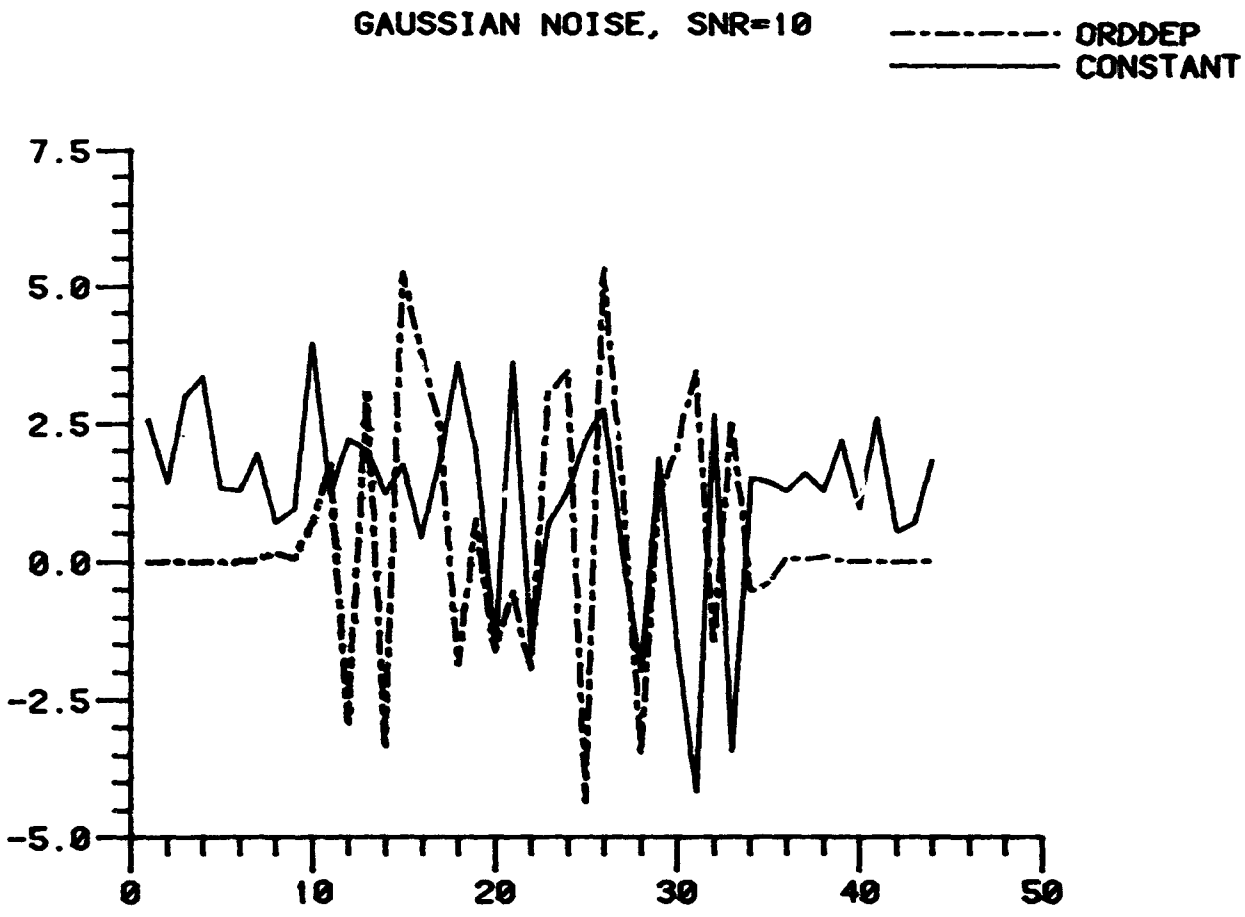

GAUSSIAN NOISE, SNR=10
ORDDEP
CONSTANT
7.5
5.0
2.5
0.0
-2.5
-5.0
0
10
20
30
40
50

FIGURE 11

H FOR SNR=5, NARROW GAUSSIAN

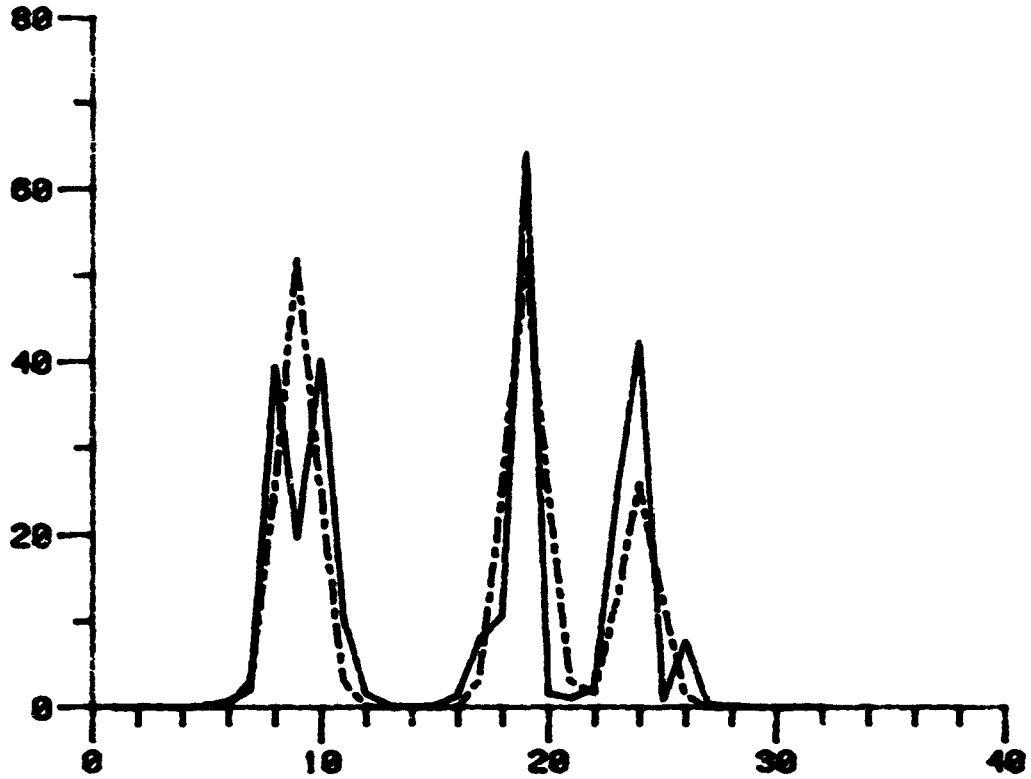


FIGURE 12

H FOR SNR=25, NARROW GAUSSIAN

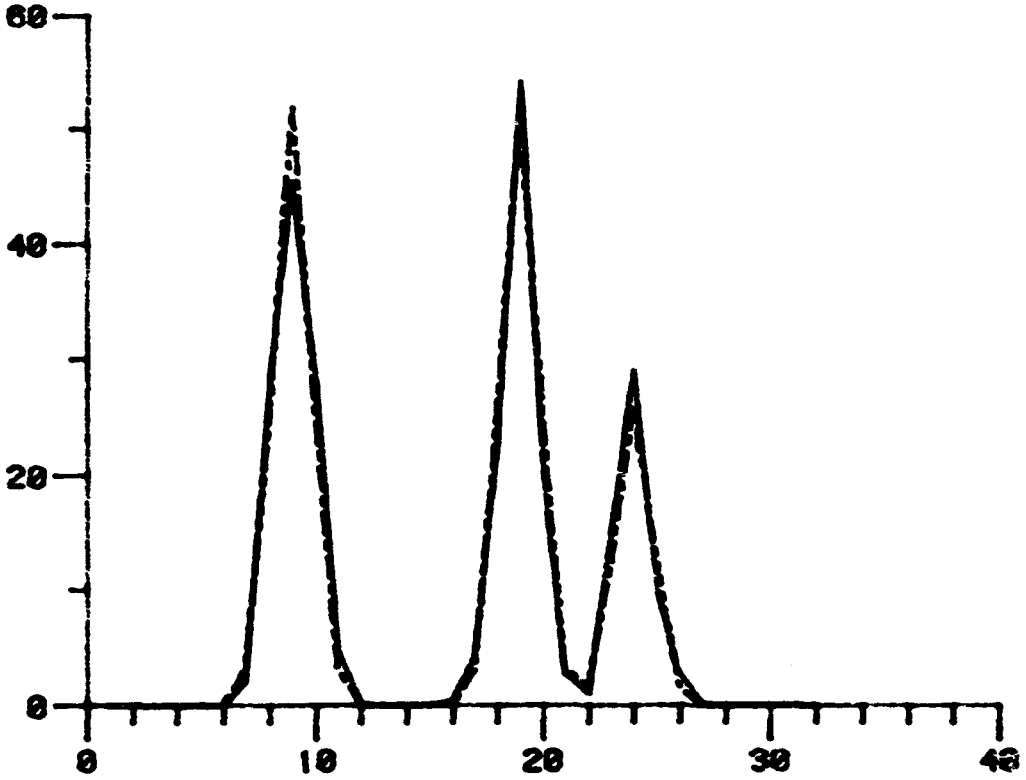


FIGURE 13

H FOR SNR=25, CONSTANT NOISE

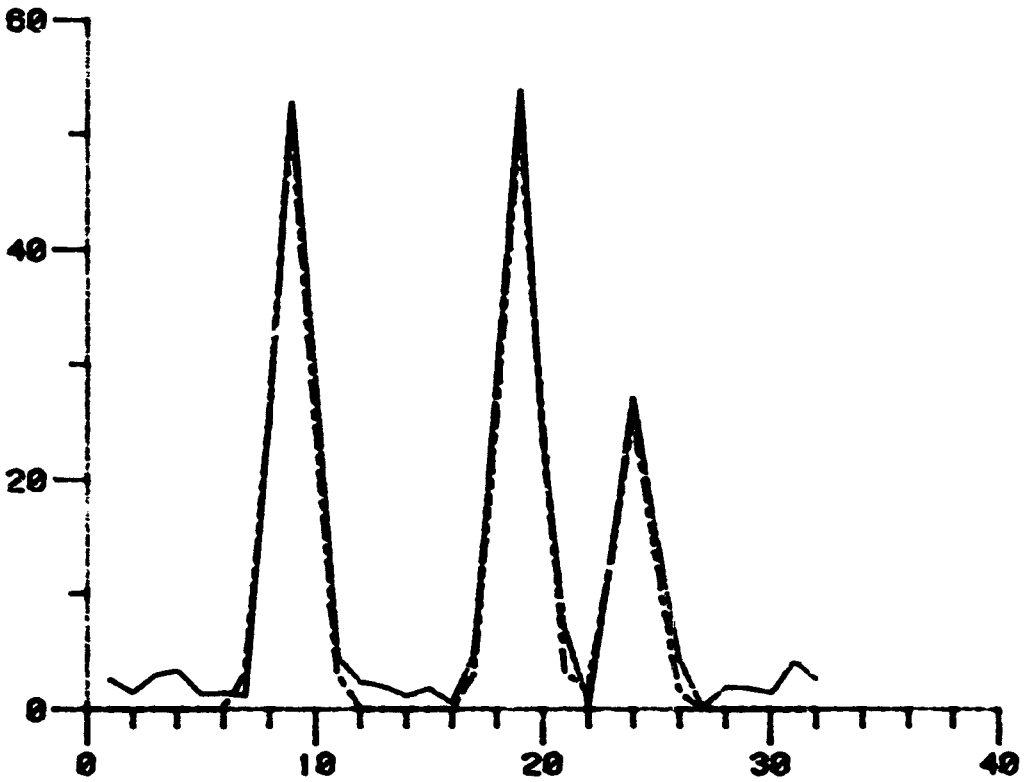


FIGURE 14

H FOR SNR=25, BROAD GAUSSIAN

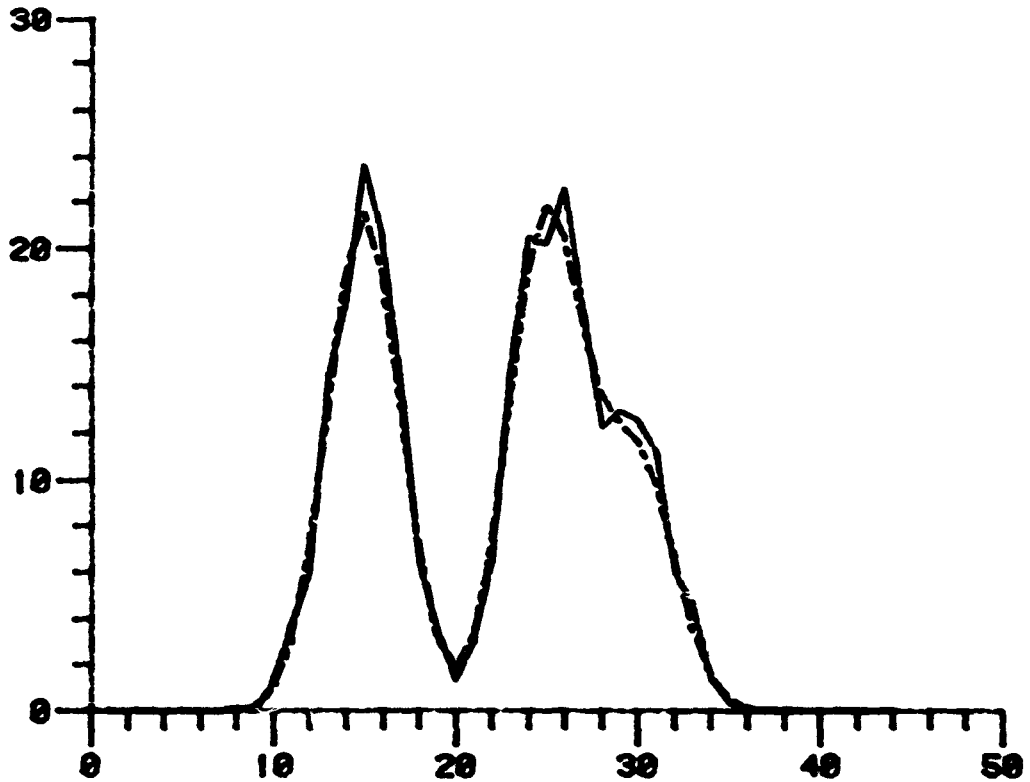


FIGURE 15

Several measures were used to characterize the noise added. The first, and most crude, was the size of the scaling factor used. The second was the root mean square deviation between the noisy and the noise free h (hereafter referred to as the RMS). The third was the signal to noise ratio (SNR) which was the maximum ordinate value of each h function divided by the corresponding RMS.

SNRs ranging from 1.0 to 2600 were used. Figure 11 illustrates how ordinate dependent gaussian noise can differ from constant gaussian noise even when both have the same RMS. Figures 12-15 show how noisy data(solid line) differs from the noise free data (dashed line) at various SNRs,h functions and noise types. Unless otherwise noted, the noise imposed is ordinate dependent.

In an actual experiment the noise free h function would not be known. In this case a reasonable test for convergence of the process would be to compare in some fashion each value of the current iteration with the corresponding value in the previous iteration. Initially, four measures of convergence were used. they were as follows:

$$CON(1) = \left\{ \sum_{x=1}^{N} \frac{(h_n(x) - h_{n-1}(x))^2}{h(x)} \right\} / N$$

$$CON(2) = \left\{ \sum_{x=1}^{N} \frac{|h_n(x) - h_{n-1}(x)|}{h(x)} \right\} / N$$

$$CON(3) = \left\{ \sum_{x=1}^{N} \left(h_n(x) - h_{n-1}(x) \right)^2 \right\} / N$$

$$CON(4) = \left\{ \sum_{x=1}^{N} \left| h_n(x) - h_{n-1}(x) \right| \right\} / N$$

where $h_n(x)$ is the current iteration, $h_{n-1}(x)$ the previous iteration, and h(x) the original noise free h function. At convergence one should find the differences between iterations becoming vanishingly small.

To characterize the noise level for each iteration it was necessary to compare each iteration with the noise free h. Three measures were used here, as follows. (Note that $h_n(x)$ is the current iteration, and h(x) is the noise free h).

$$ER(1) = \left\{ \sum_{x=1}^{N} \left| h(x) - h_n(x) \right| \right\} / N$$

$$ER(2) = \left\{ \sum_{x=1}^{N} \left(h(x) - h_n(x) \right)^2 \right\} / N$$

$$ER(3) = \sqrt{ER(2)}$$

ER(1) tended to weight all ordinates equally, while ER(2) and ER(3) tended to emphasize the larger ordinates (with larger possible differences) by squaring all terms. ER(1) can be characterized as the absolute difference, while ER(2) is the variance between the noise free h and the nth iteration, and ER(3) is the corresponding standard deviation. Note that the RMS of the added noise is the same

as the standard deviation between the noise free h and the initial noisy h.

The ordinate dependent noise case was examined first. By adjusting the noise scale factor input to the program, SNR values of h in the range of 1 to 2000 were obtained for each g function. (See Appendix 2 for discussion of MORRIS.FOR.) Convergence measures versus iteration and error measures versus iteration were plotted for each configuration. See Plots 48-74 and 1-27. Initially 100 iterations were used for each g type,but by examining the convergence data it was determined that the broad gaussian (with slow convergence) had not yet converged. The broad gaussian runs were repeated for 200 iterations and convergence was achieved. Unfortunately, the convergence plots seemed to yield no further significant information, so in the constant noise case none were plotted.

Next the constant gaussian noise case was examined. Here SNRs were varied from 1 to 1000. The narrow (fast) g and the diverging g functions were run for 100 iterations, while the broad (slow) g runs went 200 iterations. Error measure versus iteration plots were produced for the combinations of interest. See Plots 28-47.

CHAPTER 3

The data of greatest interest proved to be the plots of error measures versus iteration. (See Plots 1-47). Each error measure related the current iteration h values to the corresponding values of the noise free h. When any of those curves exhibited a minimum the noise induced error was in some sense minimized.

The error curves fell into three broad categories. Characteristic of the first category is Plot 1. Characteristic of the second category is Plot 3, while the third category follows the pattern of Plot 7. See Figure 16.

Category 1 curves all exhibited an initial minimum, then quickly rose to a constant level. This pattern occurred among low SNR runs. Morrison's method is a dual process of smoothing, then restoration. For low SNR runs (with high noise levels), the maximum improvement in the data was obtained by the initial smoothing iteration. Subsequent restoring iterations quickly restored the noise that had been smoothed out (along with some detail).

Category 2 curves had a local minimum then rose to a constant level. For this SNR range, the initial smoothing aided the data, as did the restoring iterations, up to the location of the minimum. Following that point, further

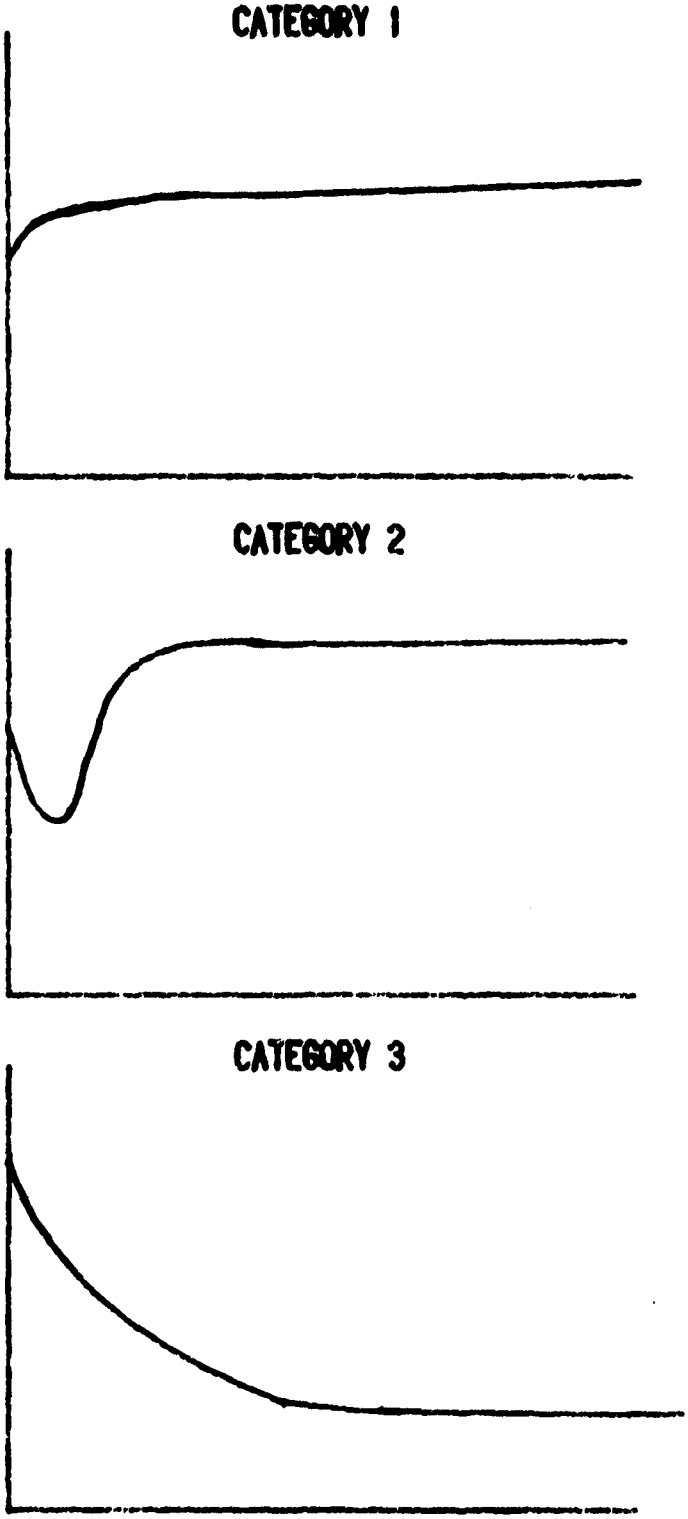


FIGURE 16

restorations continue to increase the sharpness, but noise was restored faster than the function.

Category 3 curves declined monotonically, so that the last iteration had the least error value. The curves tended to become asymptotic to a constant value. This behavior was frequent for high SNR data. In this case the noise level was so low that Morrison's method was ineffective in removing noise, but by running to convergence all sharpness was restored, so the data were not degraded. The previous statement regarding noise removal does not hold for the broad gaussian, as will be discussed subsequently.

The error curves for fast g with ordinate dependent noise all fell into these 3 categories. Category 2 (local minima) occurred for SNR=10 to 100.

Slow g with ordinate dependent noise exhibited each kind of behavior, with local minima for SNR=5 to 15.

Diverging g with ordinate dependent noise never exhibited Category 3 (flat curve) behavior, because each iteration caused the h function to diverge. The break between categories 1 and 2 was at SNR=10 to 25.

Slow g with constant noise had no perceptible SNR range wherein local minima occurred. The transition between Category 1 and Category 3 curves occurred at SNR=5. Indeed at SNR=5 all error measures were quite flat for the full range of 200 iterations. There must have occurred

(accidentally) a nice balance between restoration of detail and noise reduction, such that no one level of each was preferable to any other.

Diverging g with constant noise plots all followed categories 1 and 2, as in the ordinate dependent case, with the break at SNR=10.

It was desirable to characterize the ability of Morrison's method to remove noise. Towards this end the minimum value of ER(3), the noise RMS, was compared to the initial RMS for each combination. Table 2 contains the results. The entries are the percent improvement of the minimum RMS over the initial RMS. A negative entry corresponds to a worsening of the data.

Several features of Table 2 are noteworthy. The figures quoted are subject to some error range, since they arise from randomly generated noise. Were a different random seed to be used the results could be slightly altered. For each noise type at high SNR values the fast g had no improvement (but no worsening) of the data. This indicates that the process converged entirely back to the noisy h function. This behavior is consistent with the consideration of the transform of the fast g. The transform is broad, so there is little range for incompatible noise. All noise is compatible, and all noise is restored.

Table 2

Percent improvement of Minimum RMS over Initial RMS

Ordinate Dependent Noise

	Fast		Slow		Divergent	
SNR	%	Iter.	%	Iter.	%	Iter.
1	6.3	1	23	1	29	1
5	38	1	67	1	45	2
10	27	3	63	3	42	5
25	14	7	50	34	36	9
50	6.9	12	49	55	57	19
100	1.8	20	48	83	-140	29
500	0.0	100	40	200	-220	41
1000	0.0	100	31	200	-313	41

Constant Noise

	Fast		Slow		Divergent	
SNR	%	Iter.	%	Iter.	%	Iter.
1	8.6	1	14	1	11	1
5	6.8	1	20	3	14	3
10	3.6	3	24	25	13	6
25	1.6	11	25	55	11	15
50	.36	22	26	79	9.7	24
100	0.0	42	26	99	6.9	31
500	0.0	100	24	200	-82	41
1000	0.0	100	16	200	-185	41

In contrast, all observed cases of the slow g resulted in significant improvement of the RMS level. Even those cases where the error measures became asymptotic (indicating convergence) showed improvement, so the converged h differed substantially from the original noisy h. Viewed in the transform domain it became apparent that the narrowness of the slow g's transform allowed for appreciable incompatible noise. This noise was not restored at convergence, and accounted for the difference between the two h values. The diverging g case was not clear cut, except that in the SNR value range of 1 to 50 this diverging g function could be applied for a net improvement of the RMS level. Above SNR=100 use of this diverging g in Morrison's method caused rapid deterioration of the RMS level. It should be noted that these results are probably highly dependent upon the rate at which a g function diverges.

Within each g category there is a tendancy for Morrison's method to cause more improvement in the ordinate dependent noise case, than in the corresponding constant noise case. This implies that the noise added in the ordinate dependent case will have a larger high frequency and a smaller low frequency content than the constant noise. (See Figure 11.) (High frequencies are more likely to be incompatible and so not to be restored.) This frequency distribution is the result of several experimental parameters chosen when deciding how to shape the experiment.

First, the choice of f as three narrow gaussians means that the large ordinates (with correspondingly large noise) permits high frequencies, even after h is produced by convolving f with g. Second, and related to the first point, data were not allowed to become negative. When noise subtraction resulted in a negative signal the sign was switched, rendering it positive. In this manner sharp changes across the dc level were rendered less sharp (and so had a larger low frequency content). This second point affects mostly the constant gaussian noise, since the ordinate dependent noise near the dc level was constrained to be small. It is quite possible that were the previously mentioned parameters to be changed, Morrison's method would no longer work better for ordinate dependent noise than for constant.

Having established that Morrison's method results in noise removal for certain SNR ranges for each choice of g function and noise type, it would be desirable to establish the number of iterations necessary to achieve the optimum result. Towards this end the minimum for each error measure was tabulated as a function of iteration number. Since ER(3) is the square root of ER(2), their minima coincide and they were tabulated as one. Figures 17-22 show the results. In each figure C1 is the curve for minimizing the absolute difference and C2 for minimizing the variance or RMS (since RMS=standard deviation=ER(3)).

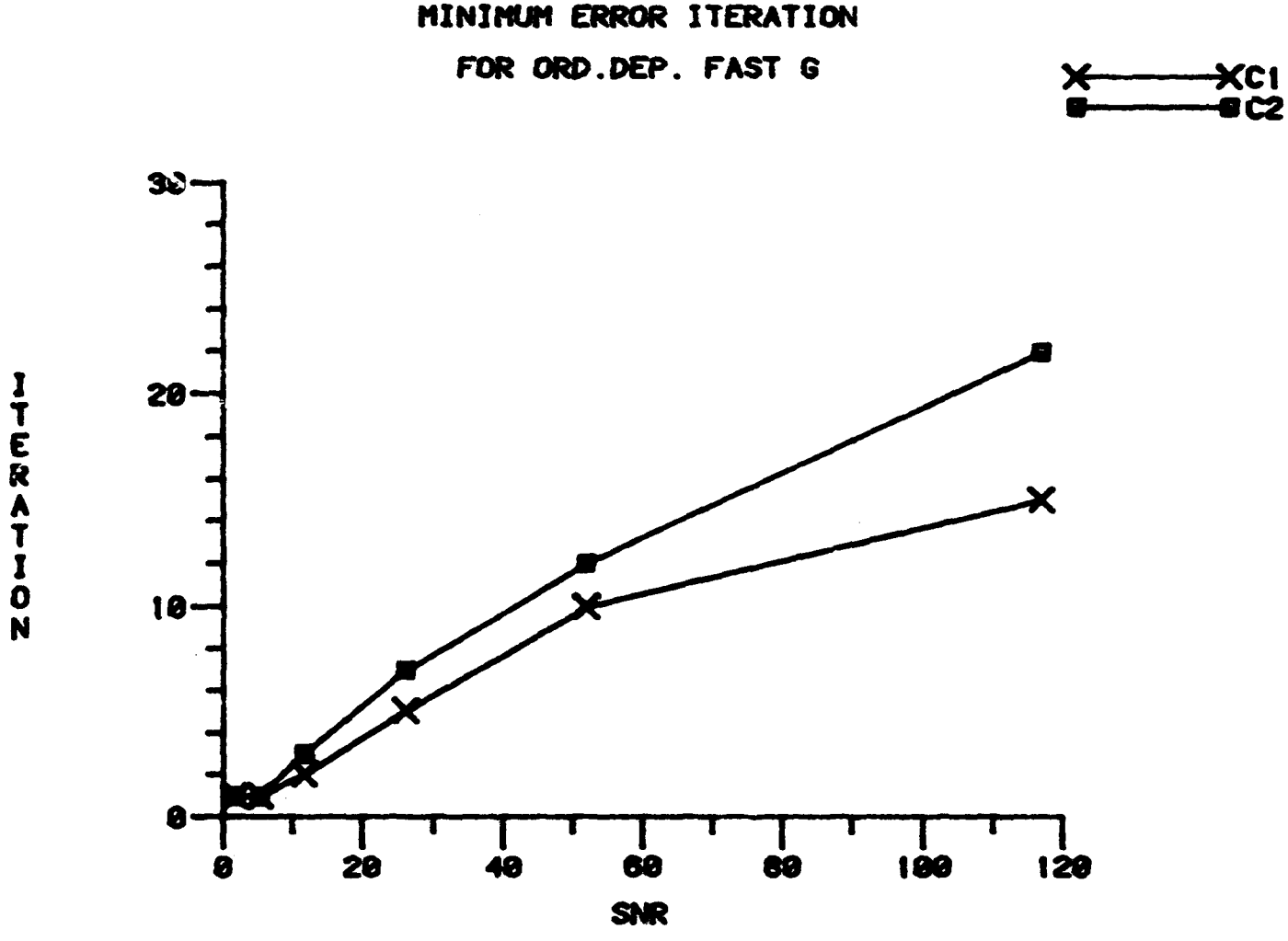

MINIMUM ERROR ITERATION
FOR ORD.DEP. FAST G
C1
C2
ITERATION
30
20
10
0
0
20
40
60
80
100
120
SNR

FIGURE 17

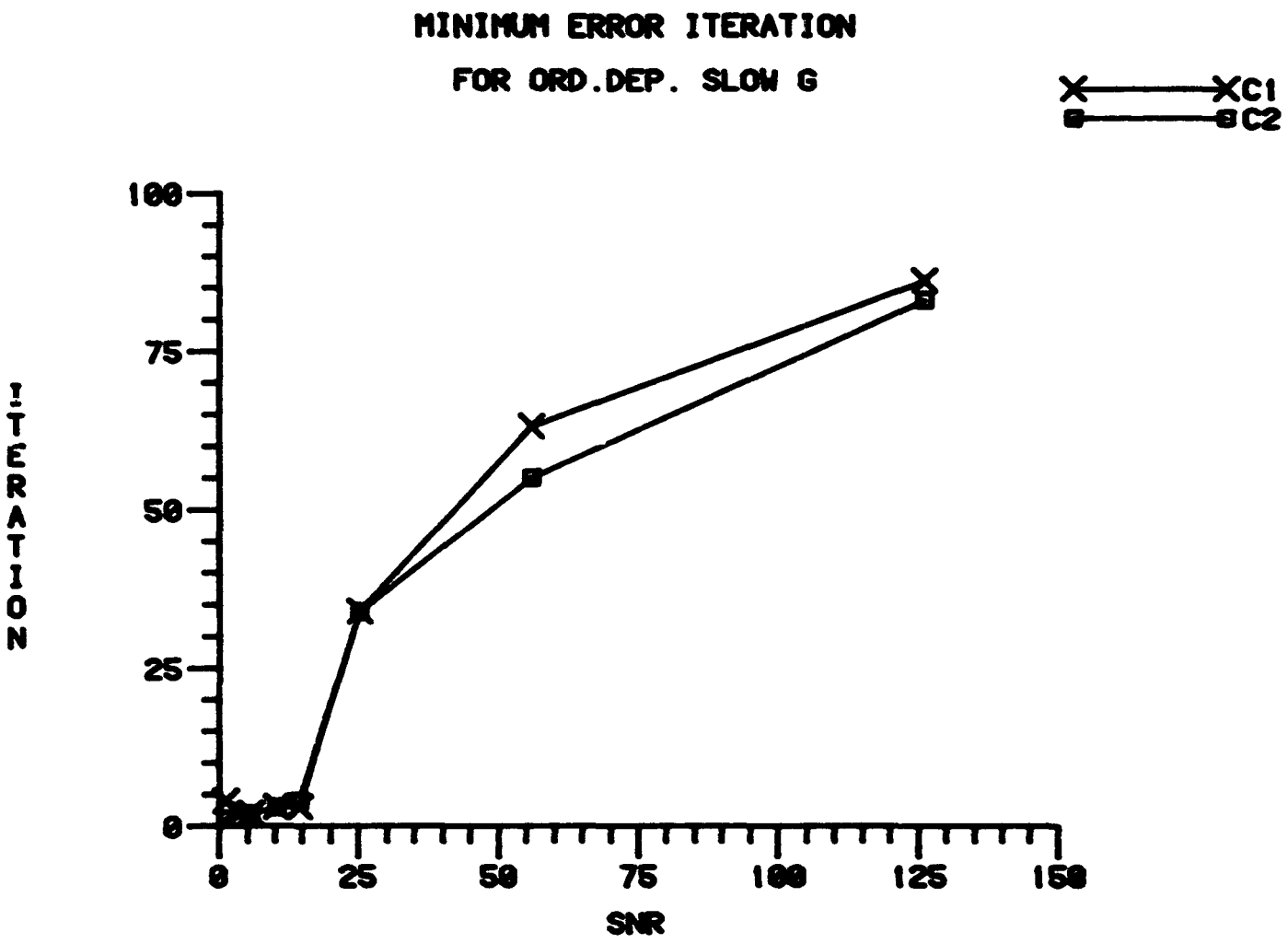

MINIMUM ERROR ITERATION
FOR ORD.DEP. SLOW G
C1
C2
ITERATION
100
75
50
25
0
0
25
50
75
100
125
150
SNR

FIGURE 18

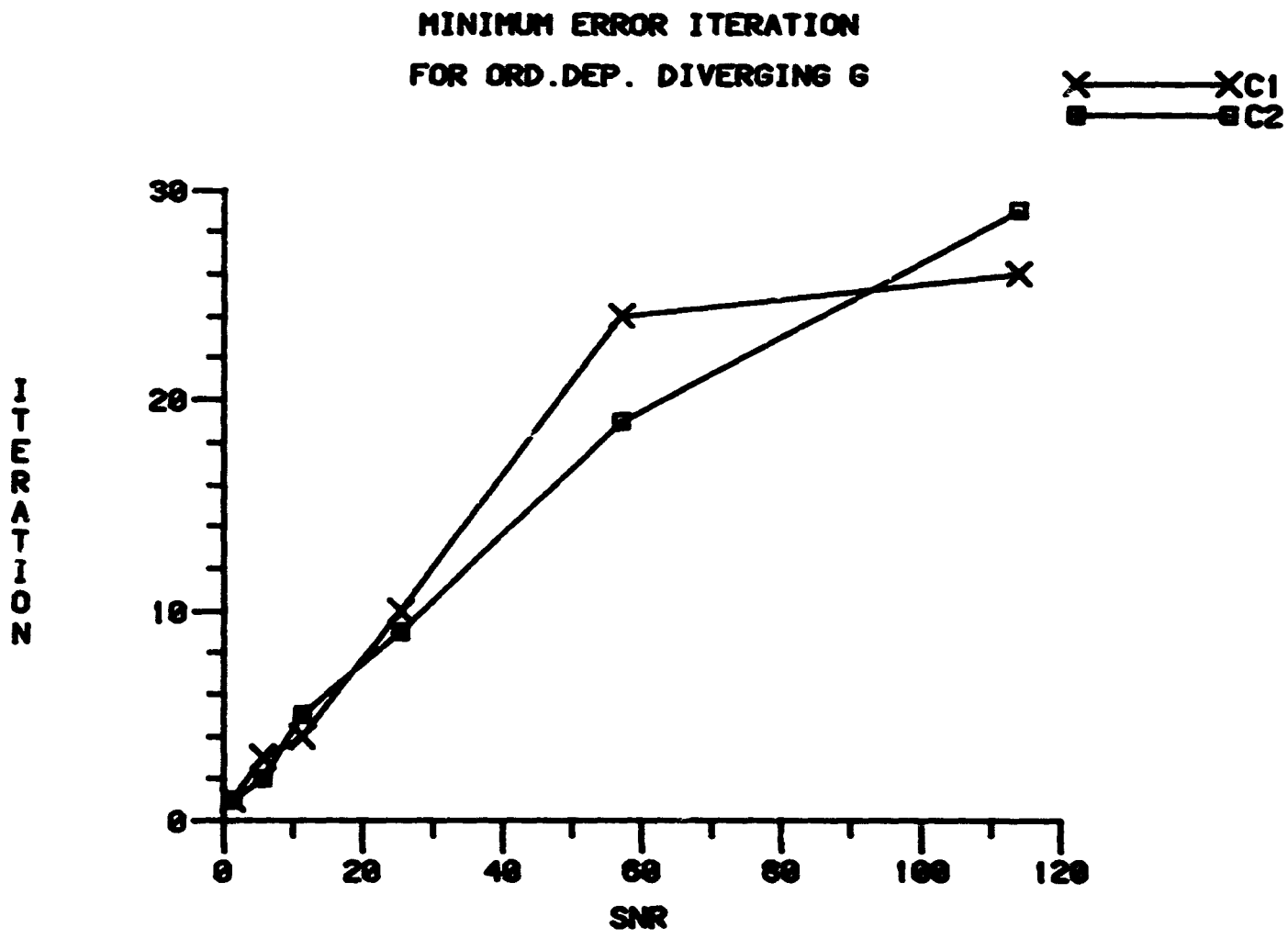

MINIMUM ERROR ITERATION
FOR ORD.DEP. DIVERGING G
C1
C2
ITERATION
SNR
0
10
20
30
0
20
40
60
80
100
120

FIGURE 19

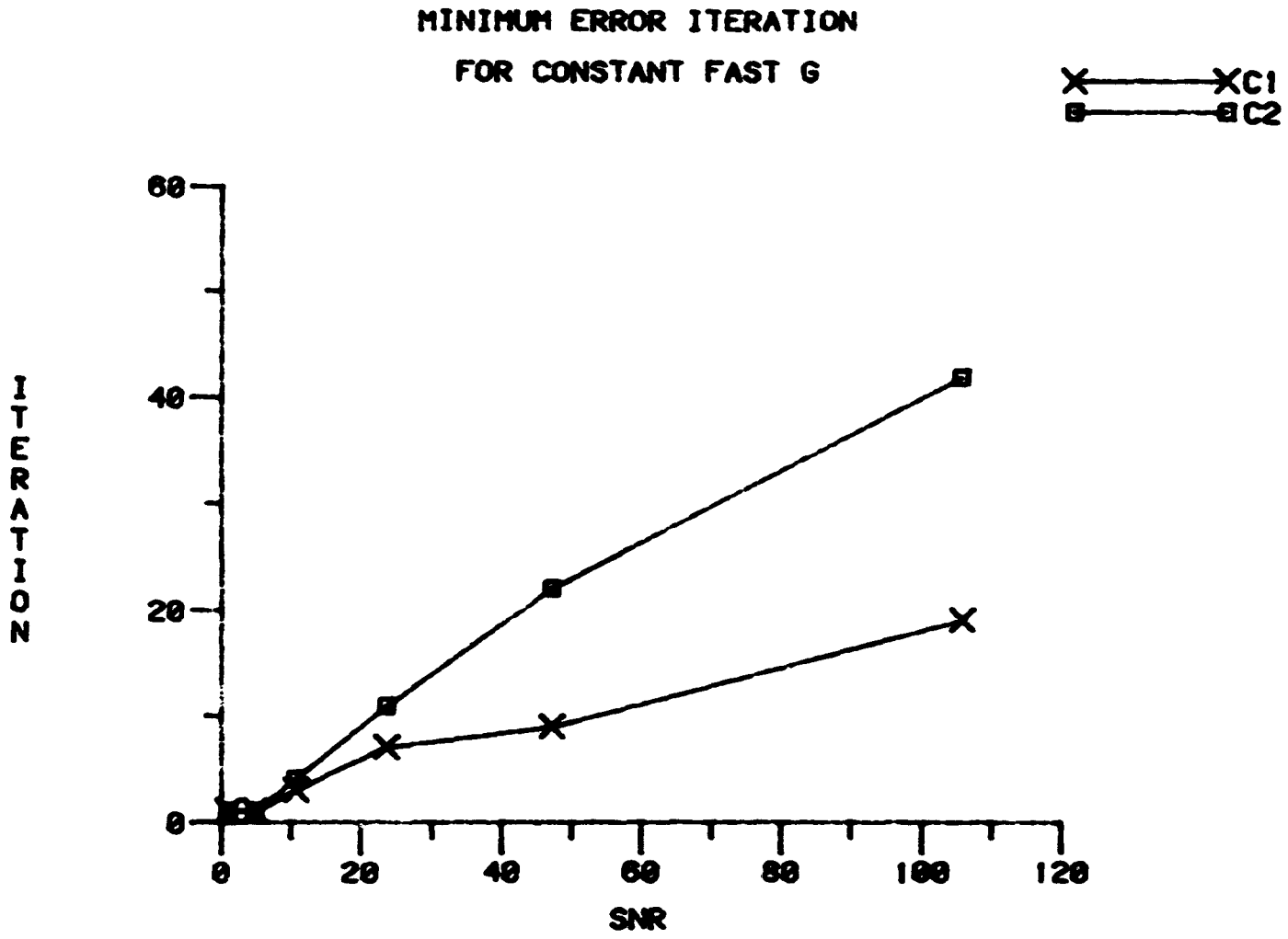

MINIMUM ERROR ITERATION
FOR CONSTANT FAST G
C1
C2
ITERATION
60
40
20
0
0
20
40
60
80
100
120
SNR

FIGURE 20

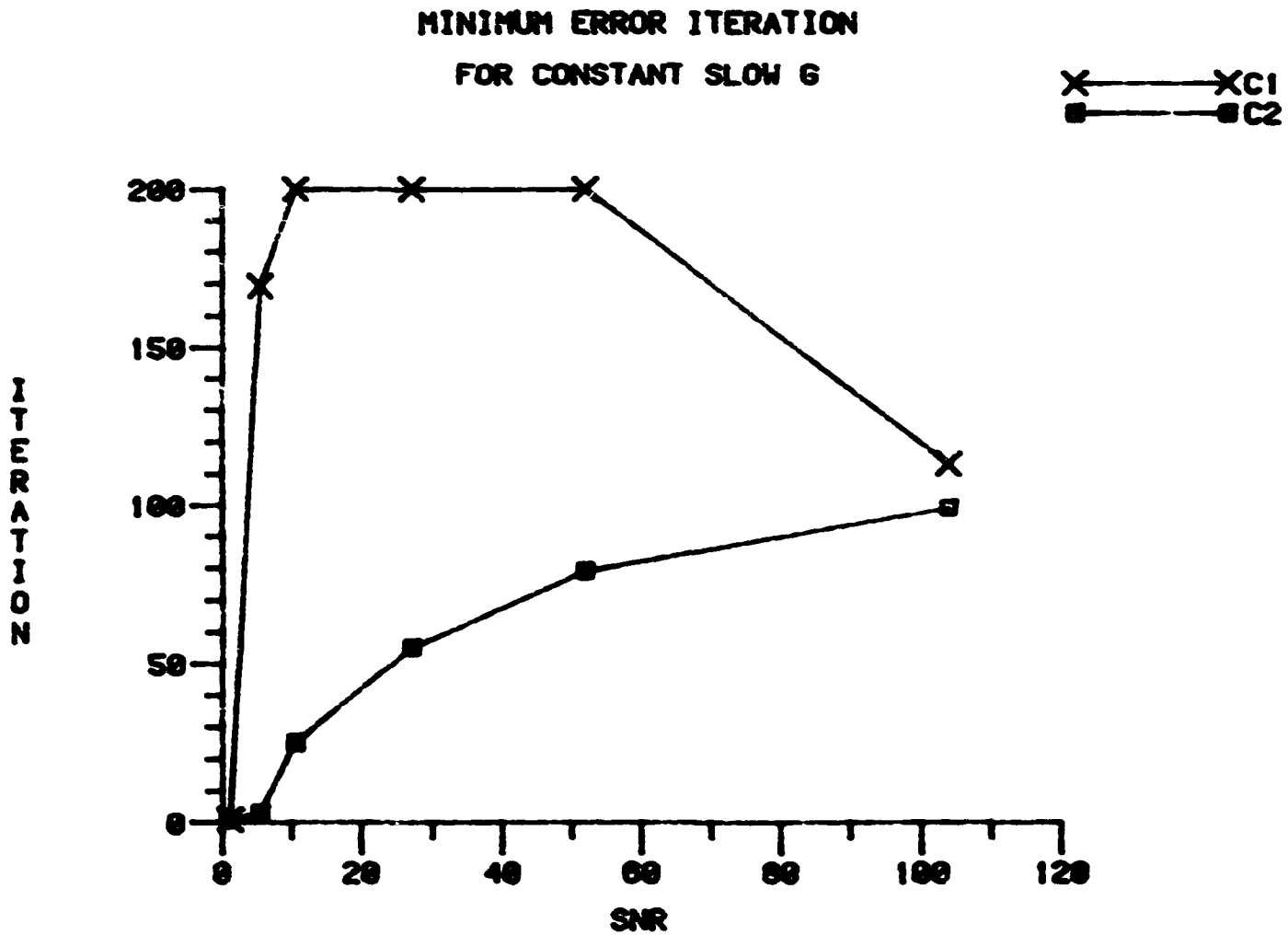

MINIMUM ERROR ITERATION
FOR CONSTANT SLOW G
C1
C2
200
150
100
50
0
ITERATION
0
20
40
60
80
100
120
SNR

FIGURE 21

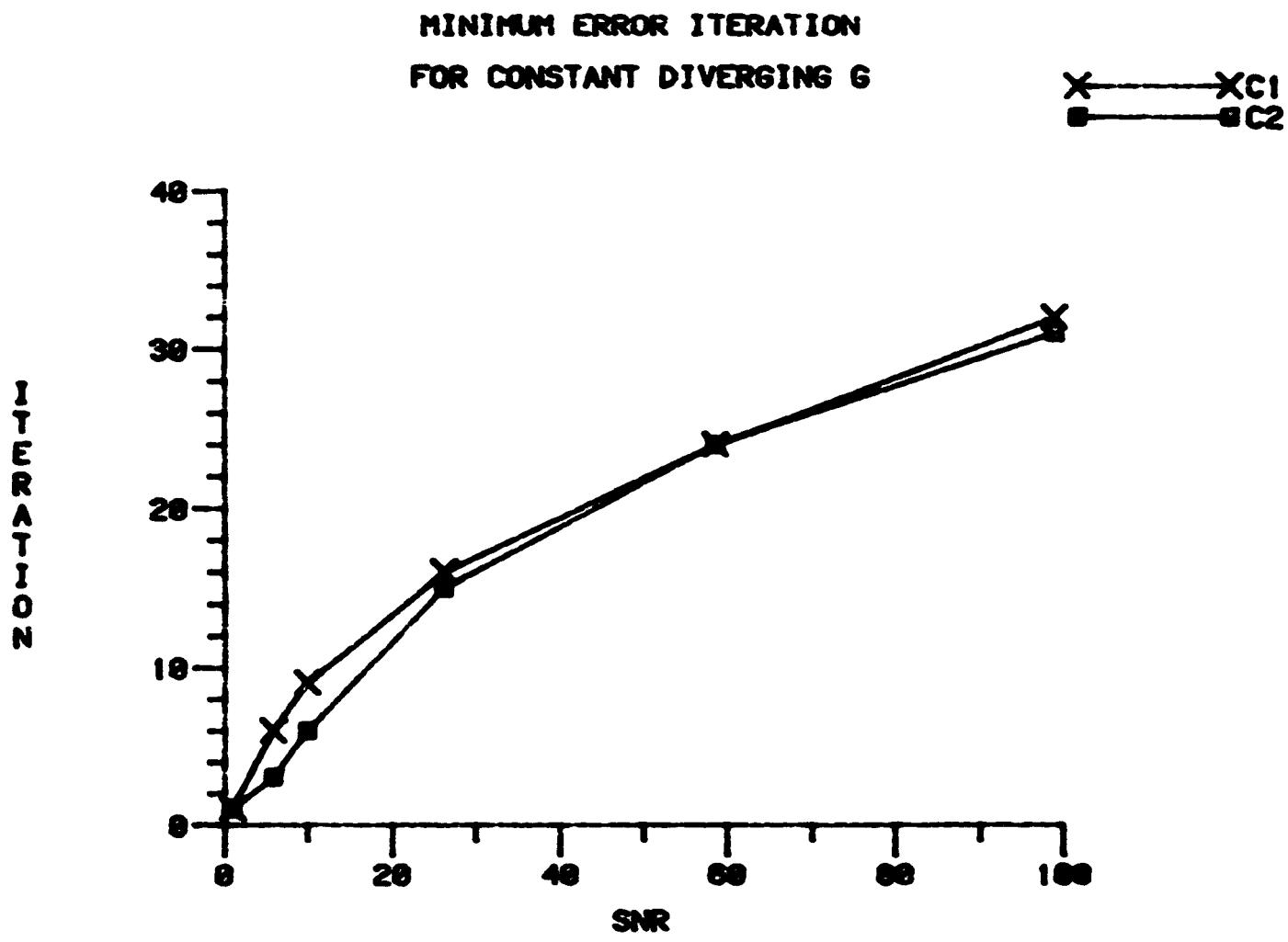

MINIMUM ERROR ITERATION
FOR CONSTANT DIVERGING G
C1
C2
ITERATION
40
30
20
10
0
0
20
40
60
80
100
SNR

FIGURE 22

For ordinate dependent noise C1 and C2 track fairly well. (In Figure 19 the SNR=100 data points can be disregarded since at this SNR level with the diverging g function, Morrison's method no longer helps.)

For constant noise the absolute difference curve became quite erratic, and a poor predictor of iteration number. In Figure 21 the large decline in C1 at SNR=110 is probably an artifact of round off error. The minimum in ER(1) giving rise to that data point occurred in the eighth decimal place of the data.

In all such cases the C2 (variance and RMS) curve was smooth and similar to $y=x^{1/2}$ in shape. For both cases of noise the fast g reaches its optimum RMS value in about half the iterations the slow g requires. This is consistant with the known convergence behavior.

This thesis has demonstrated the ability of Morrison's iterative noise removal technique to reduce noise under certain circumstances. This result is quantified in Table 2. It has also established a relationship between minimum variance and iteration number for certain SNR levels. These relationships are given in Figures 17-22. These results may be used in the following manner.

It is necessary to know the speed of convergence for the g function under consideration. The easiest way to characterize this is by Fourier transforming the g function

and examining its width in the transform domain. On this basis it should be possible to classify (roughly) the g function as "fast", "intermediate", "slow", or "diverging". Then it is necessary to know the SNR of the h function. In this study SNR was calculated as the maximum ordinate divided by the RMS of the noise. If the SNR is not known, it could be determined by taking a statistical number of measurements at the peak, and at a low point of the h function. From these measurements one could determine a mean value for the maximum ordinate, the RMS of the noise, the corresponding SNR, and whether the noise is constant or ordinate dependent.

Knowing the characteristics of the g function and the RMS one may consult the appropriate graph in Figures 17-22. If the g function in question has been classified as intermediate an average of the values specified for the number of iterations for the fast and the slow cases is recommended. If the SNR of the data is not in the range plotted, Morrison Smoothing will not help the data, except for slow g, where beyond this range Morrison's method is still beneficial when run to convergence. By finding the iteration number corresponding to the SNR in question for curve C2 one may determine how many iterations of Morrison's method to use to minimize the variance between the actual h and a noise free h. It is not recommended to use C1 to determine the appropriate number of iterations, as this measure exhibited some disturbing inconsistencies. It is

not really necessary to classify the noise of the experiment in question as ordinate dependent gaussian or as constant gaussian, as in most cases the C2 curves were remarkably similar for both noise types. The fast g results were the only ones to show significant variation between the noise types in minimum error iteration number.

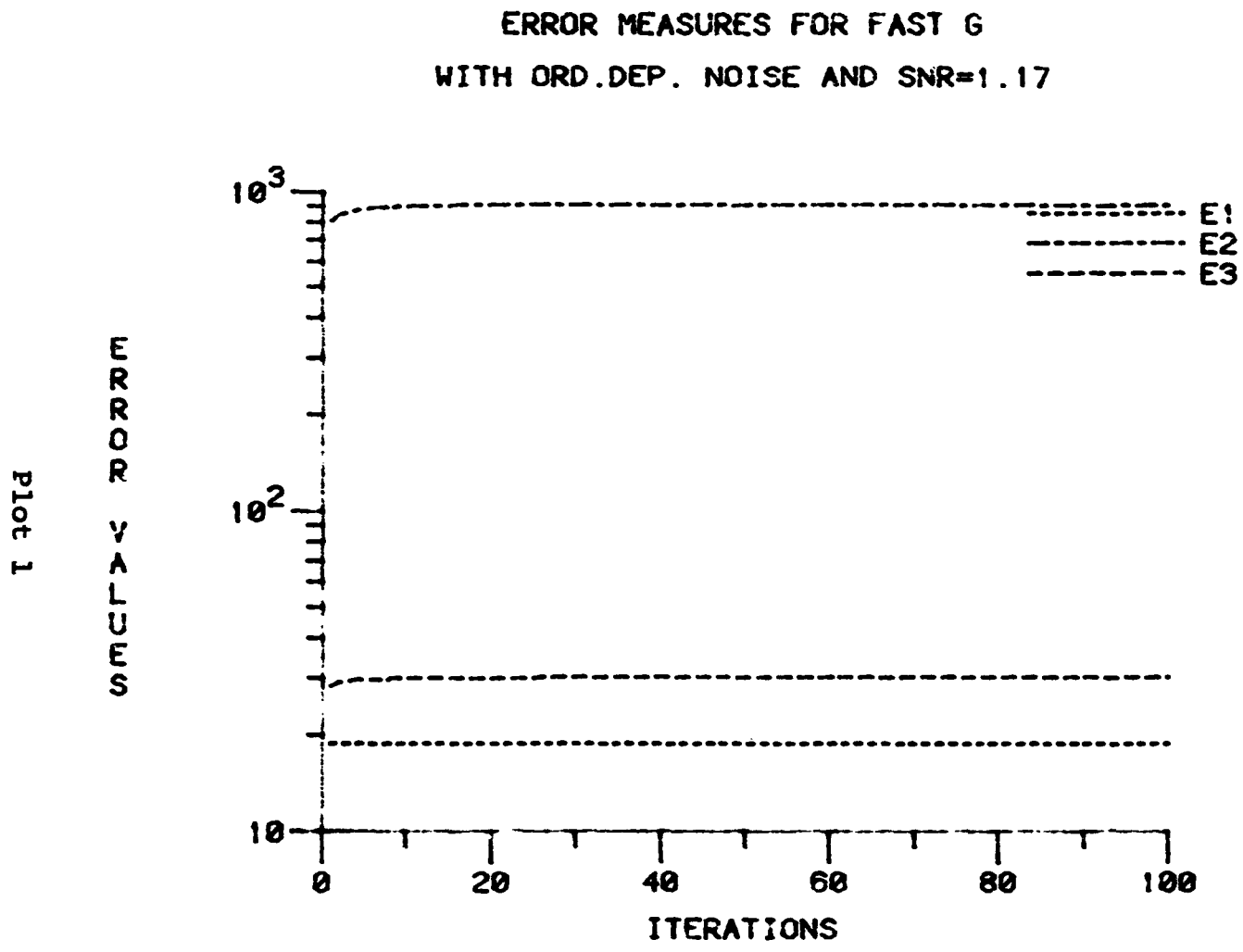
There are several interesting ways in which the scope of this work might be expanded. A few of the possibilities are as follows.

One rationale for Morrison Smoothing is the benefits obtained when the function is ultimately deconvolved. It would be interesting to compare deconvolutions of the same h function when it has been subjected first to the minimum error number of iterations of Morrison's method and second, to a significantly different number of iterations. It is hoped that the first case would result in a better deconvolution.

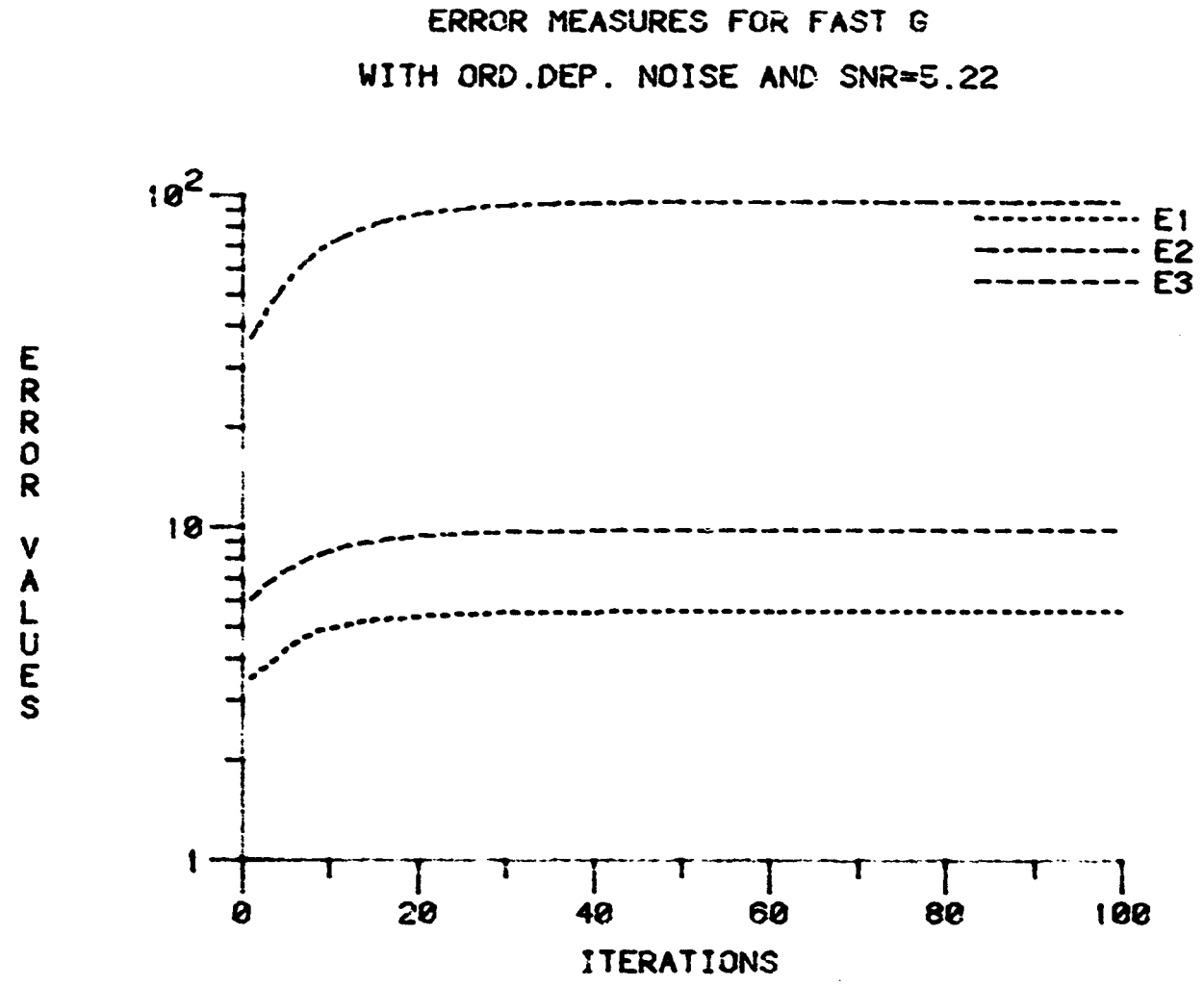
It would be enlightening to repeat this study after changing the seed for the generation of random noise. This would help establish the consistency and the fluctuations of the results.

The noise cases studied, ordinate dependent and constant represent extremes in the types of noise occurring in data. It would be interesting to study the effects of Morrison Smoothing on h functions with combinations of each

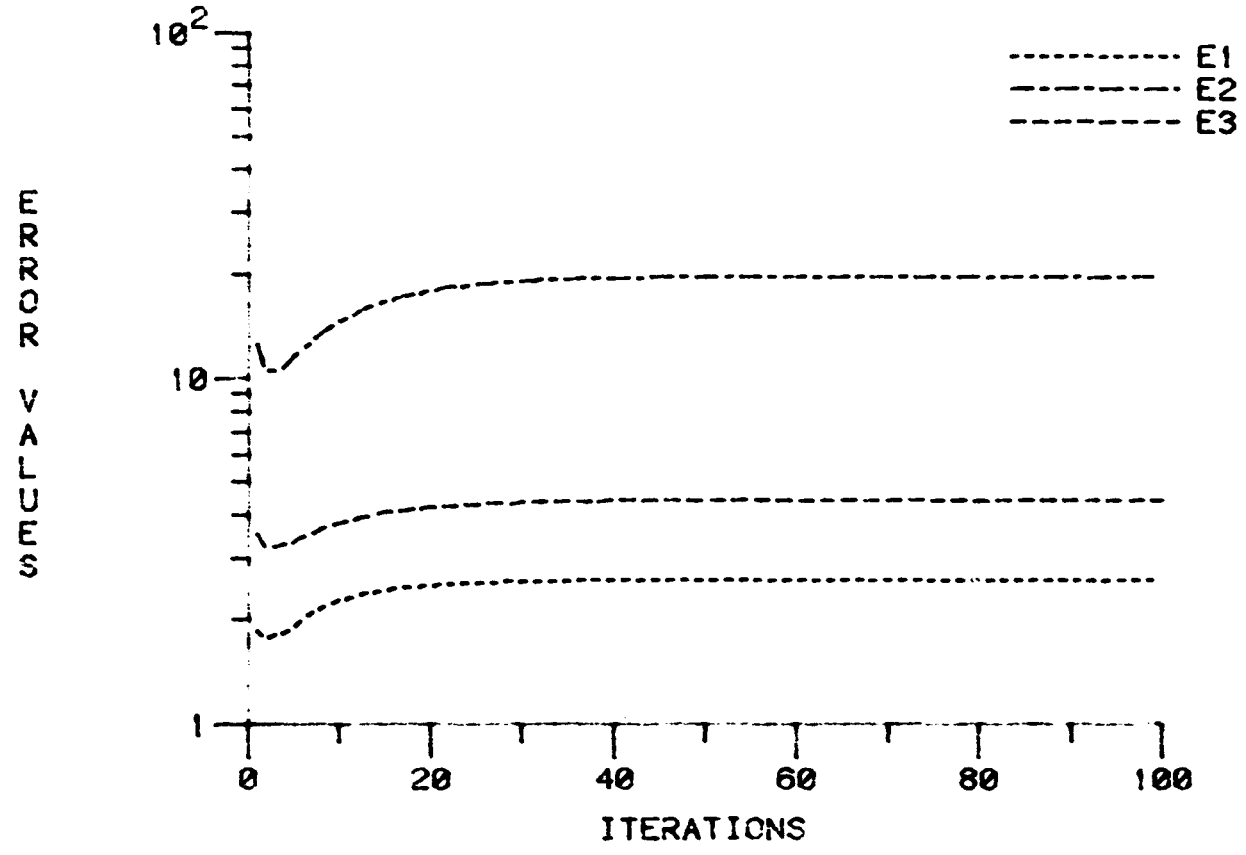
type of noise.


ERROR MEASURES FOR FAST G
WITH ORD.DEP. NOISE AND SNR=1.17
10^3
10^2
10
E1
E2
E3
ERROR VALUES
0
20
40
60
80
100
ITERATIONS

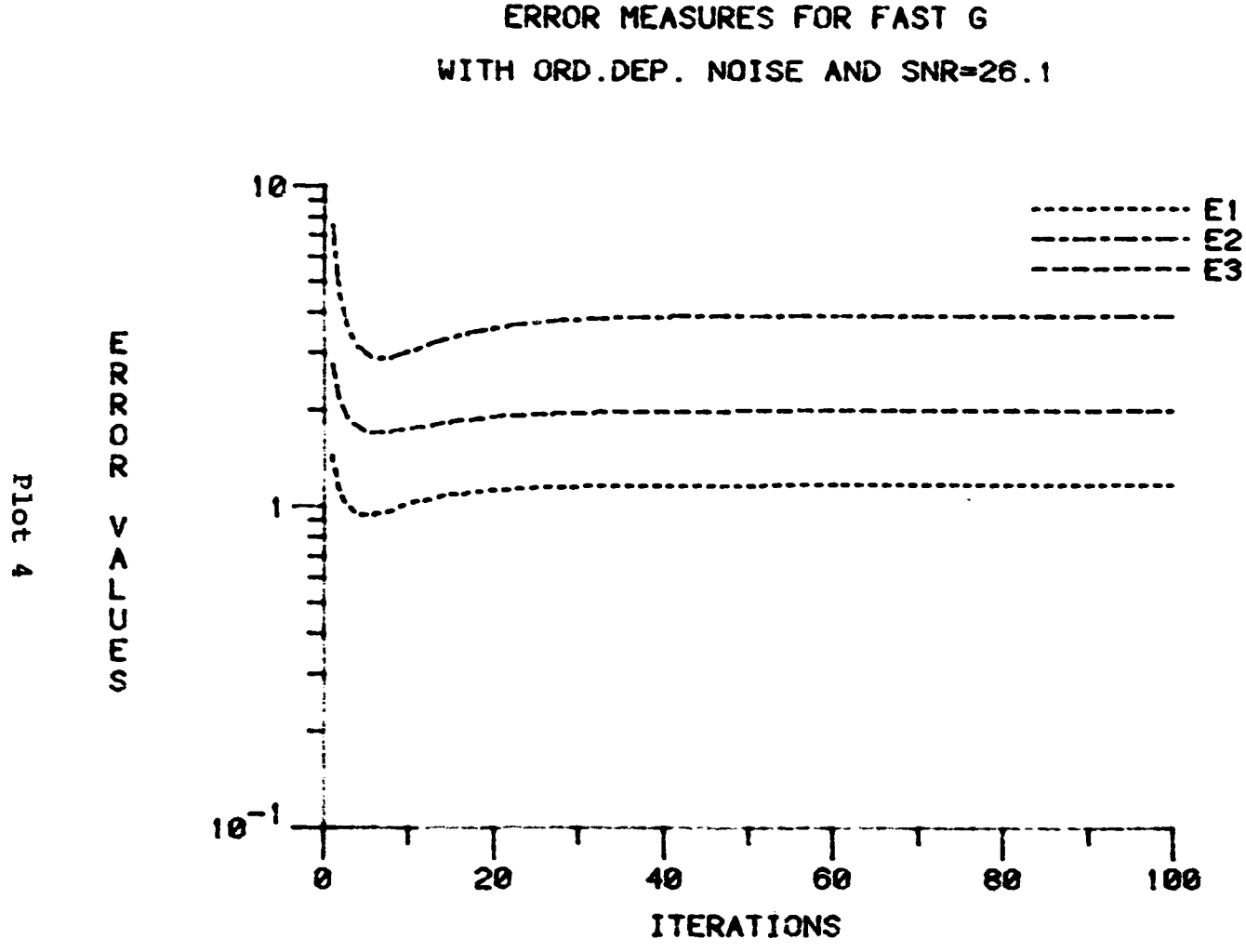
Plot 1


ERROR MEASURES FOR FAST G
WITH ORD.DEP. NOISE AND SNR=5.22
10^2
10
1
ERROR VALUES
E1
E2
E3
0
20
40
60
80
100
ITERATIONS

Plot 2

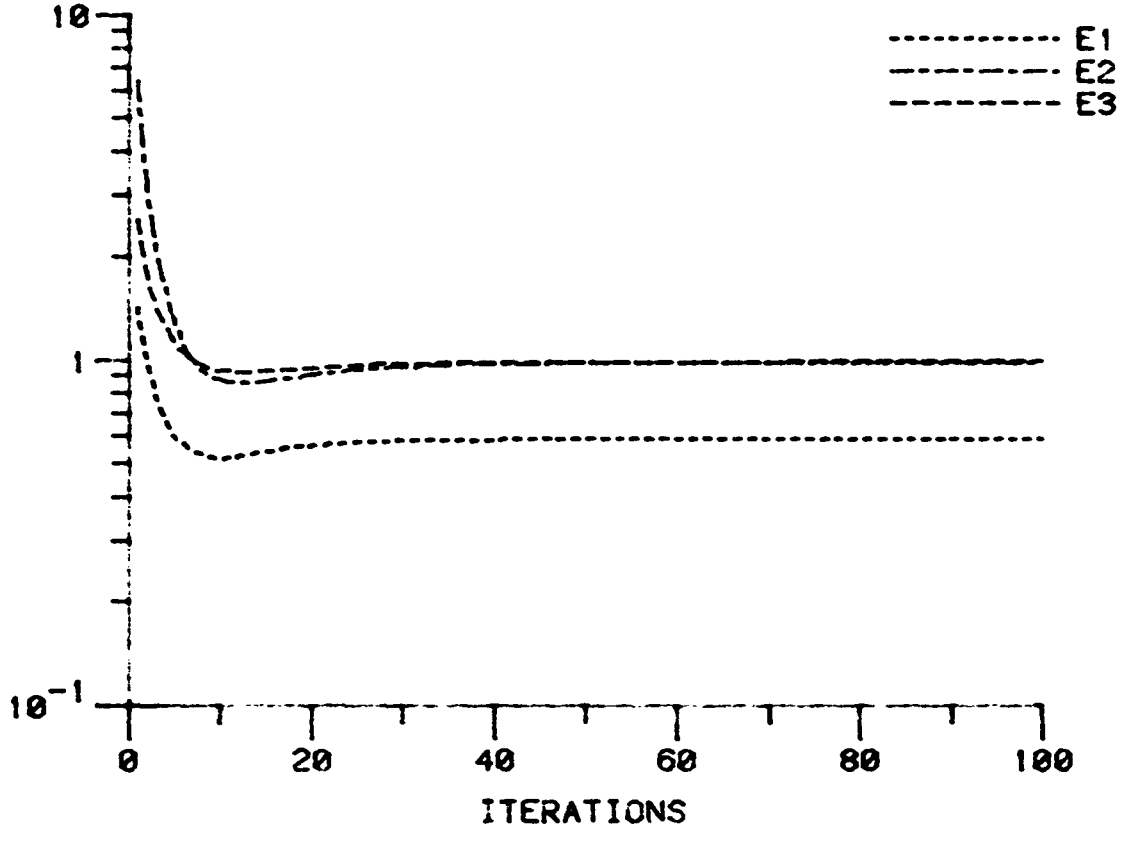

ERROR MEASURES FOR FAST G
WITH ORD. DEP. NOISE AND SNR=11.7
10^2
10
1
E1
E2
E3
ERROR VALUES
0
20
40
60
80
100
ITERATIONS

Plot 3

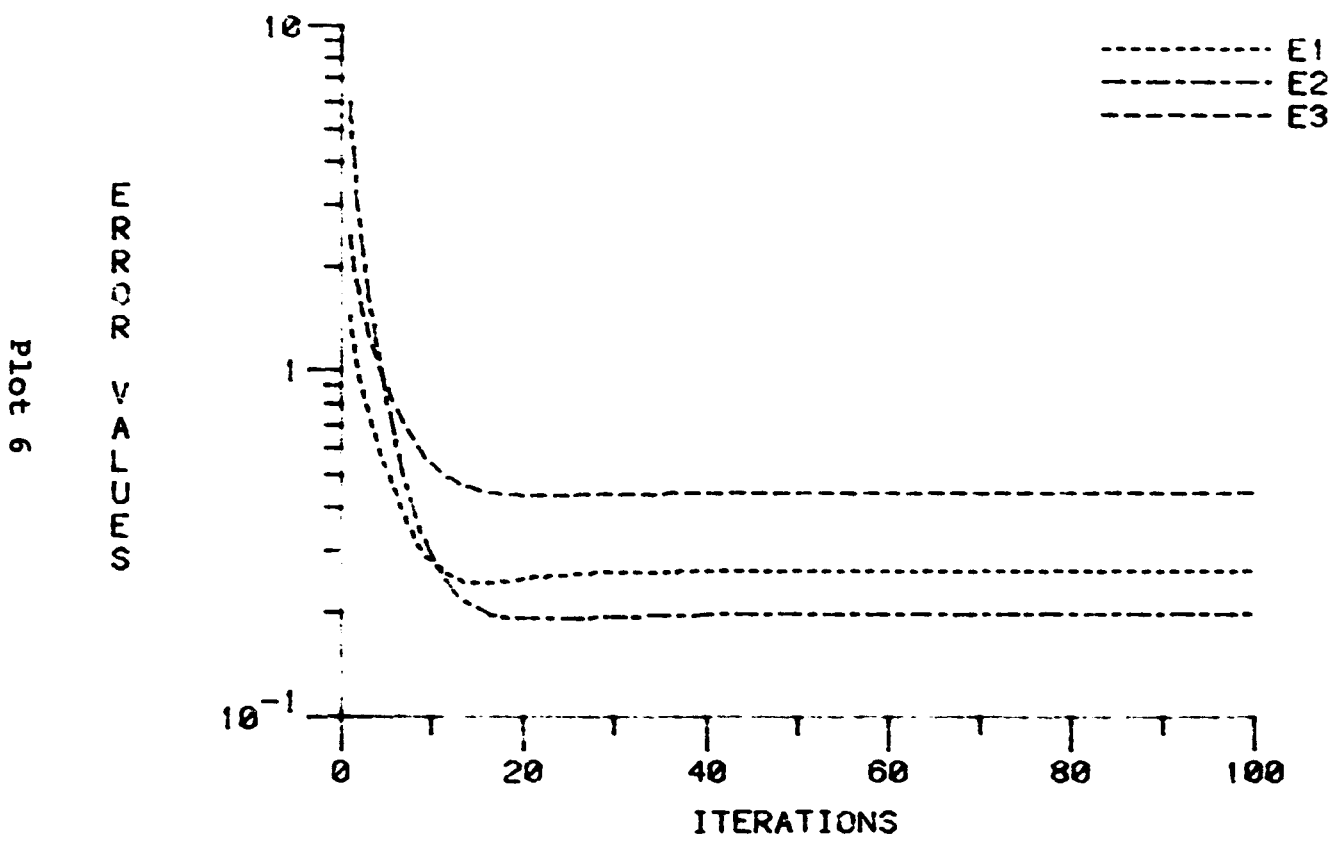

ERROR MEASURES FOR FAST G
WITH ORD.DEP. NOISE AND SNR=26.1
E1
E2
E3
10
1
10^{-1}
ERROR VALUES
0
20
40
60
80
100
ITERATIONS
Plot 4

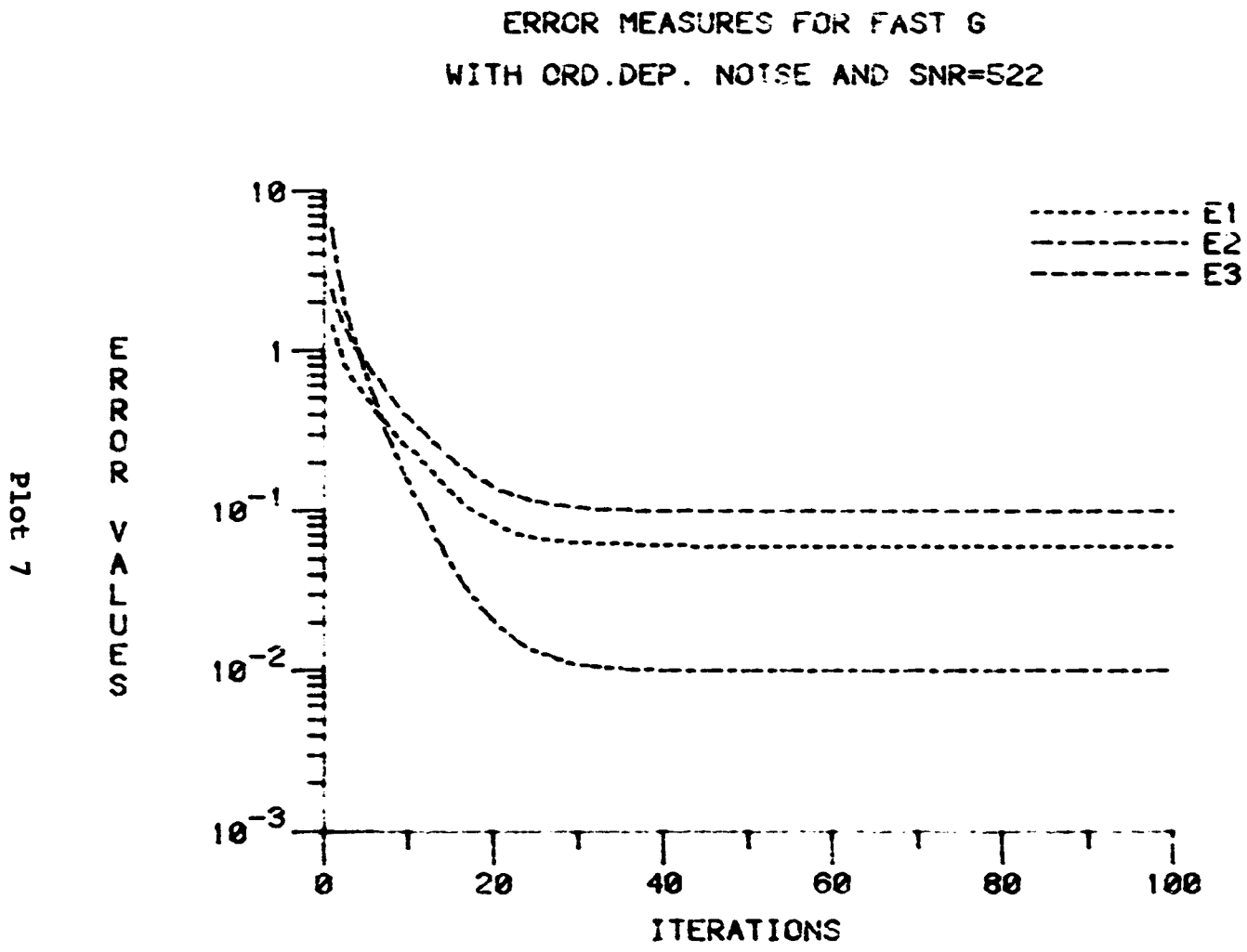
ERROR MEASURES FOR FAST G

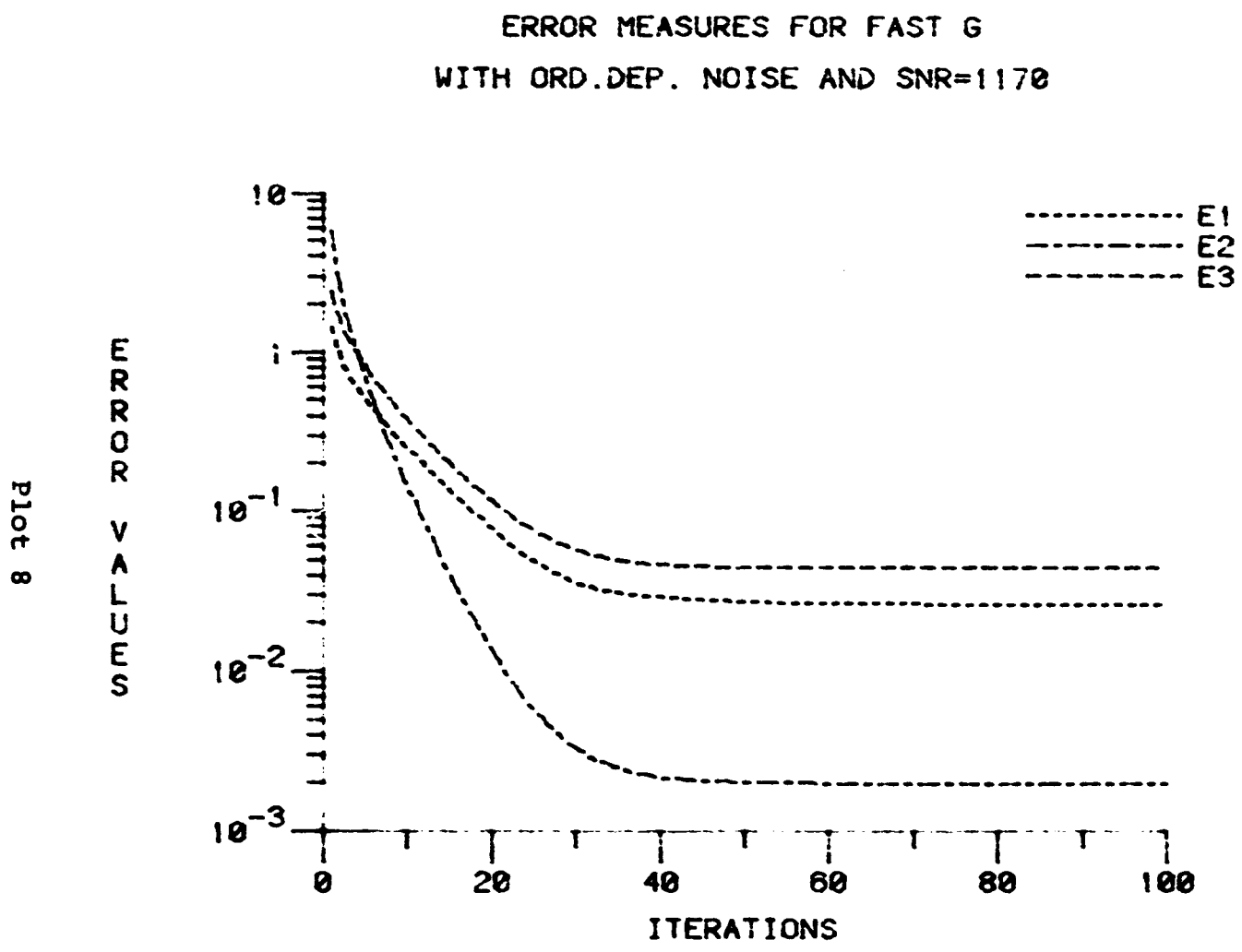
WITH ORD. DEP. NOISE AND SNR=52



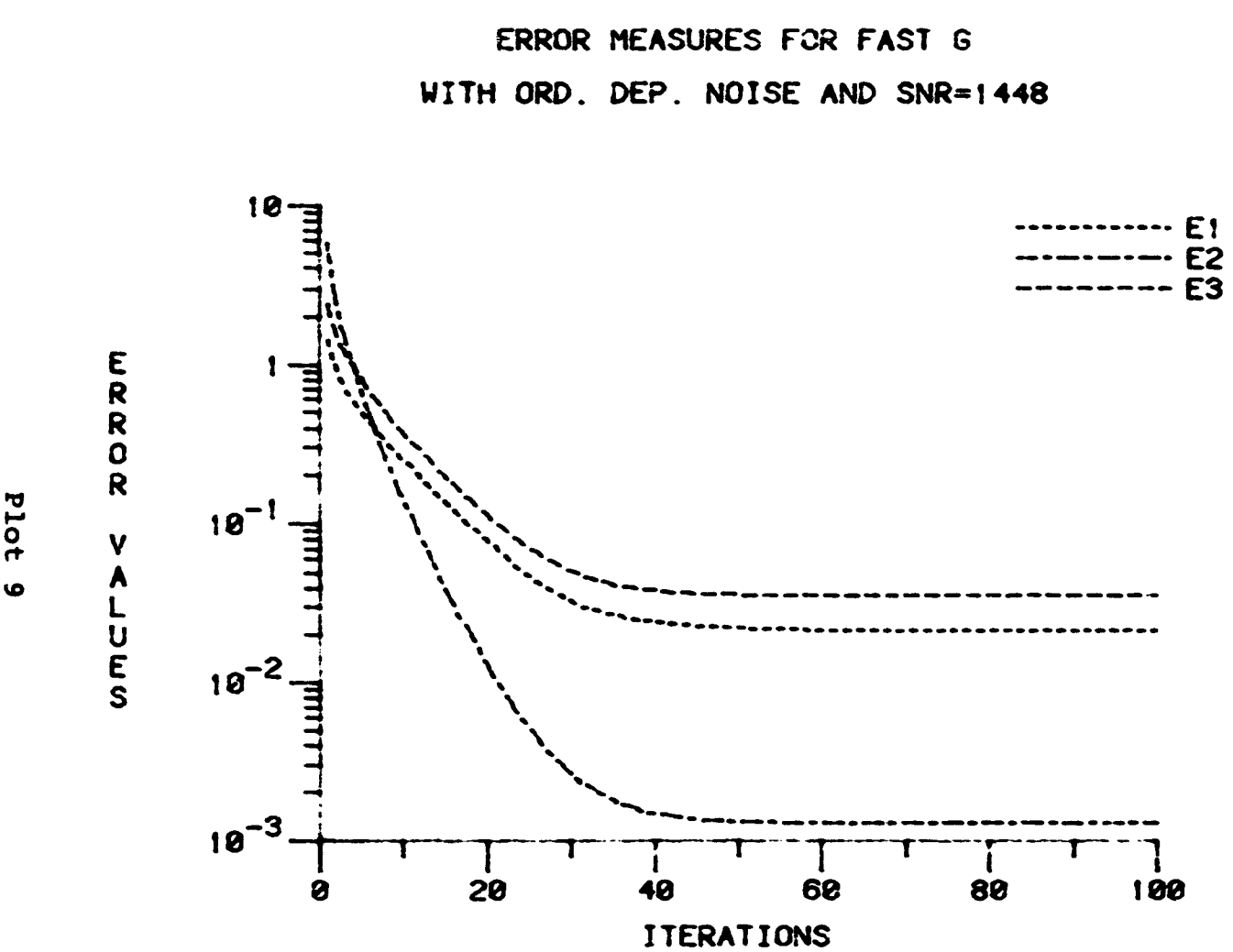
Plot 5


ERROR MEASURES FOR FAST G
WITH ORD.DEP. NOISE AND SNR=117
E1
E2
E3
10
1
10^{-1}
ERROR VALUES
0
20
40
60
80
100
ITERATIONS
PLOT 6

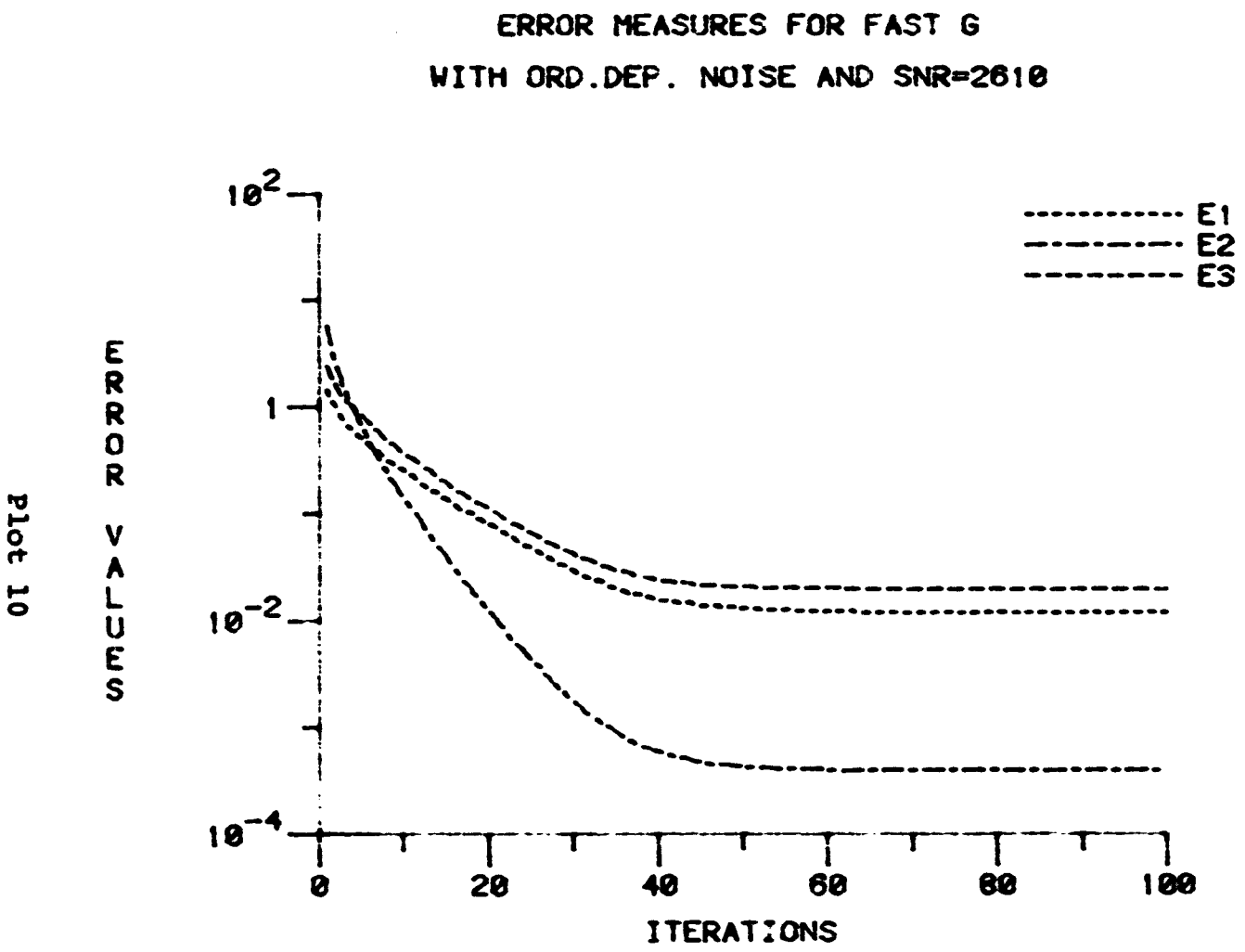

ERROR MEASURES FOR FAST G
WITH ORD.DEP. NOISE AND SNR=522
E1
E2
E3
10
1
10^{-1}
10^{-2}
10^{-3}
ERROR VALUES
0
20
40
60
80
100
ITERATIONS
PLOT 7

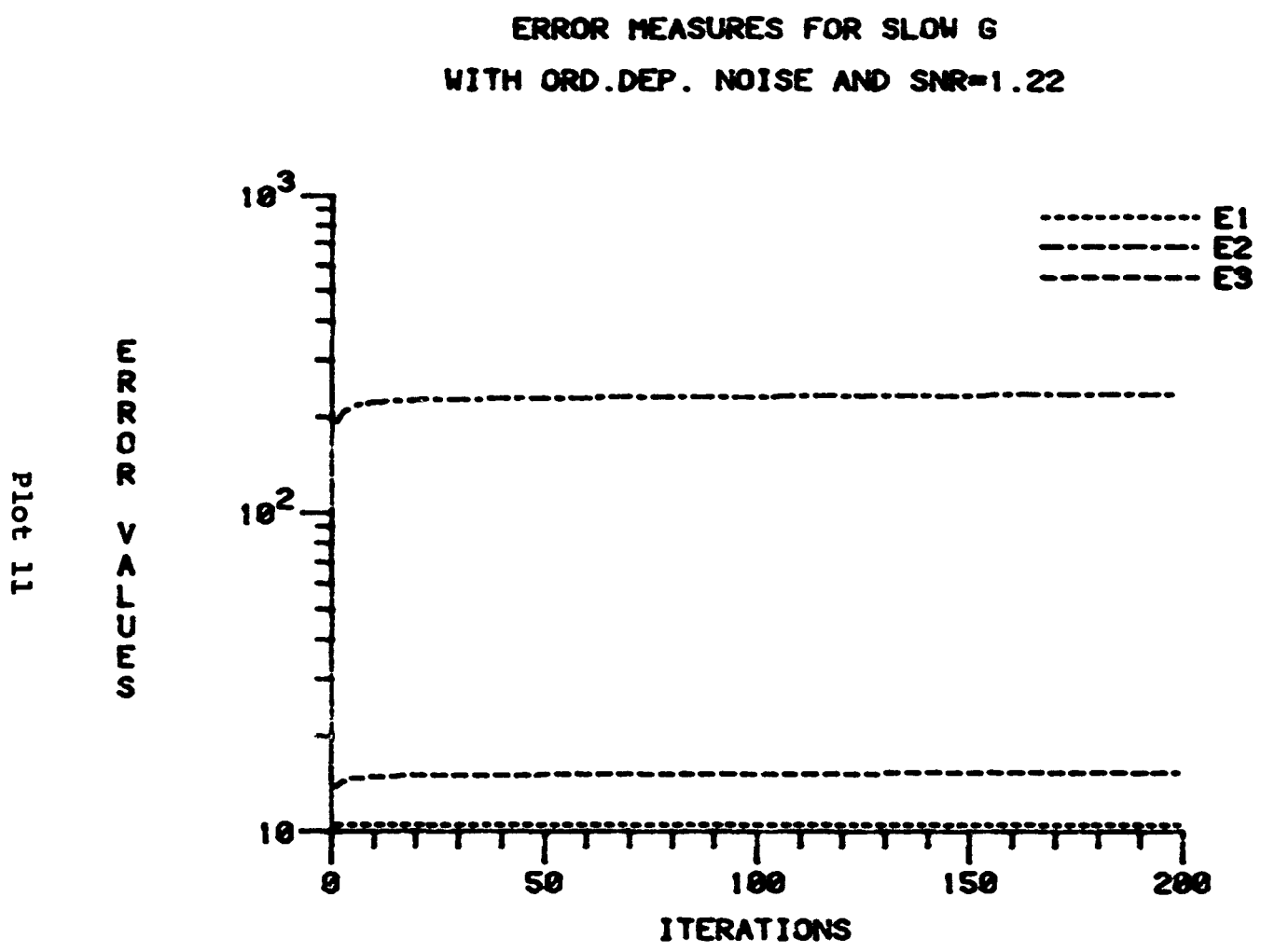

ERROR MEASURES FOR FAST G
WITH ORD.DEP. NOISE AND SNR=1170
E1
E2
E3
ERROR VALUES
10
10^{-1}
10^{-2}
10^{-3}
0
20
40
60
80
100
ITERATIONS

plot 8


ERROR MEASURES FOR FAST G
WITH ORD. DEP. NOISE AND SNR=1448
E1
E2
E3
ERROR VALUES
10
1
10^-1
10^-2
10^-3
0
20
40
60
80
100
ITERATIONS

Plot 9

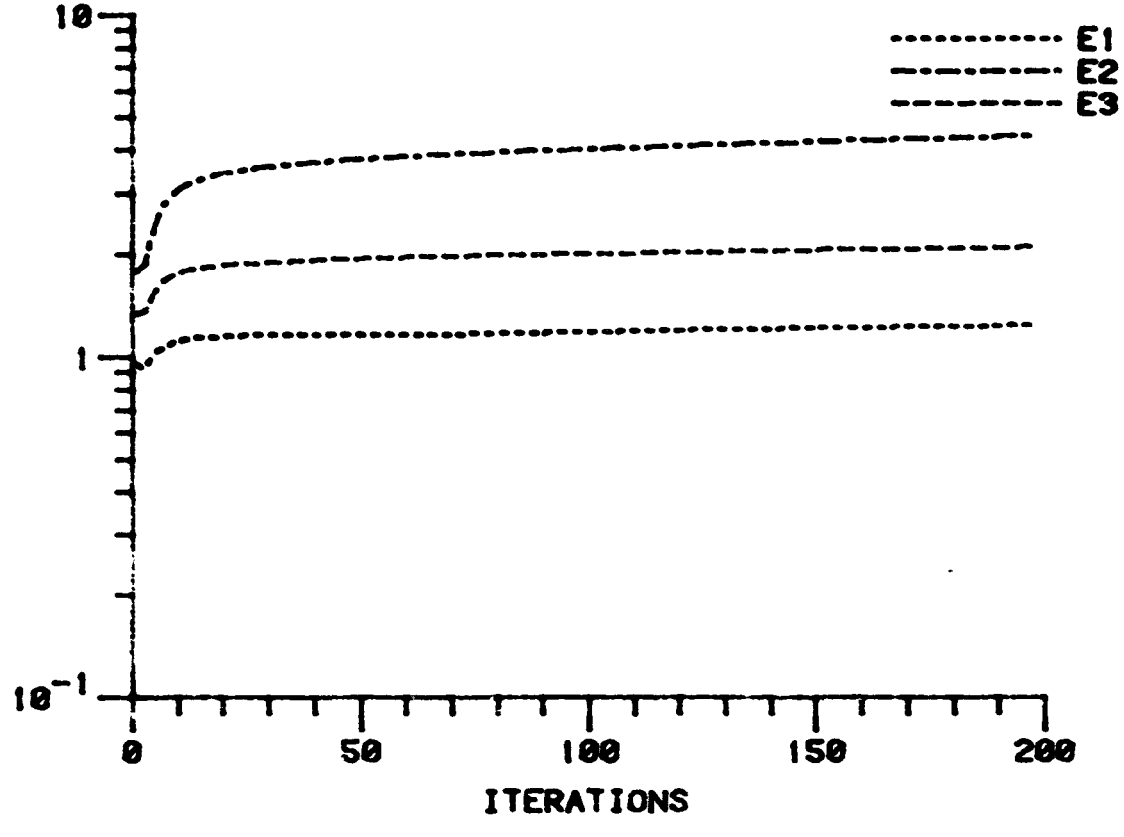

ERROR MEASURES FOR FAST G
WITH ORD.DEP. NOISE AND SNR=2610
E1
E2
E3
10^2
1
10^{-2}
10^{-4}
ERROR VALUES
0
20
40
60
80
100
ITERATIONS
plot 10


ERROR MEASURES FOR SLOW G
WITH ORD.DEP. NOISE AND SNR=1.22
E1
E2
E3
10^3
10^2
10
ERROR VALUES
0
50
100
150
200
ITERATIONS

Plot 11

ERROR MEASURES FOR SLOW G

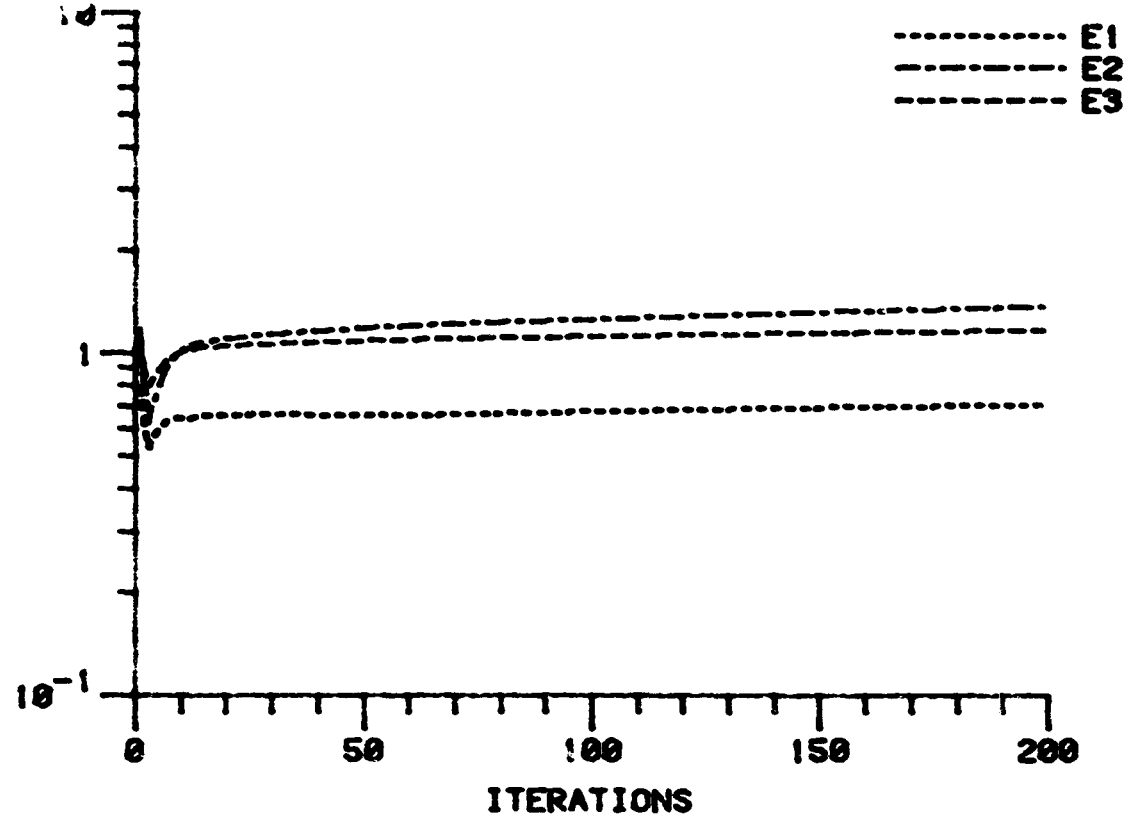
WITH ORD.DEP. NOISE AND SNR=5.7



Plot 12

ERROR MEASURES FOR SLOW G

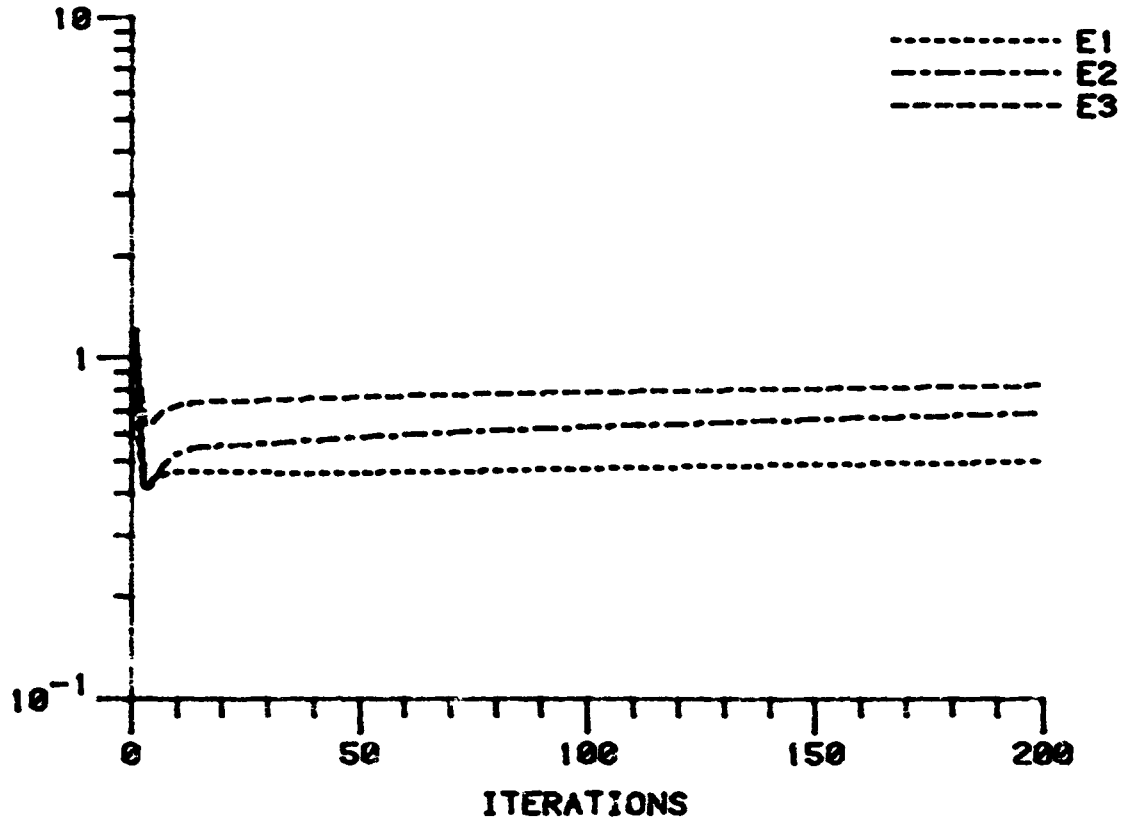
WITH ORD.DEP. NOISE AND SNR=10.3



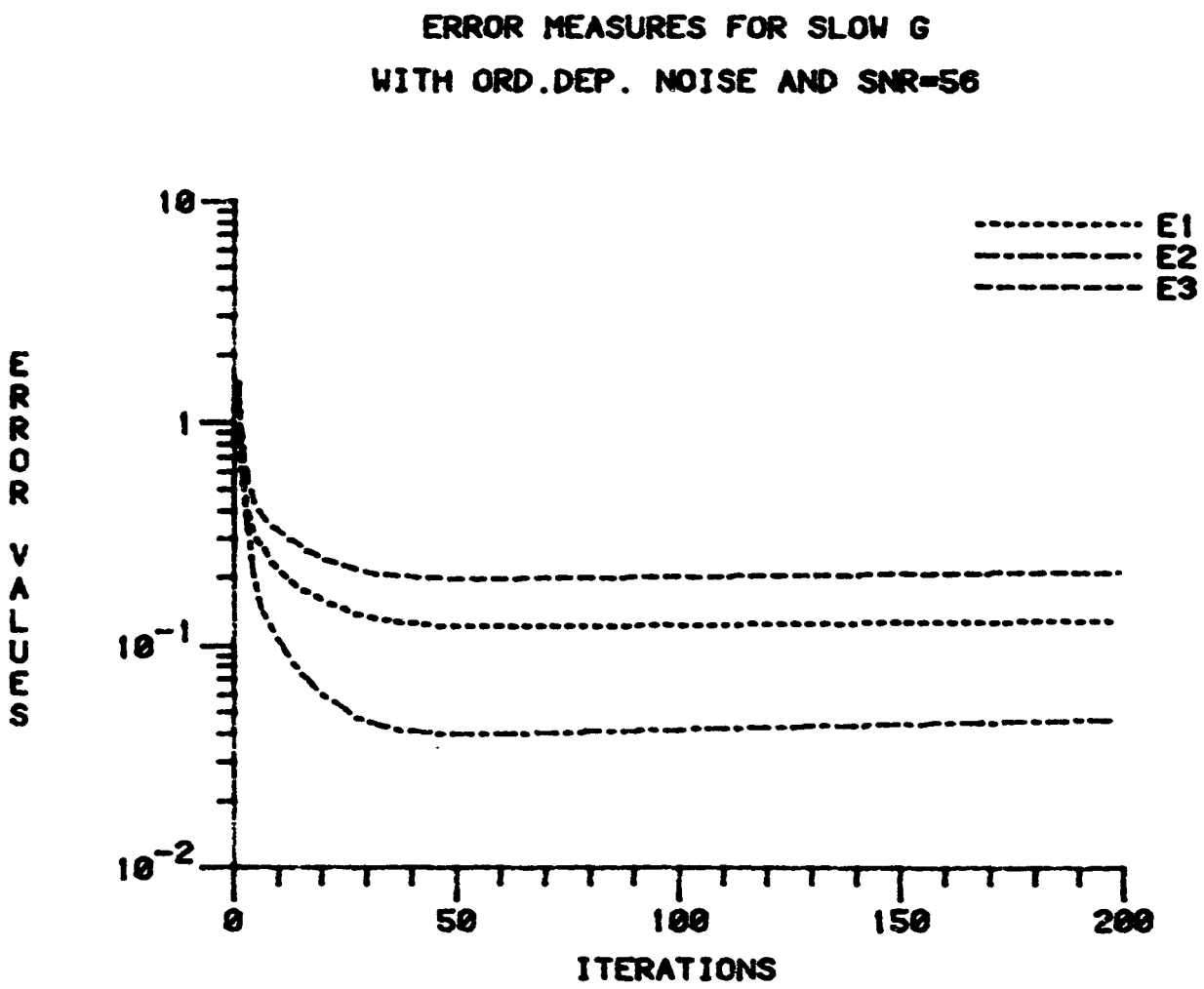
Plot 13

ERROR MEASURES FOR SLOW G

WITH ORD.DEP. NOISE AND SNR=14.5



Plot 14

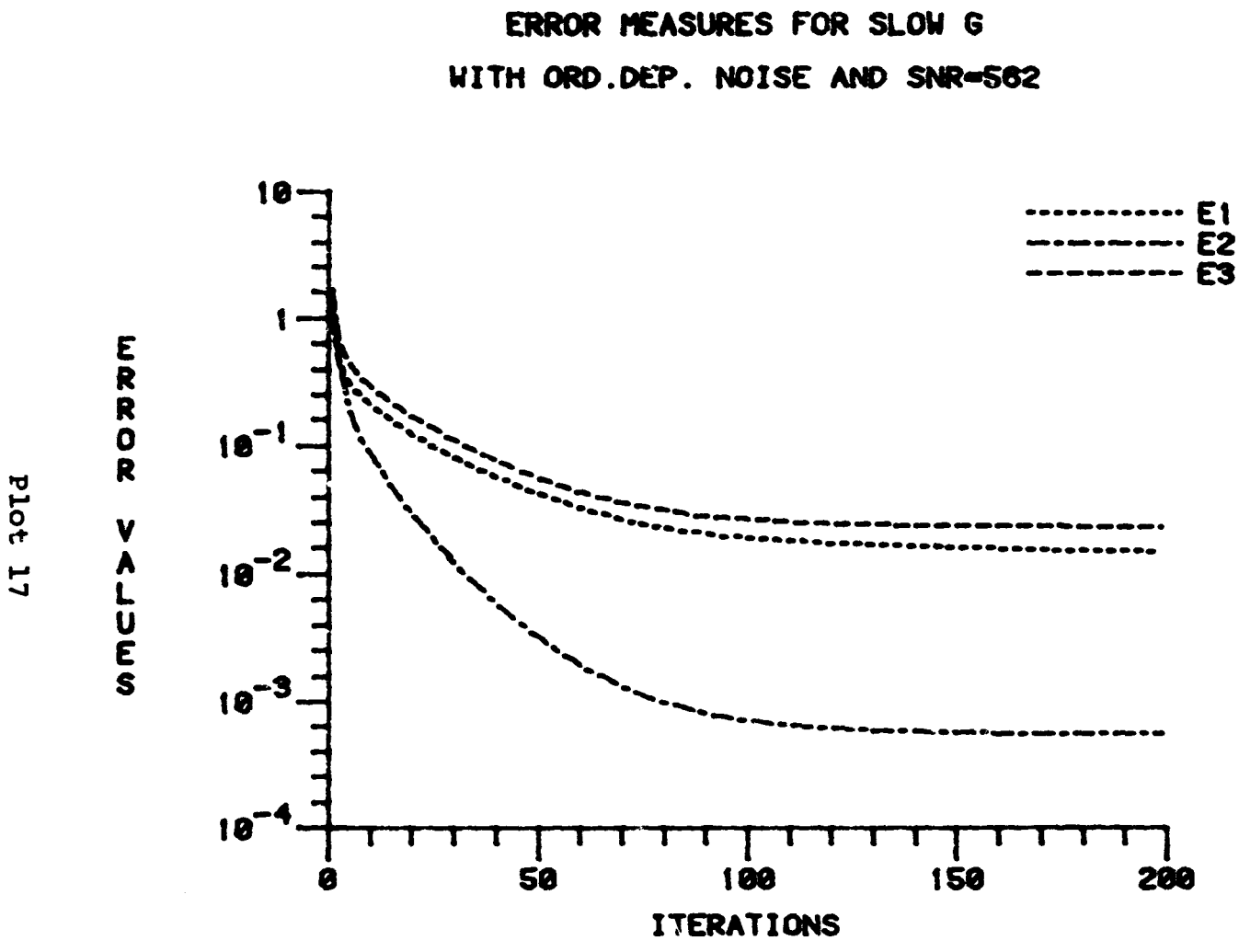

ERROR MEASURES FOR SLOW G
WITH ORD.DEP. NOISE AND SNR=56
E1
E2
E3
10
1
10^{-1}
10^{-2}
ERROR VALUES
0
50
100
150
200
ITERATIONS

Plot 15

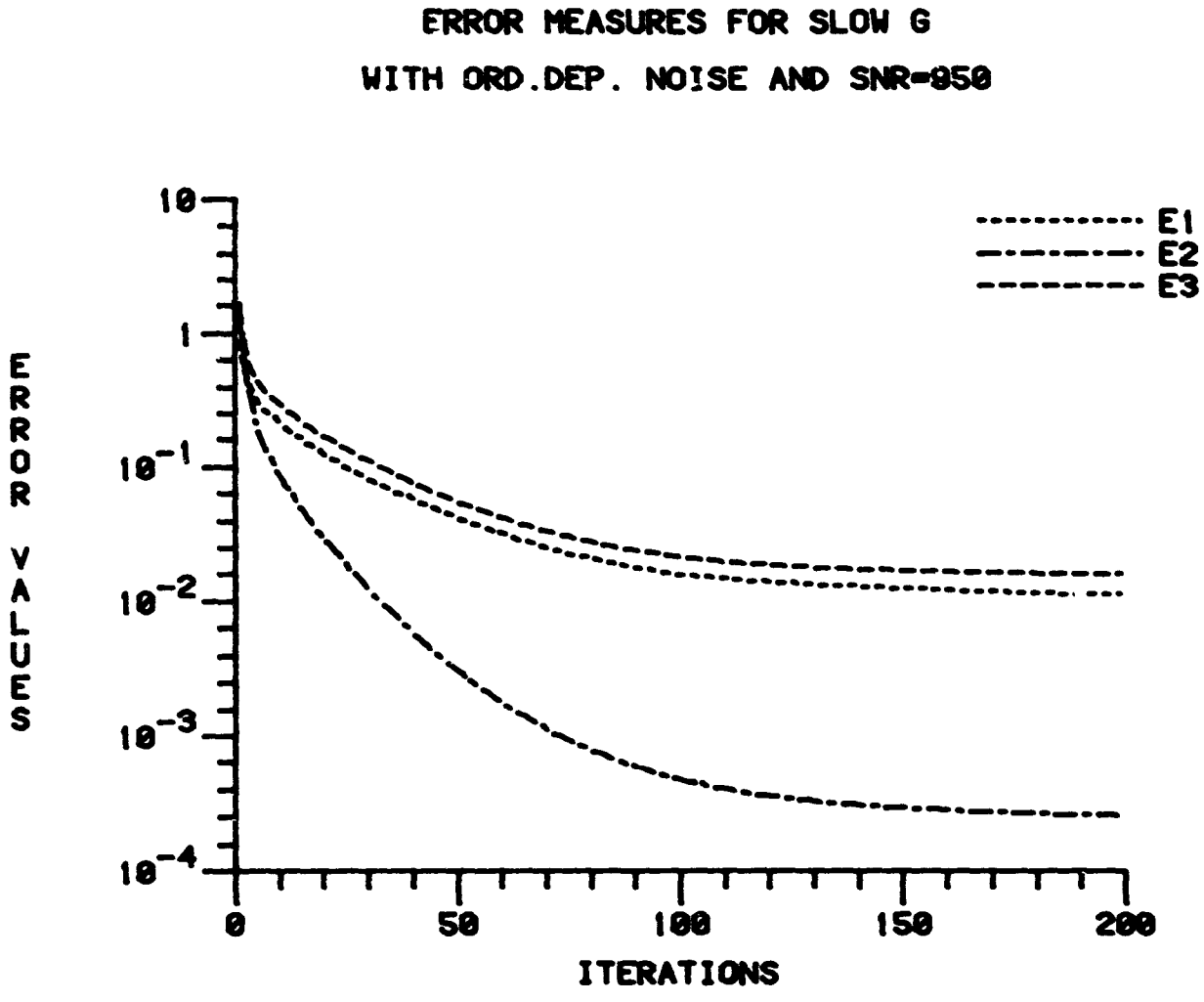
ERROR MEASURES FOR SLOW G

WITH ORD.DEP. NOISE AND SNR=126

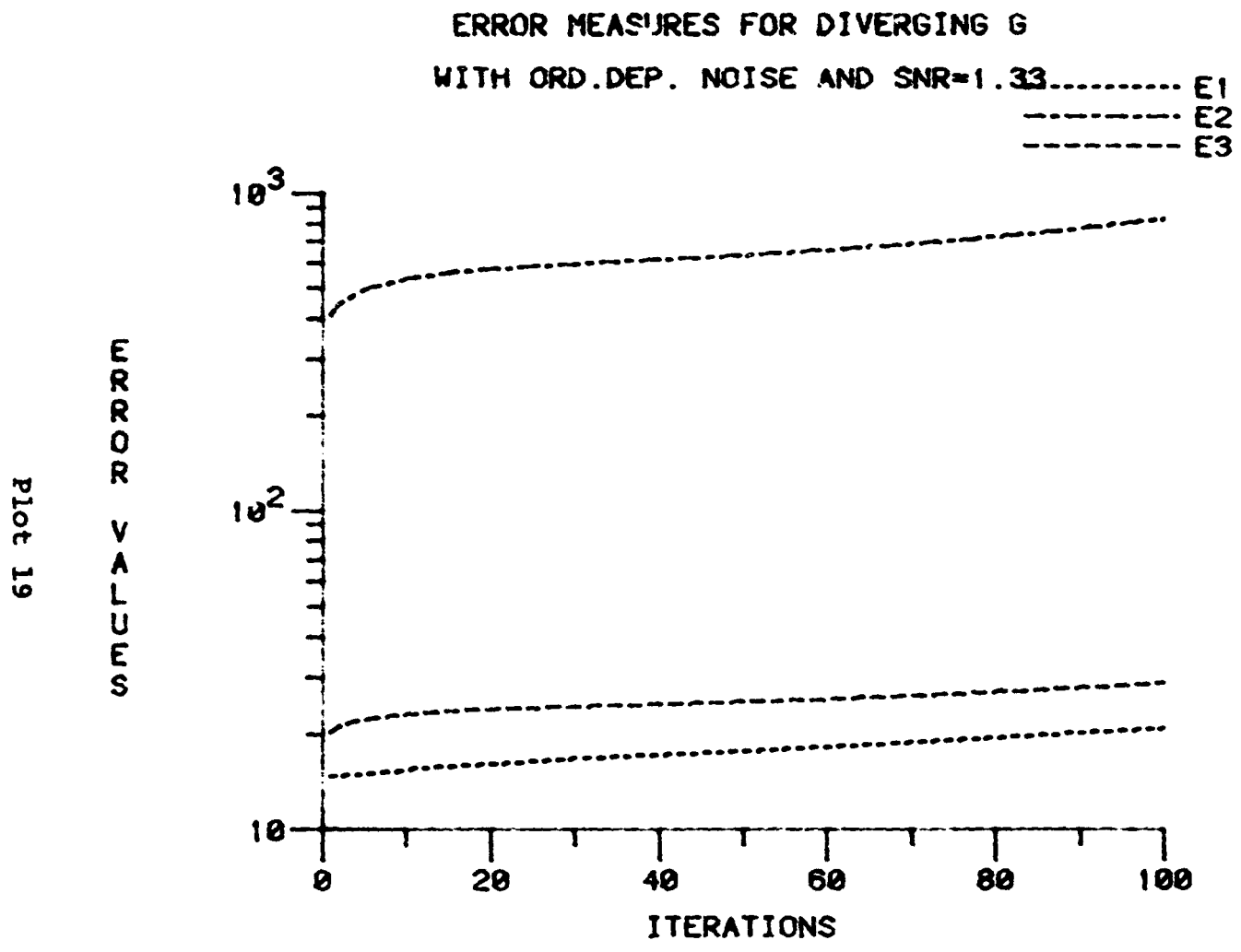
Plot 16


ERROR MEASURES FOR SLOW G
WITH ORD.DEP. NOISE AND SNR=562
E1
E2
E3
ERROR VALUES
10
1
10-1
10-2
10-3
10-4
0
50
100
150
200
ITERATIONS

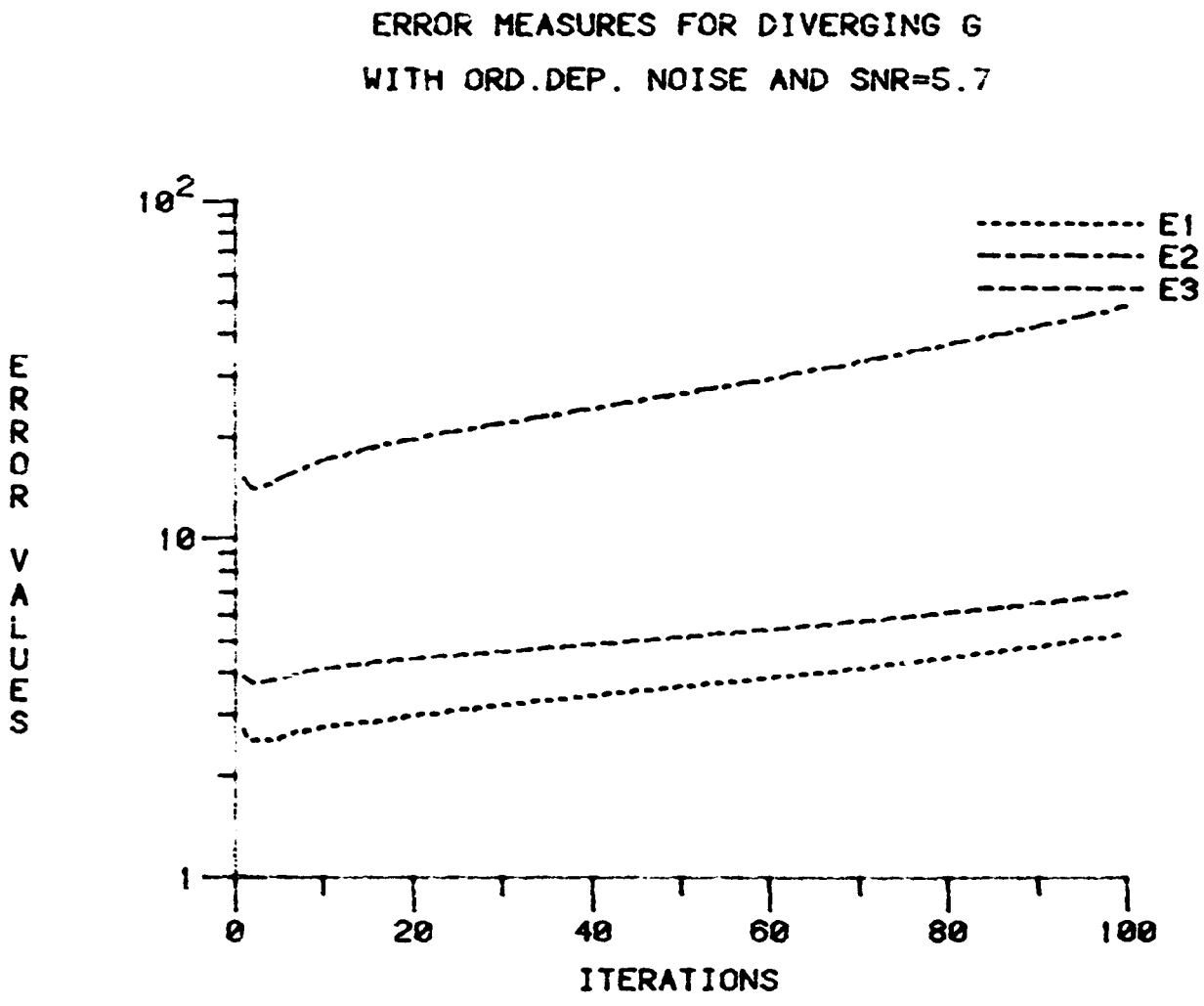
Plot 17


ERROR MEASURES FOR SLOW G
WITH ORD.DEP. NOISE AND SNR=850
E1
E2
E3
ERROR VALUES
10
1
10-1
10-2
10-3
10-4
0
50
100
150
200
ITERATIONS

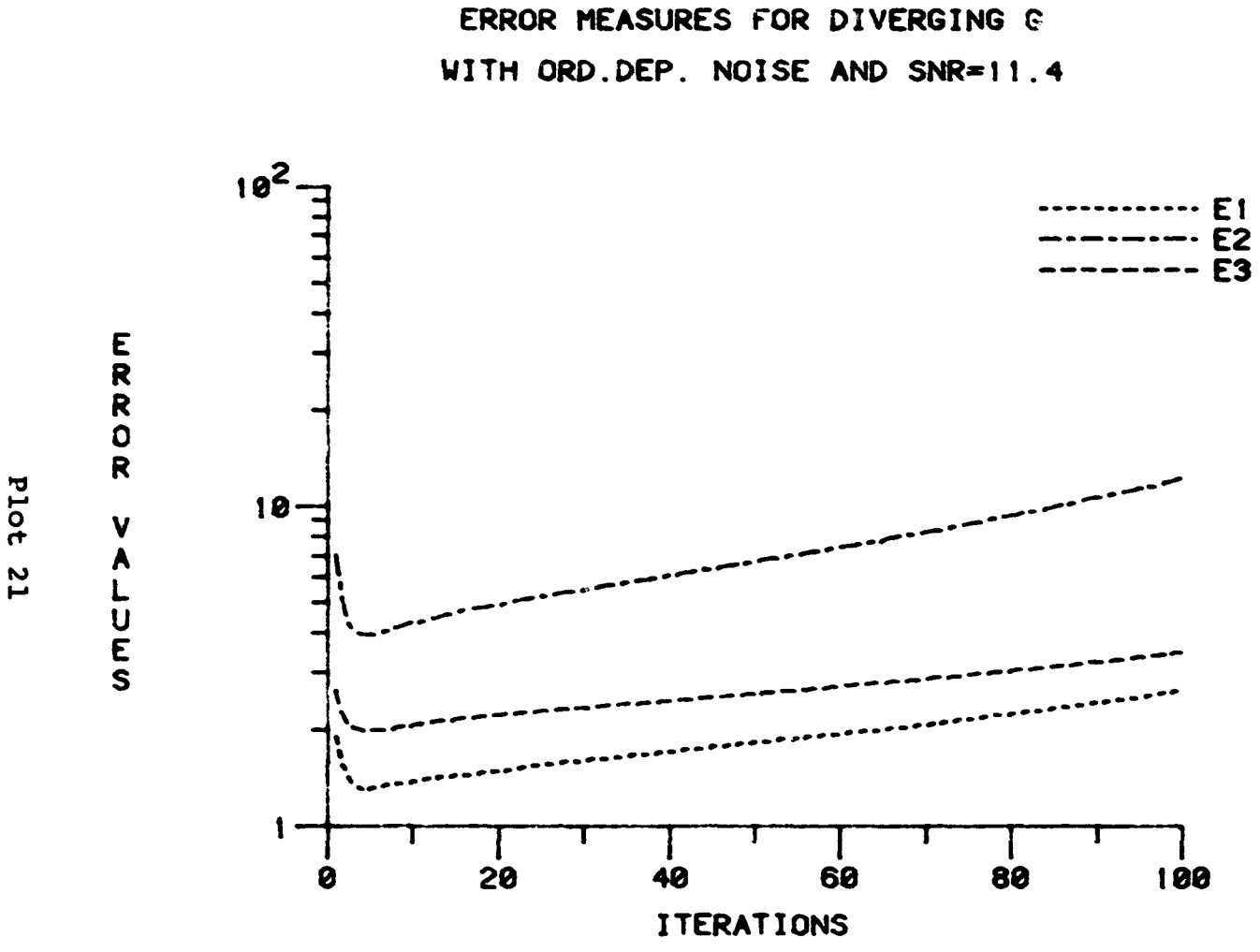
Plot 18


ERROR MEASURES FOR DIVERGING G
WITH ORD.DEP. NOISE AND SNR=1.33
E1
E2
E3
10^3
10^2
10
ERROR VALUES
0
20
40
60
80
100
ITERATIONS

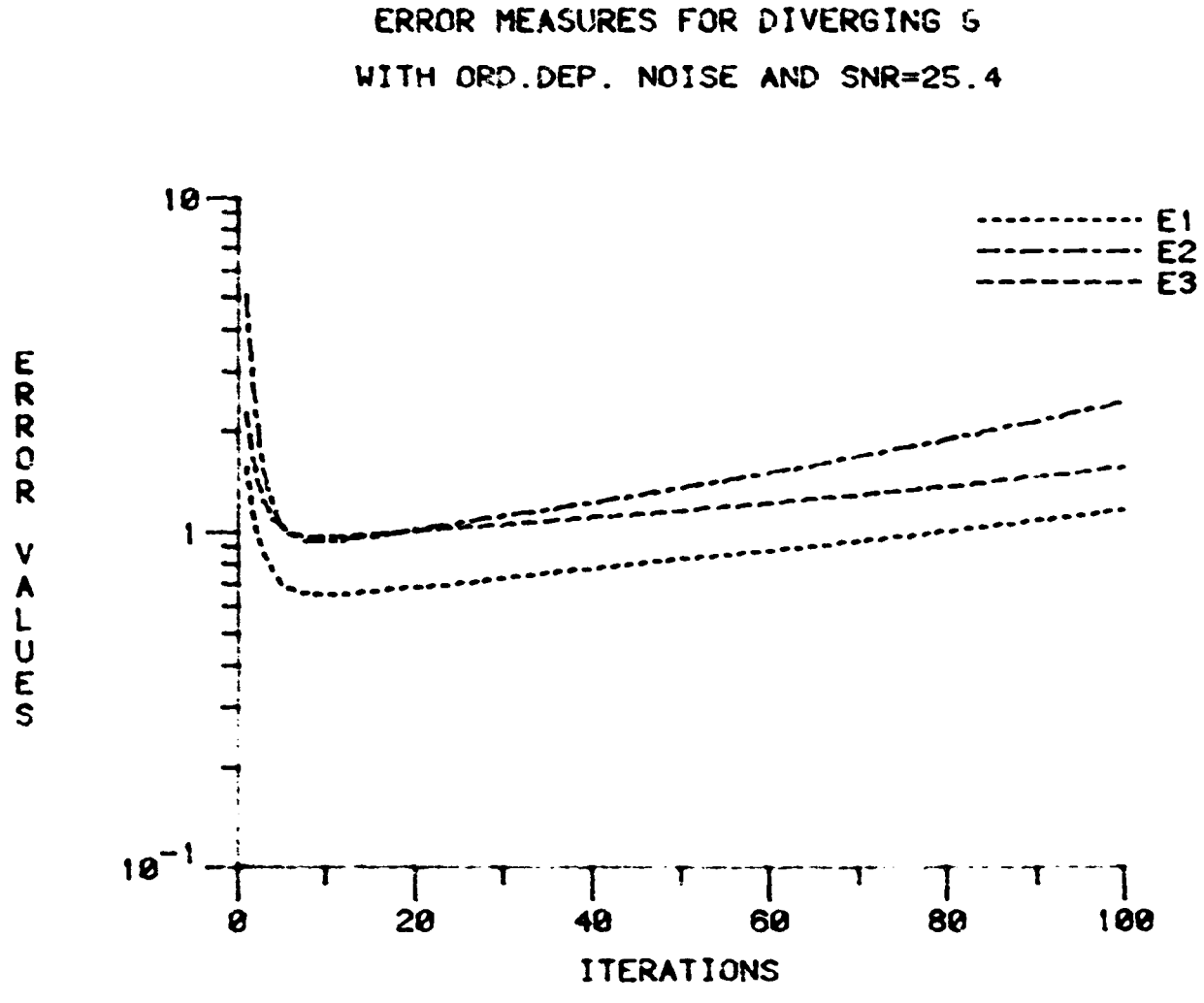
Plot 19


ERROR MEASURES FOR DIVERGING G
WITH ORD.DEP. NOISE AND SNR=5.7
10^2
10
1
ERROR VALUES
E1
E2
E3
0
20
40
60
80
100
ITERATIONS

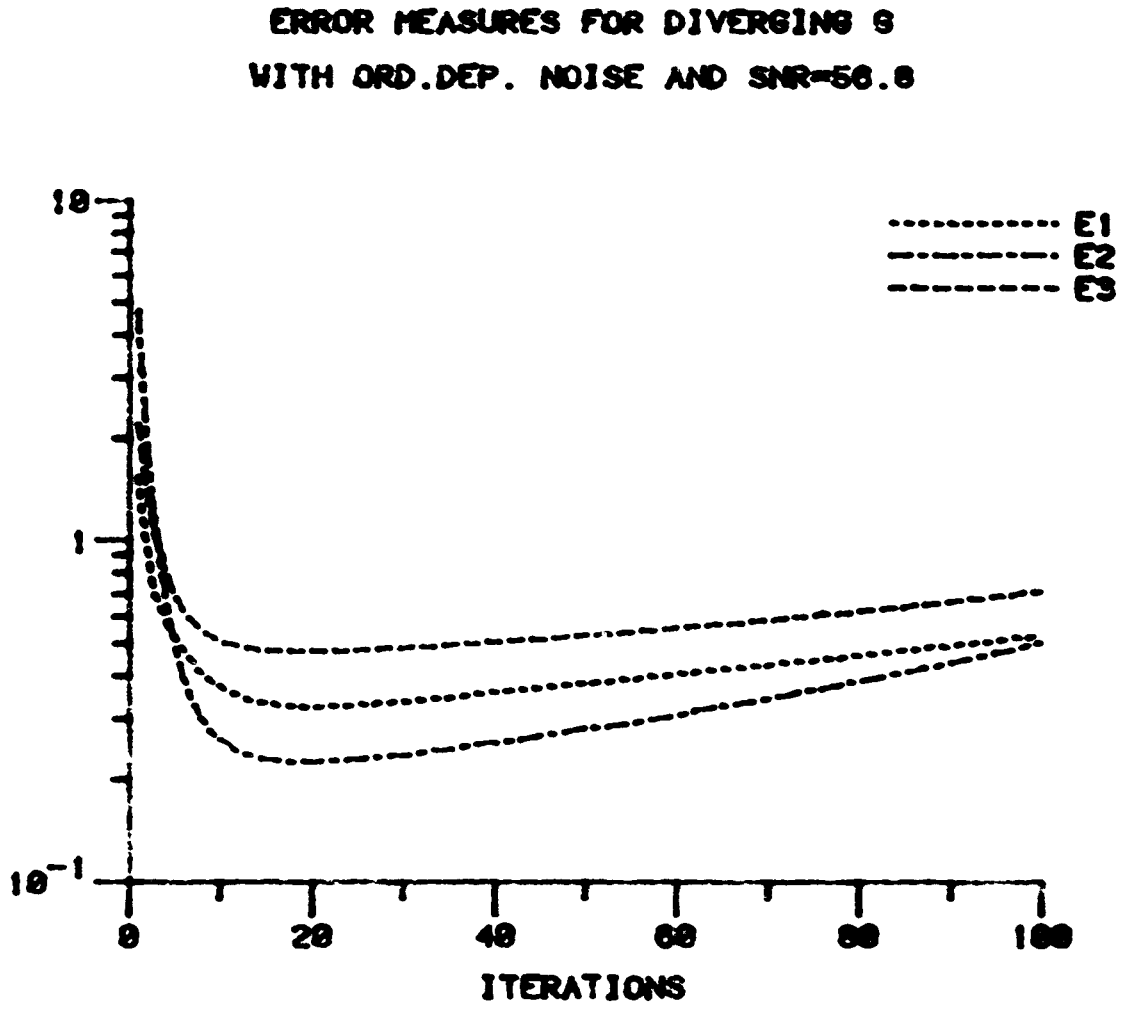
Plot 20


ERROR MEASURES FOR DIVERGING G
WITH ORD.DEP. NOISE AND SNR=11.4
E1
E2
E3
ERROR VALUES
10^2
10
1
0
20
40
60
80
100
ITERATIONS

Plot 21

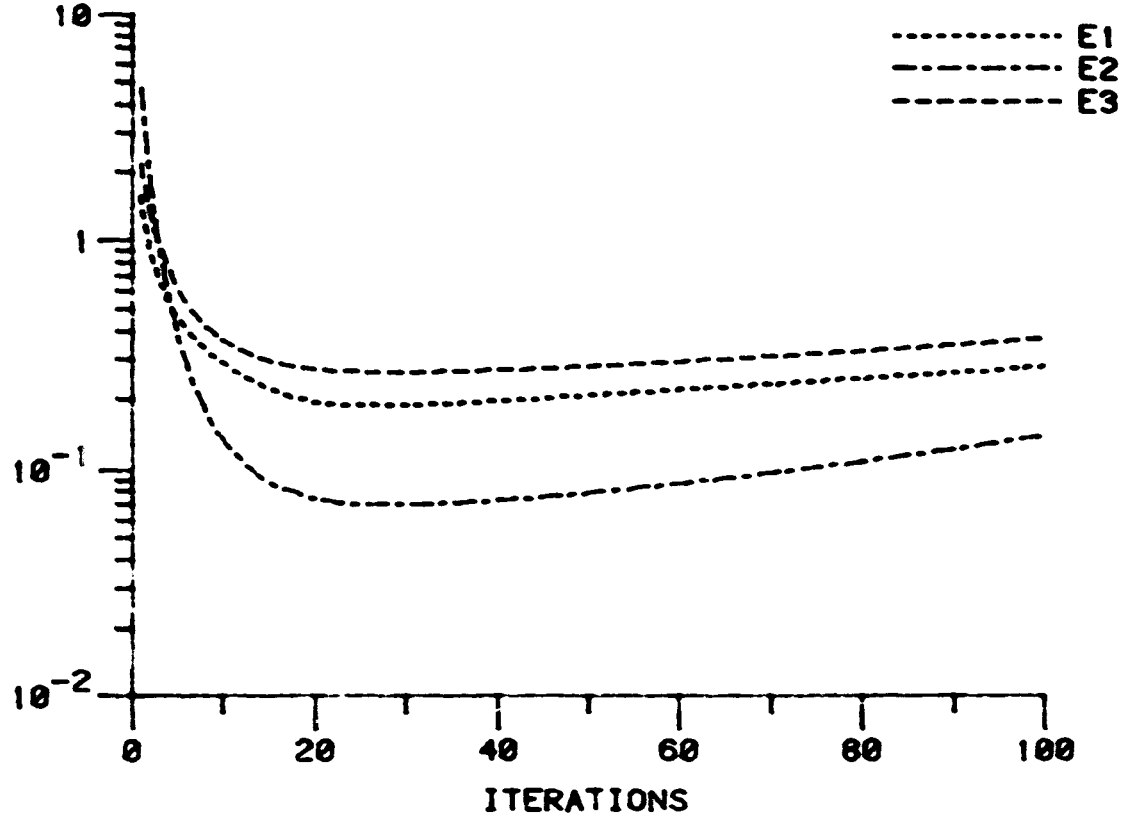

ERROR MEASURES FOR DIVERGING G
WITH ORD.DEP. NOISE AND SNR=25.4
10
1
10^{-1}
E1
E2
E3
ERROR VALUES
0
20
40
60
80
100
ITERATIONS

Plot 22

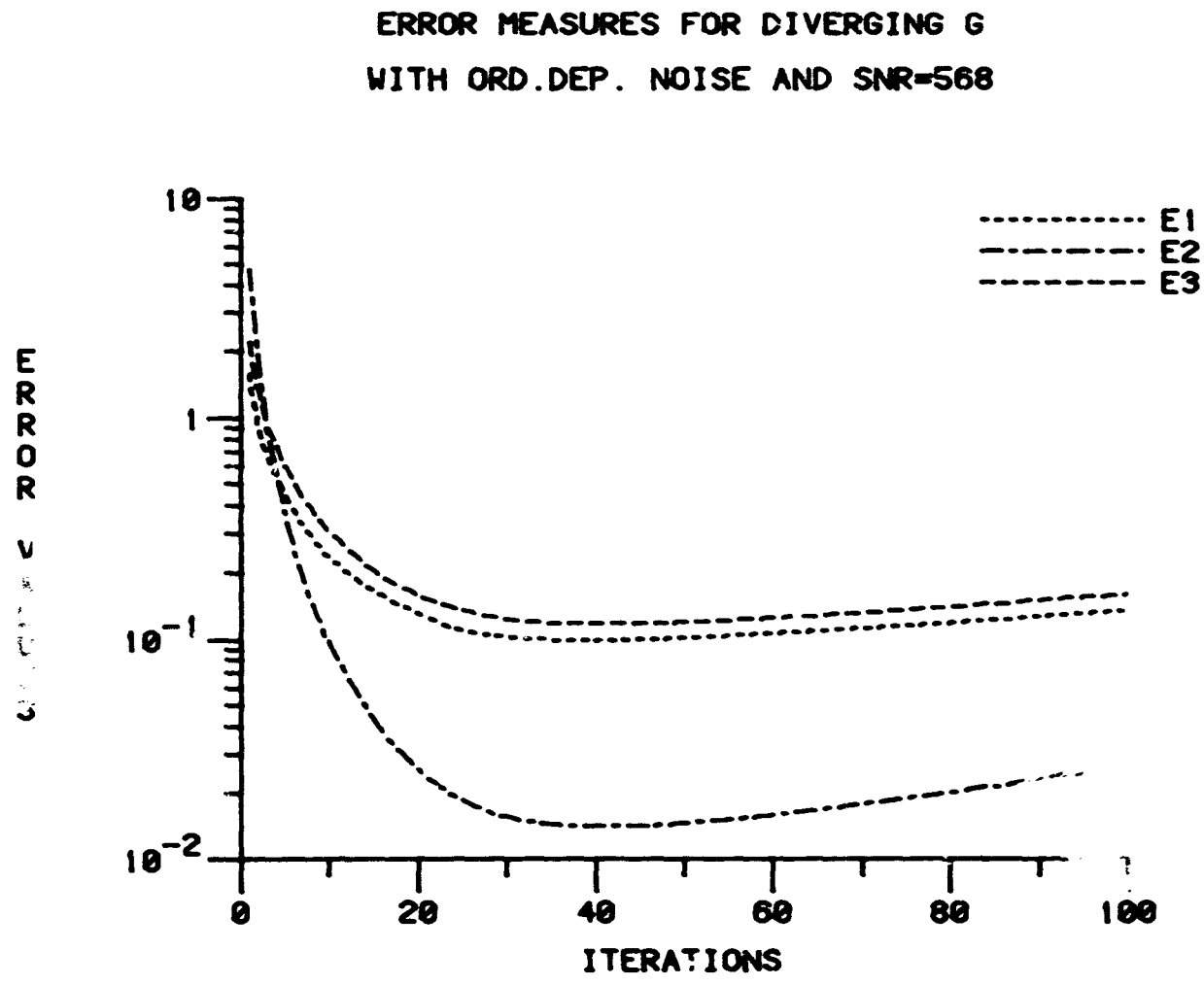

ERROR MEASURES FOR DIVERGING G
WITH ORD.DEP. NOISE AND SNR=56.8
E1
E2
E3
ERROR VALUES
10
1
10^-1
0
20
40
60
80
100
ITERATIONS

Plot 23

ERROR MEASURES FOR DIVERGING G
WITH ORD.DEP. NOISE AND SNR=114

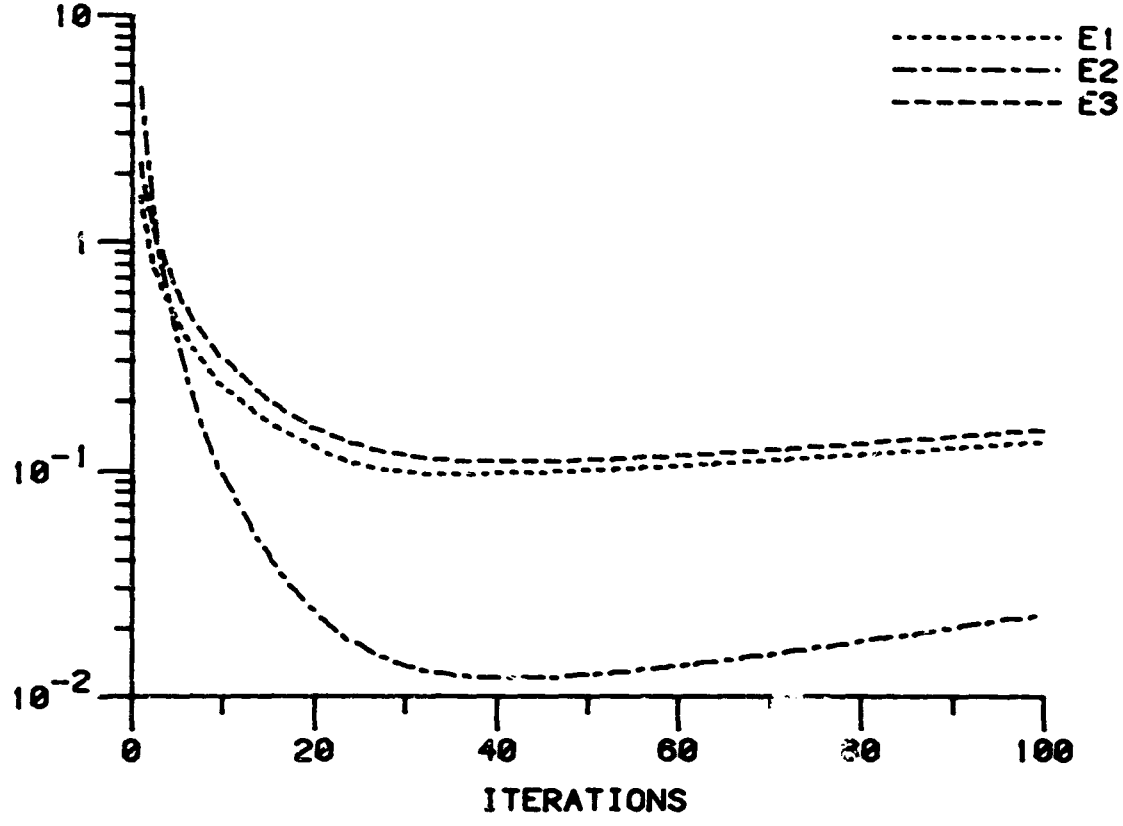


Plot 24

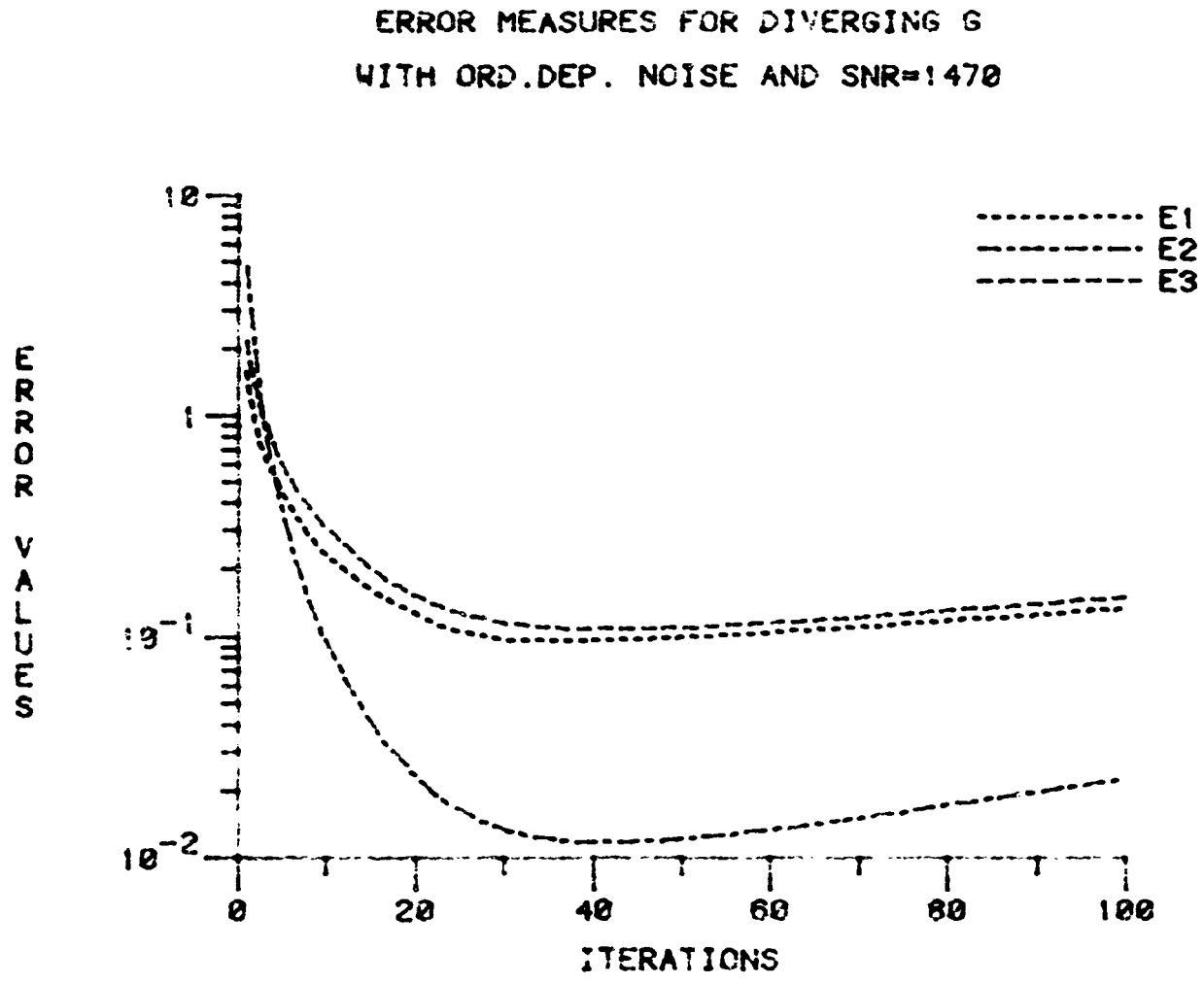

ERROR MEASURES FOR DIVERGING G
WITH ORD.DEP. NOISE AND SNR=568
E1
E2
E3
10
1
10^-1
10^-2
0
20
40
60
80
100
ITERATIONS

Plot 25

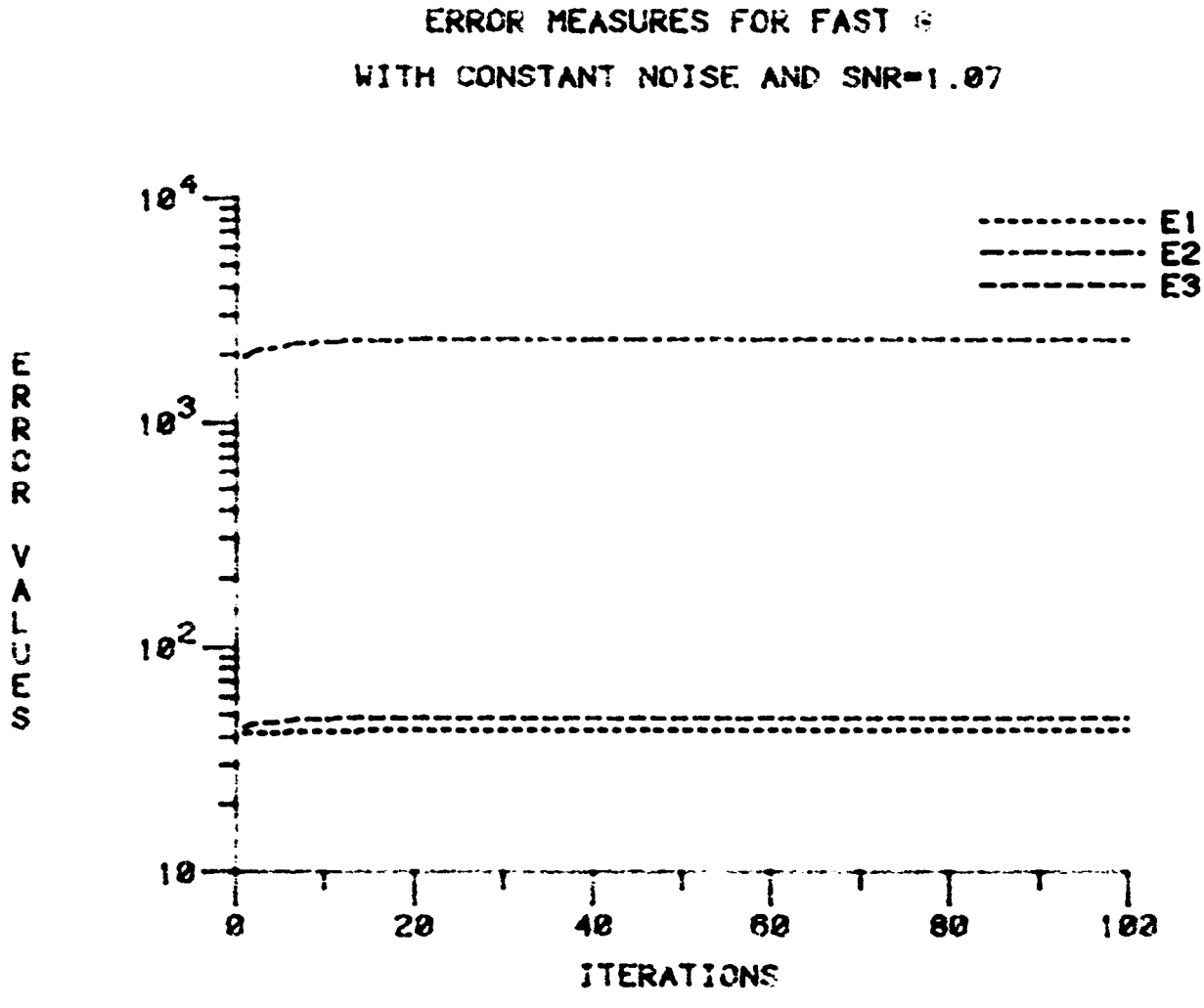
ERROR MEASURES FOR DIVERGING G
WITH ORD.DEP. NOISE AND SNR=1140



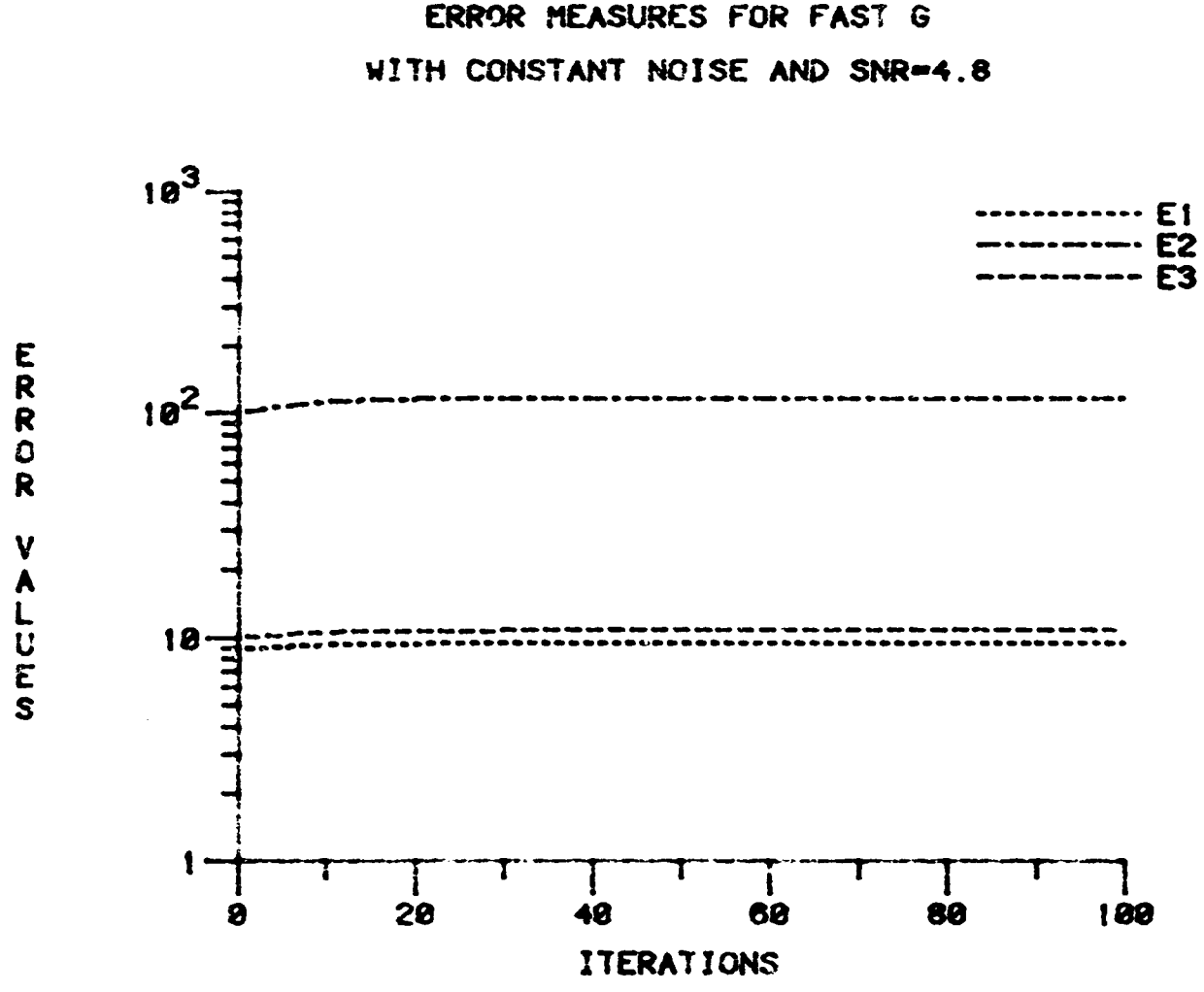
Plot 26


ERROR MEASURES FOR DIVERGING G
WITH ORD.DEP. NOISE AND SNR=1470
E1
E2
E3
ERROR VALUES
10
1
10^{-1}
10^{-2}
0
20
40
60
80
100
ITERATIONS

Plot 27


ERROR MEASURES FOR FAST
WITH CONSTANT NOISE AND SNR=1.07
E1
E2
E3
ERROR VALUES
10^4
10^3
10^2
10
0
20
40
60
80
100
ITERATIONS

Plot 28

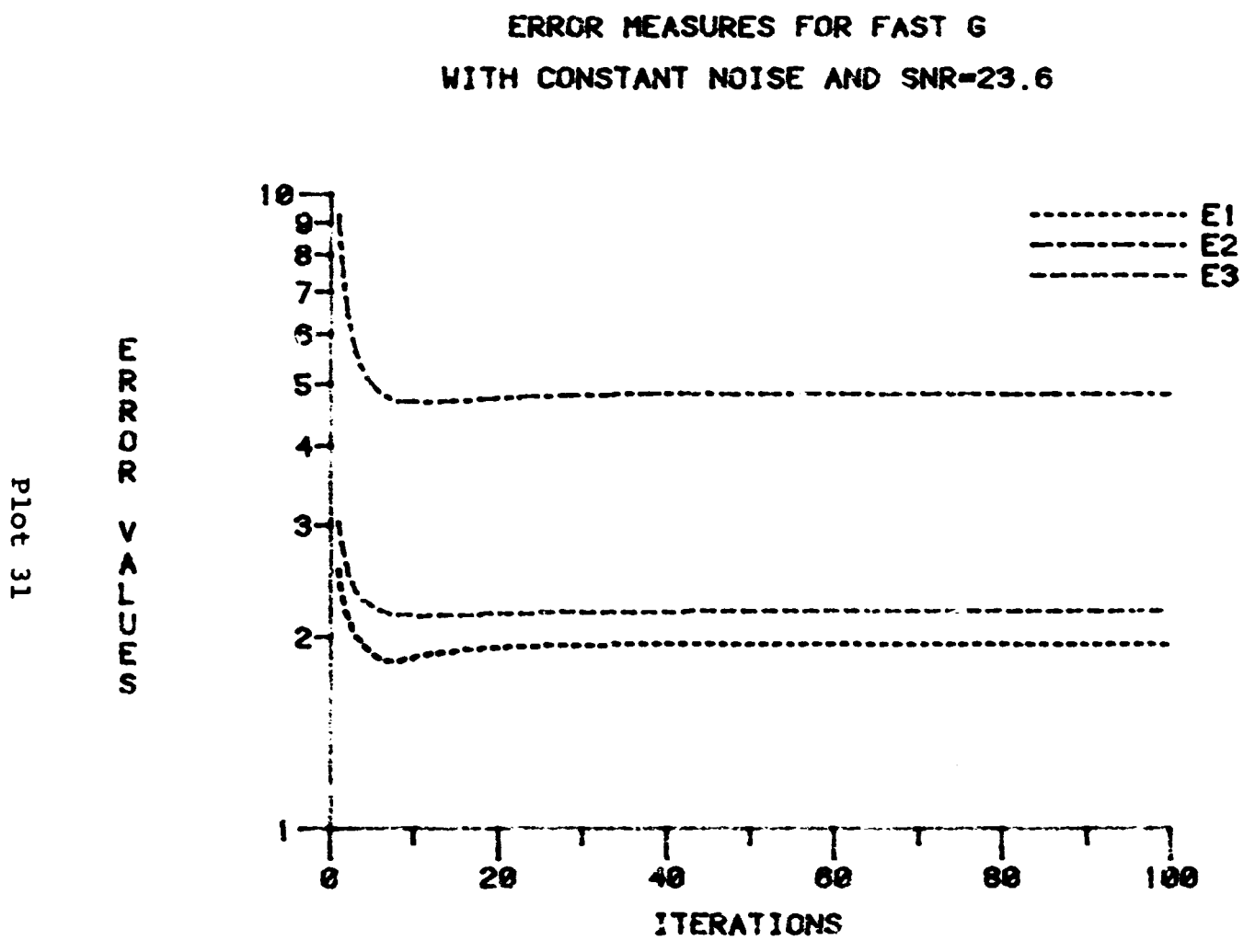

ERROR MEASURES FOR FAST G
WITH CONSTANT NOISE AND SNR=4.8
E1
E2
E3
ERROR VALUES
10^3
10^2
10
1
0
20
40
60
80
100
ITERATIONS

Plot 29

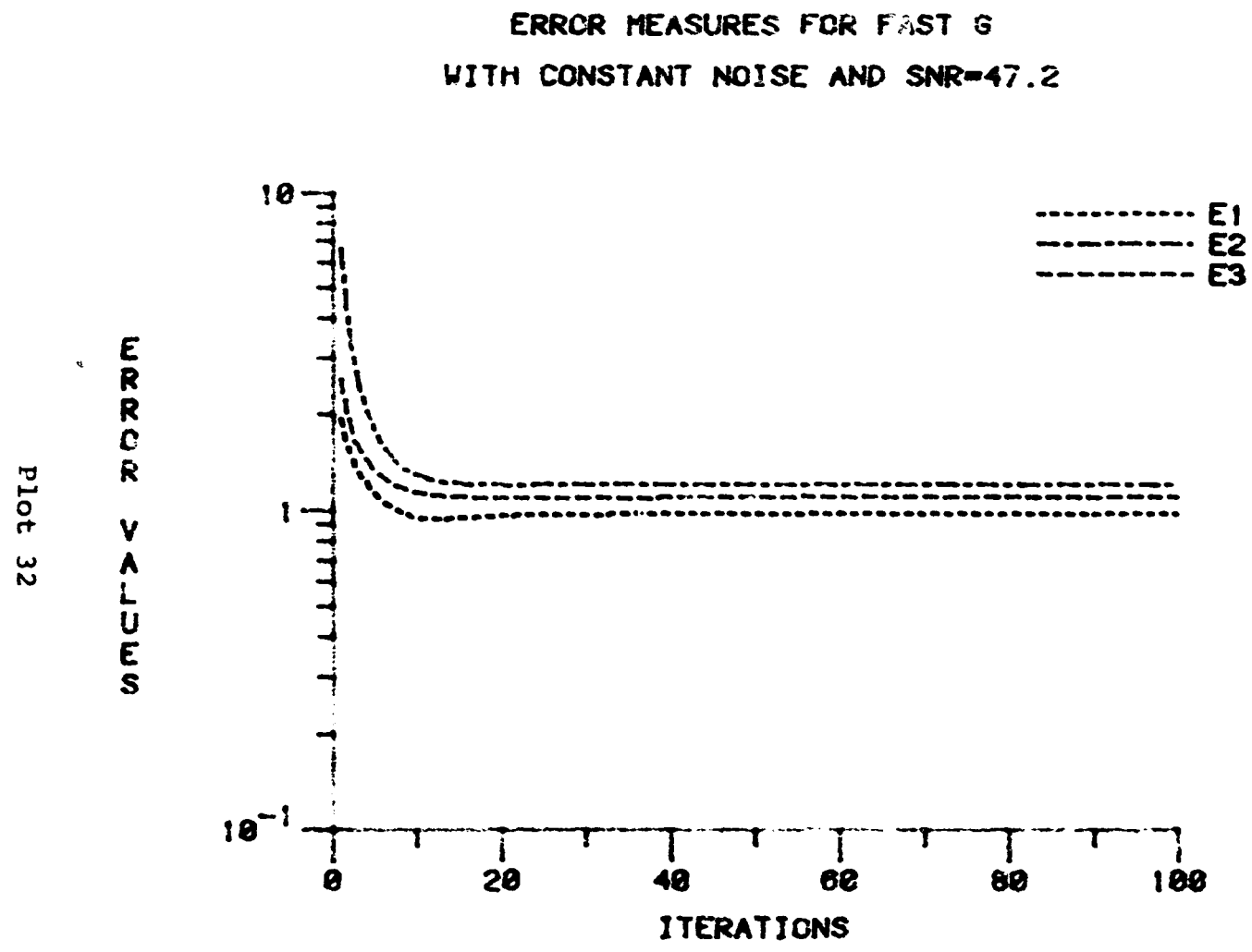
ERROR MEASURES FOR FAST G

WITH CONSTANT NOISE AND SNR=10.7

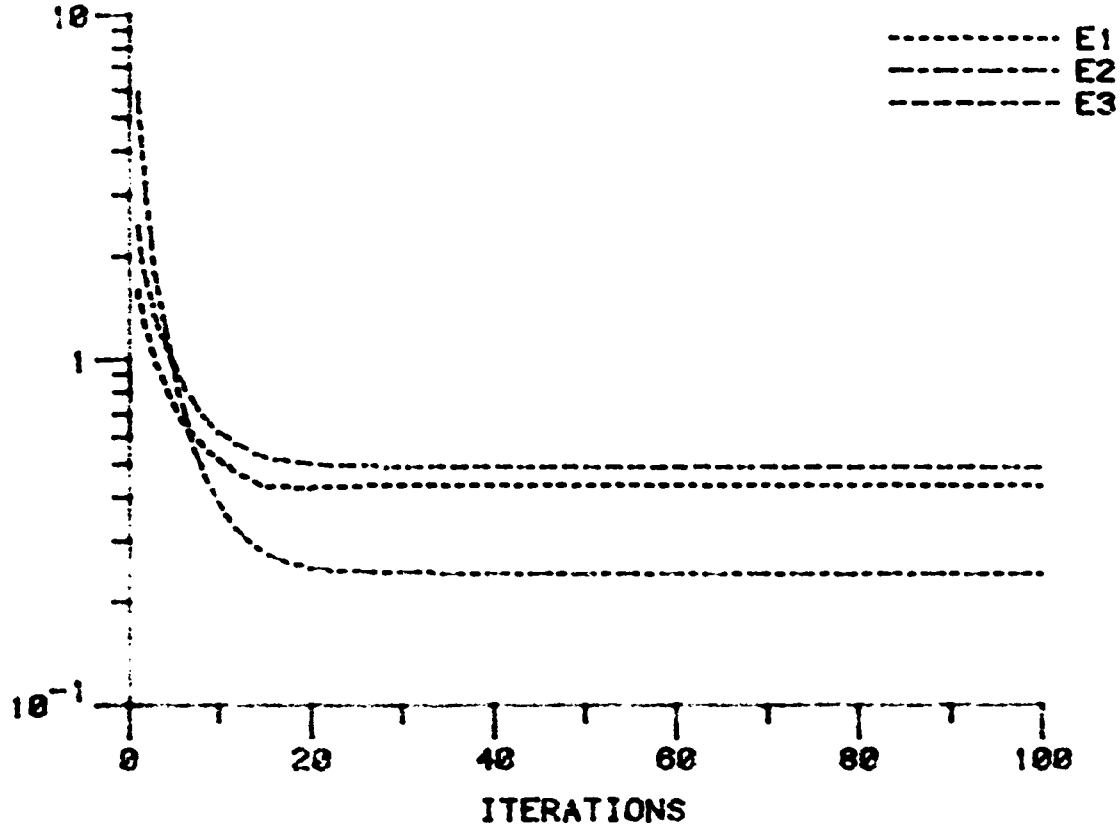
plot 30


ERROR MEASURES FOR FAST G
WITH CONSTANT NOISE AND SNR=23.6
E1
E2
E3
ERROR VALUES
10
9
8
7
6
5
4
3
2
1
0
20
40
60
80
100
ITERATIONS

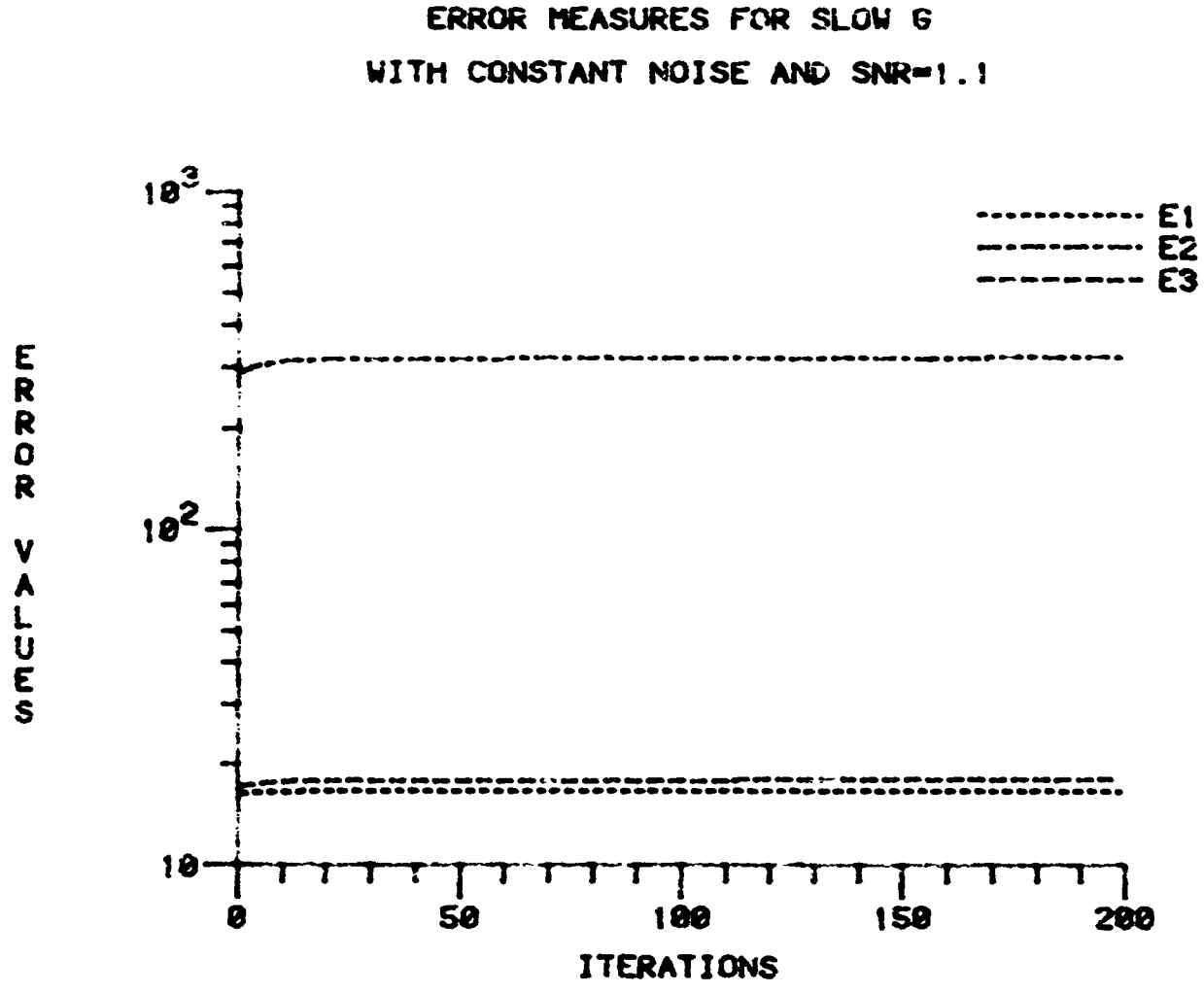
Plot 31


ERROR MEASURES FOR FAST G
WITH CONSTANT NOISE AND SNR=47.2
E1
E2
E3
10
1
10^{-1}
ERROR VALUES
0
20
40
60
80
100
ITERATIONS

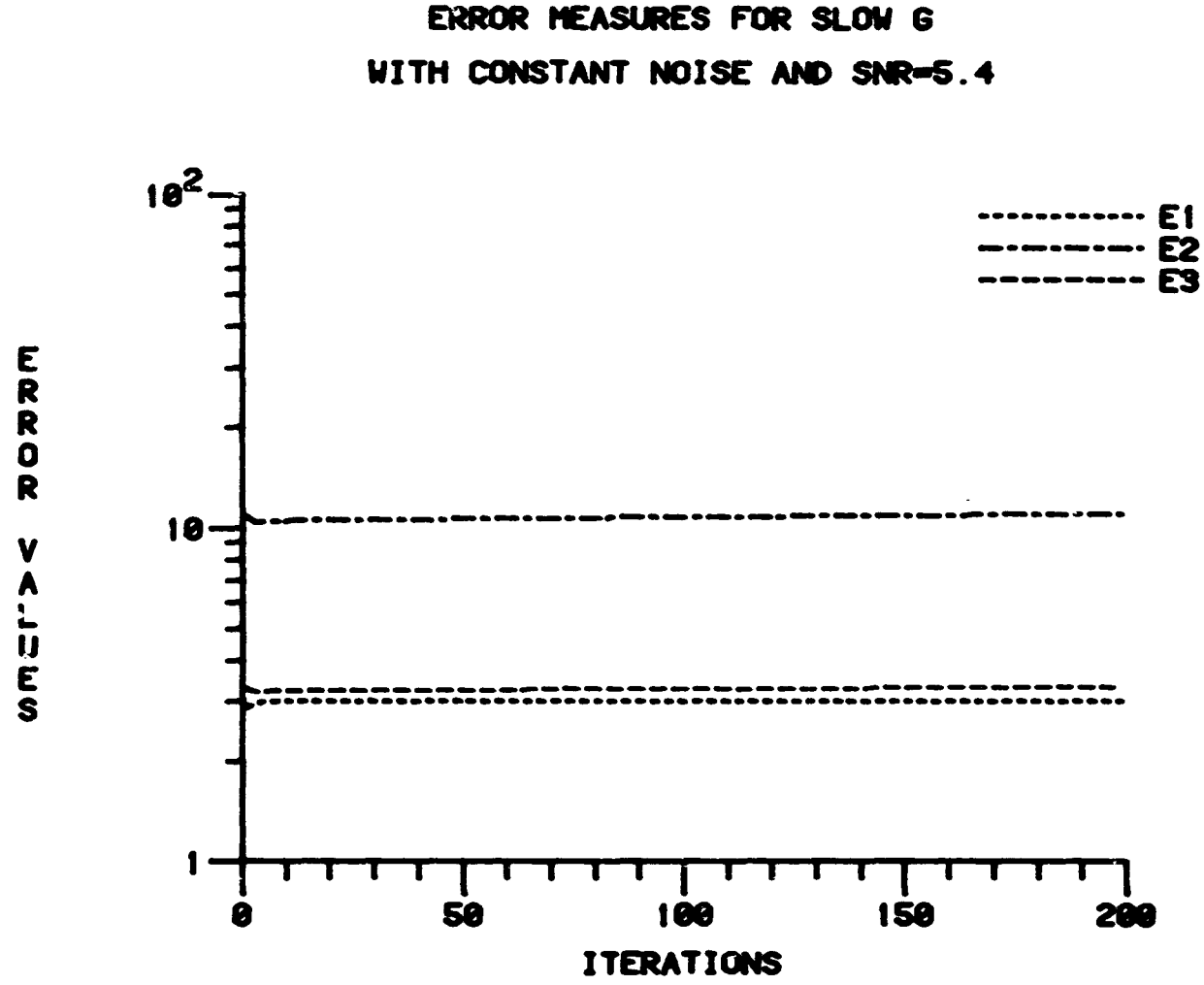
Plot 32


ERROR MEASURES FOR FAST G
WITH CONSTANT NOISE AND SNR=106
E1
E2
E3
10
1
10^{-1}
ERROR VALUES
0
20
40
60
80
100
ITERATIONS

Plot 33


ERROR MEASURES FOR SLOW G
WITH CONSTANT NOISE AND SNR=1.1
E1
E2
E3
10^3
10^2
10
ERROR VALUES
0
50
100
150
200
ITERATIONS

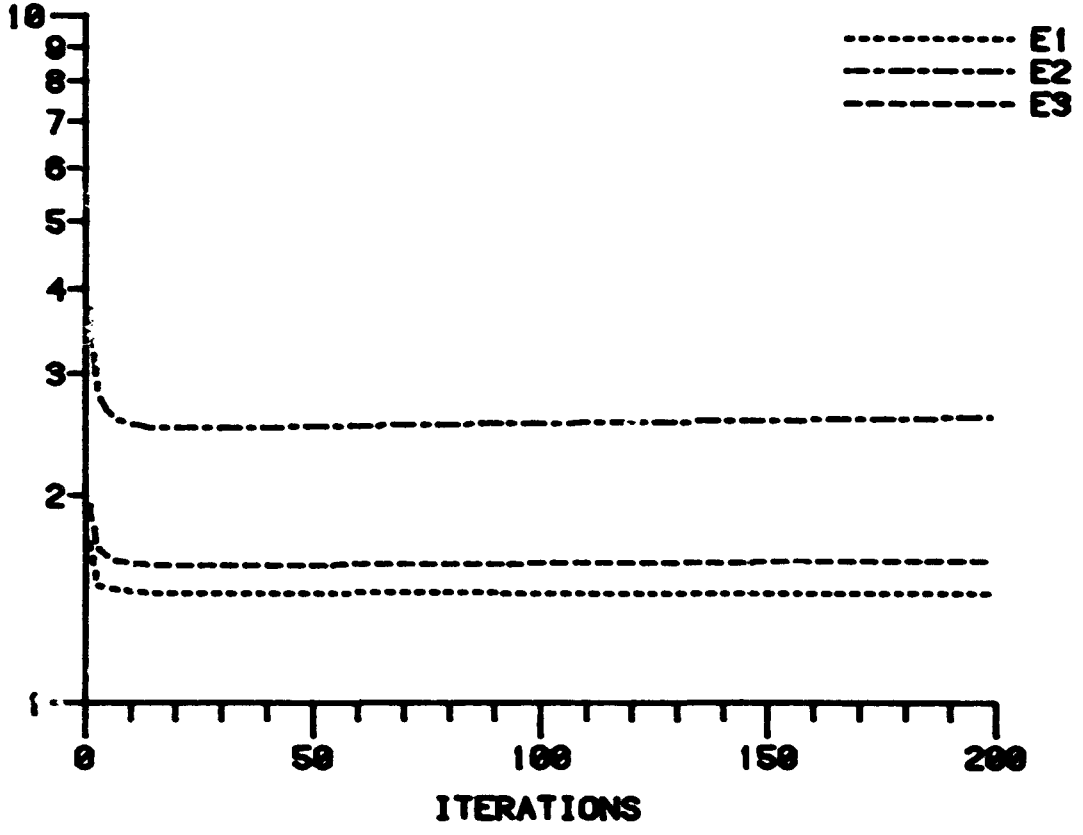
Plot 34


ERROR MEASURES FOR SLOW G
WITH CONSTANT NOISE AND SNR=5.4
10^2
10
1
ERROR VALUES
E1
E2
E3
0
50
100
150
200
ITERATIONS

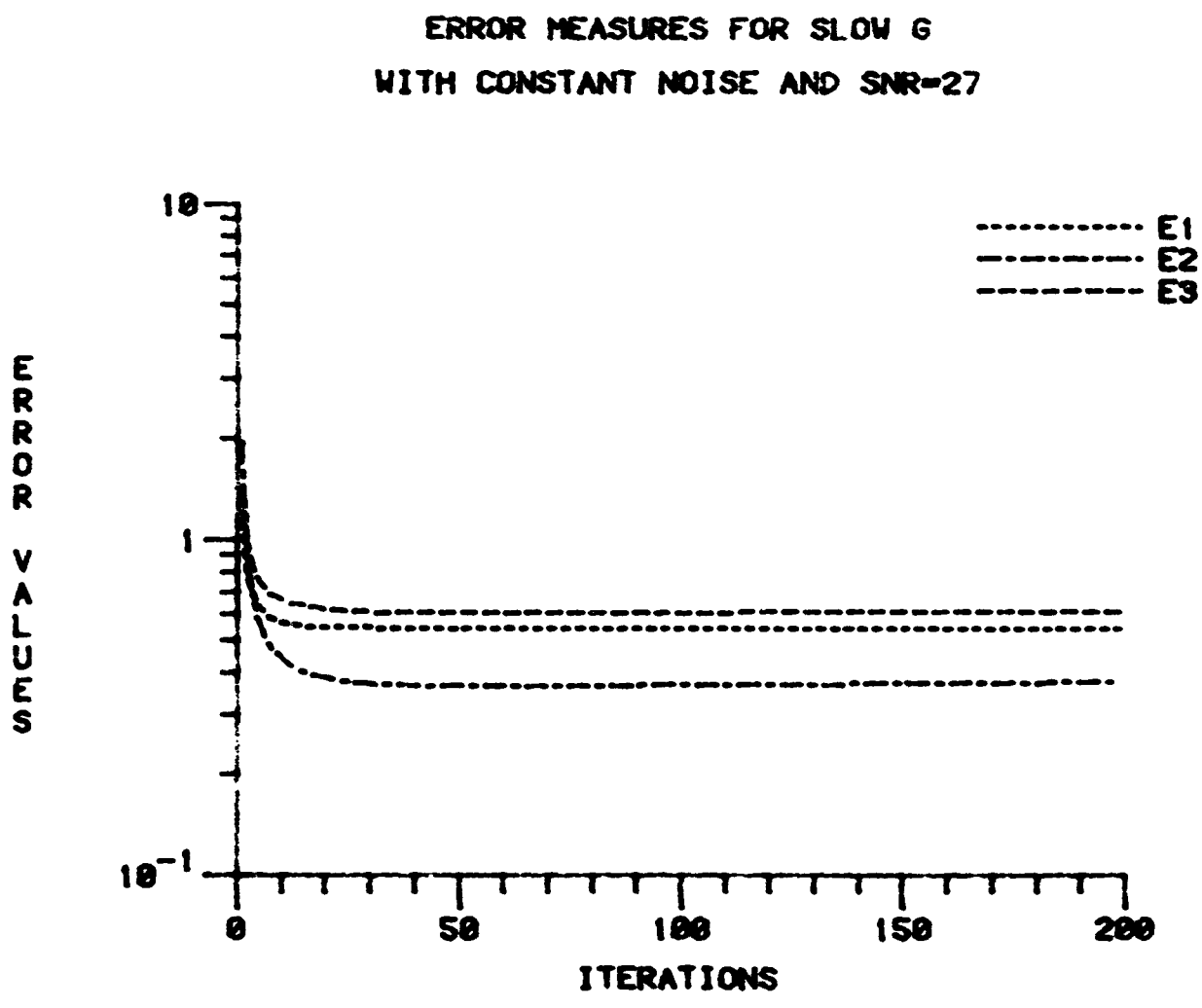
Plot 35

ERROR MEASURES FOR SLOW G

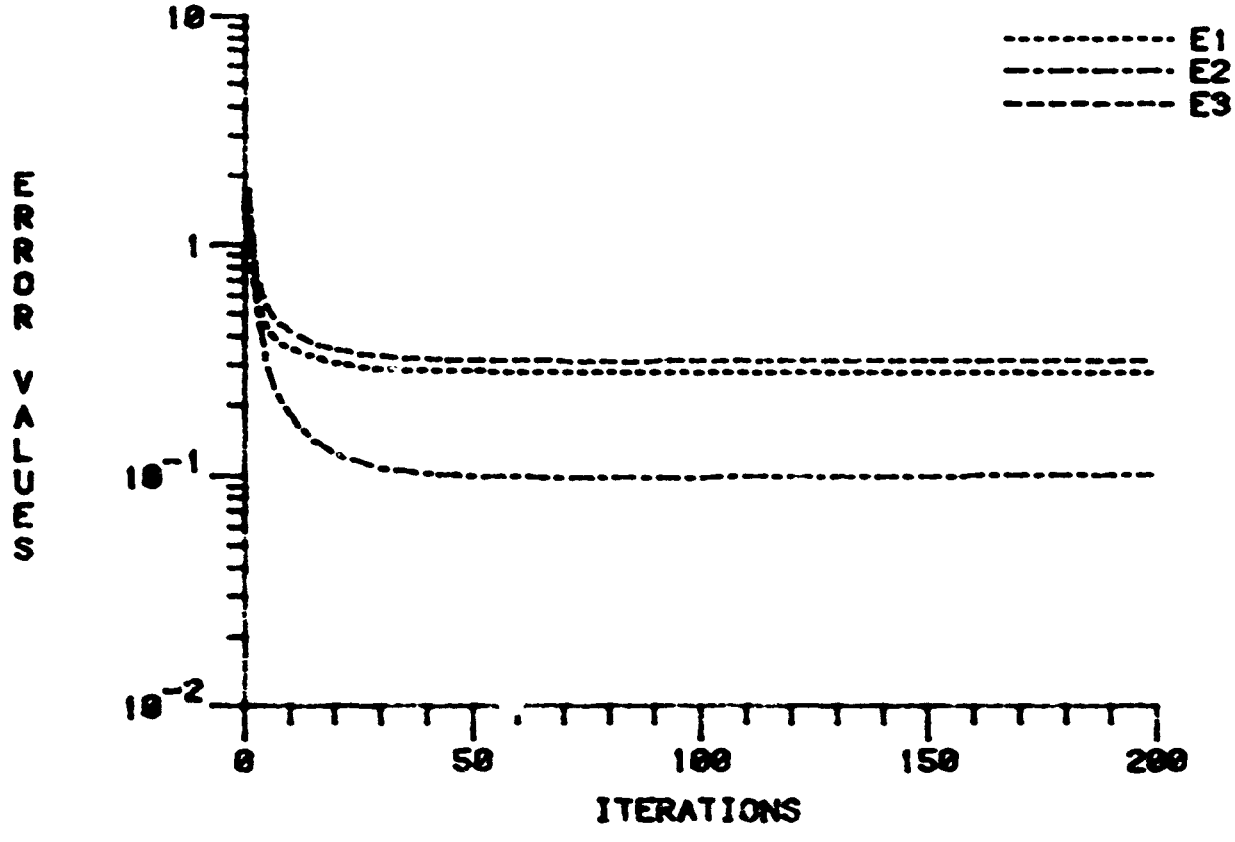
WITH CONSTANT NOISE AND SNR=10.5



Plot 36


ERROR MEASURES FOR SLOW G
WITH CONSTANT NOISE AND SNR=27
E1
E2
E3
10
1
10^{-1}
ERROR VALUES
0
50
100
150
200
ITERATIONS

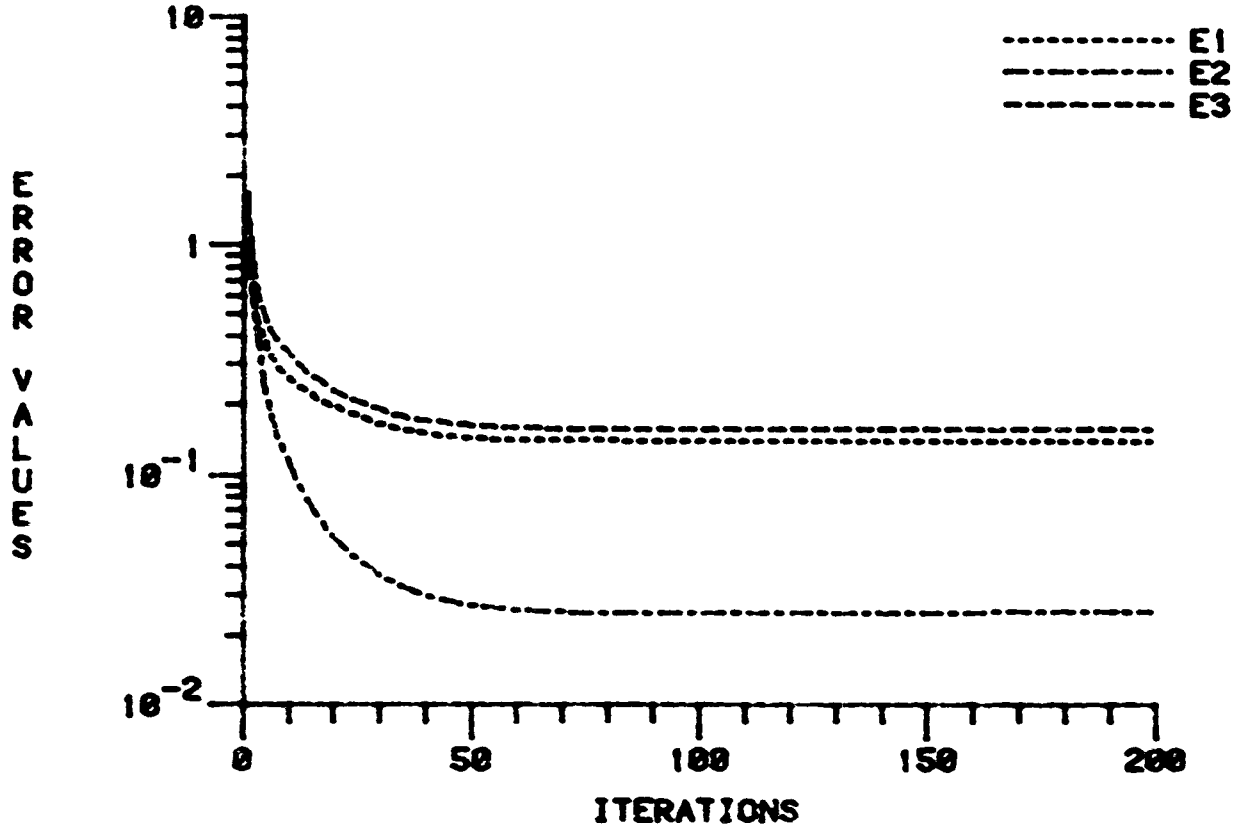
Plot 37


ERROR MEASURES FOR SLOW G
WITH CONSTANT NOISE AND SNR=52
E1
E2
E3
10
1
10^-1
10^-2
ERROR VALUES
0
50
100
150
200
ITERATIONS

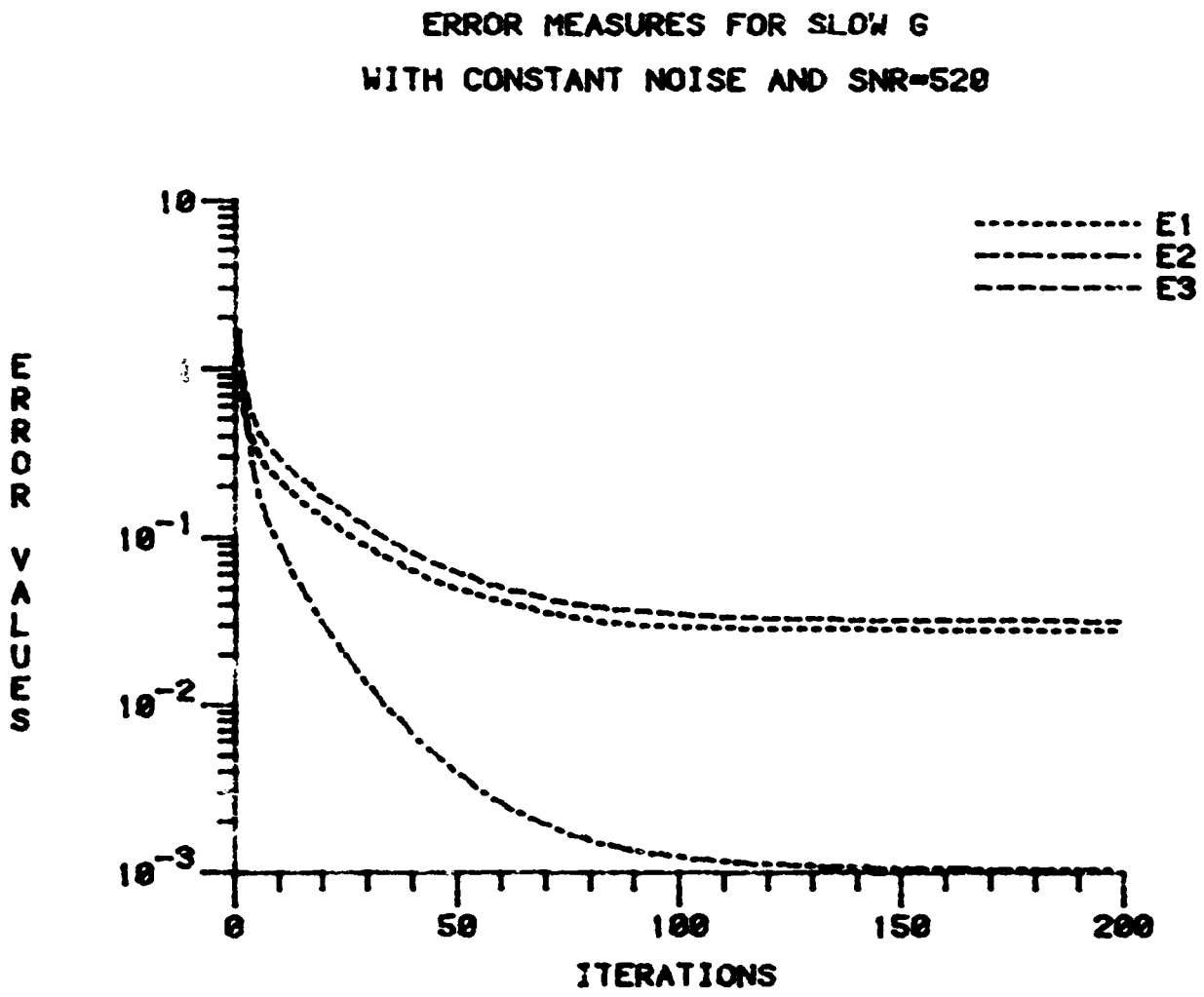
Plot 38

ERROR MEASURES FOR SLOW G

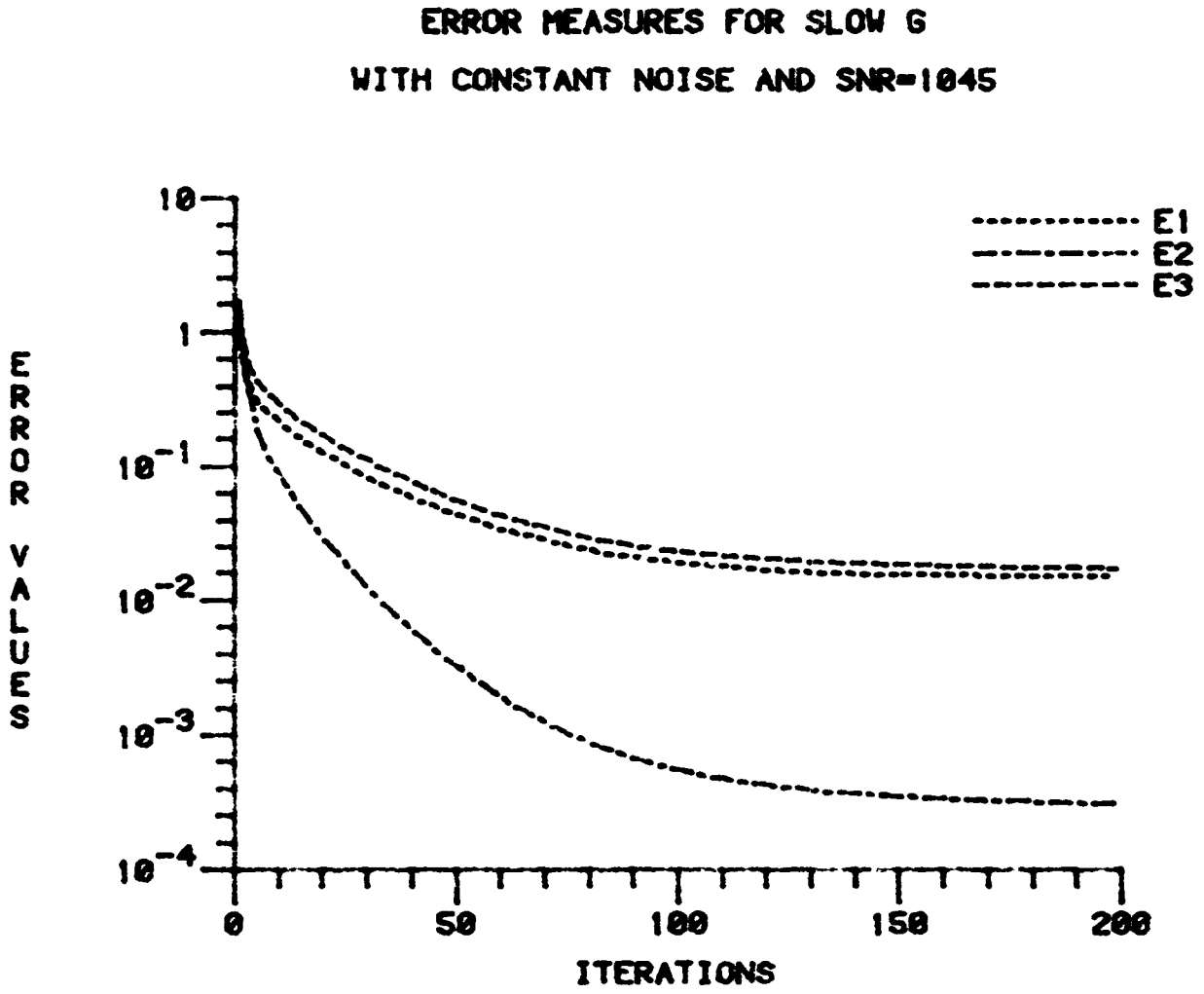
WITH CONSTANT NOISE AND SNR=104



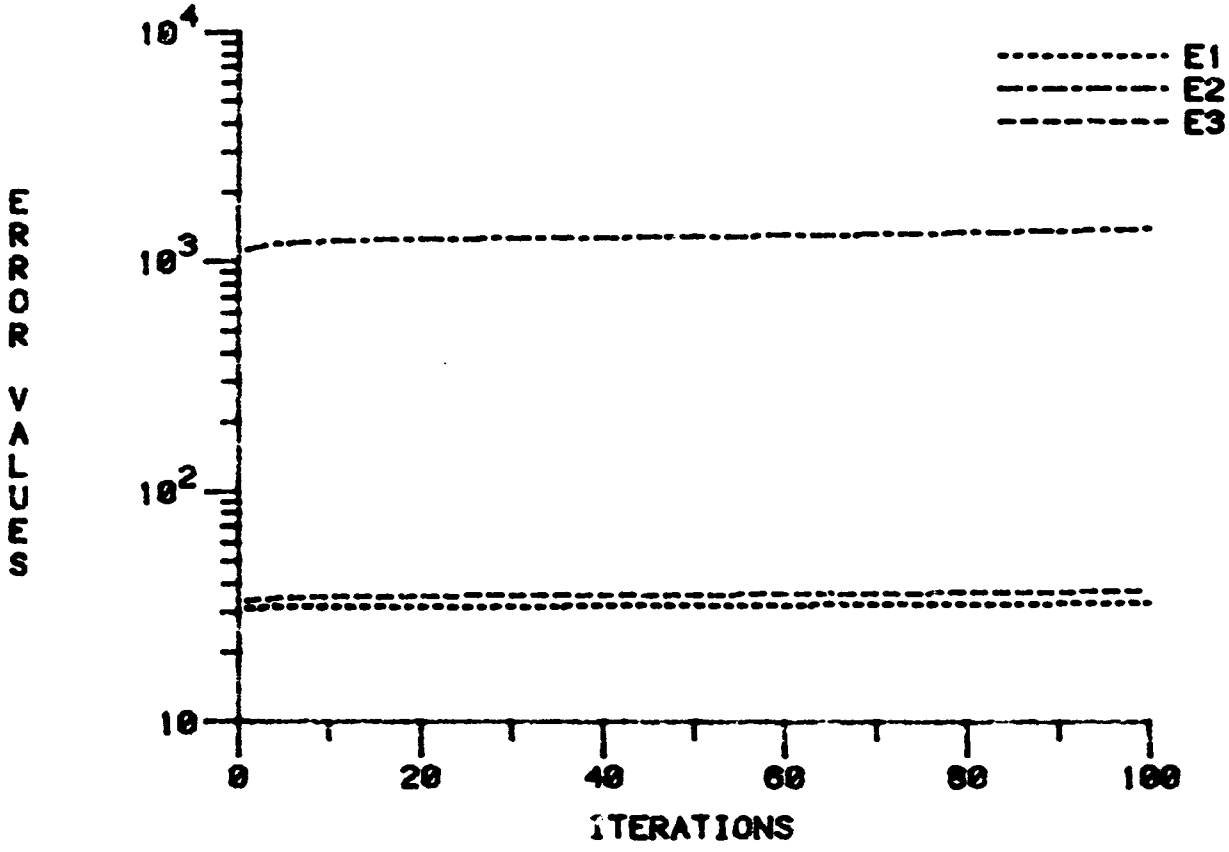
Plot 39


ERROR MEASURES FOR SLOW G
WITH CONSTANT NOISE AND SNR=520
E1
E2
E3
ERROR VALUES
10
10^{-1}
10^{-2}
10^{-3}
0
50
100
150
200
ITERATIONS

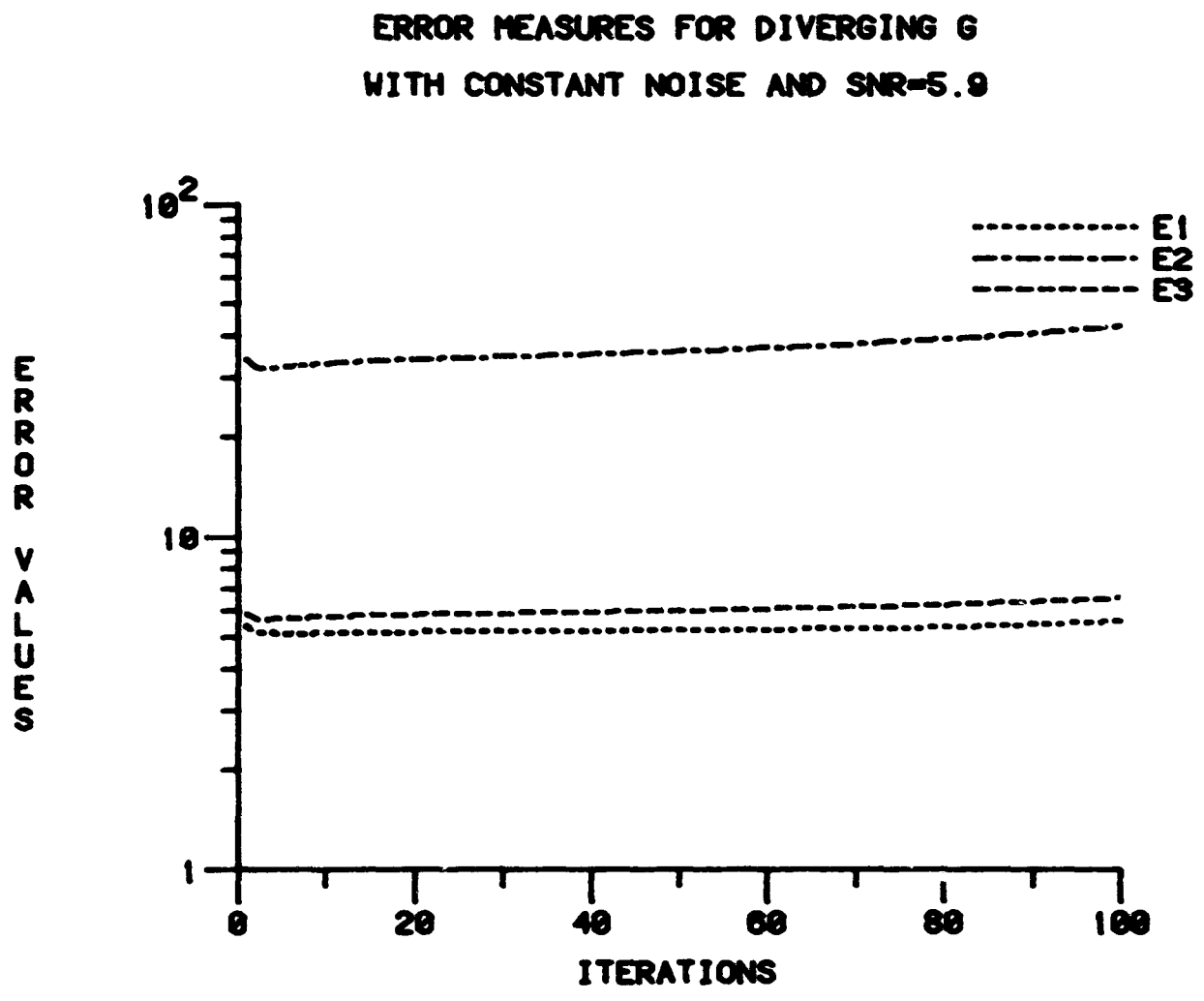
Plot 40


ERROR MEASURES FOR SLOW G
WITH CONSTANT NOISE AND SNR=1045
E1
E2
E3
ERROR VALUES
10
1
10-1
10-2
10-3
10-4
0
50
100
150
200
ITERATIONS

Plot 41

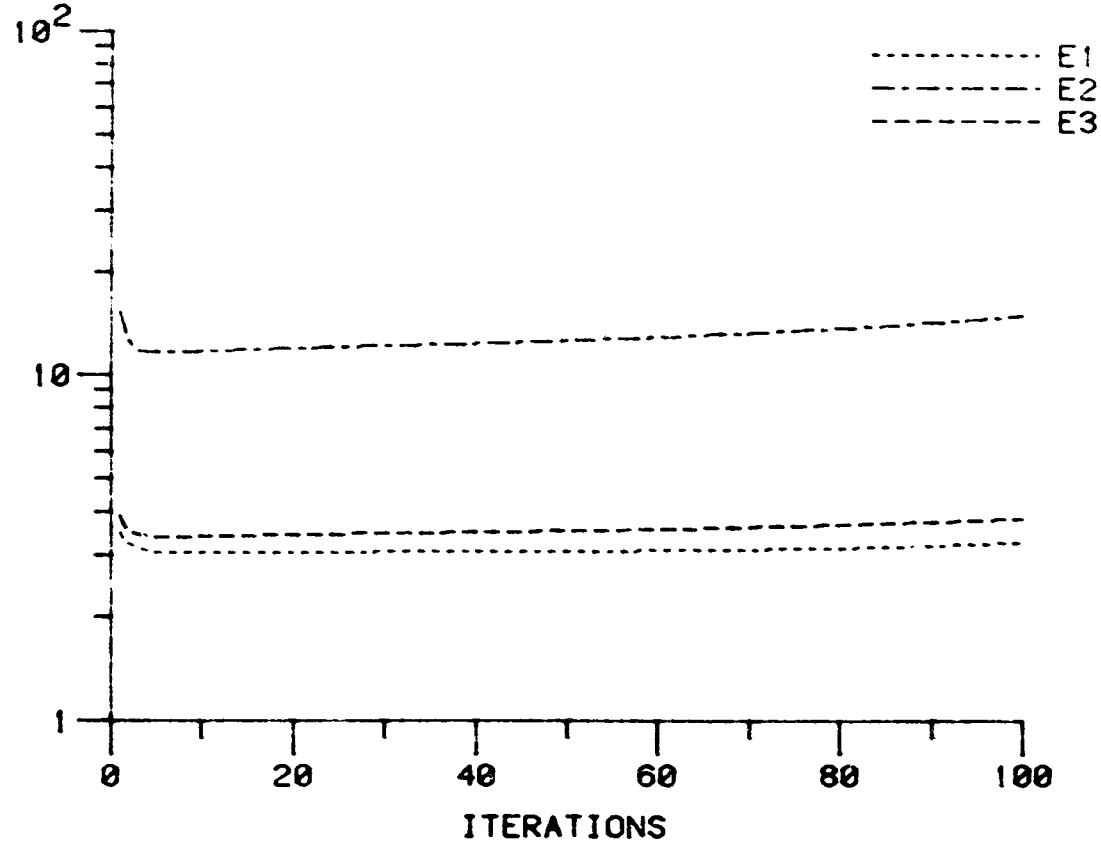

ERROR MEASURES FOR DIVERGING G
WITH CONSTANT NOISE AND SNR=1.02
E1
E2
E3
10^4
10^3
10^2
10
ERROR VALUES
0
20
40
60
80
100
ITERATIONS

Plot 42

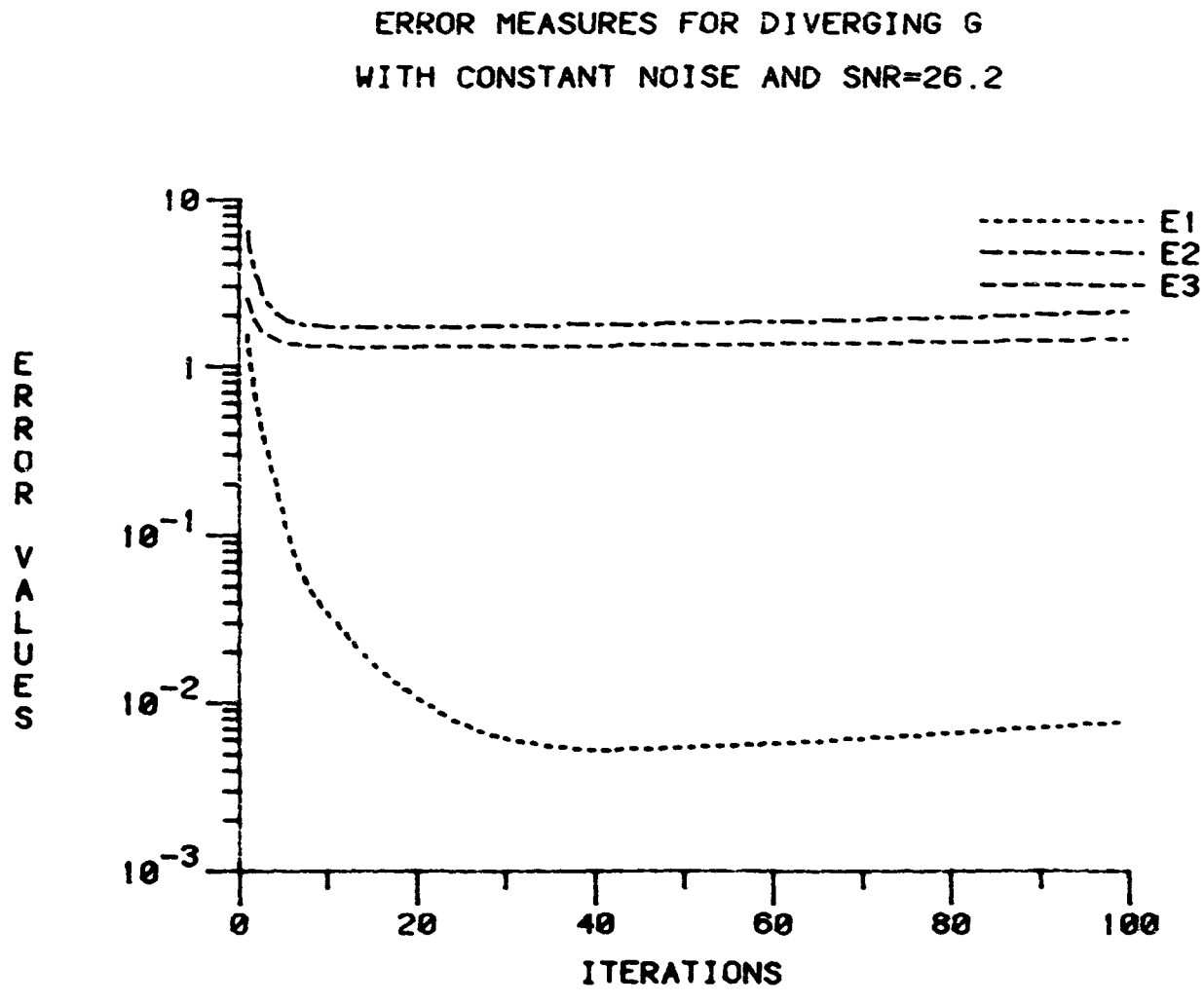

ERROR MEASURES FOR DIVERGING G
WITH CONSTANT NOISE AND SNR=5.9
10^2
10
1
ERROR VALUES
E1
E2
E3
0
20
40
60
80
100
ITERATIONS

Plot 43

ERROR MEASURES FOR DIVERGING G
WITH CONSTANT NOISE AND SNR=9.9

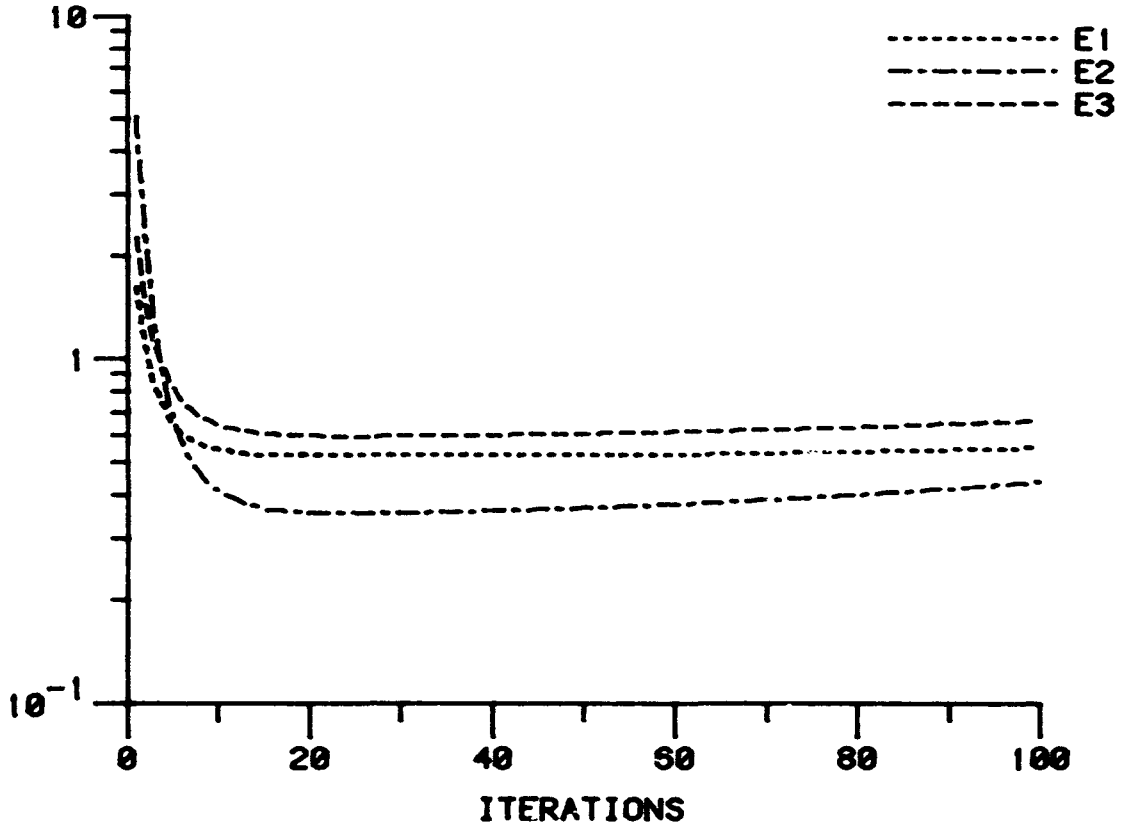


Plot 44


ERROR MEASURES FOR DIVERGING G
WITH CONSTANT NOISE AND SNR=26.2
E1
E2
E3
10
1
10^-1
10^-2
10^-3
ERROR VALUES
0
20
40
60
80
100
ITERATIONS

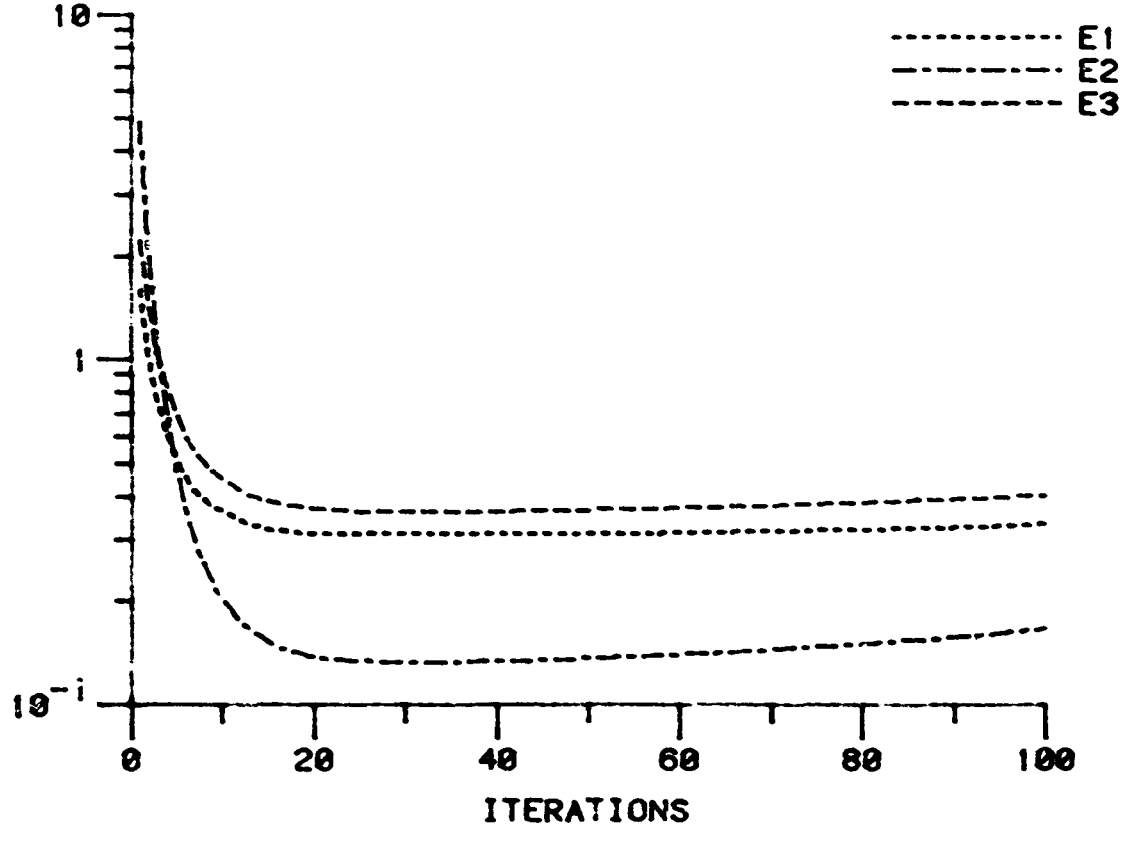
Plot 45

ERROR MEASURES FOR DIVERGING G
WITH CONSTANT NOISE AND SNR=58.6

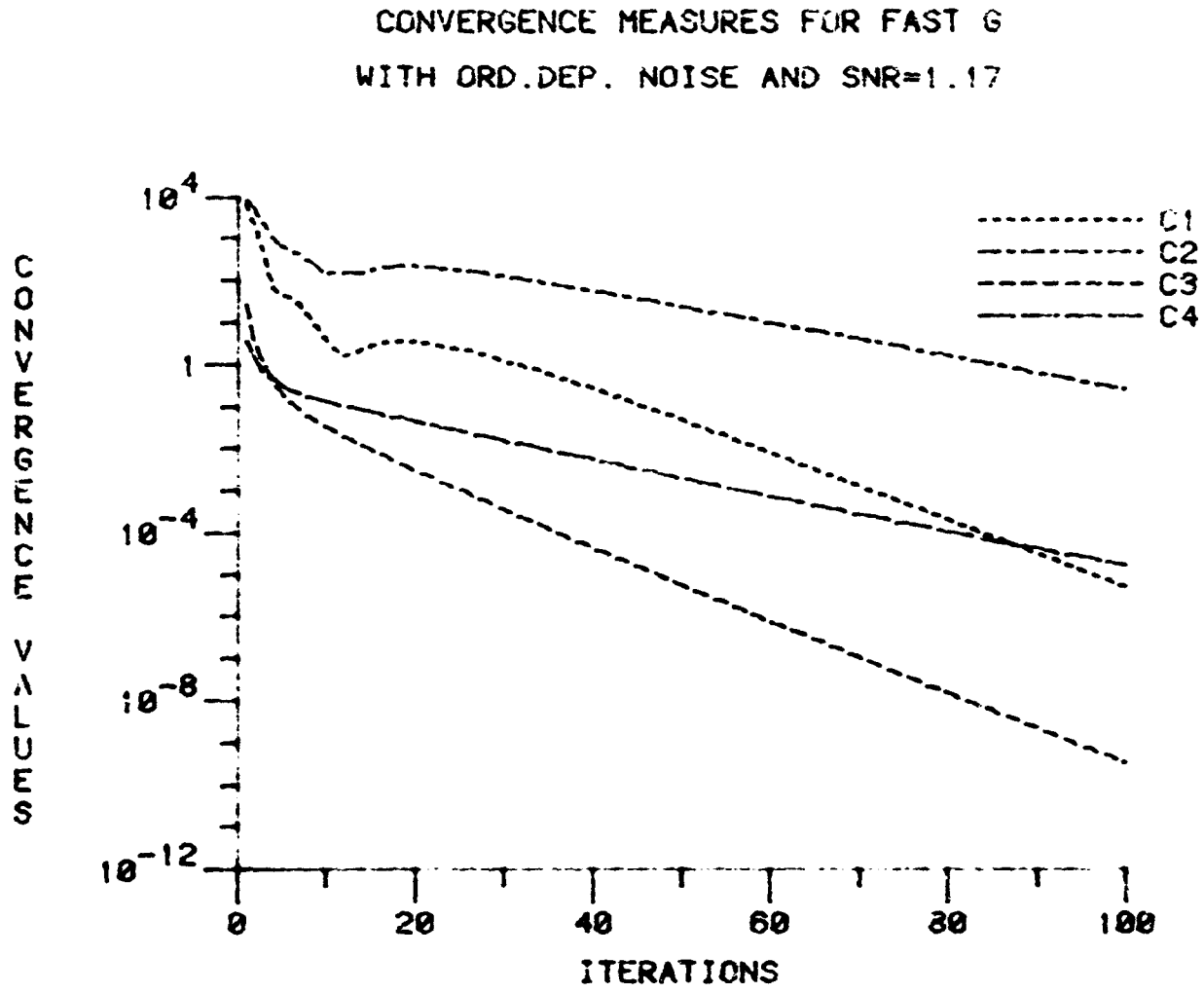


Plot 46

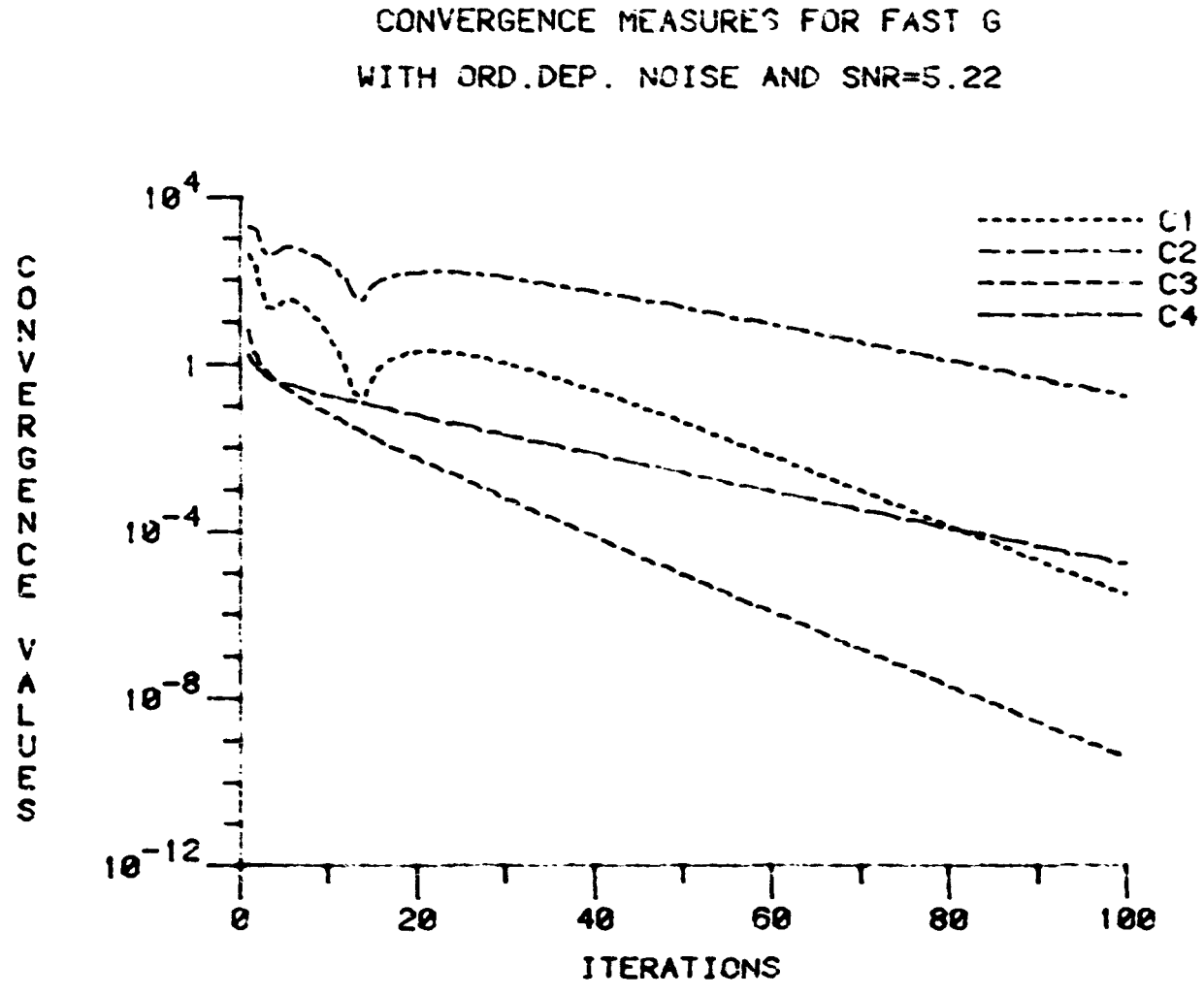
ERROR MEASURES FOR DIVERGING G
WITH CONSTANT NOISE AND SNR=99



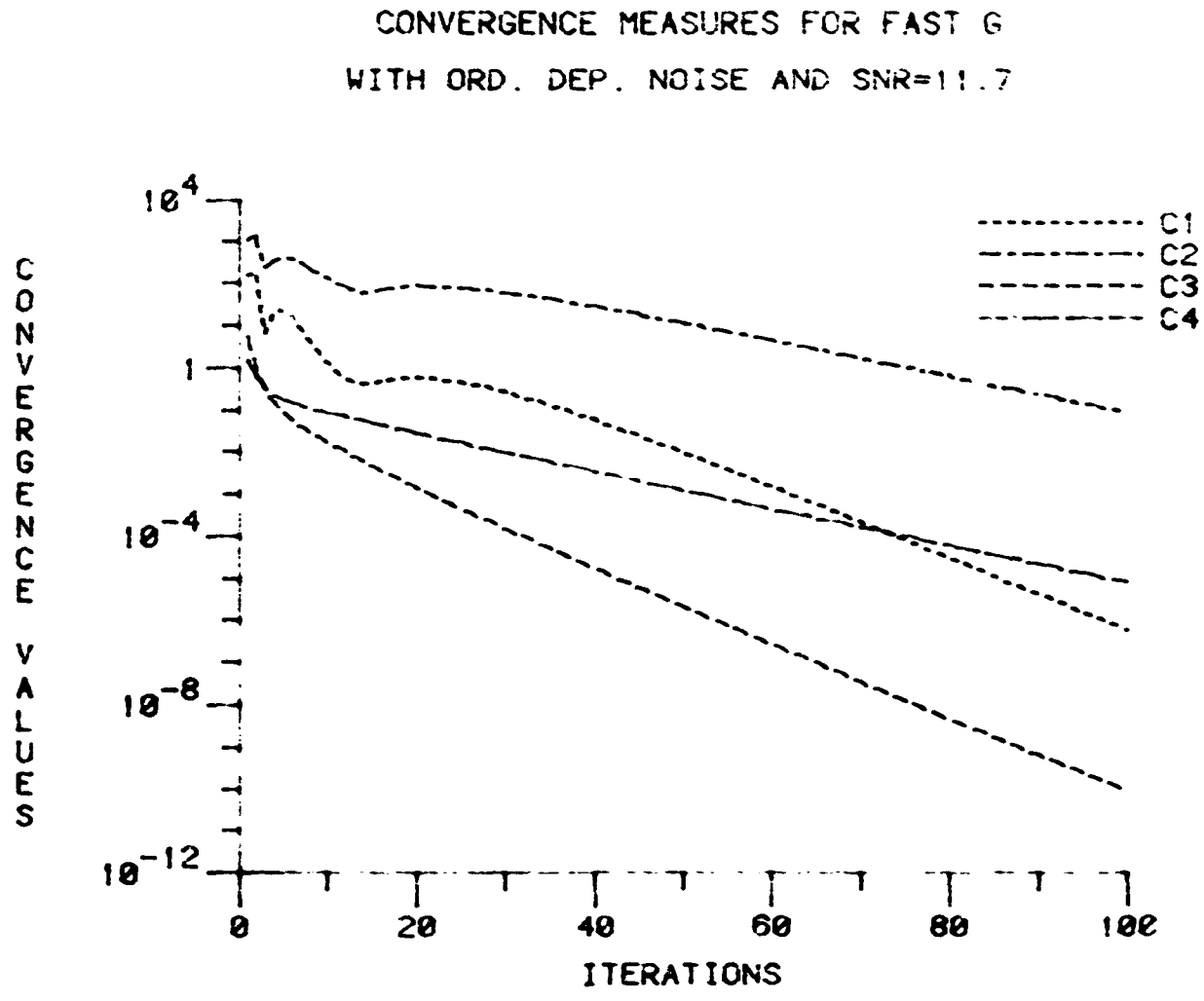
Plot 47


CONVERGENCE MEASURES FOR FAST G
WITH ORD.DEP. NOISE AND SNR=1.17
C1
C2
C3
C4
CONVERGENCE VALUES
10^{4}
1
10^{-4}
10^{-8}
10^{-12}
0
20
40
60
80
100
ITERATIONS

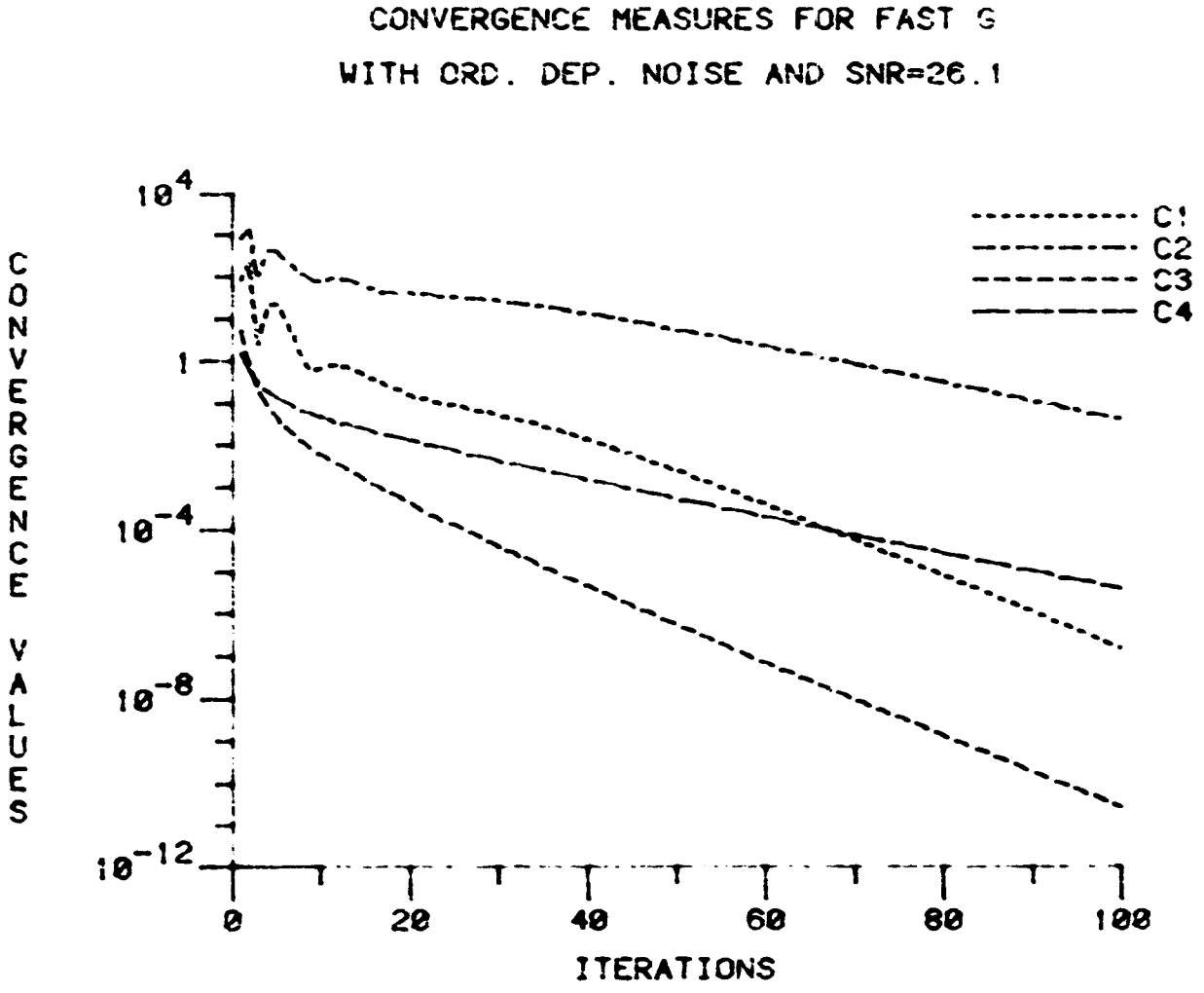
Plot 48


CONVERGENCE MEASURES FOR FAST G
WITH ORD.DEP. NOISE AND SNR=5.22
C1
C2
C3
C4
CONVERGENCE VALUES
10^{4}
1
10^{-4}
10^{-8}
10^{-12}
0
20
40
60
80
100
ITERATIONS

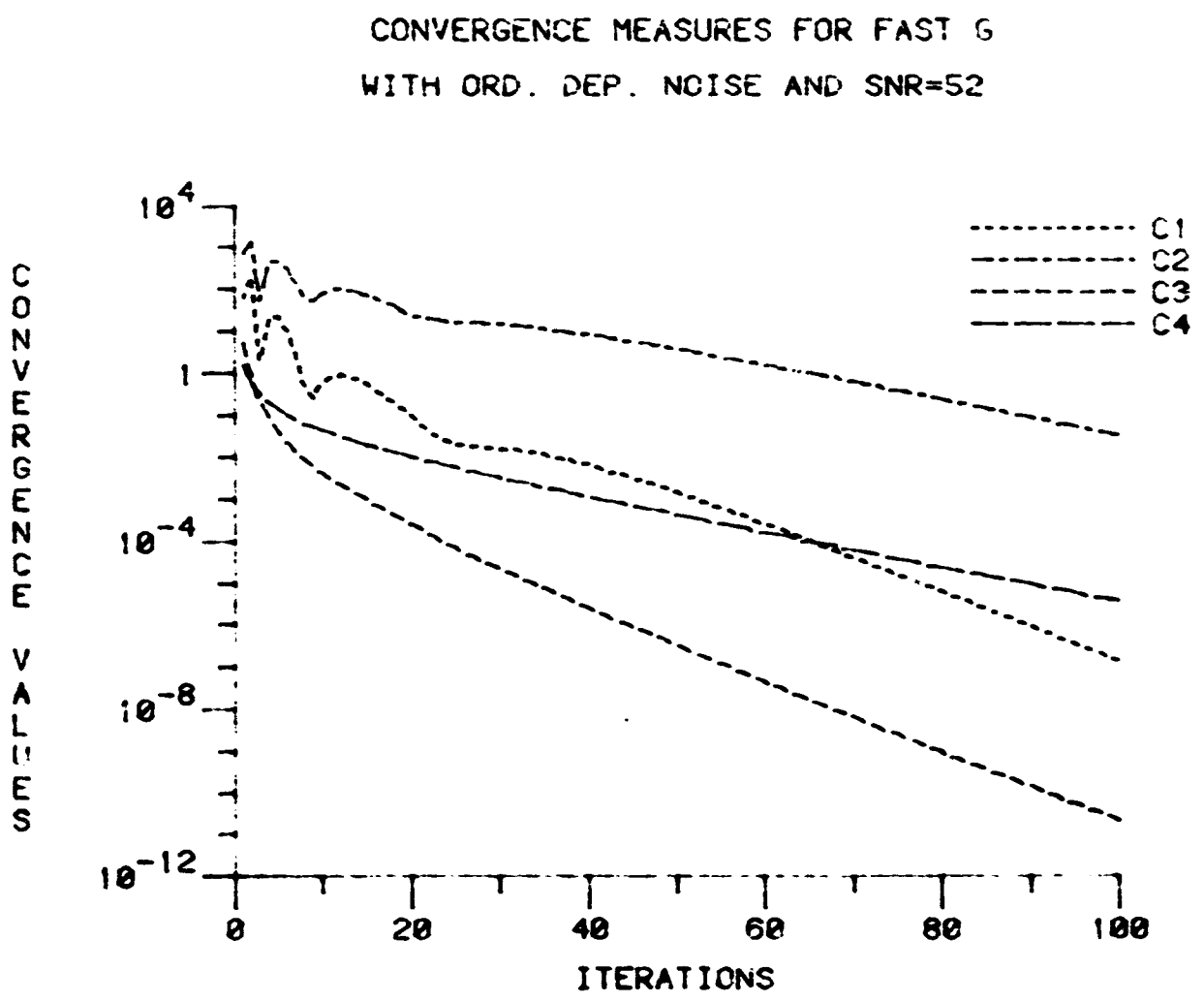
Plot 49


CONVERGENCE MEASURES FOR FAST G
WITH ORD. DEP. NOISE AND SNR=11.7
C1
C2
C3
C4
CONVERGENCE VALUES
10^{4}
1
10^{-4}
10^{-8}
10^{-12}
0
20
40
60
80
100
ITERATIONS

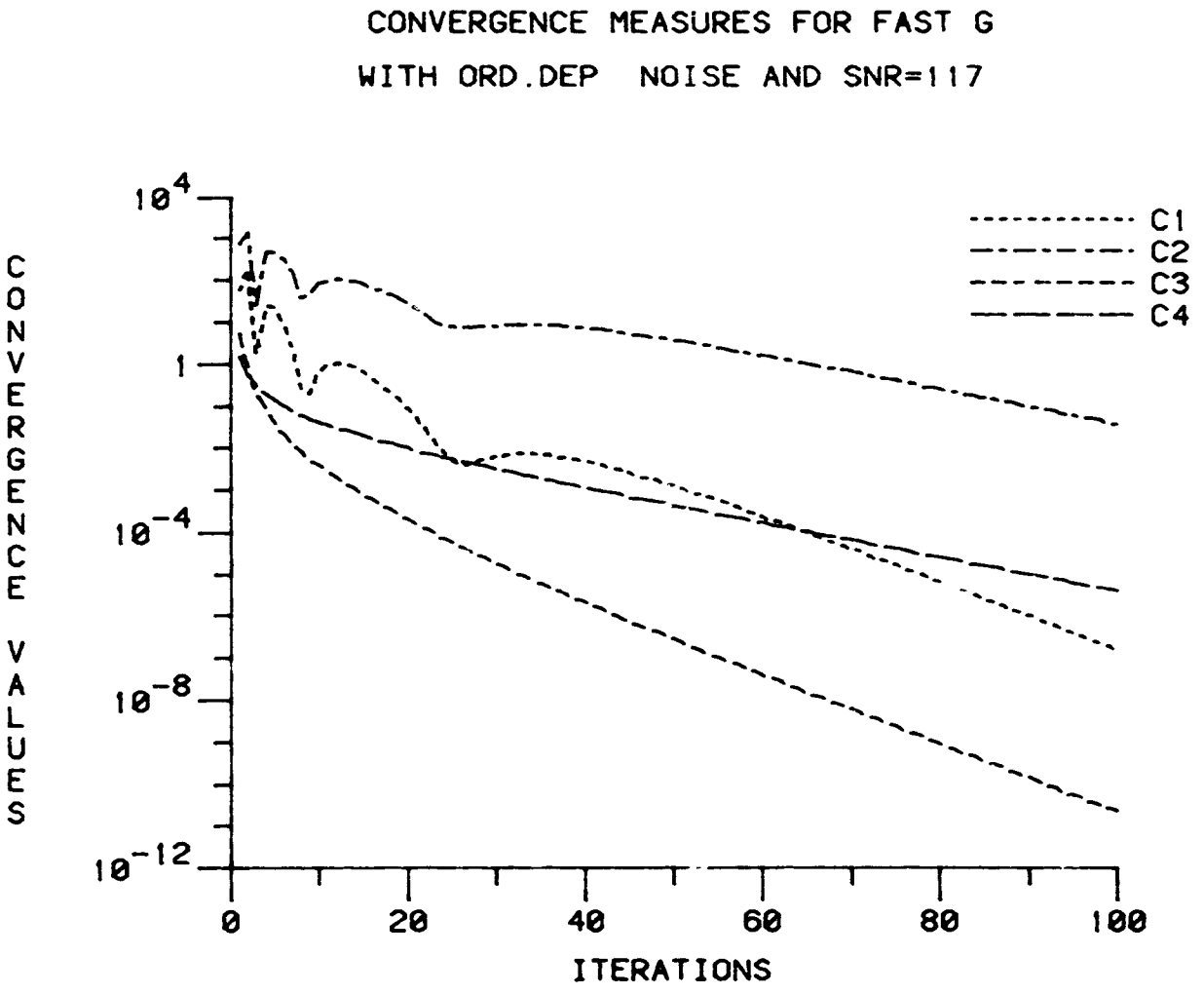
Plot 50


CONVERGENCE MEASURES FOR FAST G
WITH ORD. DEP. NOISE AND SNR=26.1
C1
C2
C3
C4
CONVERGENCE VALUES
10^{4}
1
10^{-4}
10^{-8}
10^{-12}
0
20
40
60
80
100
ITERATIONS

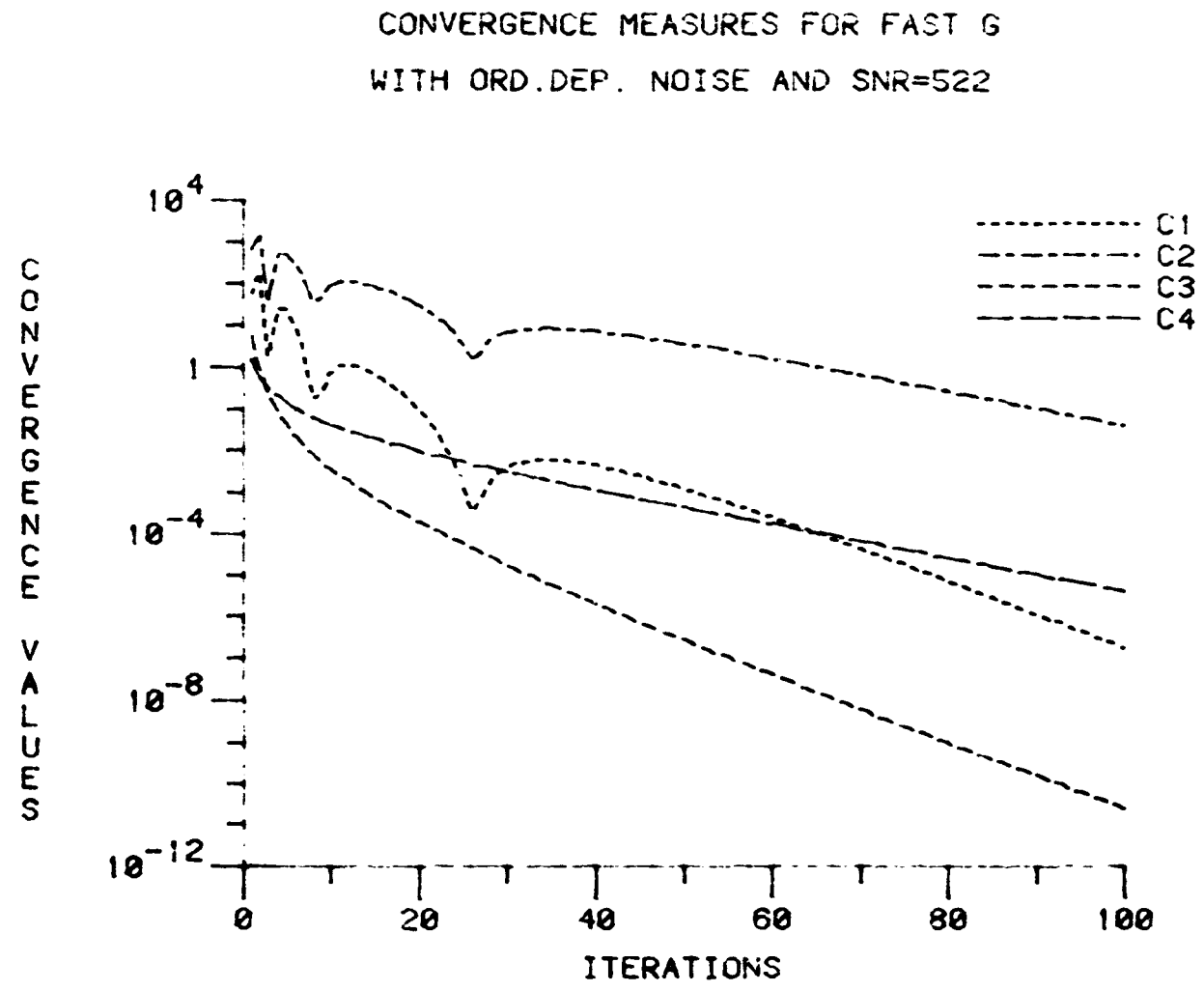
Plot 51


CONVERGENCE MEASURES FOR FAST G
WITH ORD. DEP. NOISE AND SNR=52
C1
C2
C3
C4
CONVERGENCE VALUES
10^{4}
1
10^{-4}
10^{-8}
10^{-12}
0
20
40
60
80
100
ITERATIONS

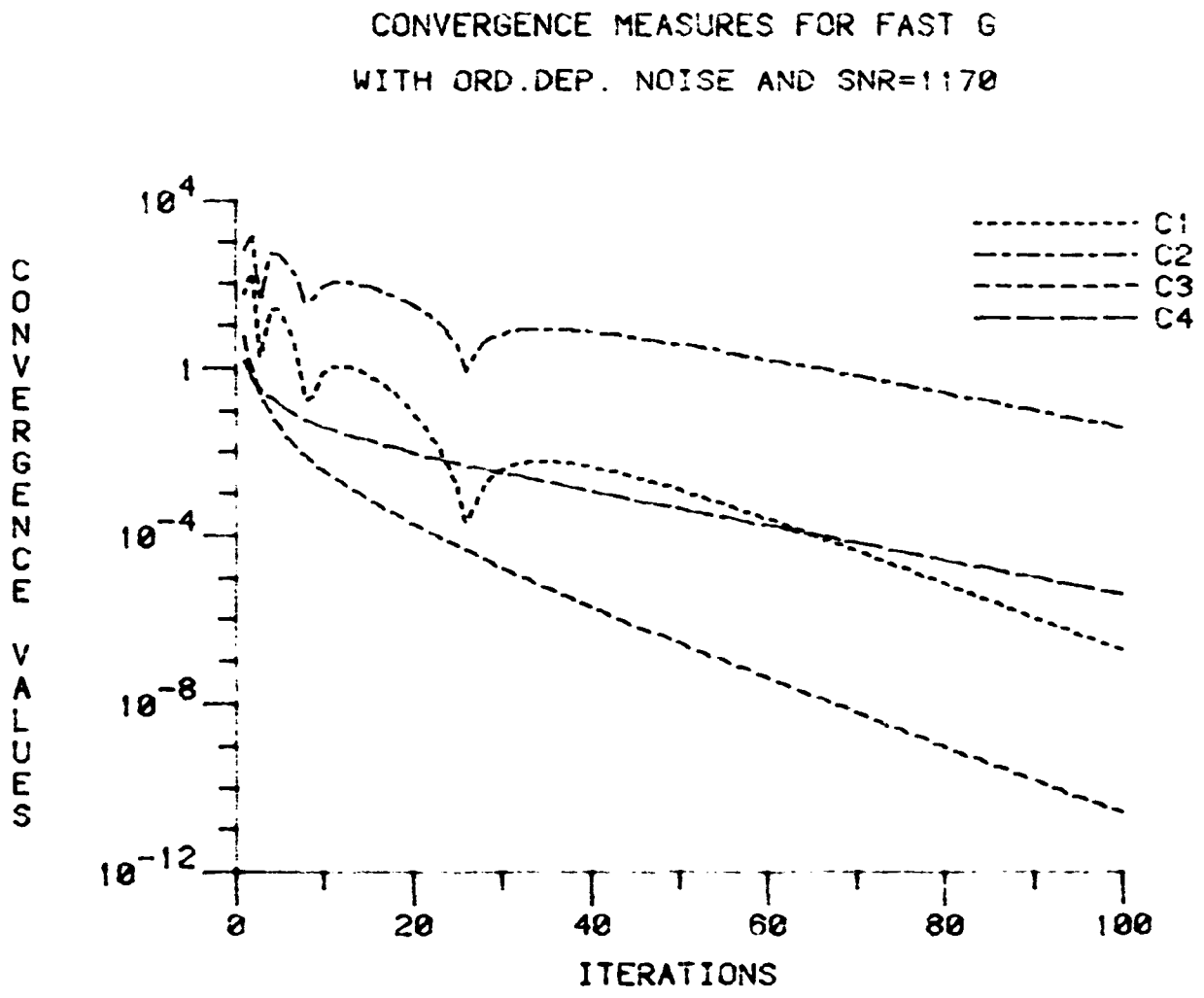
Plot 52


CONVERGENCE MEASURES FOR FAST G
WITH ORD.DEP NOISE AND SNR=117
C1
C2
C3
C4
CONVERGENCE VALUES
10^{4}
1
10^{-4}
10^{-8}
10^{-12}
0
20
40
60
80
100
ITERATIONS

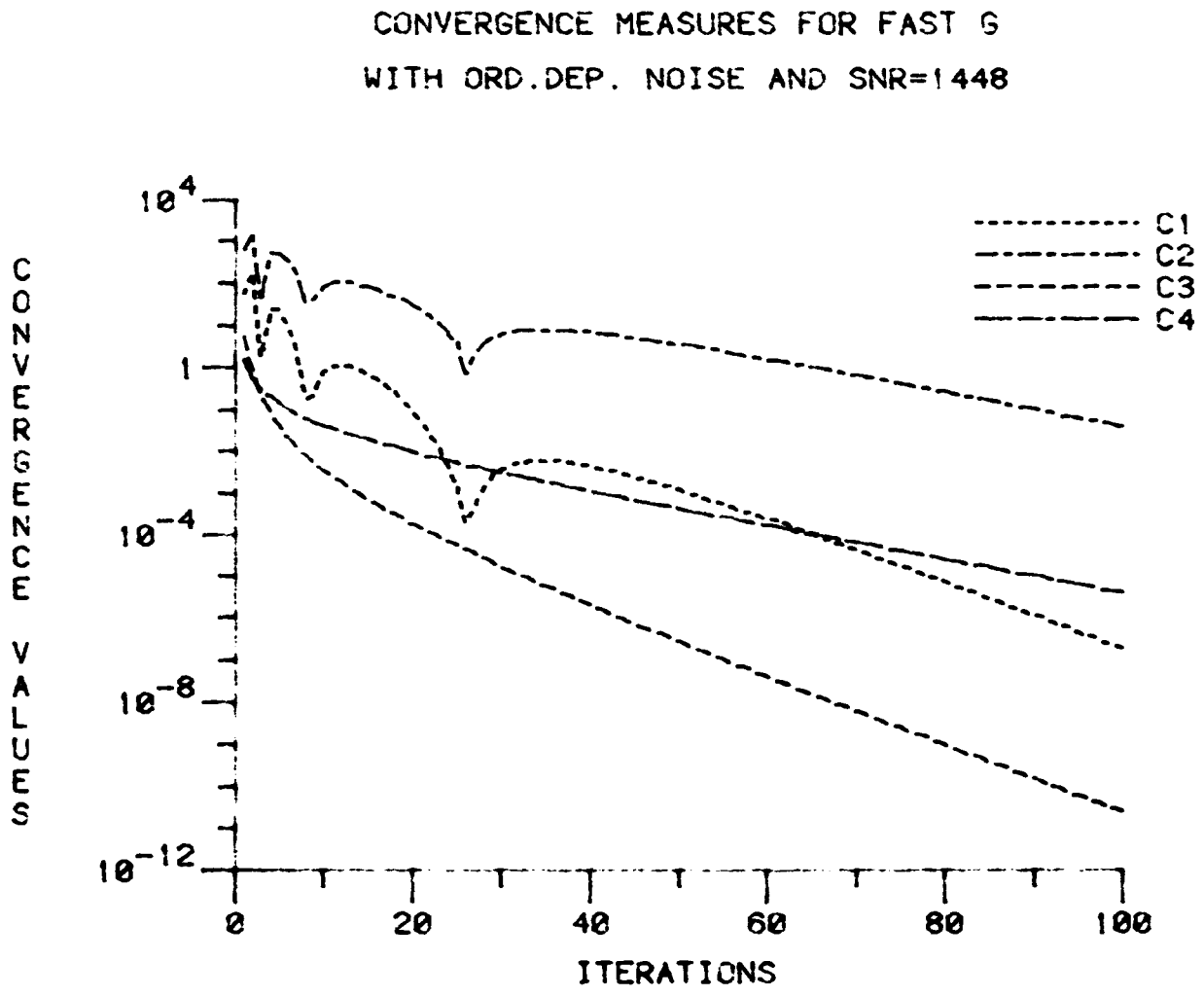
Plot 53


CONVERGENCE MEASURES FOR FAST G
WITH ORD.DEP. NOISE AND SNR=522
C1
C2
C3
C4
CONVERGENCE VALUES
10^{4}
1
10^{-4}
10^{-8}
10^{-12}
0
20
40
60
80
100
ITERATIONS

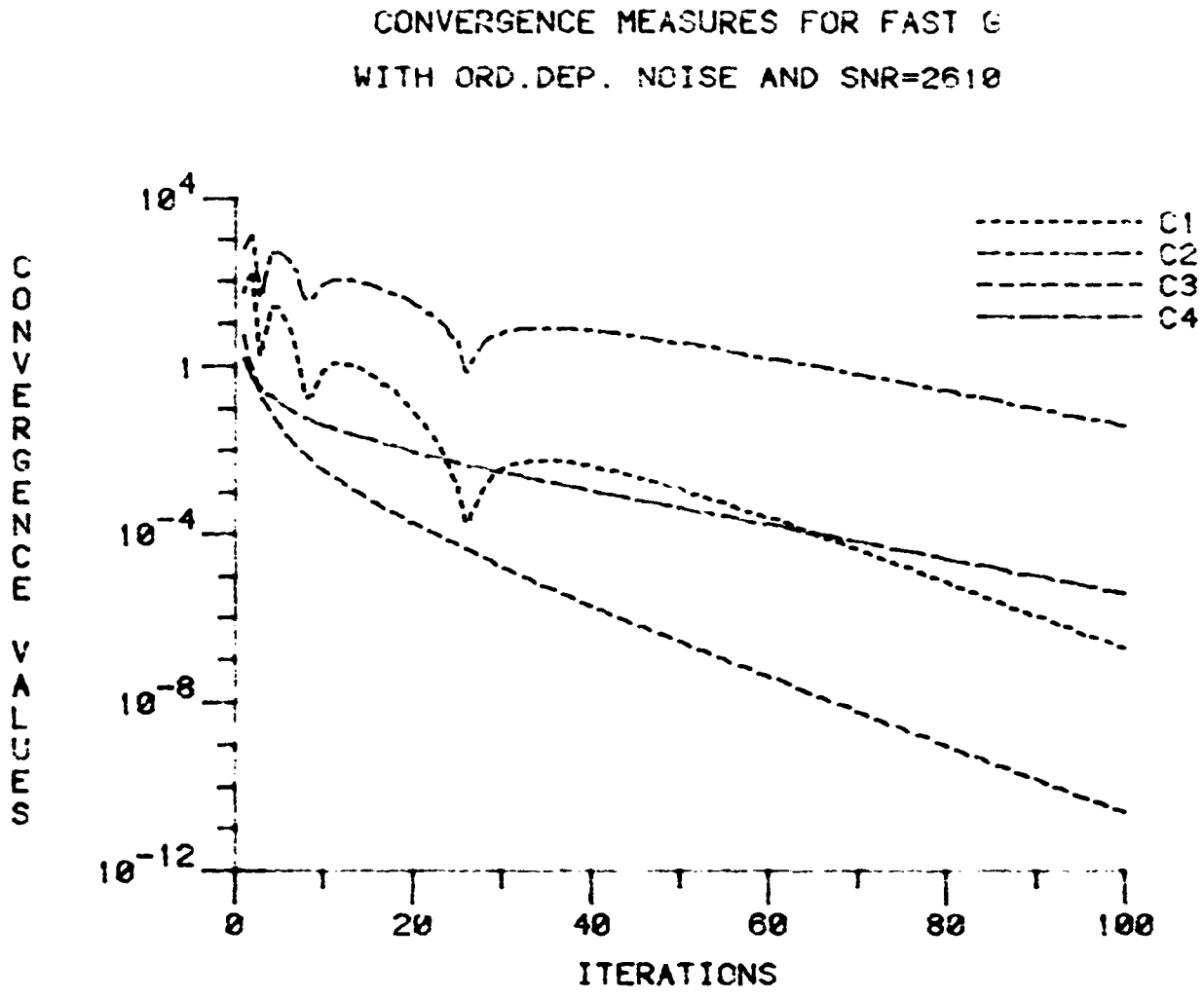
Plot 54


CONVERGENCE MEASURES FOR FAST G
WITH ORD.DEP. NOISE AND SNR=1:70
C1
C2
C3
C4
CONVERGENCE VALUES
10^{4}
1
10^{-4}
10^{-8}
10^{-12}
0
20
40
60
80
100
ITERATIONS

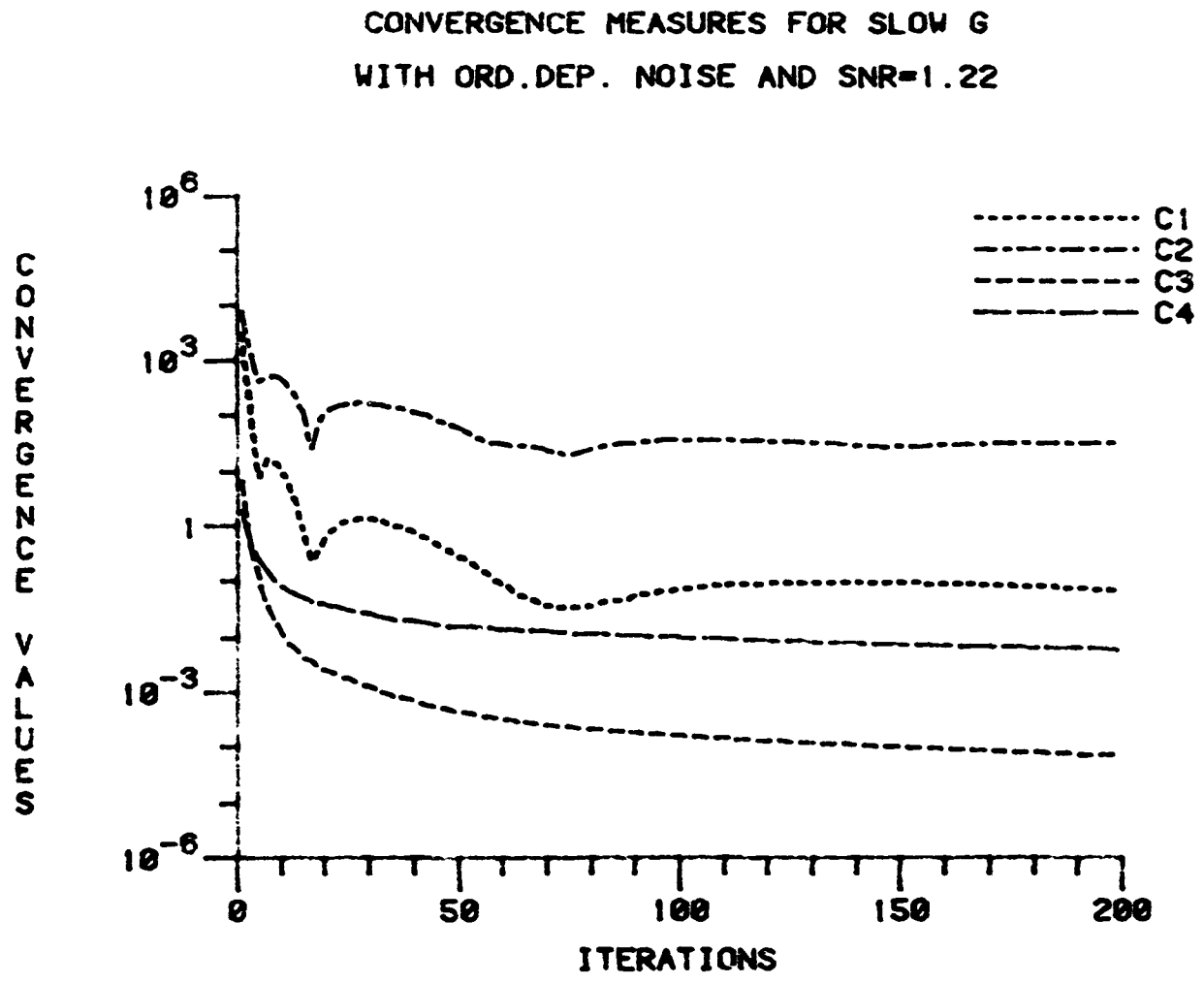
Plot 55


CONVERGENCE MEASURES FOR FAST G
WITH ORD.DEP. NOISE AND SNR=1448
C1
C2
C3
C4
CONVERGENCE VALUES
10^{4}
1
10^{-4}
10^{-8}
10^{-12}
0
20
40
60
80
100
ITERATIONS

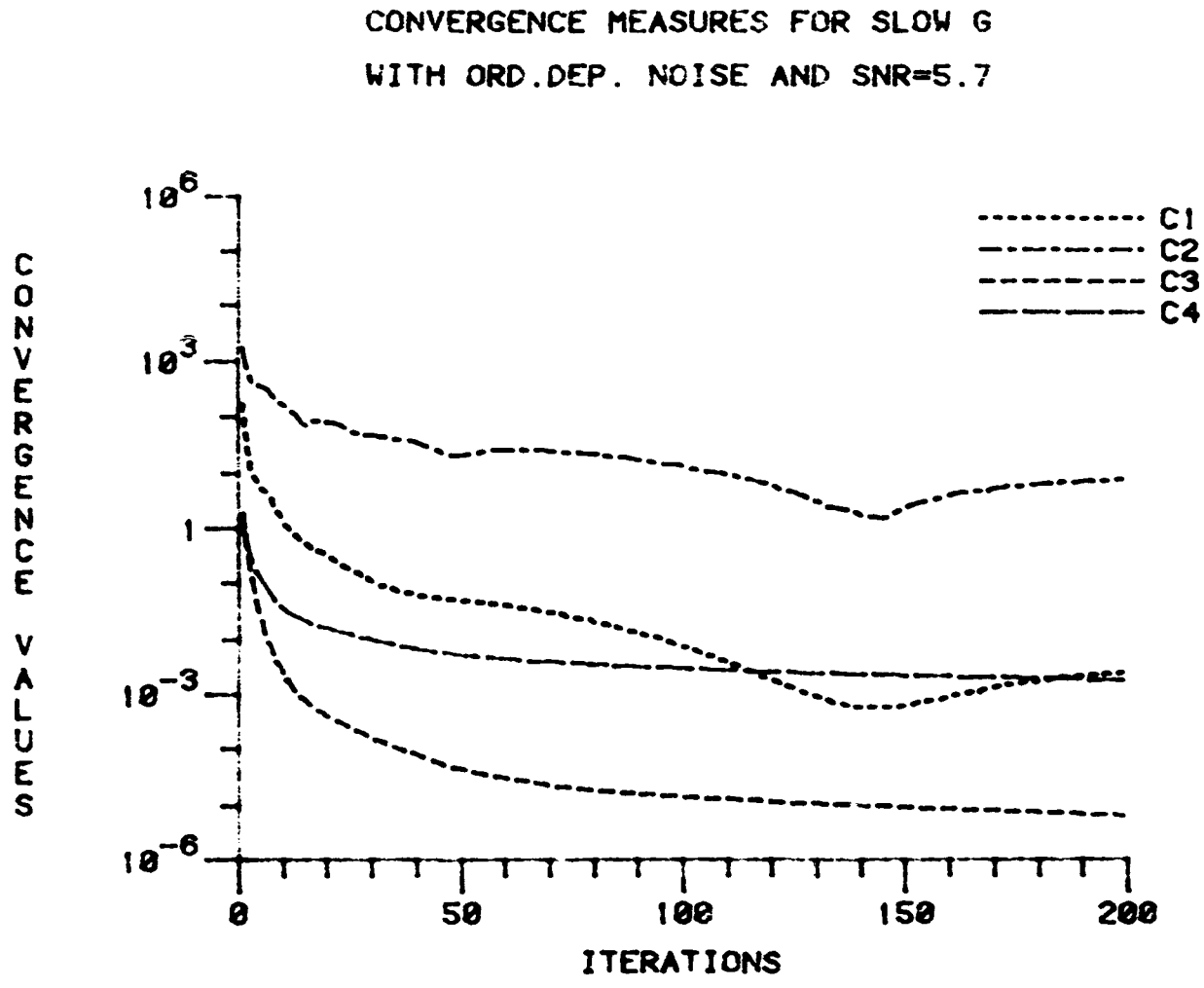
Plot 56


CONVERGENCE MEASURES FOR FAST G
WITH ORD.DEP. NOISE AND SNR=2610
C1
C2
C3
C4
CONVERGENCE VALUES
10^{4}
1
10^{-4}
10^{-8}
10^{-12}
0
20
40
60
80
100
ITERATIONS

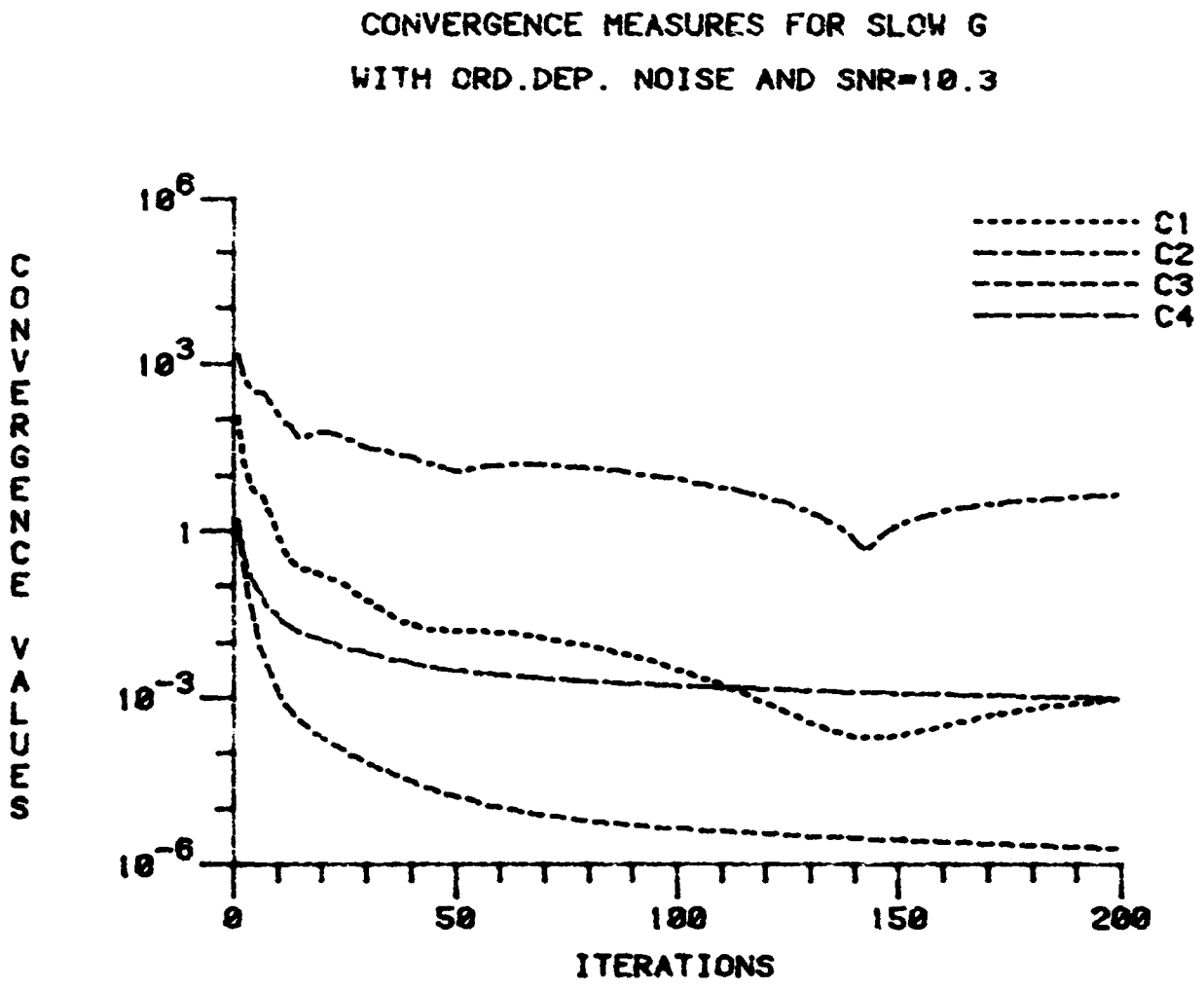
Plot 57


CONVERGENCE MEASURES FOR SLOW G
WITH ORD.DEP. NOISE AND SNR=1.22
C1
C2
C3
C4
CONVERGENCE VALUES
10^{6}
10^{3}
1
10^{-3}
10^{-6}
0
50
100
150
200
ITERATIONS

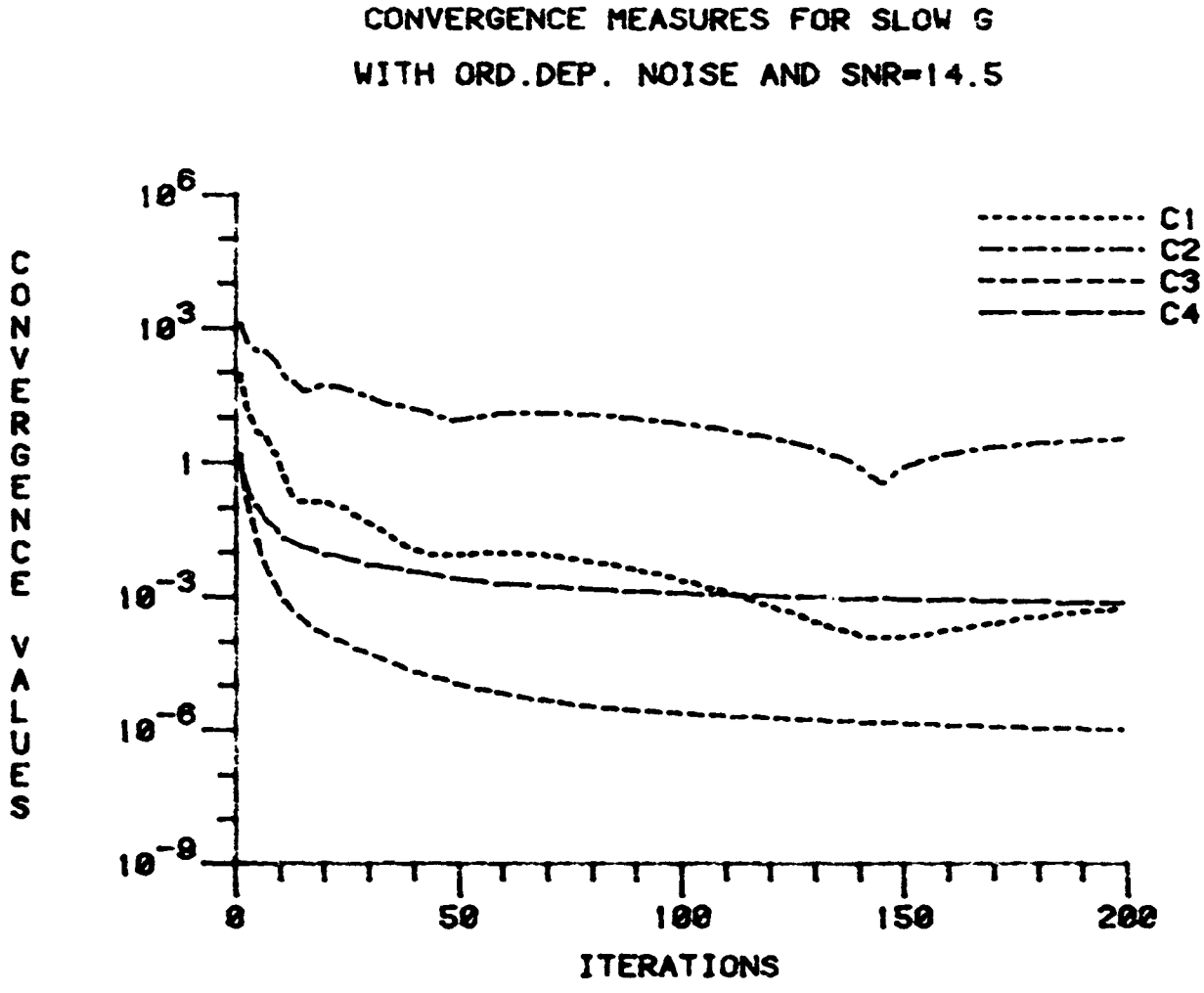
Plot 58


CONVERGENCE MEASURES FOR SLOW G
WITH ORD.DEP. NOISE AND SNR=5.7
C1
C2
C3
C4
CONVERGENCE VALUES
10^6
10^3
1
10^-3
10^-6
0
50
100
150
200
ITERATIONS

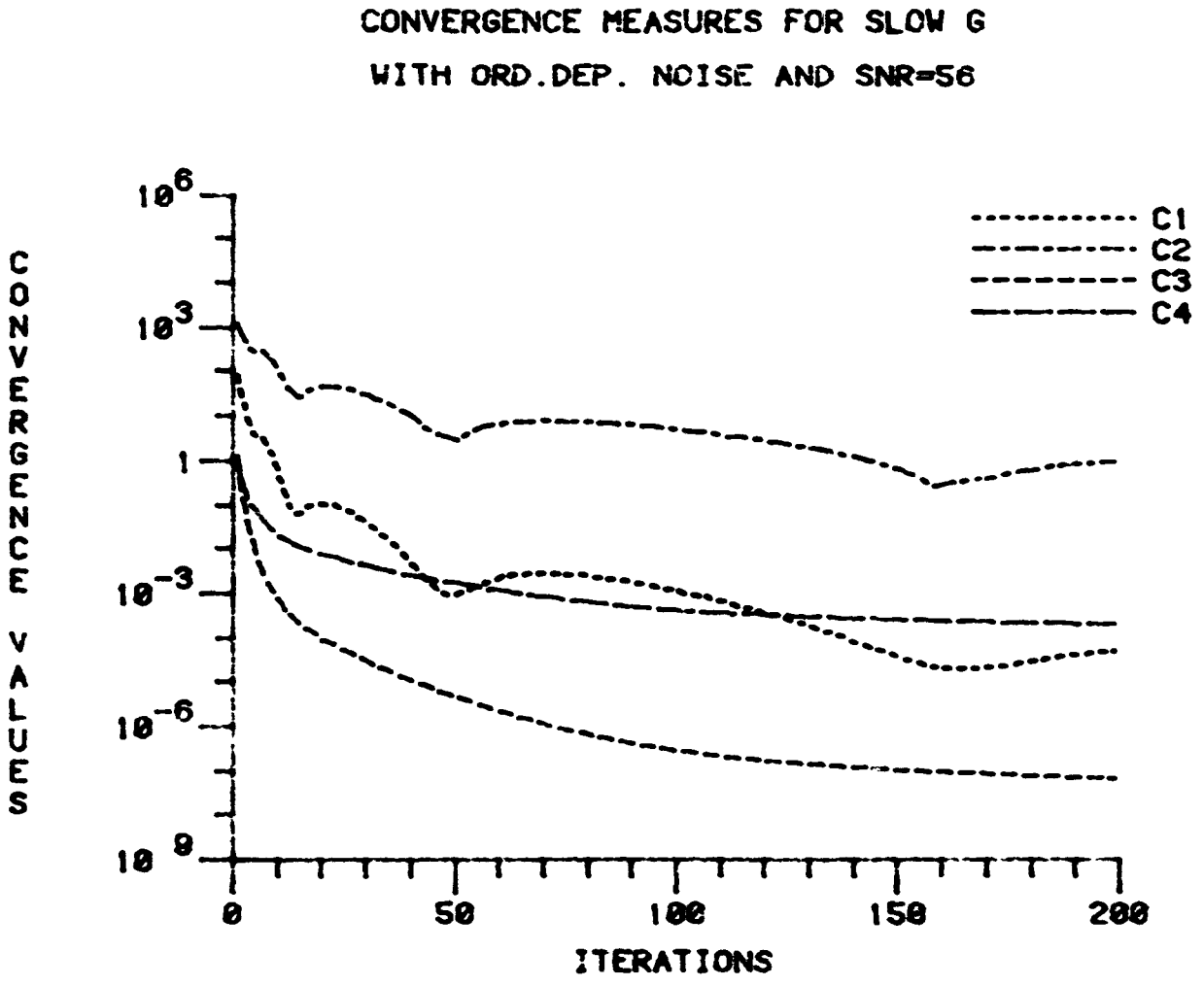
Plot 59


CONVERGENCE MEASURES FOR SLOW G
WITH ORD.DEP. NOISE AND SNR=10.3
C1
C2
C3
C4
CONVERGENCE VALUES
10^6
10^3
1
10^-3
10^-6
0
50
100
150
200
ITERATIONS

Plot 60

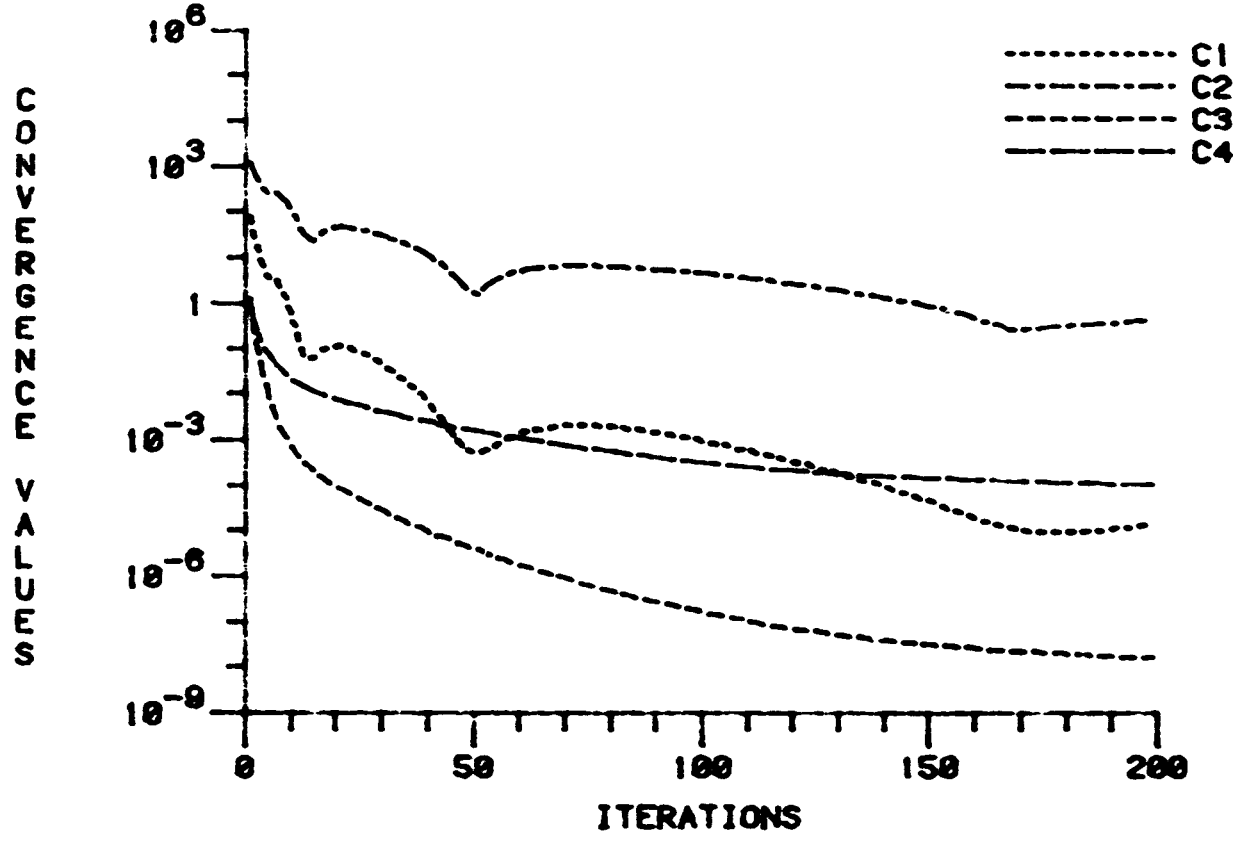

CONVERGENCE MEASURES FOR SLOW G
WITH ORD.DEP. NOISE AND SNR=14.5
C1
C2
C3
C4
CONVERGENCE VALUES
10^{6}
10^{3}
1
10^{-3}
10^{-6}
10^{-9}
0
50
100
150
200
ITERATIONS

Plot 61

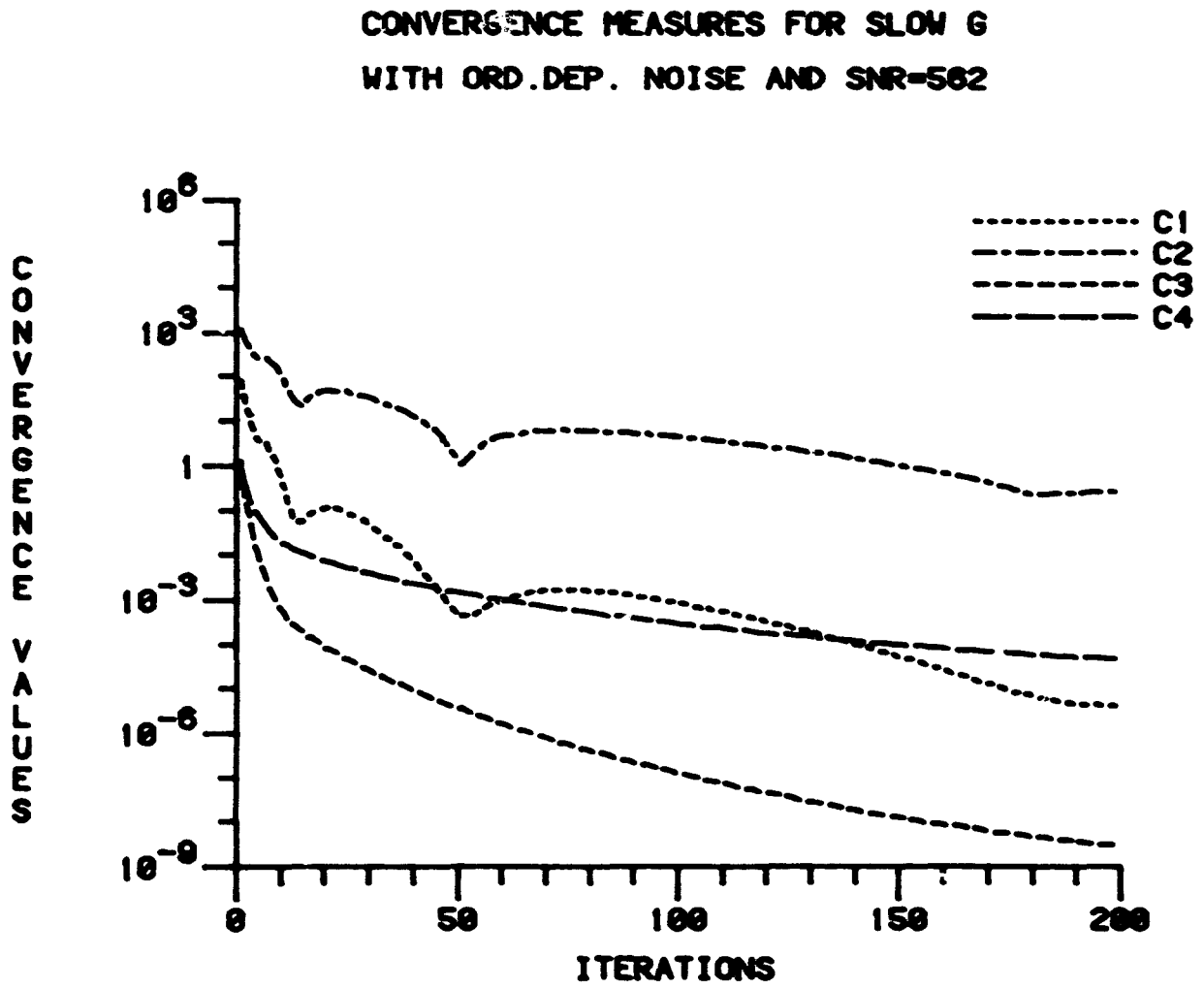

CONVERGENCE MEASURES FOR SLOW G
WITH ORD.DEP. NOISE AND SNR=56
C1
C2
C3
C4
CONVERGENCE VALUES
10^6
10^3
1
10^-3
10^-6
10^-9
0
50
100
150
200
ITERATIONS

Plot 62

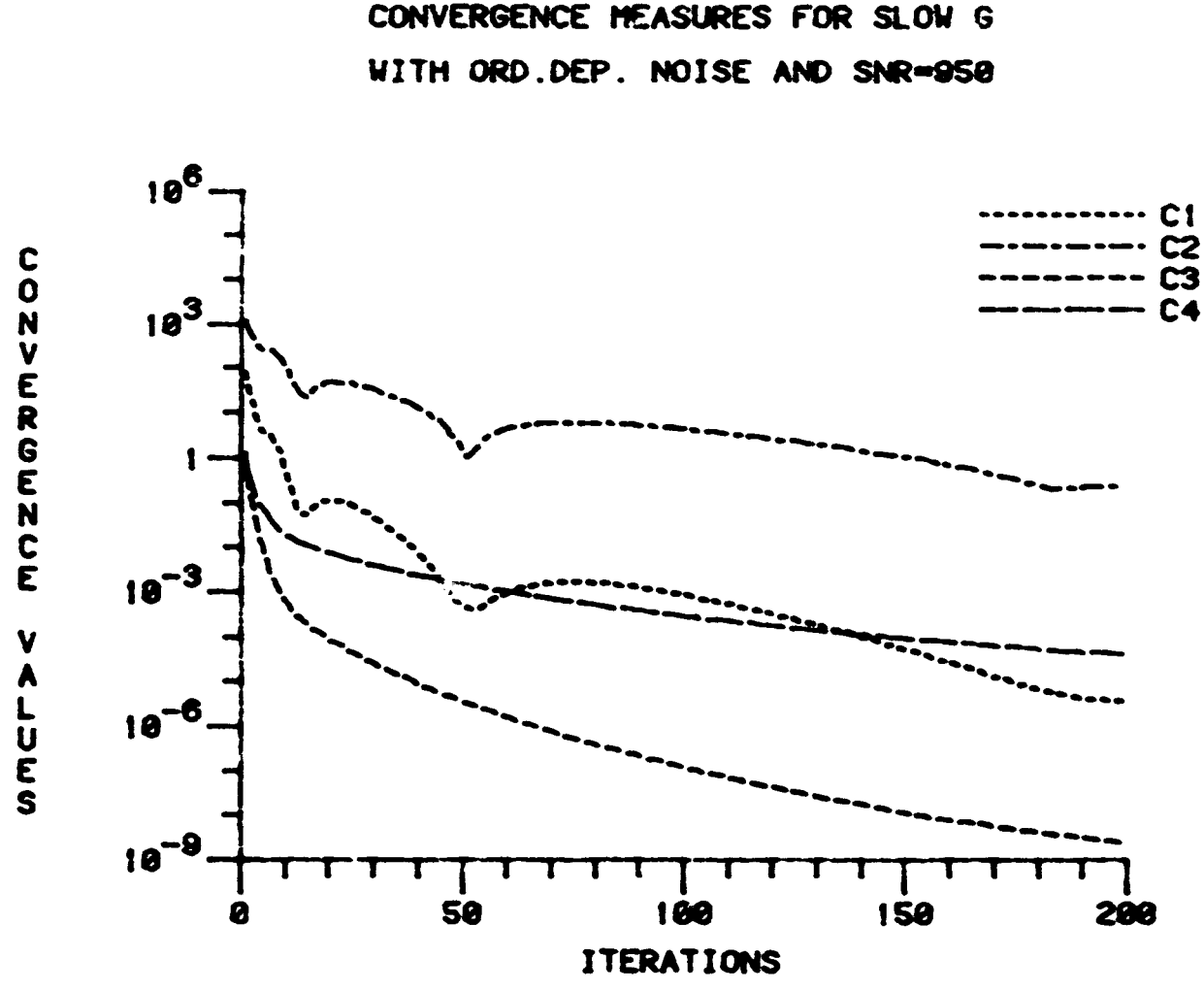
CONVERGENCE MEASURES FOR SLOW G
WITH ORD.DEP. NOISE AND SNR=126



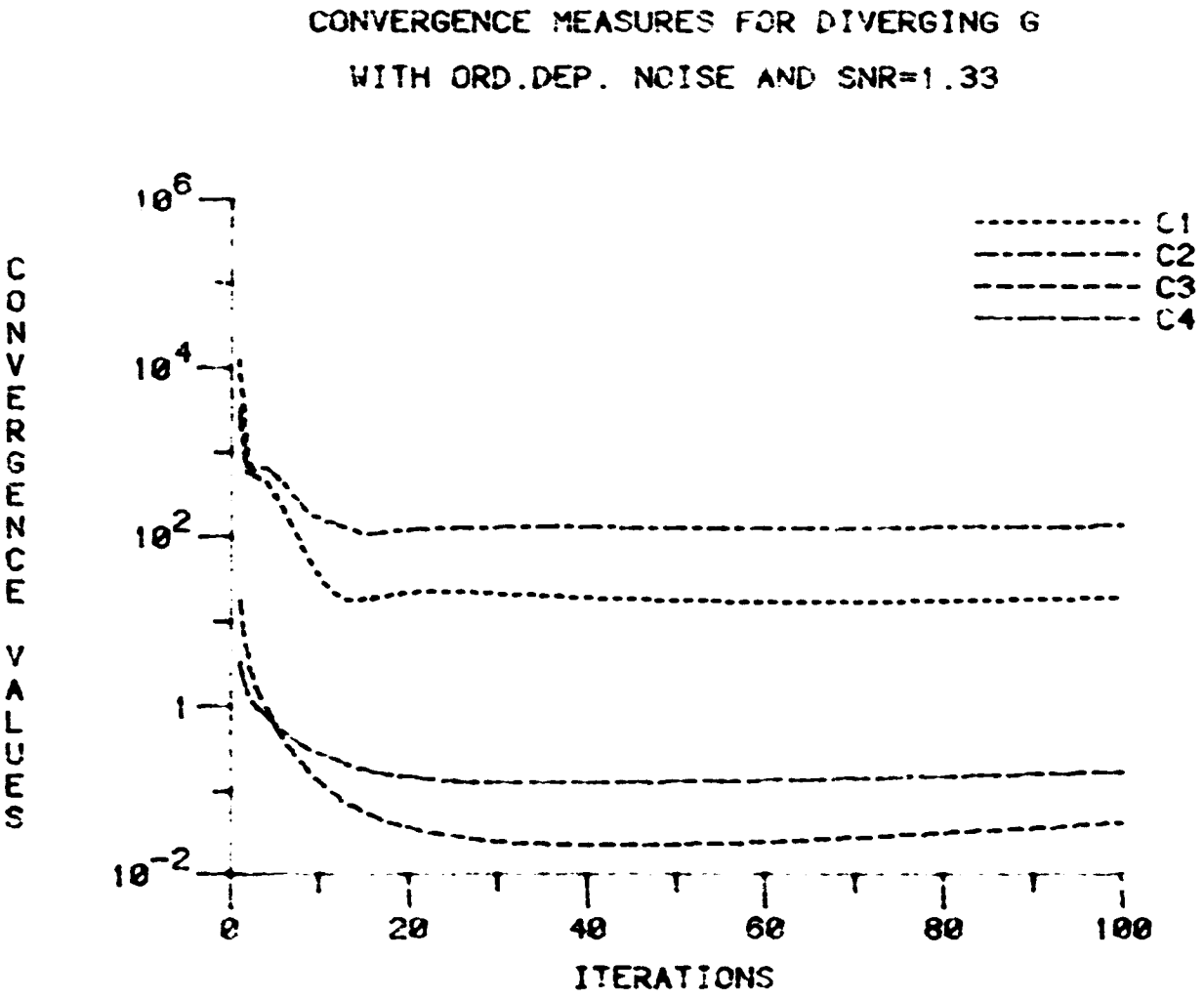
Plot 63


CONVERGENCE MEASURES FOR SLOW G
WITH ORD.DEP. NOISE AND SNR=562
C1
C2
C3
C4
CONVERGENCE VALUES
10^6
10^3
1
10^-3
10^-6
10^-9
0
50
100
150
200
ITERATIONS

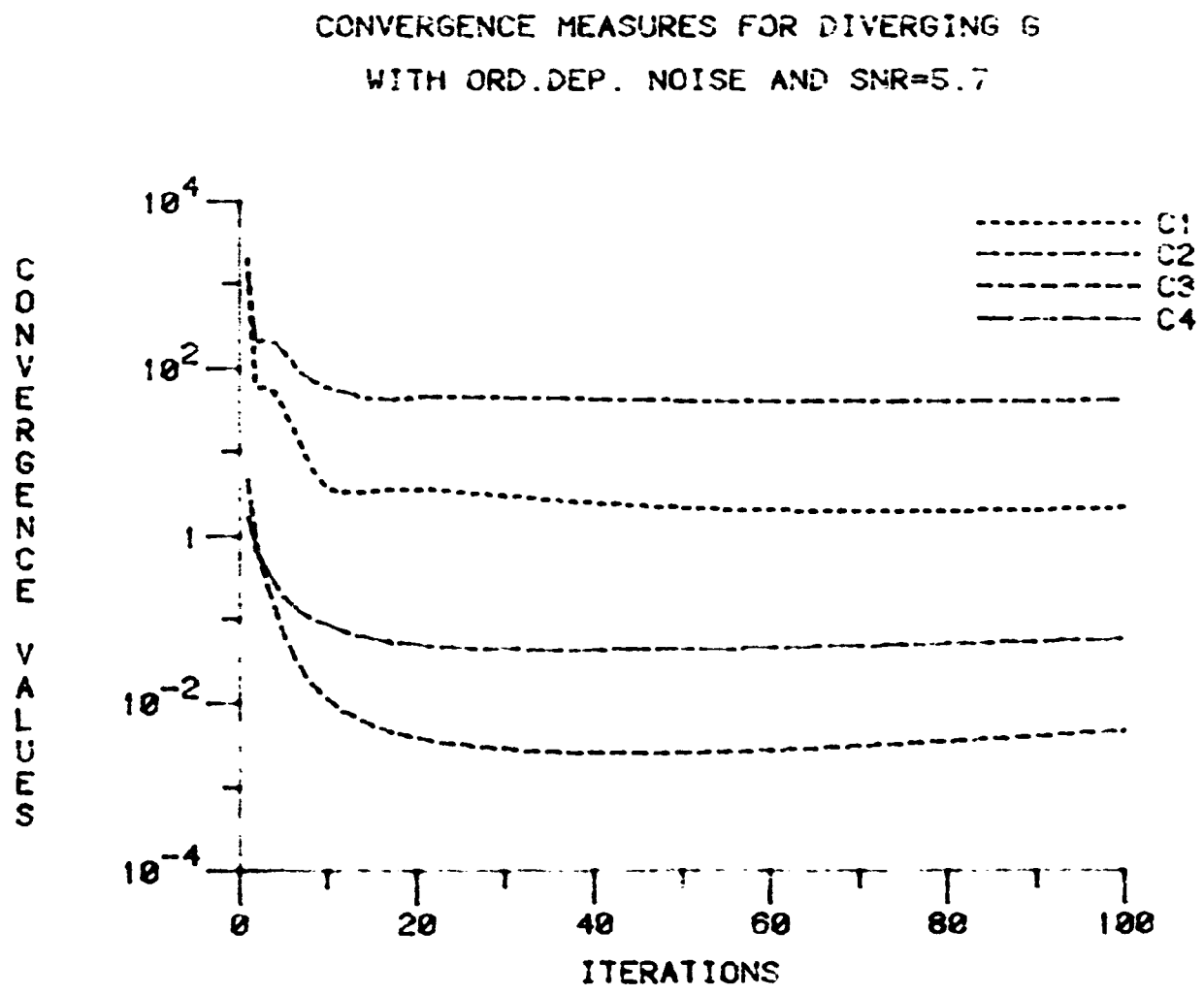
Plot 64


CONVERGENCE MEASURES FOR SLOW G
WITH ORD.DEP. NOISE AND SNR=950
C1
C2
C3
C4
CONVERGENCE VALUES
10^{6}
10^{3}
1
10^{-3}
10^{-6}
10^{-9}
0
50
100
150
200
ITERATIONS

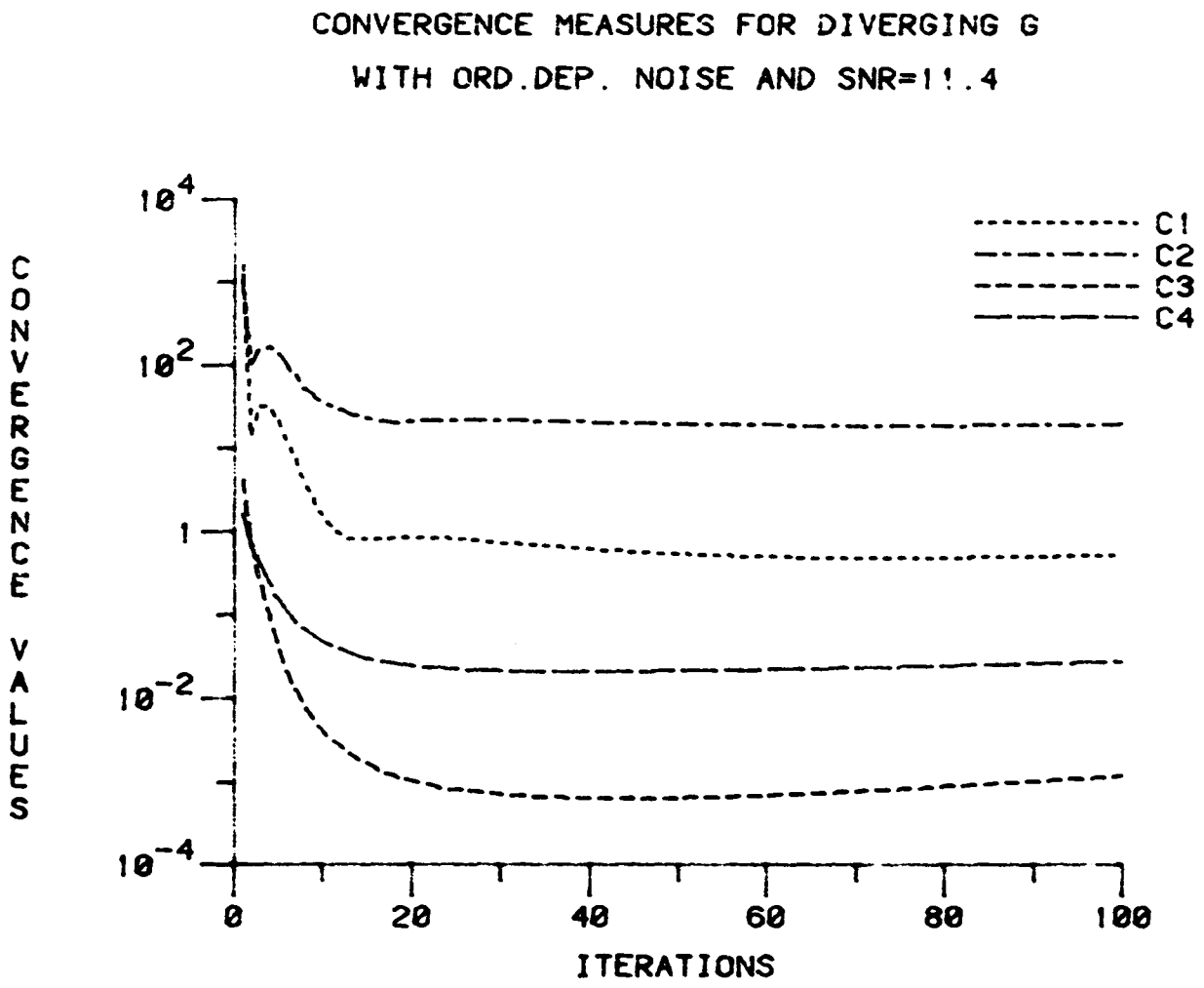
Plot 65


CONVERGENCE MEASURES FOR DIVERGING G
WITH ORD.DEP. NOISE AND SNR=1.33
C1
C2
C3
C4
CONVERGENCE VALUES
10^6
10^4
10^2
1
10^{-2}
0
20
40
60
80
100
ITERATIONS

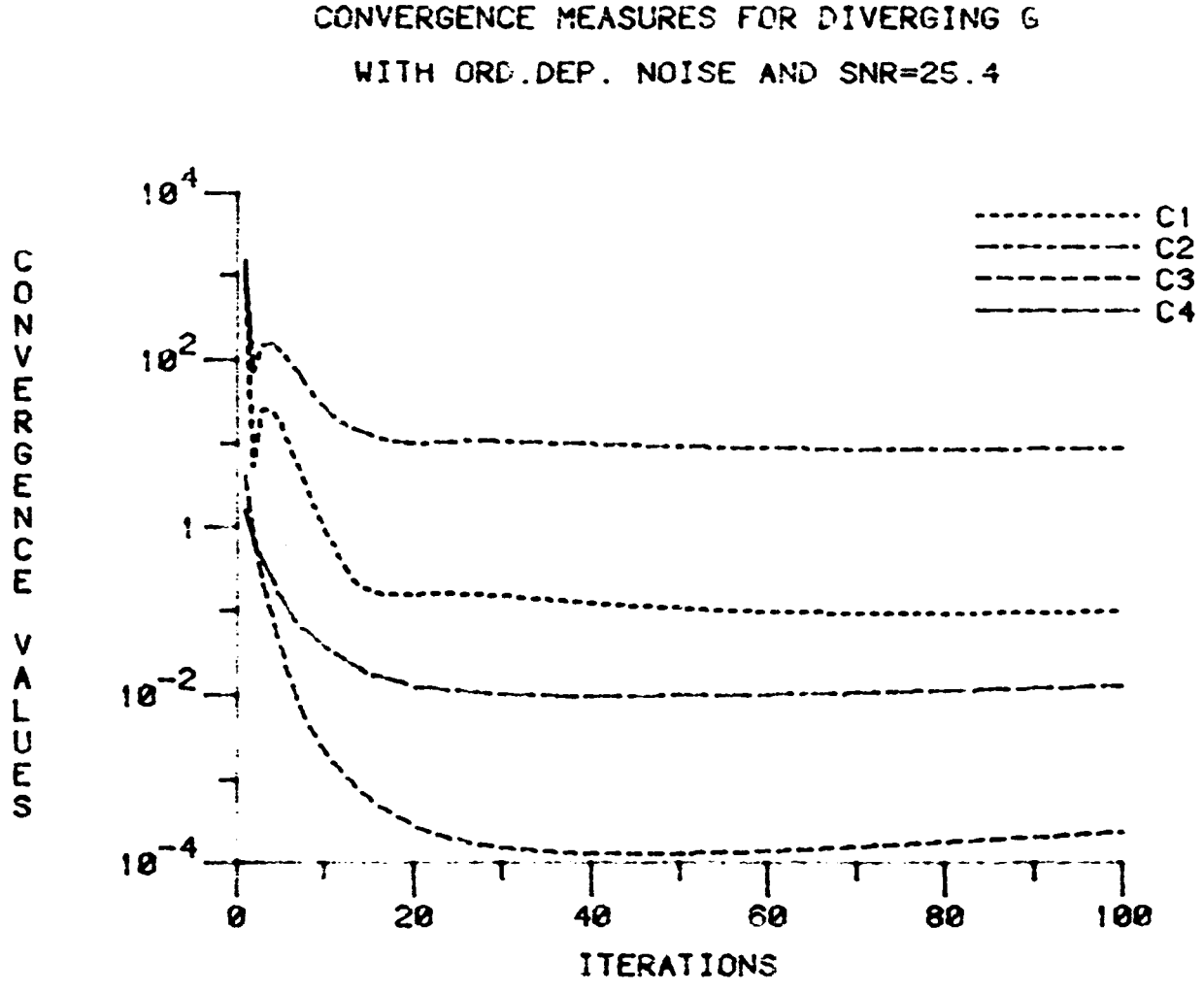
Plot 66


CONVERGENCE MEASURES FOR DIVERGING G
WITH ORD.DEP. NOISE AND SNR=5.7
C1
C2
C3
C4
10^4
10^2
1
10^{-2}
10^{-4}
CONVERGENCE VALUES
0
20
40
60
80
100
ITERATIONS

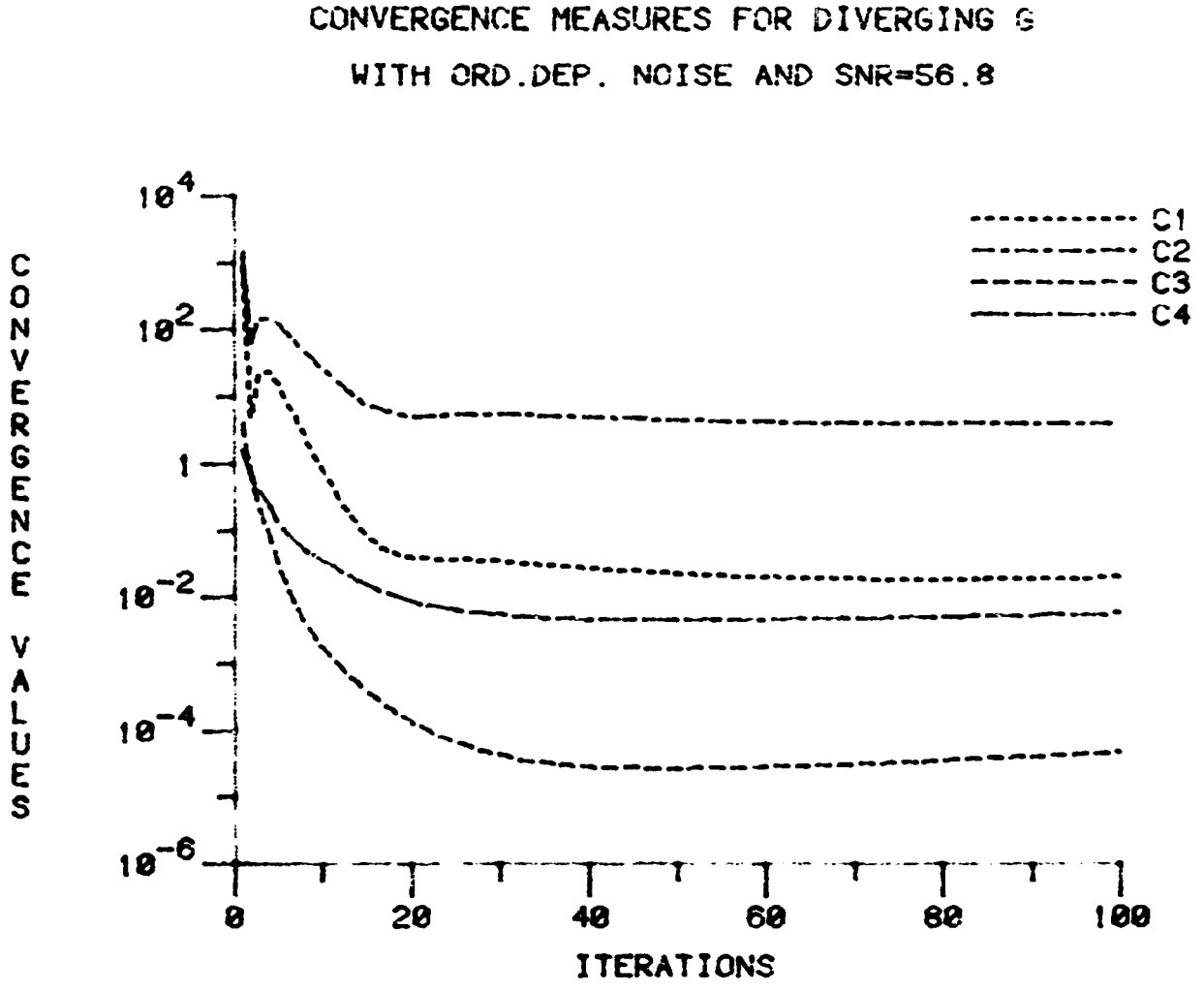
Plot 67


CONVERGENCE MEASURES FOR DIVERGING G
WITH ORD.DEP. NOISE AND SNR=1!.4
C1
C2
C3
C4
CONVERGENCE VALUES
10^4
10^2
1
10^{-2}
10^{-4}
0
20
40
60
80
100
ITERATIONS

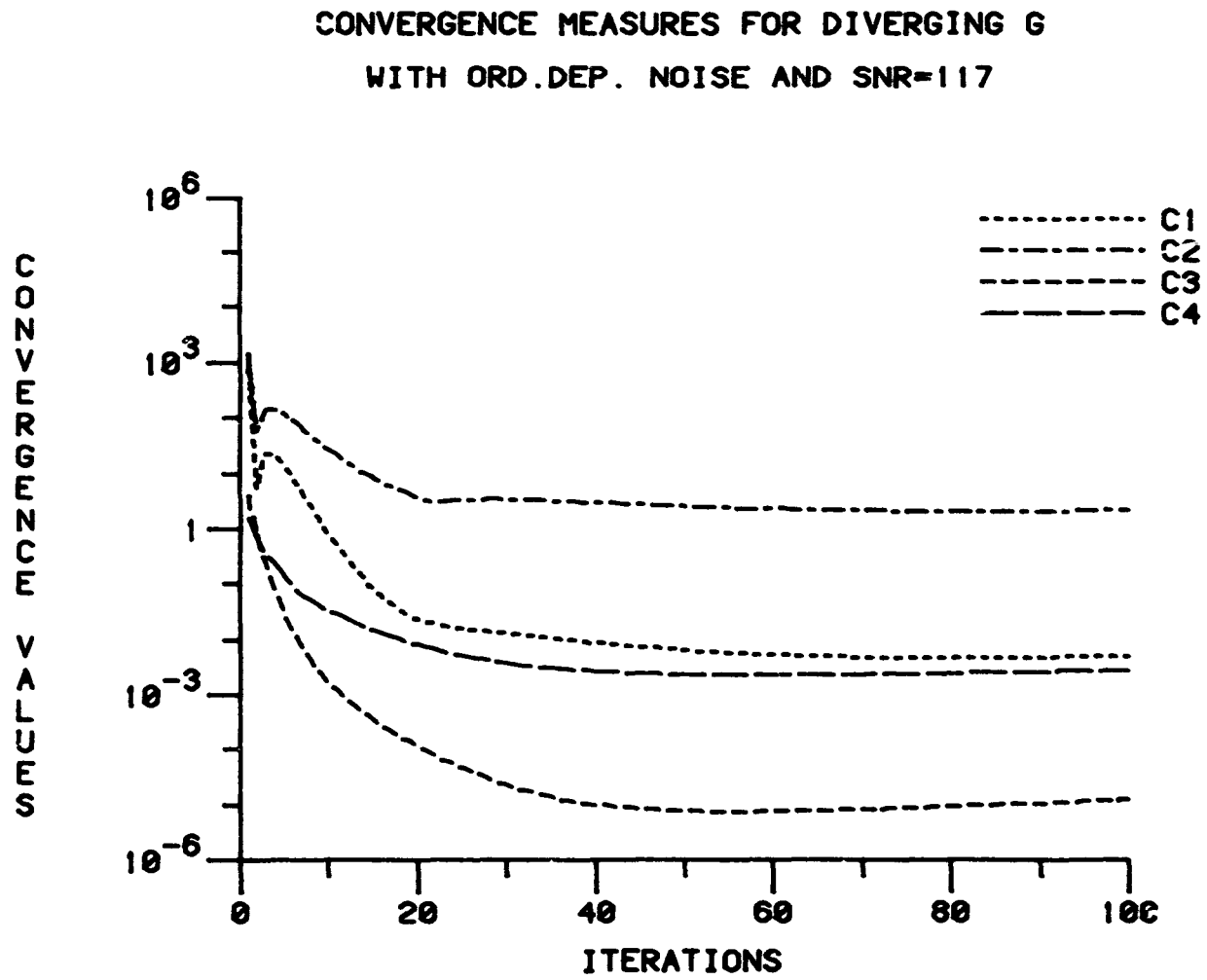
Plot 68


CONVERGENCE MEASURES FOR DIVERGING G
WITH ORD.DEP. NOISE AND SNR=25.4
C1
C2
C3
C4
10^4
10^2
1
10^{-2}
10^{-4}
CONVERGENCE VALUES
0
20
40
60
80
100
ITERATIONS

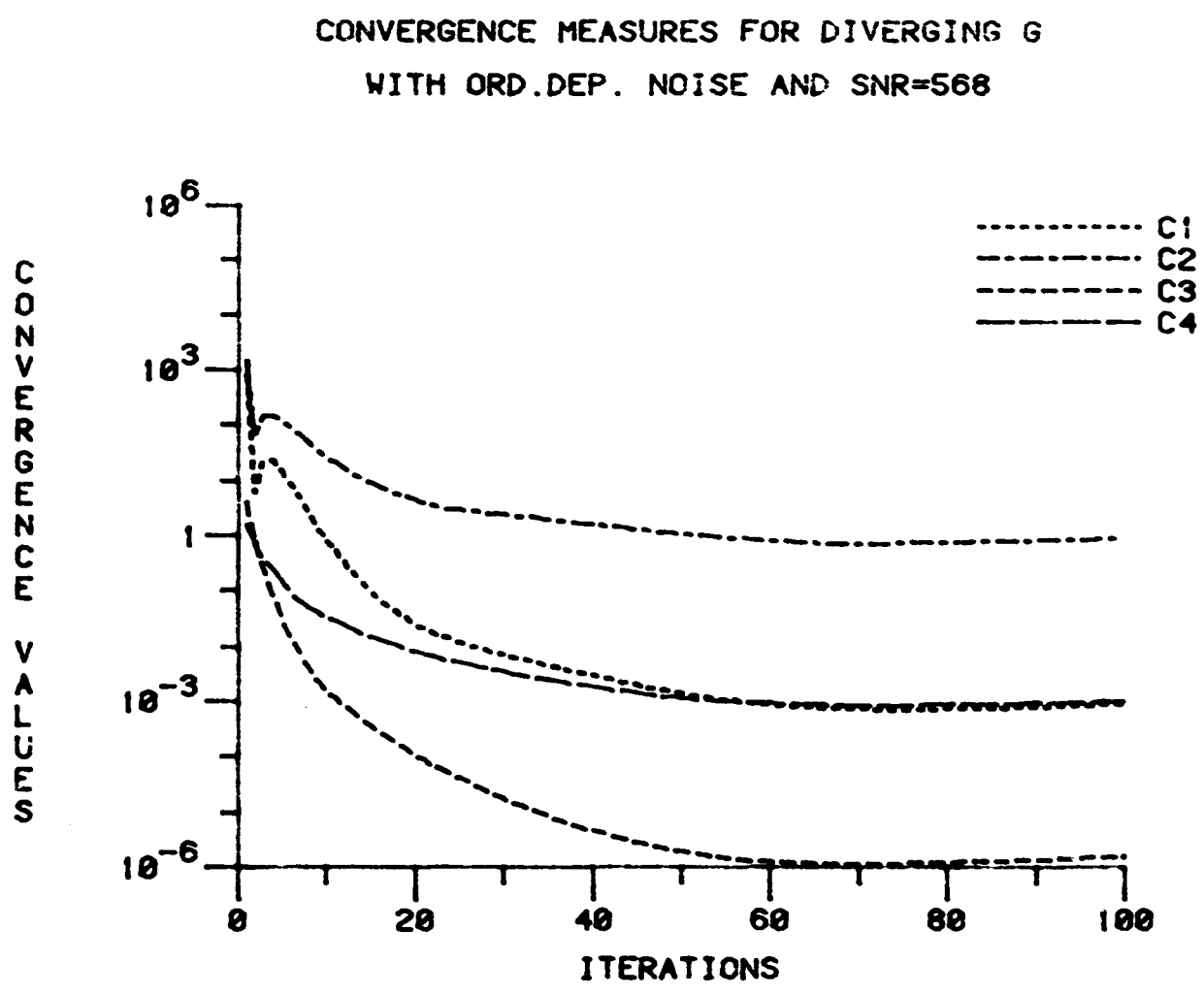
Plot 69


CONVERGENCE MEASURES FOR DIVERGING G
WITH ORD.DEP. NOISE AND SNR=56.8
C1
C2
C3
C4
CONVERGENCE VALUES
10^4
10^2
1
10^{-2}
10^{-4}
10^{-6}
0
20
40
60
80
100
ITERATIONS

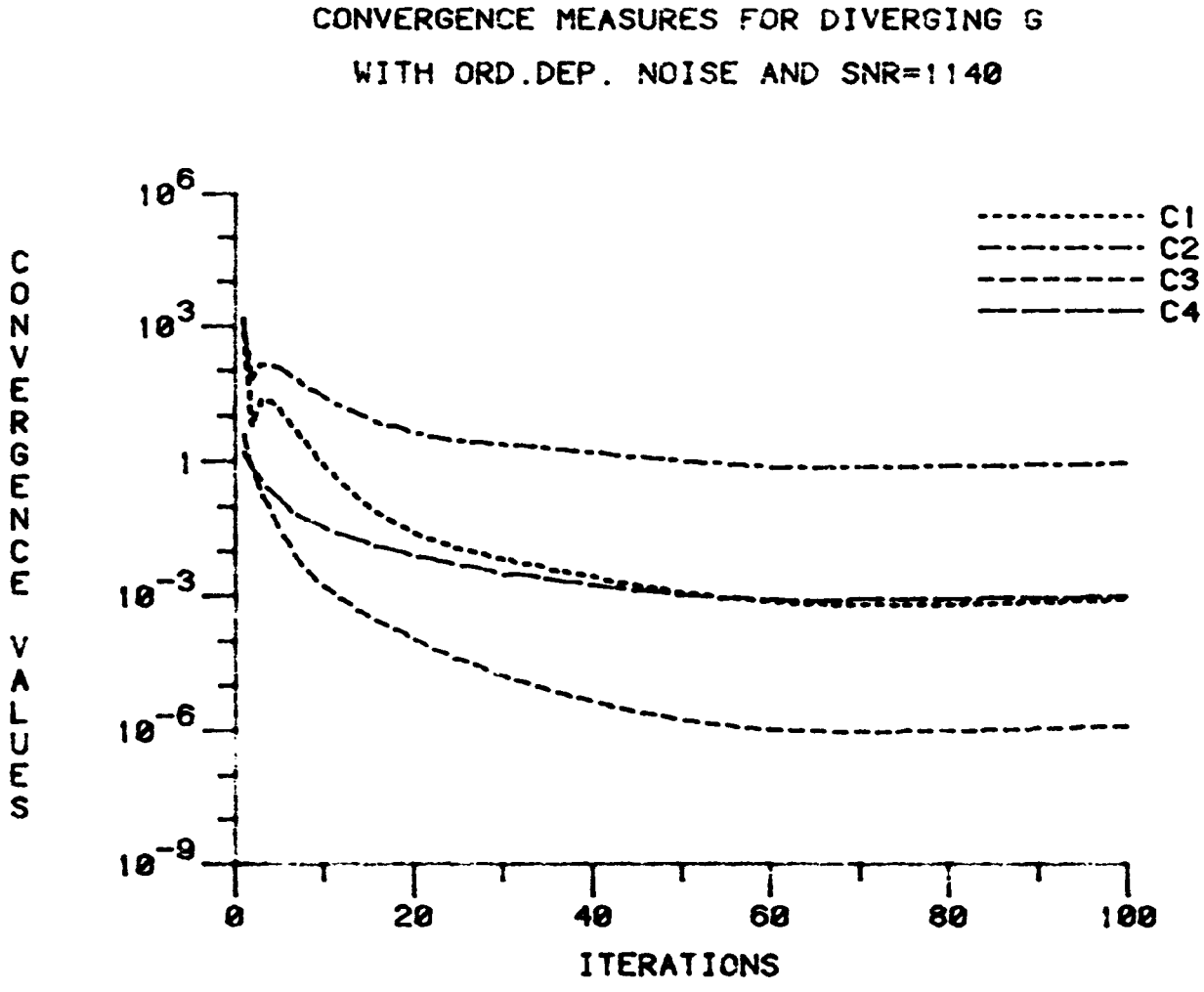
Plot 70


CONVERGENCE MEASURES FOR DIVERGING G
WITH ORD.DEP. NOISE AND SNR=117
C1
C2
C3
C4
CONVERGENCE VALUES
10^{6}
10^{3}
1
10^{-3}
10^{-6}
0
20
40
60
80
100
ITERATIONS

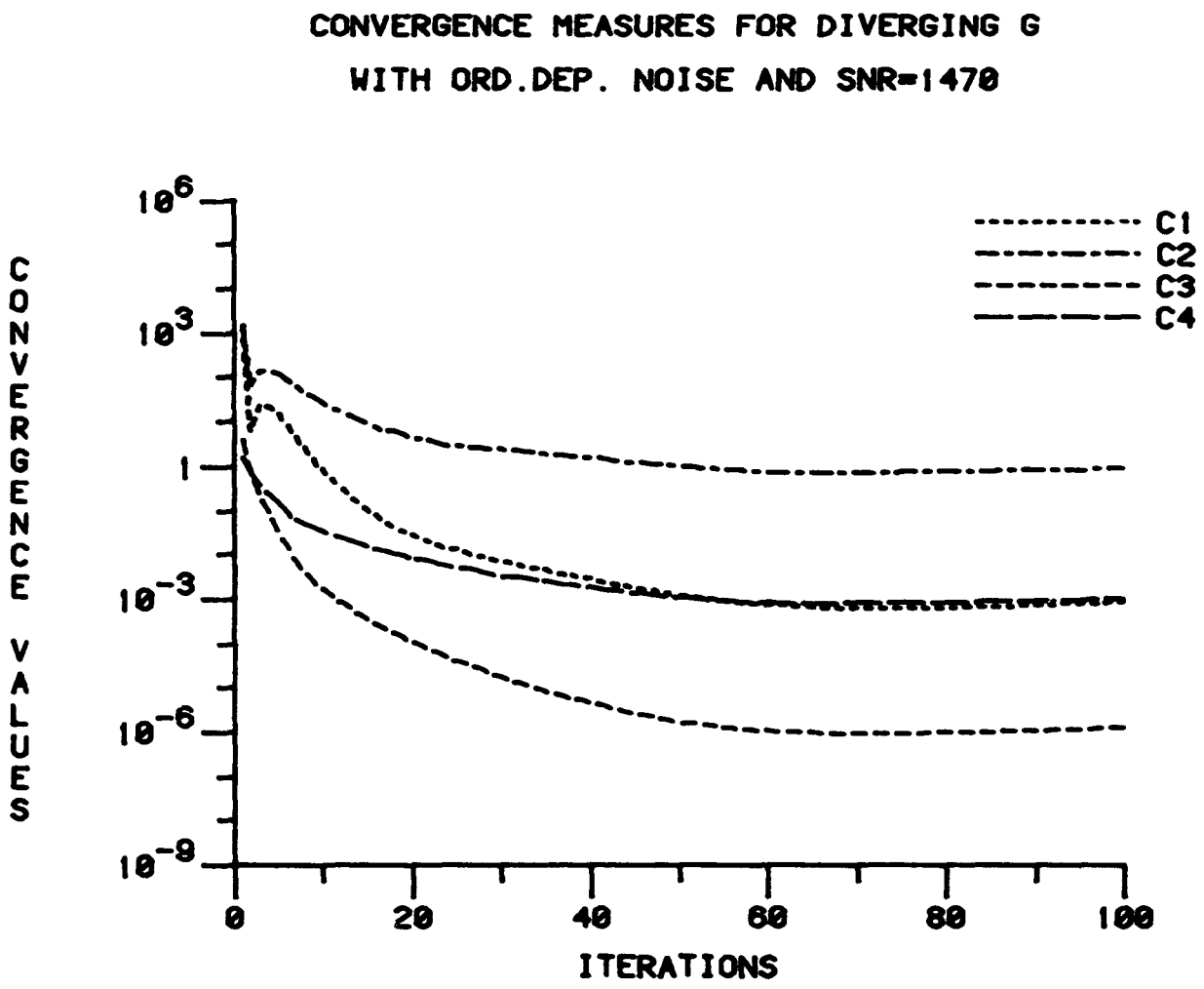
Plot 71


CONVERGENCE MEASURES FOR DIVERGING G
WITH ORD.DEP. NOISE AND SNR=568
C1
C2
C3
C4
CONVERGENCE VALUES
10^{6}
10^{3}
1
10^{-3}
10^{-6}
0
20
40
60
80
100
ITERATIONS

Plot 72


CONVERGENCE MEASURES FOR DIVERGING G
WITH ORD.DEP. NOISE AND SNR=1140
C1
C2
C3
C4
CONVERGENCE VALUES
10^6
10^3
1
10^-3
10^-6
10^-9
0
20
40
60
80
100
ITERATIONS

Plot 73


CONVERGENCE MEASURES FOR DIVERGING G
WITH ORD.DEP. NOISE AND SNR=1470
C1
C2
C3
C4
CONVERGENCE VALUES
10^{6}
10^{3}
1
10^{-3}
10^{-6}
10^{-9}
0
20
40
60
80
100
ITERATIONS

Plot 74

APPENDIX 1

Proof of Theorems

i Definate Integral, Central Ordinate Theorem

Theorem:

$$f(x) \supset F(s) \rightarrow \int_{-\infty}^{\infty} f(x)dx = F(0)$$

Proof:

$$F(s) = \int_{-\infty}^{\infty} f(x) e^{-i2\pi xs} dx$$

$$F(s)|_{s=0} = \left[\int_{-\infty}^{\infty} f(x) e^{-i2\pi xs} dx\right]|_{s=0}$$

$$F(0) = \int_{-\infty}^{\infty} f(x) e^{-0} dx$$

$$= \int_{-\infty}^{\infty} f(x)\, dx$$

ii Abscissas of centroids add under convolution.

Theorem:

$$\langle x \rangle_{f*g} = \langle x \rangle_f + \langle x \rangle_g$$

where

$$\langle x \rangle_f = \frac{\int_{-\infty}^{\infty} x f(x)\, dx}{\int_{-\infty}^{\infty} f(x) dx}$$

$$= \frac{\int_{-\infty}^{\infty} x f(x) dx}{A_f}$$

Proof (15):

$$\langle x \rangle_{f*g} = \frac{\int_{-\infty}^{\infty} x \left[\int_{-\infty}^{\infty} f(u)\, g(x-u)\, du \right] dx}{\int_{-\infty}^{\infty} \left[\int_{-\infty}^{\infty} f(u)\, g(x-u)\, du \right] dx}$$

$$= \frac{\int_{-\infty}^{\infty} f(u) \left[\int_{-\infty}^{\infty} x\, g(x-u)\, dx \right] du}{\left(\int_{-\infty}^{\infty} f(x)\, dx \right) \left(\int_{-\infty}^{\infty} g(x)\, dx \right)}$$

$$= \frac{\int_{-\infty}^{\infty} f(u) \left[\langle x \rangle_g A_g + u A_g \right] du}{A_f A_g}$$

$$= \frac{\langle x \rangle_g A_g A_f + \langle x \rangle_f A_f A_g}{A_f A_g}$$

$$= \langle x \rangle_g + \langle x \rangle_f$$

APPENDIX 2

Programs Used

The following programs were written by the author in the course of this study.

FFT.FOR (16)

CONVOL.FOR

MORRIS.FOR

GRAPH.FOR

FFT.FOR is a program which takes the fast Fourier transform of a data set. It was used to ascertain the transforms of the g functions used, so that their rates of convergence could be determined.

CONVOL.FOR takes the serial product of two sequences. It was used to convolve the chosen f function with each g function to produce to corresponding h function.

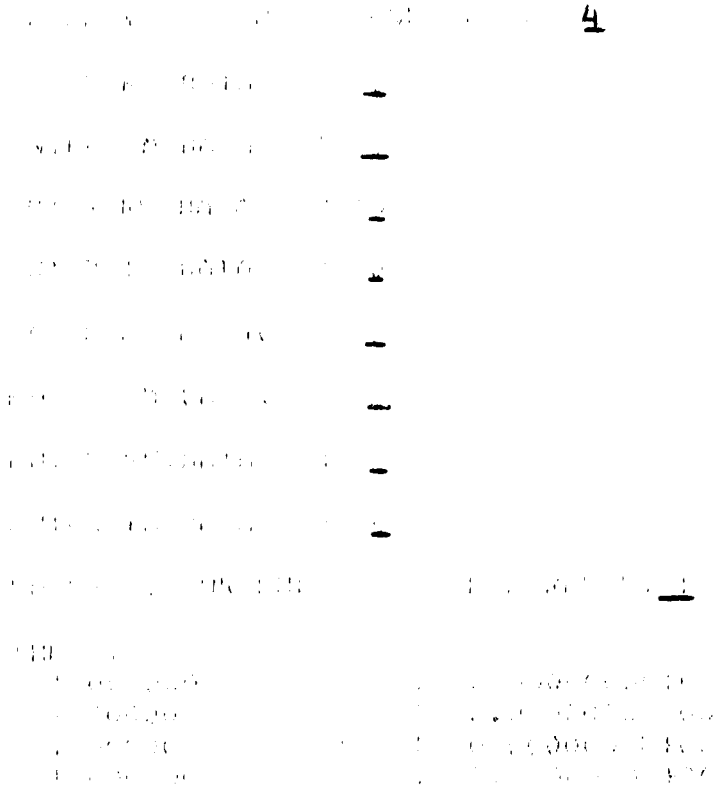
MORRIS.FOR is the primary processing program. It takes a g and an h sequence as input, adds the specified gaussian noise type (ordinate dependent, constant, or both) to the h then performs Morrison Smoothing upon the noisy h function. Several measures of convergence and error are calculated and stored for each iteration.

GRAPH.FOR is actually a small utility program that sorts the data stored by MORRIS.FOR into a form easily accessible to ICP. The Interactive Graphics Program was actually used to produce all graphs in this thesis.

Listings of each program comprise the last part of this appendix. The following is a brief overview of how those programs may be used. In each case the quantities underlined are those that were being entered.

FFT.FOR

The following is an example of how data is entered into FFT.FOR



The first entry is the length of the real or complex sequence to be transformed. It must be of length 2**m, where m is an integer. If the data are real, enter 0, zero for each imaginary part. The third input is either -1, specifying the forward transform, or +1 for the inverse transform.

The complex output is the Fourier transform.

CONVOL.FOR

The following is an example of the use of CONVOL.FOR

```
ENTER LENGTHS OF F & G
3
5
ENTER F TERMS
1
2
1
ENTER G TERMS
1
2
3
[illegible]
1
  THE RESULT IS
   1.000000
   2.000000
   1.000000
   1.000000
   1.000000
   2.000000
   1.000000
 ENTER OUTPUT FILE #/5
```

The first entries are the lengths of each sequence involved. Then as prompted, enter the elements of each sequence, in the order specified in entry 1. The result is printed out. The program the requests a number for the output file. If file storage is not required, type CONTROL C. If it is needed, enter an integer between 21 and 63.

MORRIS.FOR

The following is a sample of MORRIS.FOR.

The first entry is for labelling purposes only. Enter a 1 if the g function to be used will converge quickly, 2 if slowly and 3 if it diverges. The second entry is the number of elements in the h to be considered. The third is the number of elements of the g used, which for this program must be an odd number. The fourth entry requests the file number for the file containing the g and h functions. The fifth entry asks the user to specify the type of noise that MORRIS.FOR will add to h. 1 is for ordinate dependent noise, 2 for constant noise and 3 for both. The sixth entry requests a scale factor for the noise. This factor is highly empirical but as the value entered increases, so does the noise, as shown later by the corresponding increase in the RMS and decrease in the SNR. The seventh entry asks for the number of restoring iterations desired. The eighth entry is for the file number wherein the data will be stored. A sample of the data follows. The convergence and error measures referred to are those discussed in the body of the thesis, in the order they were presented.

GRAPH.FOR

GRAPH.FOR is poorly named. This program merely arranges the data output by MORRIS.FOR into a form accessible to IGP , the system's own graphics package. Regardless, here is a sample of its use:

ORIGINAL PAGE IS
OF POOR QUALITY

```
ENTER [illegible] FILE NUMBER
[illegible]

THE [illegible] IS [illegible] WITH NOISE TYPE    ORU.P
SF  [illegible] RMS  0.9951544     SNR     52.22293    K=  100
 DONE READING IN
ENTER THE OUTPUT FILE FOR CON# 1
21

ENTER THE OUTPUT FILE FOR CON# 2
22

ENTER THE OUTPUT FILE FOR CON# 3
23

ENTER THE OUTPUT FILE FOR CON# 4
24

ENTER OUTPUT FILE FOR FR# 1
[illegible]

ENTER OUTPUT FILE FOR FR# [illegible]
26

ENTER OUTPUT FILE FOR FR# 3
27
```

The first entry is the desired output file number from MORRIS.FOR. There follows a heading, giving the specifics of the file accessed. The second entries are output file numbers for each measure, as prompted.

```
C
C          FFT PROGRAM
C
          DIMENSION DATA(256)
          INTEGER N,NN,ISIGN
          TYPE 15
15        FORMAT(' ENTER   DATA PTS, POWER OF 2',$)
          ACCEPT 20,NN
20        FORMAT(I)
          N=2*NN
          DO 40 I=1,N,2
          J=(I+1)/2
          TYPE 25,J
25        FORMAT(' ENTER RE(DATA(',I3,'))',$)
          ACCEPT 30,DATA(I)
30        FORMAT(G)
          TYPE 35,J
35        FORMAT(' ENTER IM(DATA(',I3,'))',$)
40        ACCEPT 30,DATA(I+1)
          TYPE 45
45        FORMAT(' ENTER -1 FOR FWD FT, +1 FOR REV FT',$)
          ACCEPT 20,ISIGN
          J=1
          DO 5 I=1,N,2
          IF (I.GE.J) GOTO 2
1         TEMPR=DATA(J)
          TEMPI=DATA(J+1)
```

```
      DATA(J)=DATA(I)
      DATA(J+1)=DATA(I+1)
      DATA(I)=TEMPR
      DATA(I+1)=TEMPI
2     M=N/2
3     IF (J.LE.M) GO TO 5
4     J=J-M
      M=M/2
      IF (M.GE.2) GO TO 3
5     J=J+M
      MMAX=2
6     IF (MMAX.GE.N) GO TO 10
7     ISTEP=2*MMAX
      THETA=6.2831853/FLOAT(ISIGN*MMAX)
      SINTH=SIN(THETA/2.)
      WSTPR=-2.*SINTH*SINTH
      WSTPI=SIN(THETA)
      WR=1.
      WI=0.
      DO 9 M=1,MMAX,2
      DO 8 I=M,N,ISTEP
      J=I+MMAX
      TEMPR=WR*DATA(J)-WI*DATA(J+1)
      TEMPI=WR*DATA(J+1)+WI*DATA(J)
      DATA(J)=DATA(I)-TEMPR
      DATA(J+1)=DATA(I+1)-TEMPI
      DATA(I)=DATA(I)+TEMPR
```

```
8       DATA(I+1)=DATA(I+1)+TEMPI
        TEMPR=WR
        WR=WR*WSTPR-WI*WSTPI+WR
9       WI=WI*WSTPR+TEMPR*WSTPI+WI
        MMAX=ISTEP
        GO TO 6
10      TYPE 50
50      FORMAT(' THE FT IS')
        DO 55 I=1,N,2
55      TYPE 60,DATA(I),DATA(I+1)
60      FORMAT(G,'   +   I',G)
        STOP
        END
```

```
C
C
C
C
C            CONVOLUTION PROGRAM
C
C
      DIMENSION FI(0/255),G(0/255),H(0/255)
        INTEGER A,B,C,D,E,F,OFL
C       ENTER SEQUENCE LENGTHS
      TYPE 10
10      FORMAT(' ENTER LENGTHS OF F_G')
      ACCEPT 20,M
      ACCEPT 20,N
   20 FORMAT(I)
      M=M-1
      N=N-1
C       ENTER SEQUENCE MEMBERS
      TYPE 30
30      FORMAT(' ENTER F TERMS')
      DO 40 I=0,M
   40 ACCEPT 70,FI(I)
      TYPE 50
50      FORMAT(' ENTER G TERMS')
      DO 60 I=0,N
   60 ACCEPT 70,G(I)
   70 FORMAT(G)
```

```
C       MAIN PART OF COMPUTATION
      K=M+N
      DO 100 I=0,K
      H(I)=0.
      A=0
      B=I-N
      CALL MAX(A,B,C)
      D=I
      E=M
      CALL MIN(D,E,F)
      DO 90 J=C,F
      TEMP = FI(J)*G(I-J)
   90 H(I)=H(I)+TEMP
  100 CONTINUE
C       OUTPUT SECTION
      TYPE 110
110     FORMAT('  THE RESULT IS')
      DO 120 I=0,K
  120 TYPE 130,H(I)
130     FORMAT(4G)
      TYPE 140
140     FORMAT(' ENTER OUTPUT FILE  ',$)
        ACCEPT 20,OFL
150     FORMAT(4G)
        WRITE(OFL,150),(H(I),I=0,K)
        WRITE(OFL,150),(G(I),I=0,N)
      STOP
```

```
      END
      SUBROUTINE MAX(A,B,C)
C       CHOOSES LARGER PARAMETER
      IF (A.GT.B) C=A
      IF (A.LT.B) C=B
      IF (A.EQ.B) C=A
      RETURN
      END
      SUBROUTINE MIN(D,E,F)
C       CHOOSES SMALLER PARAMETER
      IF (D.LT.E) F=D
      IF (D.GT.E) F=E
      IF (D.EQ.E) F=E
      RETURN
      END
```

```
C
C
C
C           MORRISON SMOOTHING
C
C
C
C       THIS PROGRAM USES THE FOLLOWING ITERATIVE TECHNIQUE
TO SMOOTH
C       AND RESTORE DATA
C               (H=F*G)
C                H1=H*G
C                HN=HN-1 + (H - HN-1)*G
C       G IS THE RESPONSE FUNCTION.
C
C
C
        DIMENSION
H(0/255),G(0/255),HP(0/255),HZ(0/511),HN(0/511)
        DIMENSION ER(1000,3),CON(1000,4),HOLD(0/511)
        INTEGER P,Z,ANS,GTYP,OFL
C       ENTER THE DATA
        TYPE 5
5       FORMAT(' ENTER 1 FOR FAST G,2=SLOW,3=DIVERGING')
        ACCEPT 20,GTYP
        CALL INPUT(N,M,H,G)
C           ADD NOISE
```

```
      TYPE 10
10    FORMAT(' CHOOSE 1 FOR ORD DEP
NOISE,2=CONSTANT,3=BOTH')
      ACCEPT 20,ANS
20    FORMAT(I)
      TYPE 30
30    FORMAT(' ENTER NOISE SCALE FACTOR')
      ACCEPT 40,SF
40    FORMAT(G)
      IF (ANS.EQ.1) CALL ORDNOI(H,HZ,HP,SF,RMS
     * , SNR,M,N)
      IF (ANS.EQ.2) CALL CONST(H,HZ,HP,SF
     * ,RMS,SNR,M,N)
      IF (ANS.EQ.3) CALL BOTH(H,HZ,HP,SF,RMS,
     * SNR,M,N)
C       PERFORM SMOOTHING OPERATION
      CALL SMOOTH(N,M,HP,G,HN)
C       SET UP RESTORATION LOOP
      TYPE 35
35    FORMAT('  ENTER NUMBER OF RESTORATIONS')
      ACCEPT 20,Z
      IF (Z.EQ.0) GO TO 50
C        RESTORING LOOP
      DO 45 K=1,Z
      CALL RESTOR(N,M,HP,G,HN,HOLD,HZ)
C       COMPARE HN TO H
      CALL ERROR(K,M,N,H,HN,ER)
```

```
C          COMPARE HN TO HN-1
           CALL CONVER(K,M,N,HN,HOLD,CON,H)
45         CONTINUE
50         TYPE 60
60         FORMAT(' ENTER FILE   FOR OUTPUT FILE  ',$)
           ACCEPT 20,OFL
C          OUTPUT RESULTS
           CALL OUTPUT(K,M,N,H,G,ER,CON,SF,RMS,
     * SNR,GTYP,ANS,OFL)
           STOP
           END
C
C          INPUT ENTERS THE DATA
C
           SUBROUTINE INPUT(N,M,H,G)
           DIMENSION H(0/255),G(0/255)
           INTEGER IFL
           TYPE 110
110        FORMAT ('   ENTER SIZE OF H')
           ACCEPT 120,M
120        FORMAT(I)
           TYPE 130
130        FORMAT('  ENTER SIZE OF G,ODD')
           ACCEPT 120,N
           M=M-1
           N=N-1
           TYPE 140
```

```
140     FORMAT('  ENTER THE INPUT FILE     ',$)
        ACCEPT 120,IFL
        READ(IFL,160)(H(I),I=0,M)
        READ(IFL,160)(G(I),I=0,N)
160     FORMAT(4G)
        RETURN
        END
C
C         ORDNOI ADDS ORDINATE DEPENDENT NOISE
C
        SUBROUTINE ORDNOI(H,HZ,HP,SF,RMS,SNR,M,N)
        DIMENSION H(0/255),HZ(0/255),HP(0/255)
        REAL MAXIM
        INTEGER Q,L
        Q=N/2
        RMS=0.
        MAXIM=H(0)
        DO 210 I=0,M
        P=RAN(15)
        S=RAN(10)
        IF (H(I).LT..0000001) V=RAN(5)*.000001
        IF (H(I).LT..0000001) GO  TO 200
        IF (H(I).GT.MAXIM) MAXIM=H(I)
        V=SQRT(2.*SF*H(I)*(-ALOG(P)))
200     IF (S.GT..5) V=-V
        HP(I)=H(I)+V
        IF (HP(I).LT.0.) HP(I)=-HP(I)
```

```
      L=L+Q
      HZ(L)=HP(I)
      RMS=(HP(I)-H(I))**2+RMS
210   CONTINUE
      RMS=SQRT(RMS/(M+1))
      IF (RMS.EQ.0.) GOTO 215
      SNR=MAXIM/RMS
215   RETURN
      END
C
C        CONST ADDS CONSTANT NOISE
C
      SUBROUTINE CONST(H,HZ,HP,SF,RMS,SNR,M,N)
      DIMENSION H(0/255),HZ(0/511),HP(0/255)
      REAL MAXIM
      INTEGER Q,L
      Q=N/2
      RMS=0.
      MAXIM=H(0)
      DO 230 I=0,M
      P=RAN(15)
      S=RAN(10)
      IF (H(I).GT.MAXIM) MAXIM=H(I)
      V=S*T(2.*SF*(-ALOG(P)))
220   IF (S.GT..5) V=-V
      HP(I)=H(I)+V
      IF (HP(I).LT.0.) HP(I)=-HP(I)
```

```
      L=I+Q
      HZ(L)=HP(I)
      RMS=(HP(I)-H(I))**2+RMS
230   CONTINUE
      RMS=SQRT(RMS/(M+1))
      IF (RMS.EQ.0.) GO TO 235
      SNR=MAXIM/RMS
235   RETURN
      END
C
C       BOTH ADDS BOTH KINDS OF NOISE
C
      SUBROUTINE BOTH(H,HZ,HP,SF,RMS,SNR,M,N)
      DIMENSION H(0/255),HZ(0/511),HP(0/255)
      REAL MAXIM
      INTEGER Q,L
      Q=N/2
      RMS=0.
      MAXIM=H(0)
      DO 250 I=0,M
      P=RAN(15)
      S=RAN(10)
      R=RAN(12)
      IF (H(I).LT..0000001) VP=0.
      IF (H(I).LT..0000001) GO TO 240
      IF (H(I).GT. MAXIM) MAXIM=H(I)
      VP=SQRT(2.*SF*H(I)*(-ALOG(P)))
```

```
      IF (R.GT..5) VP=-VP
      T=RAN(8)
240   V=SQRT(2.*SF*(-ALOG(T)))
      IF (S.GT..5) V=-V
      HP(I)=H(I)+V+VP
      IF (HP(I).LT.0.) HP(I)=-HP(I)
      L=I+Q
      HZ(L)=HP(I)
      RMS=(HP(I)-H(I))**2+RMS
250   CONTINUE
      RMS=SQRT(RMS/(M+1))
      IF (RMS.EQ.0.) GOTO 255
      SNR=MAXIM/RMS
255   RETURN
      END
C
C                  CONVER CHECKS THE DIFFERENCE BETWEEN
ITERATIONS
C                  AS A MEASURE OF THE CONVERGENCE.
C
      SUBROUTINE CONVER(K,M,N,HN,HOLD,CON,H)
      DIMENSION HN(0/511),HOLD(0/511),CON(1000,4),H(0/255)
      INTEGER P,Q
      P=N/2
      TEMP=0.
      TEMP1=0.
      TEMP2=0.
```

```
      TEMP3=0.
      M=M+1
      DO 550 I=P,M+P
      Q=I-P
      IF (H(Q).EQ.0.) GO TO 545
      TEMP=TEMP+(HN(I)-HOLD(I))**2/H(Q)
      TEMP1=TEMP1+ABS(HN(I)-HOLD(I))/H(Q)
545   CONTINUE
      TEMP2=TEMP2+(HN(I)-HOLD(I))**2
550   TEMP3=TEMP3+ABS(HN(I)-HOLD(I))
      CON(K,1)=TEMP/M
      CON(K,2)=TEMP1/M
      CON(K,3)=TEMP2/M
      CON(K,4)=TEMP3/M
      M=M-1
      RETURN
      END
C
C     SMOOTH DOES THE INITIAL SMOOTHING, AND STORES THE
RESULT IN HN
C                        HN=H*G
C
      SUBROUTINE SMOOTH(N,M,HP,G,HN)
      DIMENSION HP(0/255),G(0/255),HN(0/511)
      INTEGER A,B,C,D,E,F
405   FORMAT(4G)
      DO 420 I=0,M+N
```

```
      HN(I)=0.
      A=0
      B=I-N
      CALL MAX(A,B,C)
      D=I
      E=M
      CALL MIN(D,E,F)
      DO 410 J=C,F
      TEMP=HP(J)*G(I-J)
      HN(I)=HN(I)+TEMP
410   CONTINUE
420   CONTINUE
      RETURN
      END
C
C       MAX RETURNS THE LARGR VALUE
C
      SUBROUTINE MAX(A,B,C)
      IF (A.GT.B) C=A
      IF (A.LT.B) C=B
      IF (A.EQ.B) C=A
      RETURN
      END
C
C       MIN RETURNS THE SMALLER VALUE
C
      SUBROUTINE MIN(D,E,F)
```

```
      IF (D.GT.E) F=E
      IF (D.LT.E) F=D
      IF (D.EQ.E) F=D
      RETURN
      END
C
C     RESTORE DOES RESTORING ITERATIONS
C       HOLD=HN-1
C       HN=HOLD+(HZ-HOLD)*G
C
      SUBROUTINE RESTOR(N,M,HP,G,HN,HOLD,HZ)
      DIMENSION
     S(512),V(1000),HP(0/255),G(0/255),HN(0/511),
     * HOLD(0/511),HZ(0/511)
      INTEGER P
      P=N/2
C       SET UP BRACKET S=(HZ-HN),  AND UPDATE HN,
     HOLD=HN-1
      DO 300 I=0,M+N
      S(I)=HZ(I)-HN(I)
300   HOLD(I)=HN(I)
C       DO CONVOLUTION  V=S*G
      DO 330 I=0,M+2*N
      V(I)=0.
      A=0
      B=I-N
      CALL MAX(A,B,C)
```

```
      D=I
      E=M+N
      CALL MIN (D,E,F)
      DO 320 J=C,F
      TEMP=S(J)*G(I-J)
320   V(I)=V(I)+TEMP
330   CONTINUE
C          ASSEMBLE HN,  HN=HOLD + V
      DO 340 I=0,M+N
340   HN(I)=HOLD(I)+V(I+P)
      RETURN
      END
C
C          ERROR COMPUTES THE ABSOLUTE DIFFERENCE, VARIANCE
C          AND DEVIATION BETWEEN EACH ITERATION AND THE
NOISE FREE
C          ORIGINAL H.
C
      SUBROUTINE ERROR(K,M,N,H,HN,ER)
      DIMENSION H(0/255),HN(0/511),ER(1000,3)
      INTEGER P,Q
      P=N/2
      SUM=0.
      SIGMA=0.
      DO 500 I= 0,M
      Q=I+P
      TEMP=H(I)-HN(Q)
```

```
        SUM=SUM+ABS(TEMP)
500     SIGMA=SIGMA+TEMP*TEMP
        ER(K,1)=SUM/(M+1)
        ER(K,2)=SIGMA/(M+1)
        ER(K,3)=SQRT(SIGMA/(M+1))
        RETURN
        END
C
C       OUTPUT
C
        SUBROUTINE OUTPUT (K,M,N,H,G,ER,CON,SF,RMS,
     * SNR,GTYP,ANS,OFL)
        DIMENSION H(0/255),G(0/255),ER(1000,3),CON(1000,4)
        INTEGER GTYP,ANS,OFL,GFUN,NTYP
        K=K-1
        IF (GTYP.EQ.1) GFUN='FAST CON'
        IF (GTYP.EQ.2) GFUN='SLOW CON'
        IF (GTYP.EQ.3) GFUN='DIVERGE'
        IF (ANS.EQ.1) NTYP='ORD.DEP'
        IF (ANS.EQ.2) NTYP='CONST'
        IF (ANS.EQ.3) NTYP='BOTH'
        TYPE 700,GFUN
700     FORMAT(' THIS G FUNCTION IS ',A8)
        TYPE 710,SF,NTYP
710     FORMAT('  THE SCALE FACTOR WAS ',G,' WITH NOISE TYPE
',A8)
        TYPE 720,RMS,SNR
```

```
720     FORMAT(' THE RMS WAS ',G,' AND THE SNR WAS ',G)
740     FORMAT(4G)
        TYPE 760
760     FORMAT(' THE MEASURES OF CONVERGENCE WERE ')
        TYPE 745,(I,CON(I,1),CON(I,2),CON(I,3),CON(I,4),
     * I=10,K,10)
745     FORMAT(I4,4G)
755     FORMAT(I4,3G)
        TYPE 770
770     FORMAT(' THE AD,VAR,_STD.DEV. WERE ')
        TYPE 755,(I,ER(I,1),ER(I,2),ER(I,3),I=10,K,10)
        TYPE 780,OFL
780     FORMAT(' THE OUTFILE IS  ',I4)
        WRITE(OFL,790) GFUN,NTYP
790     FORMAT(2A8)
        WRITE (OFL,795) SF,RMS,SNR,K
795     FORMAT(3G,I4)
        WRITE (OFL,740)(CON(I,1),I=1,K)
        WRITE(OFL,740)(CON(I,2),I=1,K)
        WRITE(OFL,740)(CON(I,3),I=1,K)
        WRITE(OFL,740)(CON(I,4),I=1,K)
        WRITE(OFL,740)(ER(I,1),I=1,K)
        WRITE(OFL,740)(ER(I,2),I=1,K)
        WRITE(OFL,740)(ER(I,3),I=1,K)
        RETURN
        END
```

```
C
C       GRAPH
C
        DIMENSION CON(1000,4),ER(1000,3)
        INTEGER GFUN,NTYP,IFL,OFL
        TYPE 100
100     FORMAT(' ENTER INPUT FILE NUMBER')
        ACCEPT 110,IFL
110     FORMAT(I)
        READ(IFL,120) GFUN,NTYP
120     FORMAT(2A8)
        READ(IFL,130) SF,RMS,SNR,K
130     FORMAT(3G,I3)
140     FORMAT(4G)
        TYPE 150,GFUN,NTYP
150     FORMAT(' THE G IS ',A8,' WITH NOISE TYPE ',A8)
        TYPE 160, SF,RMS,SNR,K
160     FORMAT(' SF=',G,' RMS=',G,' SNR= ',G,'K= ',I4)
        DO 180 J=1,4
        READ(IFL,140)(CON(I,J),I=1,K)
180     CONTINUE
        DO 190 J=1,3
        READ(IFL,140)(ER(I,J),I=1,K)
190     CONTINUE
145     FORMAT(G)
        TYPE 1000
1000    FORMAT('  DONE READING IN')
```

```
      DO 210 J=1,4
      TYPE 200,J
200   FORMAT(' ENTER THE OUTPUT FILE FOR CON ',I2)
      ACCEPT 110,OFL
210   WRITE(OFL,145)(CON(I,J),I=1,K)
      DO 230 J=1,3
      TYPE 220,J
220   FORMAT (' ENTER OUTPUT FILE FOR ER ',I2)
      ACCEPT 110,OFL
230   WRITE(OFL,145)(ER(I,J),I=1,K)
      STOP
      END
```

REFERENCES

1. R.N. Bracewell, The Fourier Transform and Its Applications (New York: McGraw-Hill Book Company, 1978) pg. 24.

2. Reference 1, pg. 32.

3. Reference 1, pg. 6.

4. Reference 1, pg. 110.

5. G.E. Ioup, Analysis of Low Energy Atom and Molecular Collisions: Semiclassical Elastic Scattering Calculations and Deconvolution of Data, Ph. D. dissertation, University of Florida(1968), pg. 73.

6. J.D. Morrison, "On the Optimum Use of Ionization Efficiency Data", J. Chem. Phys., Vol. 39, No. 1 (1963), pp. 200-207.

7. Reference 5, pg. 73.

8. For all a such that $|a|<1$, $a^n \rightarrow 0$ as $n \rightarrow \infty$, so $H = [1-0] = H$

9. Reference 5, pg. 79.

10. R.N. Bracewell and J.A. Roberts, "Aerial Smoothing in Radio Astronomy", Aust. J. Physics, Vol. 7 (1954), pp. 615-640

11. Reference 5, pg. 75.

12. N.R. Hill, Deconvolution for Resolution Enhancement, (Master's thesis, Dept. of Physics, University of New Orleans, 1973)

13. N.R. Hill and G.E. Ioup,"Convergence of the van Cittert iterative method of deconvolution", J. Opt. Soc. Am., Vol. 66, No. 5 (1976) pp. 487-489.

14. Reference 1, pg. 72.

15. G.E. Ioup, private communication.

16. Adapted from an algorithm given by:
R.J. Higgins, "Fast Fourier transform: An introduction with some minicomputer experiments", Am. J. Phys., Vol 44, No. 8 (1976) pp. 766-773.

VITA

Karin A. R. Wright was born [redacted] to Karl and Betty [redacted]. She attended numerous primary schools in both California and New Zealand, but graduated from Saddleback High School in Santa Ana, California, in 1971. She subsequently attended Santa Ana College then transferred to the University of California at Irvine. At this institution as a chemistry major she earned a B.A. in 1975 and an M.A. in 1977. During her academic career she was awarded three scholarships, one in high school, one as an undergraduate, and one as a graduate student.

Following graduation she worked as an engineer for Douglas Aircraft Company, then, desiring to study physics she enrolled in the University of New Orleans in 1978. While a graduate student for both her M.A. and her M.S. she was a graduate assistant.

She married Cary Lance Wright in 1974 and in 1979 they became the proud parents of Miranda Lorraine Wright.

Following graduation she will be employed as a geophysicist for Shell Oil Company.