View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Lehigh University: Lehigh Preserve

Lehigh University
Lehigh Preserve

Theses and Dissertations

1992

# Nonlinear CODEC in the digital domain

Francis J. Kaczmarczyk
*Lehigh University*

Follow this and additional works at: http://preserve.lehigh.edu/etd

Recommended Citation
Kaczmarczyk, Francis J., "Nonlinear CODEC in the digital domain" (1992). *Theses and Dissertations*. Paper 114.

This Thesis is brought to you for free and open access by Lehigh Preserve. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of Lehigh Preserve. For more information, please contact preserve@lehigh.edu.

AUTHOR:

Kaczmarczyk, Francis J.

TITLE:

Nonlinear CODEC in Digital Domain

DATE: October 11, 1992

# Nonlinear CODEC in the Digital Domain

by

Francis J. Kaczmarczyk

A Thesis

Presented to the Graduate Committee

of Lehigh University

in Candidacy for the Degree of

Master of Science

in

Electrical Engineering

Lehigh University

May 1992

This thesis is accepted and approved in partial fulfillment of the requirements for the Master of Science.

8/11/92
Date

Professor Douglas R. Frey
Thesis Advisor

Chairman of Department

TABLE OF CONTENTS

## LIST OF TABLES

LIST OF FIGURES

## I. ABSTRACT

This thesis discusses a nonlinear CODEC using chaos. This is the first device to nonlinearly encode in the digital domain in this fashion. This device accepts a stream of data and performs a nonlinear mapping to produce the encoded data. This data can then be transmitted over a medium and processed by the decoder which is the inverse of the encoder. The nonlinear chaotic coding allows for information to pass through the channel, but it appears to show no correlation due to its chaotic quality. A 4 bit and 10 bit prototype system are studied in detail using computer simulation. Responses due to constant, periodic, and AR process data are shown, and the chaotic properties of the encoded data are demonstrated.

## II. INTRODUCTION

Information and its transmission over a medium is vital in today's world. Unfortunately, mediums such as telephone lines and the atmosphere are accessible to anybody willing to take the trouble required to listen in. As a result, sensitive information can be picked up by people who have no authorization to use it. This can be harmful or annoying. To combat this, the sender can code his information so that people who have access to his medium cannot automatically have access to his information. Linear codes have accomplished this task for decades. Two ways of encoding signals linearly are (1) to introduce noise to the input signal, or (2) to utilize a linear map on the signal. However, today's high speed computers can "break" linear codes by churning out mathematical functions on the data.

The goal of this work is to investigate a CODEC which produces an output which appears chaotic in the channel and still is able to decode the signal. The CODEC's structure is composed of a feedback encoder and a feed forward decoder of which both incorporate a nonlinear function, a Left-Circulate block. Several plots are shown to demonstrate the randomness of the channel for a 4 bit and 10 bit system. The 4 bit system is completely explored with constant inputs and with a sine wave. A general linear process is used to gain insight into the 10 bit system. Hopefully, these promising results generate interest for further research

into the realm of nonlinear coding.

Before detailing the construction of a CODEC system, perhaps a definition and discussion of chaos is required. "Any deterministic system of equations of motions driven by either a dc or deterministic (not random) input signal is chaotic if one or more of its solution wave forms exhibit a continuous frequency spectrum."[1] One property of chaotic signals is that an attractor can be generated which is a subspace of the entire space. Knowledge of an attractor is like knowing the house in which someone lives, but having no clue as to where that someone is inside the house. Another characteristic of chaos is its extreme dependence upon initial conditions. In other words, two very close starting locations can produce very different signals.

In the digital domain, it can be shown that all signals will repeat in some finite length. In an autonomous system, the maximum length of a sequence before it starts to repeat is the number of possible states of each delay multiplied together. For example, an autonomous system with three 8 bit delays, independent of the configuration, would have at most a sequence which repeats in $(2^8*2^8*2^8)=16777216$. This is derived by counting each of the discrete state possibilities. In other words, how many ways can three numbers be chosen for each delay unit? Each delay unit has a quantity of possible choices, in this case $2^8$. When each of these discrete state possibilities are multiplied

together, they yield the total quantity of ways that numbers can be chosen for each delay unit, the size of the state space.

A digital nonautonomous system cannot exhibit a true chaotic signal either. Since the source signal must be of finite duration, there are a finite amount of possible states for the system. Using similar analysis, the maximum length of the signal can be determined by finding the maximum number of states of the autonomous system and then multiplying by the Least Common Multiple of the length of the sequence of each source. No matter how many sources are added to the system, only one more dimension is generated. For example, if there are two sources, one which alternates between 0 and 1 and the other among 2, 4, and 6, then the repetition length is 6. The sources can be thought of as one with the sequence: 0,2; 1,4; 0,6; 1,2; 0,4; 1,6; 0,2... If 0,2 is declared to be position $x_1$, 1,4 position $x_2$, etc, then the input sequence cycles through the positions of $x_1$ through $x_6$. The new input sequence has a finite length because each of the original sources were periodic.

Since the state space of any discrete system has been shown to be finite, it is impossible to have a truly digital chaotic signal. The signal's relationship to the frequency spectrum is that there must be a spectral term which is much greater than any other. This will occur at a frequency which is equal to 1/period. As the period of the repeating

sequences approaches infinity, the spectral term of greatest magnitude approaches a frequency of 0 Hertz. Note that these limits place constraints on the system, but these limits are such that the system could still appear to be chaotic.

## III. THE CODEC SYSTEM

In the design of a nonlinear CODEC which possesses properties of chaos, it is important to put some thought into the relationship between the encoder and decoder of the system. The block diagram shown below illustrates the relationship of each unit.


(figure 1)

In a successful transmission of a signal, the sequence which is entered into the encoder is the same as the sequence produced by the decoder. Also, it is hoped that the channel's sequence is sufficiently scrambled.

The relationship between the decoder and the encoder is that they are the inverse of one another. Whatever one does the other must undo. Both are essentially digital filters. In order to approach chaos in the channel, these filters incorporate a nonlinear function in their structure.

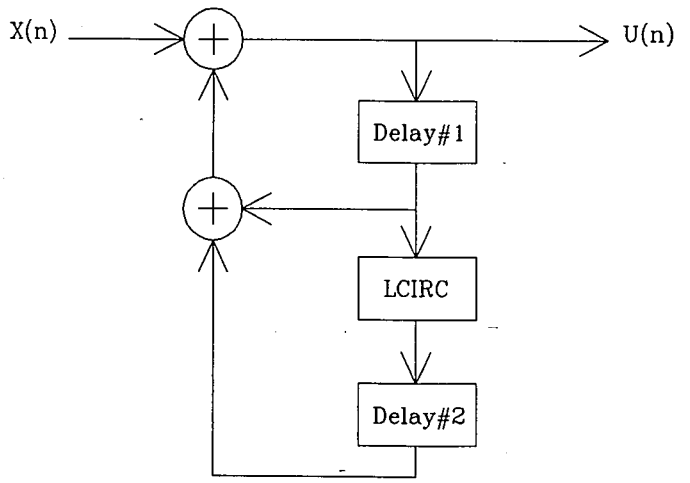
The most basic building block of each of the coders is the delay. If the encoder of the system is accomplishing its task of placing a chaotic signal in the channel, then initial conditions provide several interesting characteristics. Different initial conditions for each delay produce different signals. Hence, the decoder can only decode if it knows the proper starting conditions. In

order to avoid the complications of initial conditions, the encoder is designed to have only feedback and the decoder only feed-forward. Such a configuration allows the decoder to sync into the correct signal. The required sync time is directly proportional to the number of delays in the system. Because the decoder is not using the information in the channel in any feedback, errors in the decoder are not compounded to the point where the decoder follows some other output sequence. When the signal from the channel remains error free long enough to fill each delay with good data, the decoder will then decode the signal properly. Therefore, noise corrupts the output of the decoder, but the length of the bad output which is produced is proportional to the number of delays.

## The 4 Bit Encoder

The key to making a filter become chaotic is to find a nonlinear function which causes the filter to possess "good" properties. The Left-Circulate function, LCIRC, tends to produce the desired characteristics. The following is a block diagram of the design of the encoder:

ENCODER



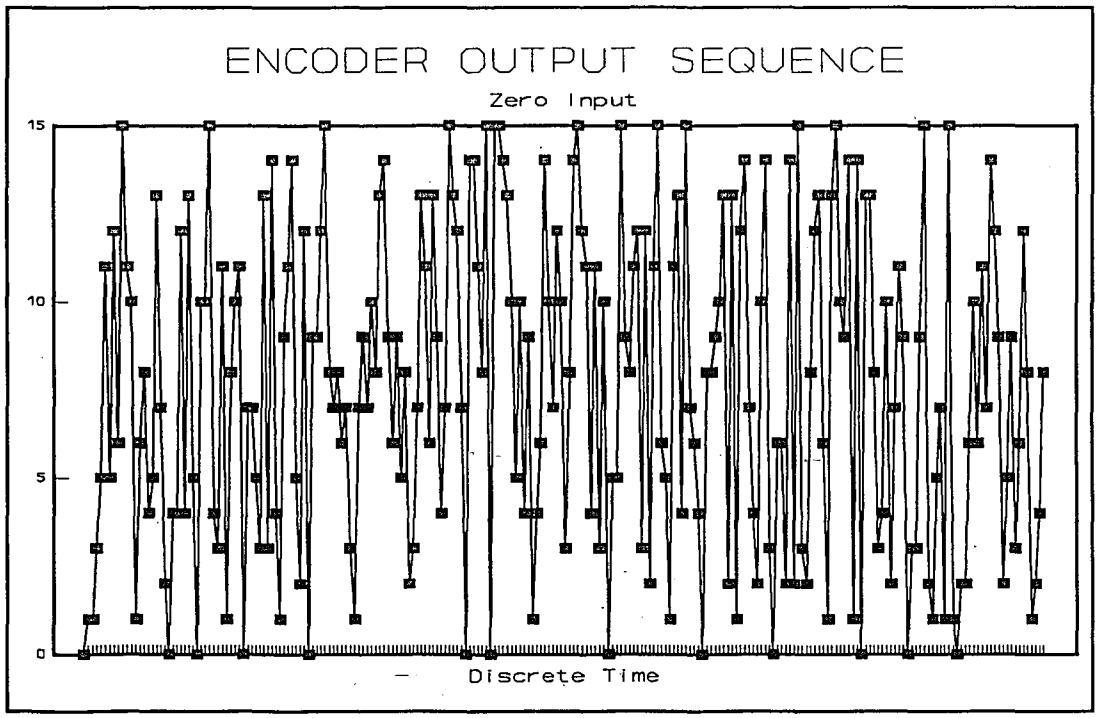

(figure 2)

In order to begin analysis on the system, a 4 bit number will be used. This means that numbers can range from 0 to 15 without using two's compliment notation. The LCIRC will map numbers by left circulating the four bits (ex LCIRC(10)=5). Also, for our beginning analysis the input sequence will be DC.

If the input sequence is set to be 0 for all time, then the filter produces some interesting results. Depending on the initial conditions, the filter produces different

sequences. There are sequences of length 232, 8, 6, 3, and 1. There are also two different sequences of length 6. This accounts for all of the 256 states of the system.

Maybe the most interesting occurrence of initial conditions is $D_1$=0 and $D_2$=0. The output of the encoder remains unchanged at a constant of zero. This is not just unique to the zero input case, since for any constant input there is a set of initial conditions which will cause the output to be a constant.



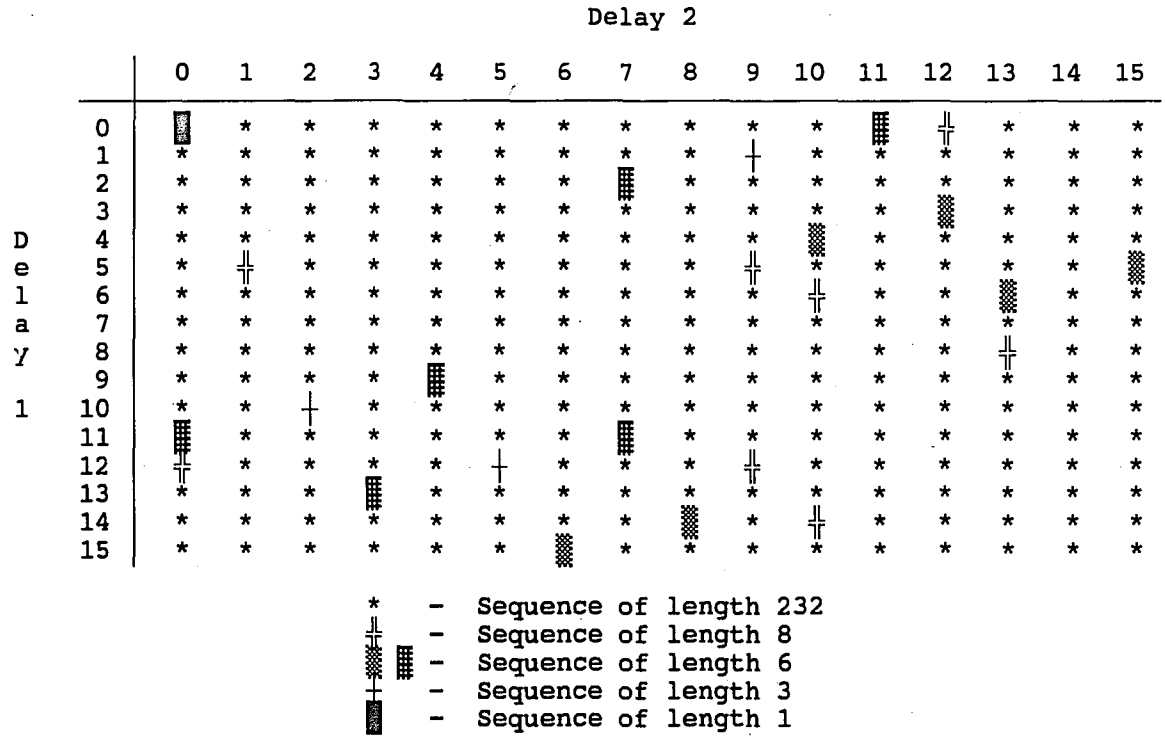

(figure 3)

In figure 3, the longest sequence is graphed versus time. At first glance it seems very unlikely that this is the output of a constant input of zeroes. The initial conditions of this sequence is $D_1$=1 and $D_2$=0. These are not

the only initial conditions which lead to this sequence. There are 231 other possible starting locations which would produce the same sequence.

As seen in figure 4, no matter what the starting states are, there are no states in which a transient is produced. Every state is part of a loop. Note that this is the case for every constant input between 0 and 15, not just the zero input. The diagram is only for the zero input, but it shows

STATE SPACE OF ZERO INPUT RESPONSE

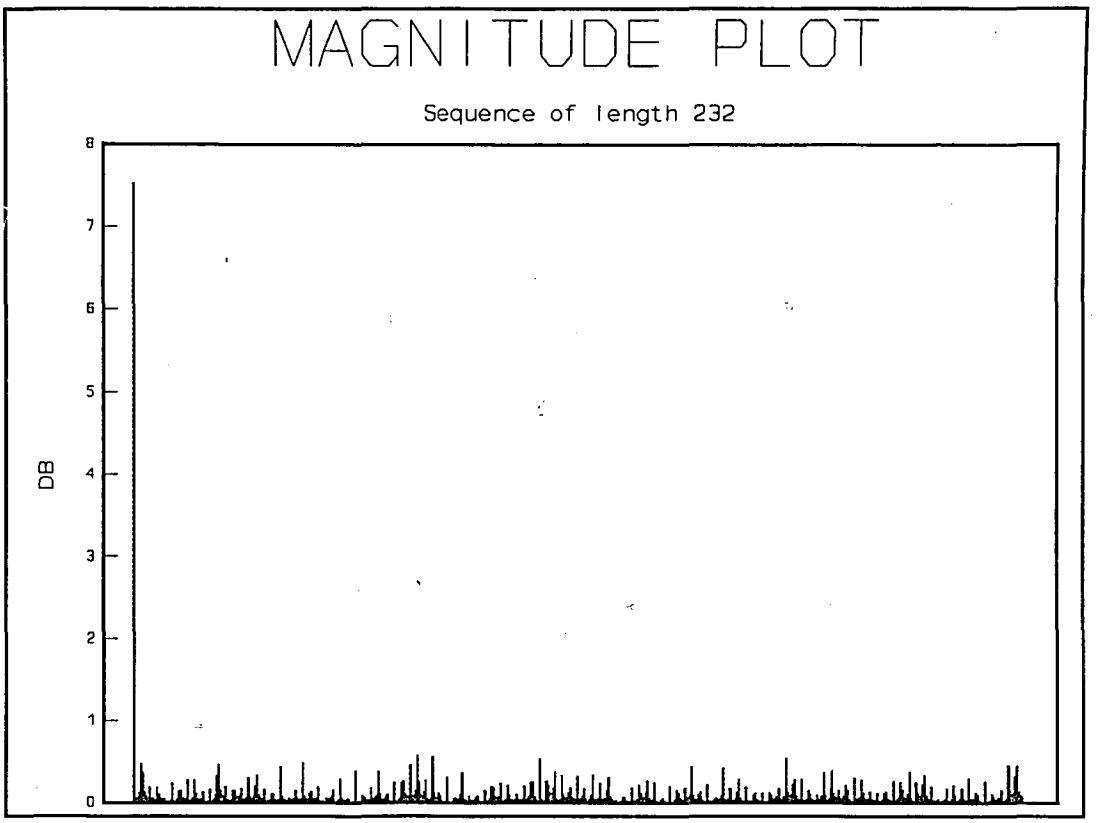



(figure 4)

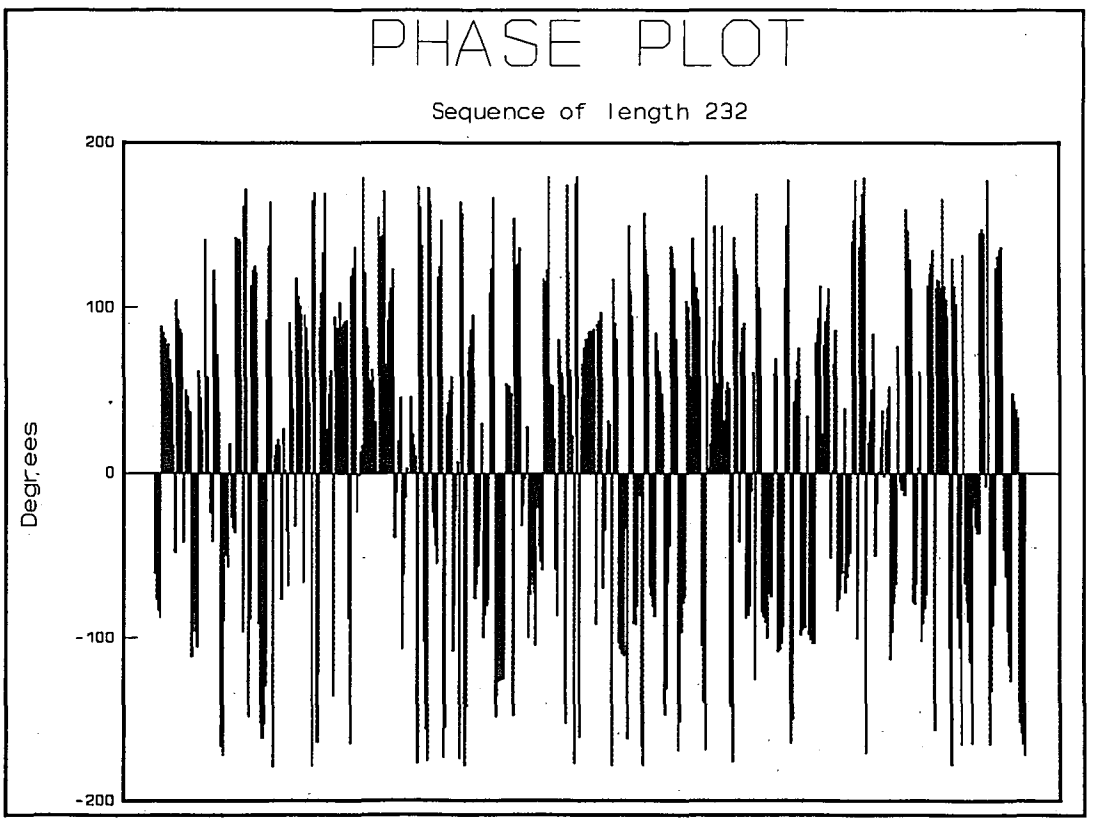
the regions which each of the sequences occupy. A property of nonlinear circuits is bifurcation. The above is not really period doubling, but a constant value shows many

different lengths of periods. All constant inputs have very similar diagrams and properties.

In order to analyze the characteristics of the largest sequence of 232, it follows that a Fourier transform would provide some useful insight. Figure 5 shows the magnitude spectrum and figure 6 the phase spectrum of the signal. Notice that the magnitude spectrum is fairly flat except at DC. This leads to an observation that this sequence does not have any discrete spectral terms associated with it, and that any section of the sequence would appear to be noise. The phase plot supports this observation since the angle is sufficiently chaotic. Such a result is very similar for any constant input in both the magnitude and phase plots.




(figure 5)




(figure 6)

After completely testing the encoder with all the possible constant inputs, the encoder was then tested with the sequence 0, 7, 0, -7, 0, 7, 0, -7, etc. This simple sine wave, $7 \sin[\frac{1}{2} \pi n]$, produces many interesting sequences. Noting that the longest possible sequence is 1024, it is found that the longest sequence which is produced is 412. However, depending on the initial conditions there are a multitude of other possible sequences.

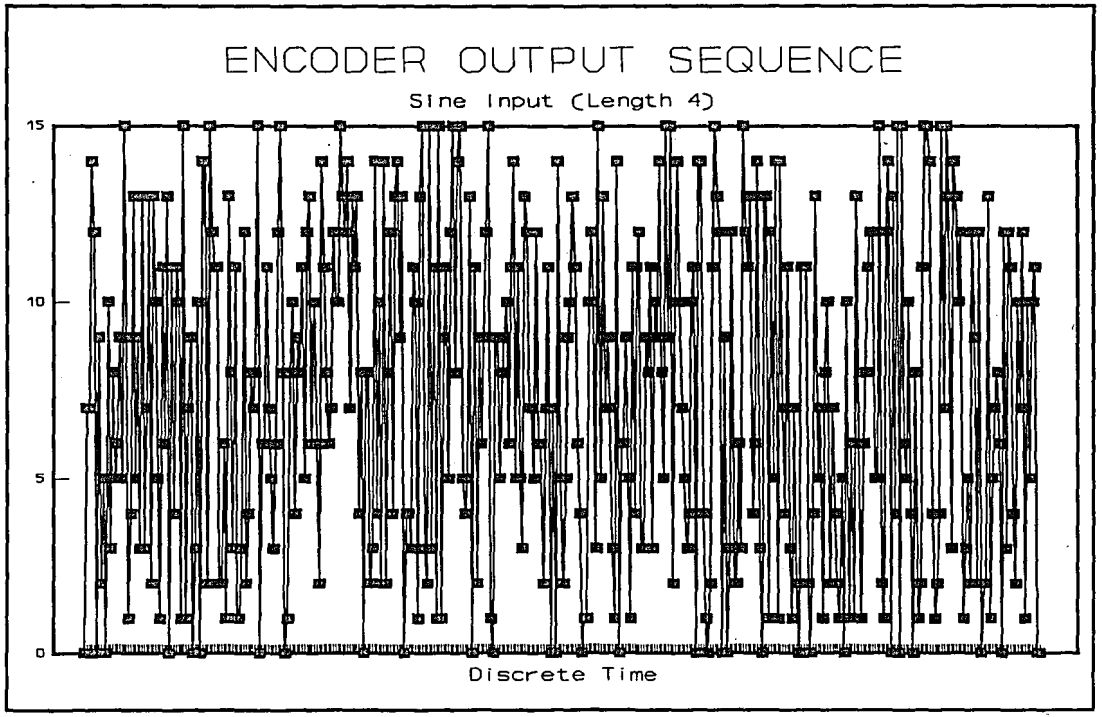
Encoder Sequence Length Table with Sine Input:

| Sequence Length | Quantity | Symbol |
|---|---|---|
| 412 | 1 | * |
| 160 | 1 | + |
| 128 | 1 | # |
| 84 | 1 | ▒ |
| 56 | 1 | ⁞ |
| 32 | 1 | ┼ |
| 28 | 1 | ╬ |
| 20 | 1 | ※ |
| 16 | 1 | § |
| 8 | 3 | ♥♣♠ |
| 4 | 16 | αßΓπΣσμτΦΘΩδ▲φϵ∩ |

(table 1)

The smallest sequence found is one of length four. There were no sequences found which were smaller than the original sequence, and all sequences of loops are divisible by four. The observations are not unique to this sample sine wave. Any repeating sequence will have encoded output sequences which are divisible by the input sequence length. Also, the smallest repeating sequence will be equal to the length of the input sequence. The encoded output sequence

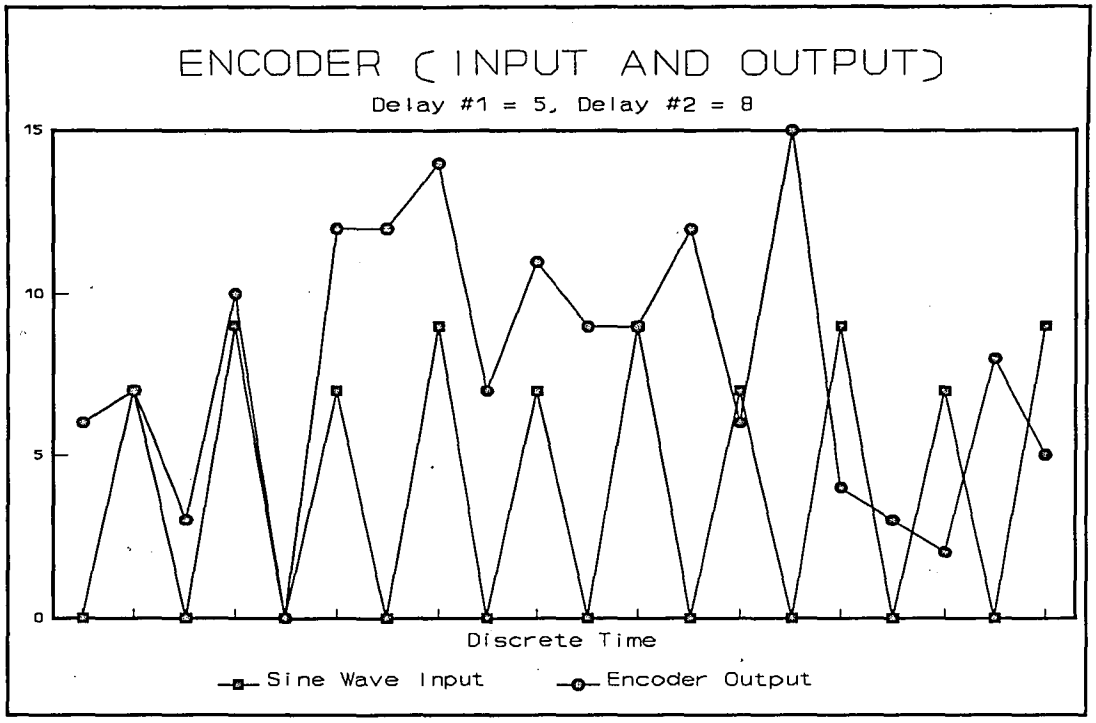
cannot be smaller in length than the input sequence which will be discussed later.




(figure 7)

The diagram above shows the output of the decoder for the longest sequence, 412. The initial conditions for the decoder are $D_1$=0 and $D_2$=0. There appears to be no correlation between this output and to that of the input of the sine wave sequence of 0, 7, 0, 9. Note that the value of 9 in two's compliment is a -7.

In figure 8, the output sequence of length 20 is shown with the input sequence in order to show the relationship between the input and output of the encoder. Notice that even in this short output sequence, the encoder produces numbers unrelated to the input.

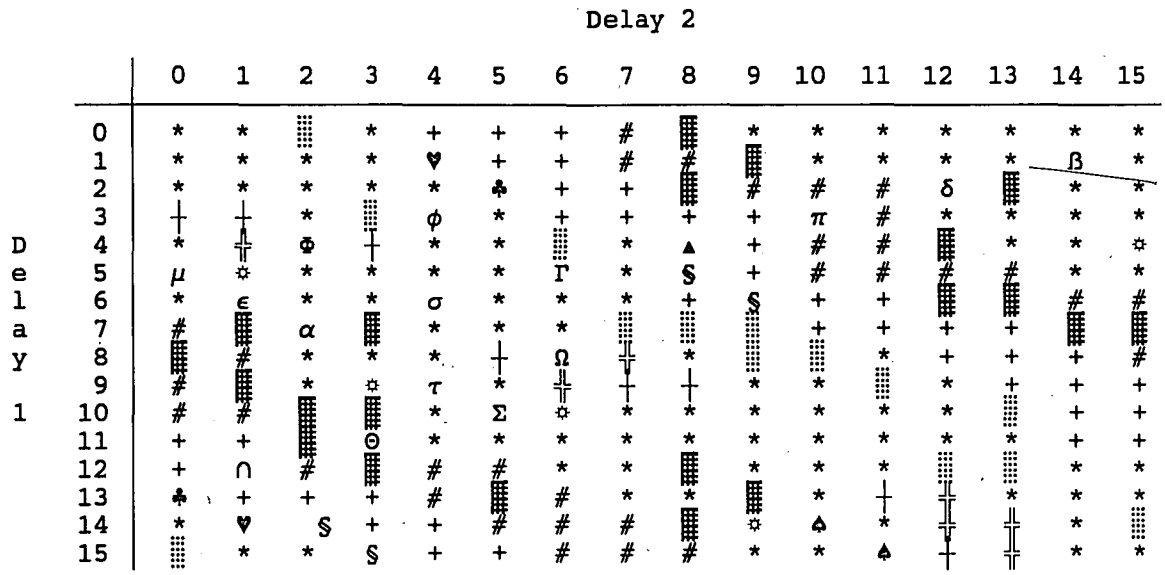



(figure 8)

The state space in figures 9 through 12 is shown with each of the possible inputs. The plots work as follows: knowing the states of the two delays and the position in the input sequence, the attractor formed can be seen on each of the two dimensional subspaces associated with the position in the input sequence. Define a "plane" to be the state space formed by each of the delay units of the encoder circuit. There is a plane associated with each input value in the sequence. Since the input sequence repeats in a length of 4, there are 4 unique planes of the state space. Depending upon the initial conditions of the encoder, different attractors are found in the state space.

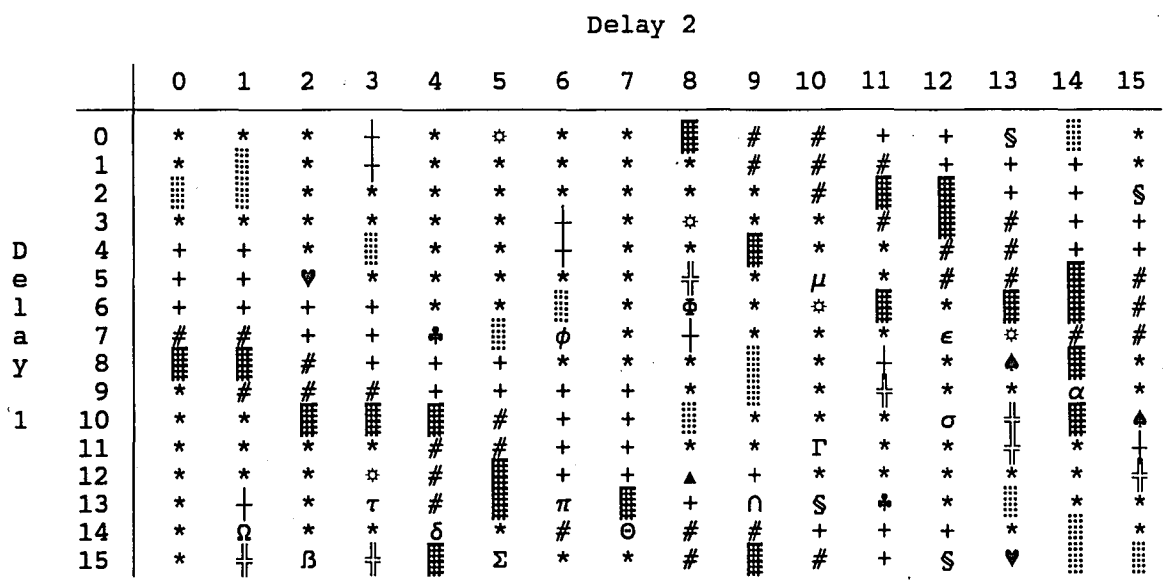
## STATE SPACE OF SINE WAVE INPUT RESPONSE

Input = 7 Sin[½ π 0] = 0




(figure 9)

Input = 7*Sin[½ π 1] = 7

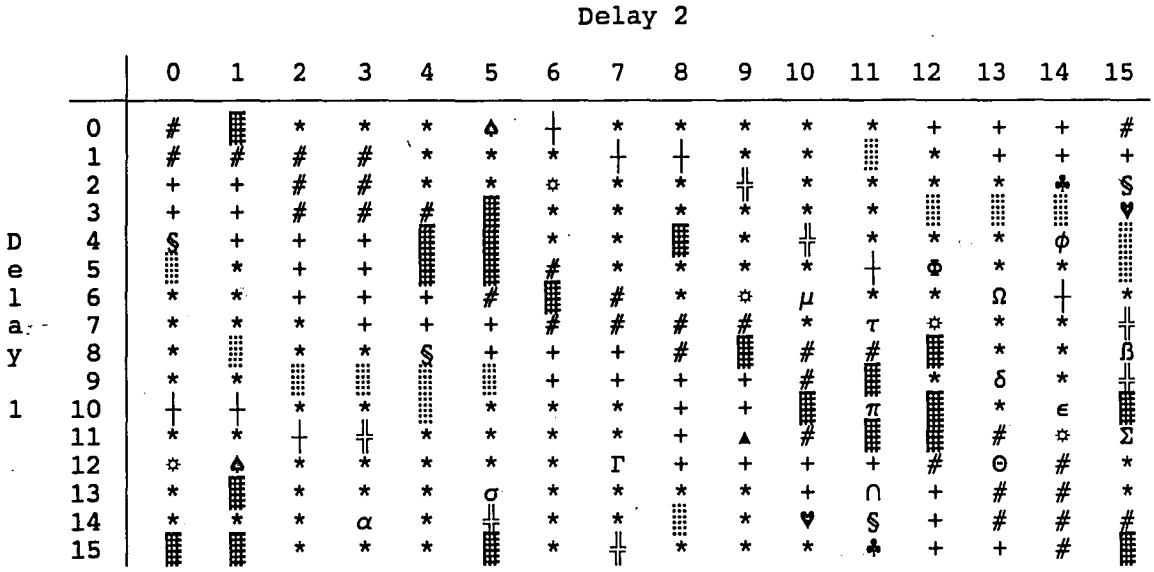



(figure 10)

Note: See table 1 for symbol reference

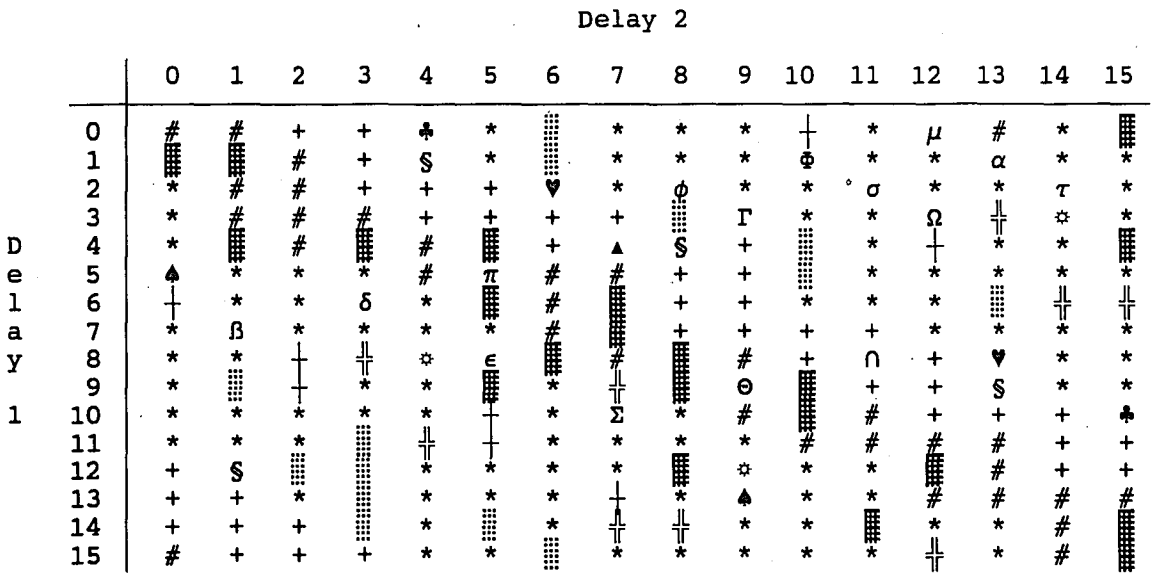
# STATE SPACE OF SINE WAVE INPUT RESPONSE

Input = 7*Sin[½ π 2] = 0




(figure 11)

Input = 7*Sin[½ π 3] = 9




(figure 12)

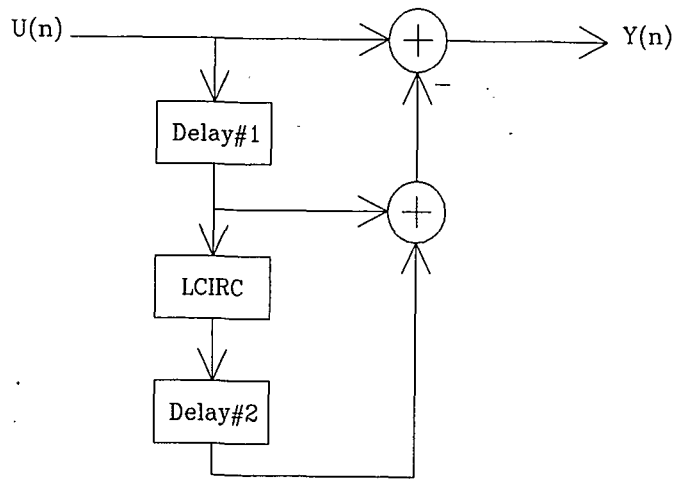
Note: See table 1 for symbol reference

Longer input sequences would only generate more planes. This partly explains why the length will always be evenly divisible by the length of the input sequence. A better understanding can be obtained by studying the decoder.

## The 4 Bit Decoder

The decoder circuit is very important in the CODEC's operation, even though it appears rather simple in nature. Without the ability to decode in a straightforward manner the system loses practicality. Also, the decoder's best characteristic is that one piece of bad data does not keep it from producing the correct decoded sequence. This is not to say that corrupted input does not produce corrupted output. Rather, after the input data stream is no longer corrupted, the decoder will be able to lock up such that the decoded message is completely correct after a finite error block.

DECODER




(figure 13)

The decoder shown above operates perfectly with the encoder to successfully transmit messages in simulation. This 4 bit CODEC system worked without flaws for all the

data tested. These promising results encourage further testing of the encoder to better understand its behavior with a greater variety of inputs.

Working backwards with the knowledge of the decoder above, it can be shown that a constant input causes the decoder to produce a constant output. This is due to it's feed forward structure. These results can be applied to any sequence entering the decoder. If a sine wave of length 4 enters the decoder, it will produce a repeating sequence of no longer than length four. As proven below, to enter a sequence of numbers into the decoder and have it produce a longer stream than was entered, is impossible.

In the decoder circuit in figure 13, the equation for y(n) is: y(n)=u(n)-u(n-1)-LCIRC[u(n-2)]. Since the value of y(n) is a memoryless mapping of the inputs, the equation can be reformulated to y(n)=F[**u**] where F[·] is a mapping a the vector **u**, $\{u(n),u(n-1),u(n-2)\}^T$. A given vector of inputs, **u**, always maps to the same scalar, y. Therefore, the maximum output sequence of y(n) cannot be greater than u(n). In general, any feed forward system has the form $y(n)= C_0\,u(n) + C_1\,u(n-1) + \ldots C_N\,u(n-N) + F[u(n), u(n-1), \ldots u(n-N)]$. This memoryless mapping of a vector of inputs to a scalar output, always has the property that the length of the output sequence cannot be longer than the input sequence.

However, it is possible for the decoder to take a long stream of information and decode it into a smaller sequence. Since the encoder is the inverse of the decoder, the encoder cannot produce a smaller sequence than the one which was entered. If the encoder could produce a stream with a smaller period, then the decoder would have to be able to increase to stream back to its original length. This has been shown not to be the case.

## IV. THE 10 BIT CODEC

The width of the bit bus has a direct relation to the length of the output signal and to the number of discrete states of the system. A revised encoder is now considered with a 10 bit bus. The allowable numbers are defined to be in the range of -512 to +511. This assumes that the ten bits are thought of as two's complement where the first bit is the sign bit.

Using the same encoder configuration as before, the task of displaying the state space is more difficult due to the increased number of states, $1024^2=1048576$. Once again, it is important to note that no matter what the encoder is doing it cannot produce a completely chaotic sequence, but the output produced has an almost flat magnitude spectrum.

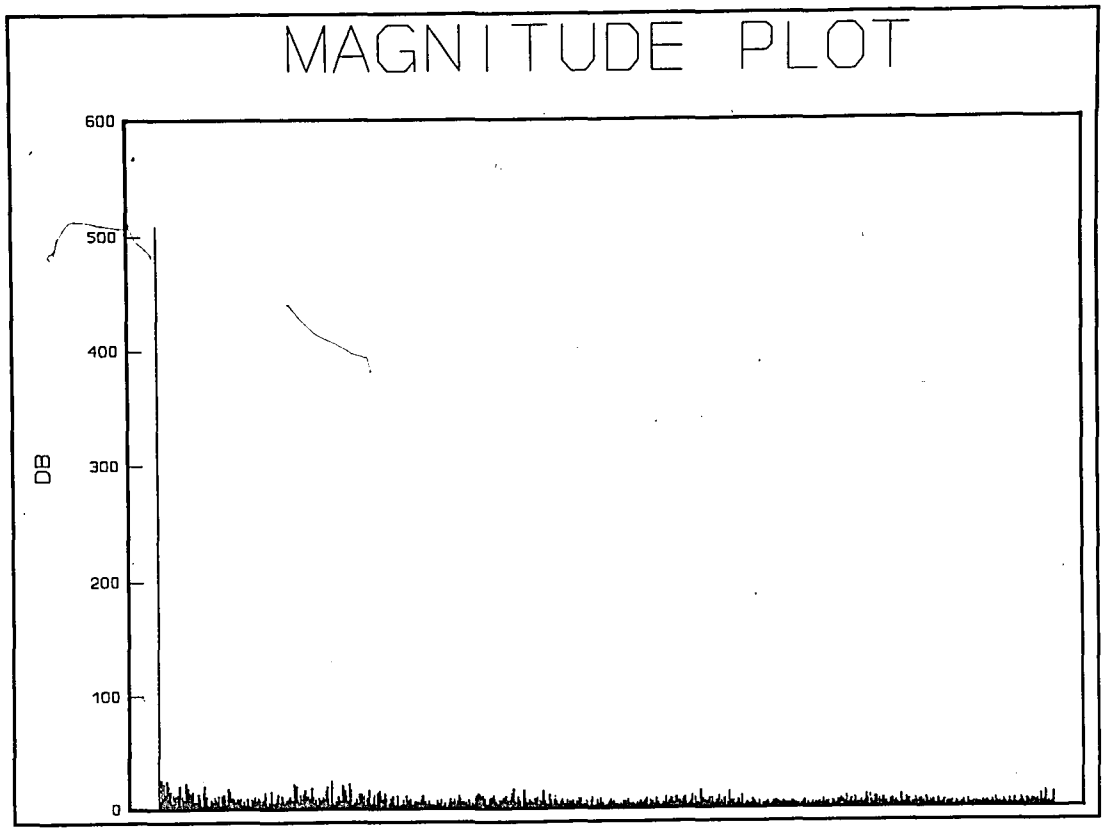
The autonomous system of the encoder will still produce a constant stream of zeroes if the initial states are both zero ($D_1=0$, $D_2=0$). The possible initial conditions for this simple system has now jumped to the millions. This complicates the problem of analyzing the system completely for each of the states.

Similarly, the encoder mentioned above with constant inputs is very hard to analyze. However, some generalities between the 4 bit and 10 bit system can be made. There are attractors in the state space which are of various sizes. Furthermore, there is always the possibility that there is a

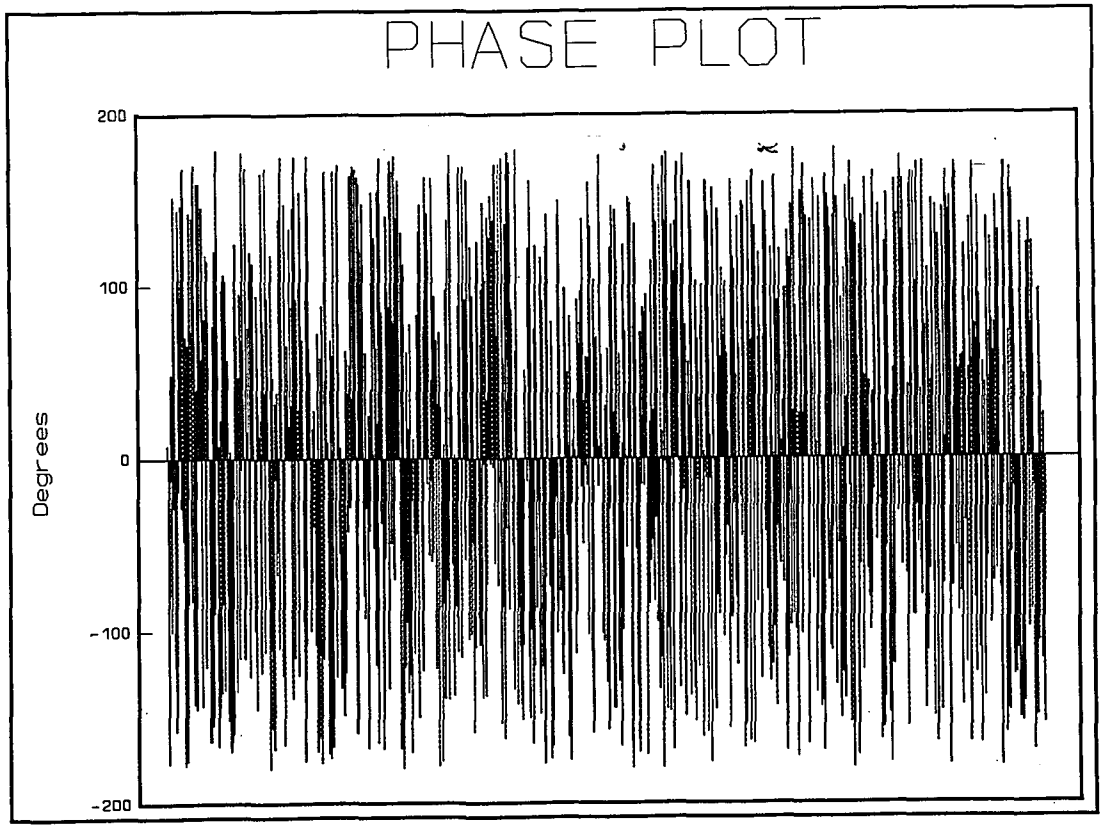
set of initial conditions in which a constant input produces a coded sequence which is a constant output.

As the sequences increase in length, it is known that the properties are the same as the 4 bit system. Using the same argument as above, the length of the encoded sequence cannot be smaller than the input sequence.

One test, as discussed before, is to analyze the output of the encoder by running it into a magnitude and phase spectrum analyzer. A variety of initial conditions were tested with the autonomous encoder. In figure 14, the magnitude plot is shown and in figure 15 the phase plot is shown for an initial state of $D_1$=1, $D_2$=0. With the increase in precision (10 bits), the length of the possible sequences is greatly increased. Since the sequence is too long for the program to analyze, a section of the sequence was used in order to determine the magnitude and phase. The diagrams show that the magnitude spectrum is rather flat, except for at DC. This observation suggests that the system output is very close to white noise. In addition, the phase spectrum does not have a pattern associated with it.



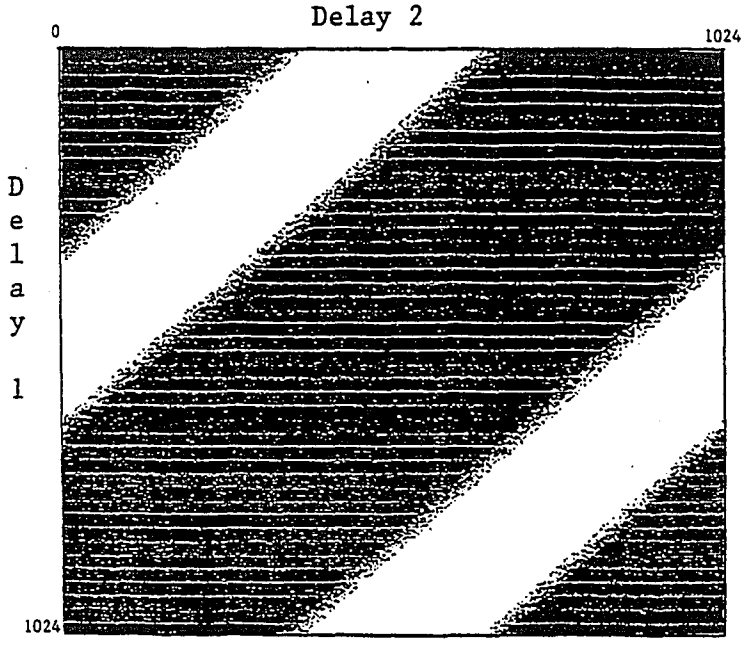

(figure 14)




(figure 15)

The state space allows attractors to be shown for different combinations of initial conditions. The quantity of possible states is sufficiently large such that not all the subspaces can be shown on the same plane without confusion. In figure 16, the subspace shown is for the autonomous encoder if the initial conditions are set to $D_1=1$, $D_2=0$.

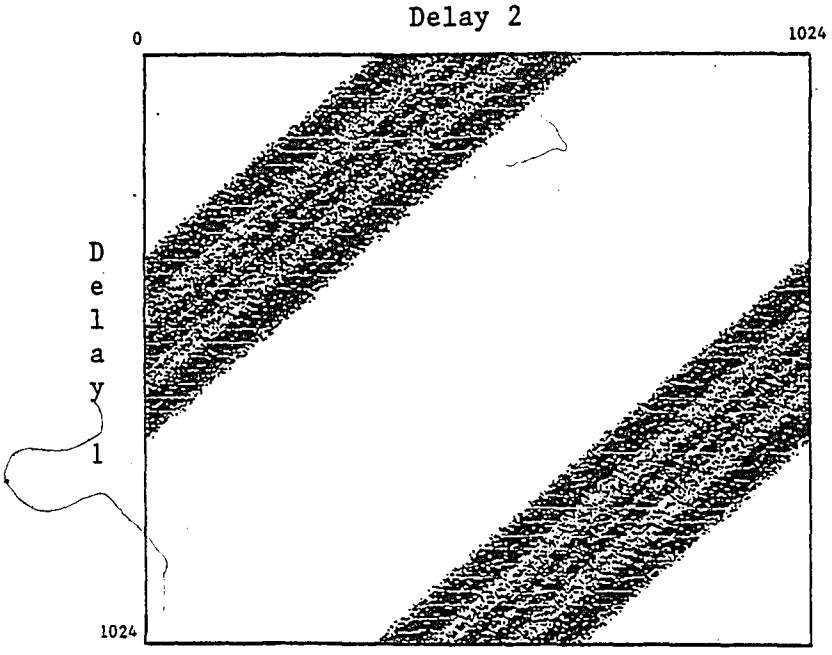
STATE SPACE OF ENCODER
($D_1=1$, $D_2=0$)




(figure 16)

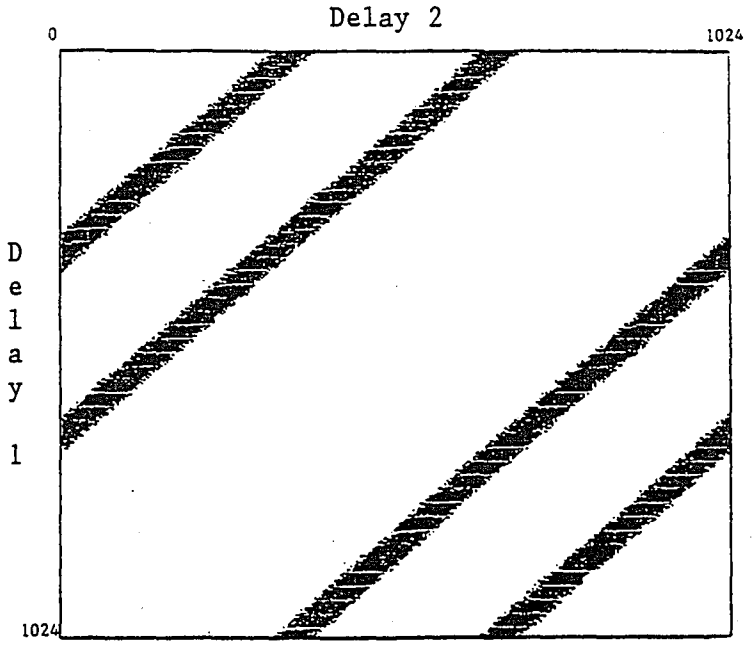
The subspace shown above is an attractor which occupies 488854 states. It is important to compare this subspace with some of the other possibilities that the autonomous encoder produces. Figures 17 and 18 show other attractors with different initial conditions.

STATE SPACE OF ENCODER
($D_1$=600, $D_2$ =0)




(figure 17)

STATE SPACE OF ENCODER
($D_1$=380, $D_2$ =0)

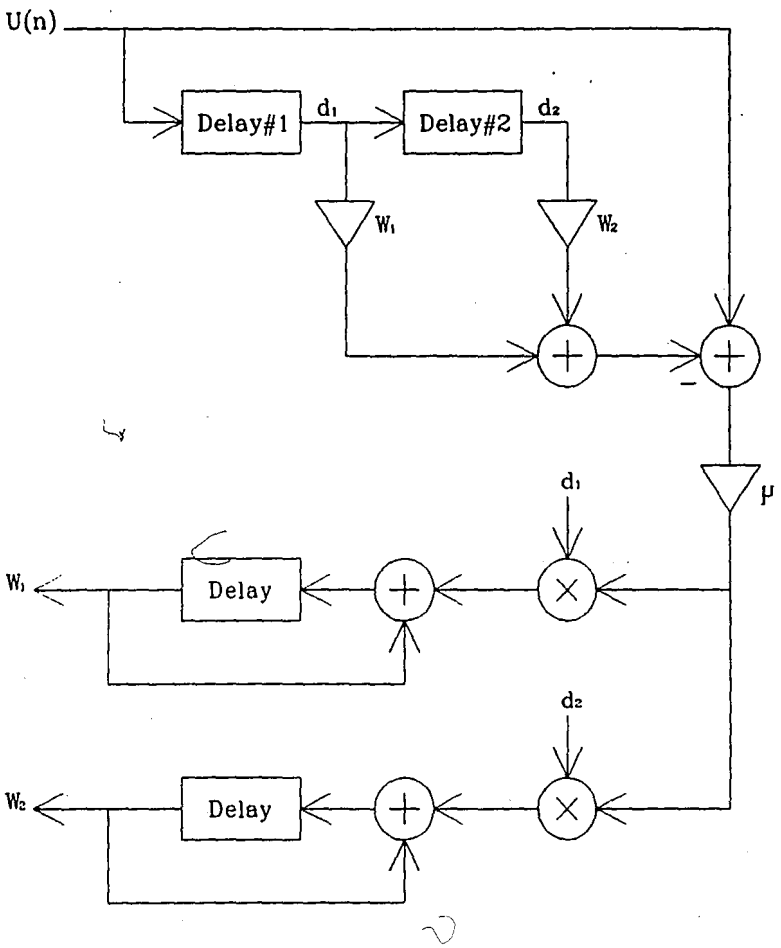



(figure 18)

## Security - Are There Detectable Statistics in the Channel?

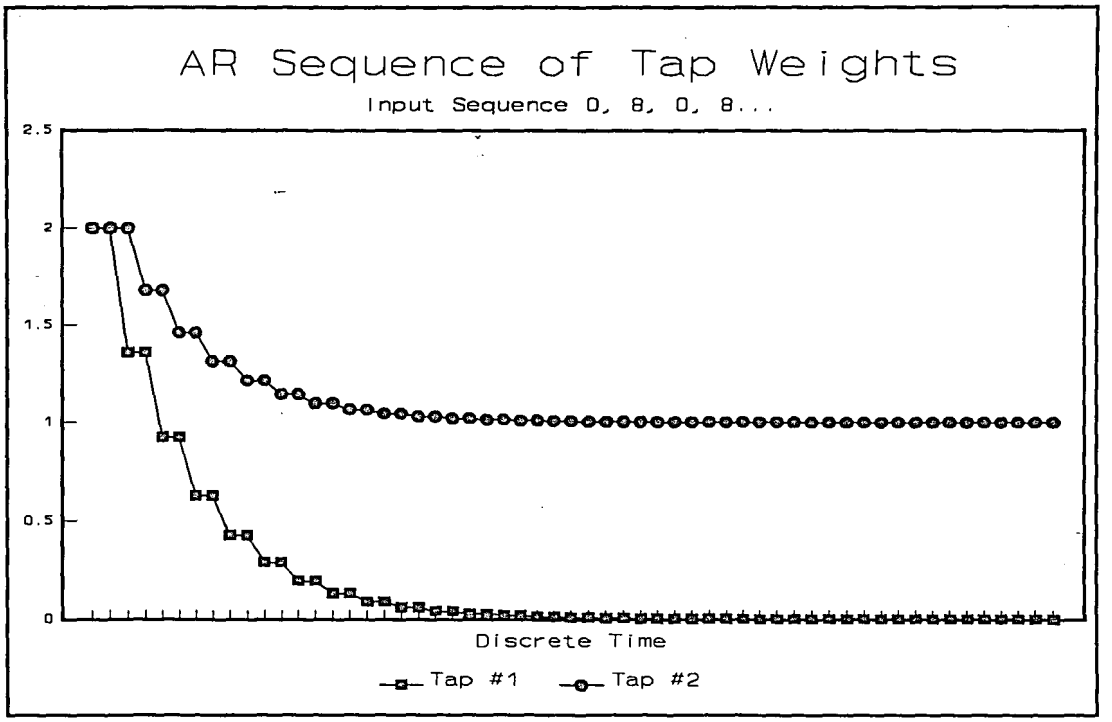
The desirable features of an adaptive filter which make it a powerful device for signal-processing and control applications are its abilities to operate satisfactorily in an unknown environment and to track time variations of input statistics. An adaptive filter is used to provide a linear model that represents the best fit to an unknown collection of data that the encoder is producing in the channel. This linear filter analyzes the data utilizing a Least-Mean-Square adaptive algorithm. By automatically changing the tap weights in the filter, the algorithm is able to successfully compute the best set of tap weights. These tap weights show the statistics of the data entering the adaptive filter. If the tap weights, which can be thought of as a vector, approach zero in every position then the data entering the adaptive filter does not show much correlation. The tap weights approaching zero also signifies that the low order correlation of the data is zero. Another possible occasion where a chaotic sequence is detected by the adaptive filter is when the tap weight vector never becomes stable. One could conclude that the signal might have some local correlation. However, the forward prediction error filter is probably of insufficient length or the constant $\mu$ is too large, causing instability in the Least-Mean-Square algorithm.

ADAPTIVE FILTER STRUCTURE



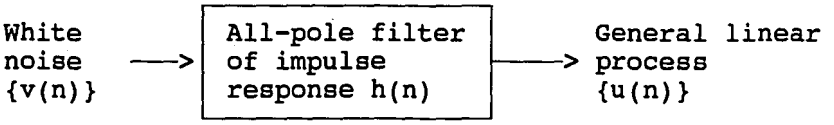

(figure 19)

In the diagram shown above of the forward prediction adaptive filter, the feedback structure allows for the readjusting of the tap weights. In order to show how the filter works under perfect conditions, the sequence of 0, 8, 0, 8... is entered into the filter. The graph in figure 20 shows how the adaptive filter starting with tap weights of 2 and 2 converge to a new setting of 0 and 1.




(figure 20)

The above sequence is highly correlated, however, which is rarely the case in the real world. By feeding an all pole filter with white noise, it is possible to generate a general linear process.

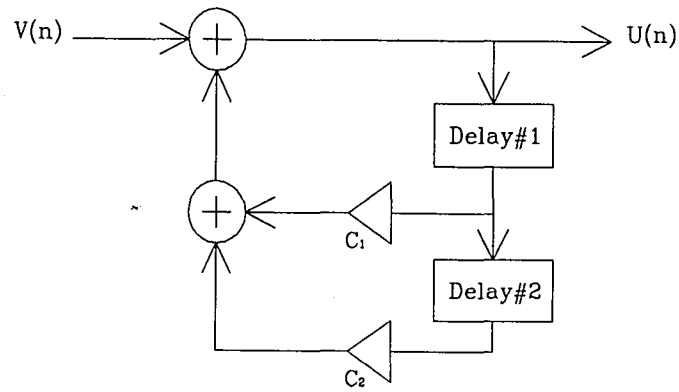



(figure 21)

This process can then be thought of as a speech signal because this signal has correlation due to the filter. The feedback coefficients of the all-pole filter which is producing this response from white noise are unique. If the adaptive filter is operating properly, it should lock into

feedback coefficients of the all-pole filter. Figure 22, shows the configuration of the all pole filter.

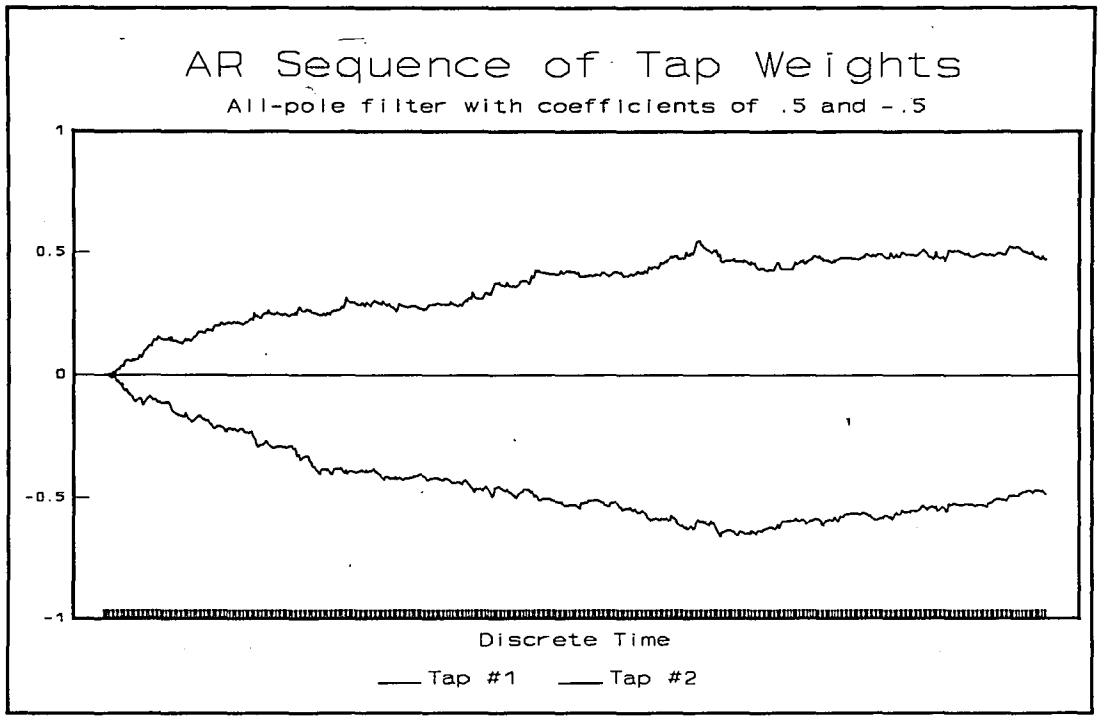
All-Pole Filter Structure




(figure 22)

In order to utilize this filter with a computer, the white noise can be generated by allowing the computer to produce random numbers. The output of the filter is a linear process. This process can then be fed into the same adaptive filter as described before. If the adaptive filter is operating properly, the tap weights will approach the value of the feedback coefficients of the all-pole filter.

If the coefficients of the all-pole filter are set to .5 and -.5, and the initial states of the adaptive filter are set to be 0 and 0, then the following tracking of the adaptive filter is observed in figure 23. Notice how the adaptive filter is approaching the correct values for the coefficients. The rate of convergence to the correct solution is affected by the adaptation constant, $\mu$.



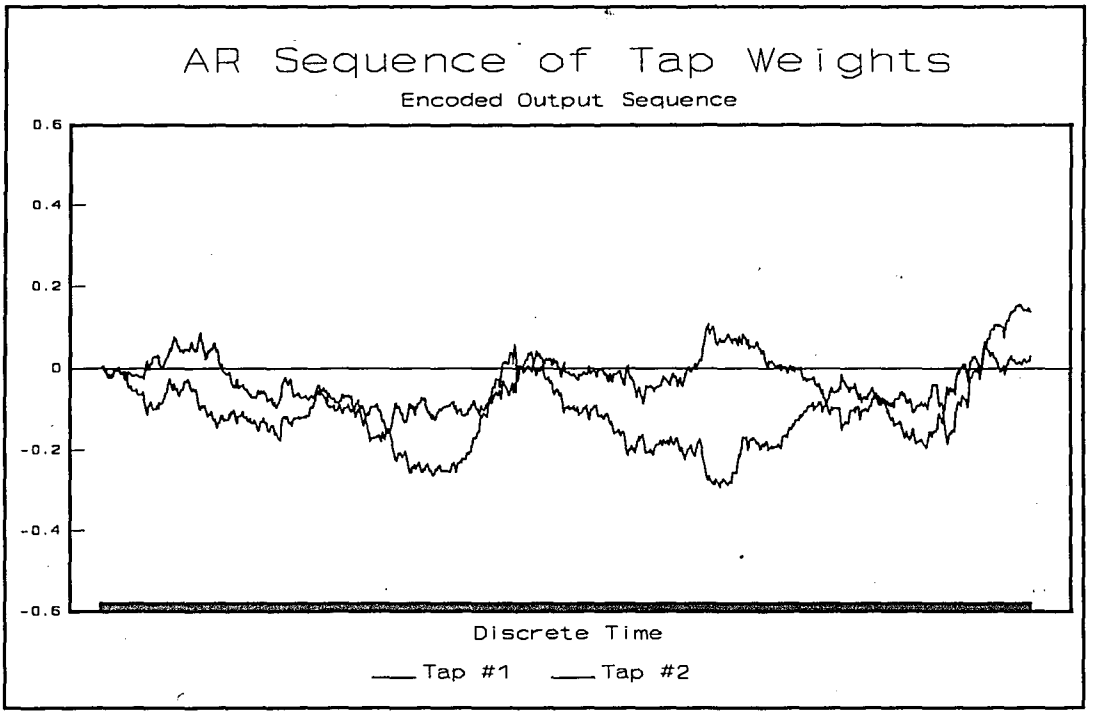

(figure 23)

However, if $\mu$ is not small enough, the filter's tap weights may not converge. Since the filter is constantly trying to correct itself, the correct solution will never exactly be observed, instead it will be randomly approximated to within some small error.

By utilizing the general linear process generator with two taps of .5 and -.5 to simulate some characteristic signal such as speech, the signal can be fed into the ten bit encoder. The coded signal can then be tested with the adaptive filter as a check to determine the characteristics of the chaotic signal in the channel.

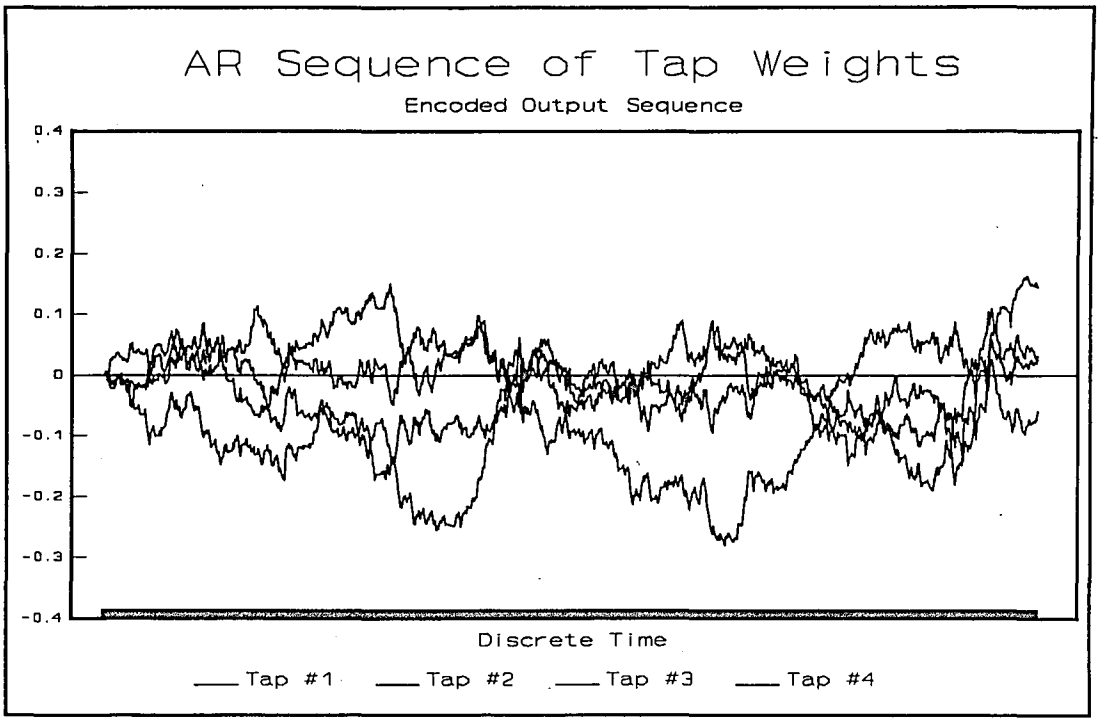
The adaptive filters' initial conditions are set to zero, and the algorithm attempts to determine tap weights for the process. In the graph shown below, several

observations can be made about the encoded signal. The tap weights do tend to show some local correlation of the encoded signal. By decreasing the value of $\mu$, the correlation is less apparent. Also, the adaptive filter only has two taps.




(figure 24)

If the number of taps are increased on the adaptive filter to 4, then some information might be able to be concluded based upon the same linear autoregressive process encoded and fed into the adaptive filter. The graph seen in figure 25 shows that there is no further information to be gained by increasing the number of taps. The tap weights seem to have some local correlation, but for the most part the vector weight is less than .1 in magnitude for the data shown in the above graph.



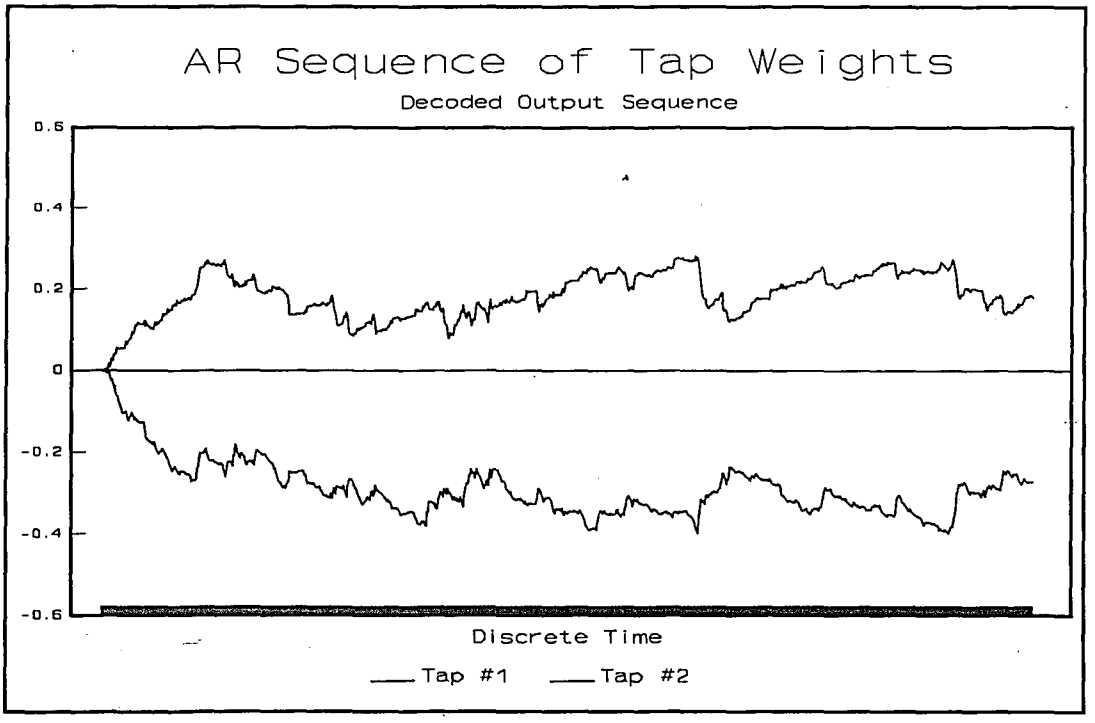

(figure 25)

The same sequence was also fed into an adaptive filter with eight tap weights utilizing the Least Mean Square adaptive algorithm. The results were similar to the four tap filter except with even smaller values of tap weights.

Another way of looking at the encoder circuit is to consider the LCIRC function as a multiplication of 2 and an occasional 1 added in as a noise component. Treating the encoder in this manner describes the encoder as an unstable system because there is a pole outside the unit circle. Due to encoder's configuration it is possible to imagine occasions when the system appears to be stable and the amount of noise introduced into the system is relatively low in magnitude.

However, no matter how one views the encoder, the results are very typical. This is the case for more than the general linear process with the taps of .5 and -.5; several other general linear processes were tested to demonstrate similar results of the encoded sequence. Some of these graphs are included in the appendix for further evaluation.

In a way similar to the testing of the encoder, the decoder was tested with the same general linear process with taps of .5 and -.5. By applying a general linear process to the decoder, a surprising result is discovered when the adaptive filter analyzes the output of the decoder. The adaptive filters tap weights approach a constant value in the range of .2 and -.3 as shown in figure 26.




(figure 26)

Remembering that the analysis performed is on the decoder circuit, the main determination is that the decoder should never be used as the encoder of the system. This is because the decoder would not produce a chaotic signal in the channel.

The principle of the decoder is to take a chaotic sequence in the channel and decode the sequence back into the original. Therefore, a signal which has some correlation already associated with it should become more correlated. This is due to the decoder's finite impulse response structure. Remember, the decoder system is incapable of producing longer sequences. Even though the adaptive filter approaching constant values from the decoder's output sequence was surprising, it might have been expected.

The decoder circuit was checked with several general linear processes. These additional adaptive filter tests on the decoder always caused the top weights to approach a constant value. This effectively guarantees that the results of the decoder correlating signals are consistent regardless of the general linear process chosen.

# V. CONCLUSION

Since most of the time the encoder is producing nearly chaotic signals in the channel, it is very important to have as noiseless a channel as possible. If the channel is extremely noisy, the probability that enough correct information will be passed is quite low. Even though the decoder has the capability of self correction, it only exists when data entering is relatively error free. It takes a finite amount of time before any error entering the decoder is no longer corrupting its own output.

Incorporating error correction codes in the transmission of the channel can aid in the ability of the decoder to produce the correct output. Note that there is always a practical limit as to how much error correction can be provided; thus, corrupted data can still enter the decoder. Therefore, careful consideration must be taken when actually implementing the system.

The CODEC which has been described so far has a multitude of exciting features, most of which center around the encoder's ability to produce nearly chaotic signals in the channel in real time. However, contingent upon the state of the delay units and the input signal, the encoder's performance, i.e., its ability to produce a chaotic-like output sequence, may not be very good. Therefore, important transmissions might not always be encoded in a highly chaotic manner. A slight modification to the encoder, as

described below, can allow the encoded signal to retain its high non-correlation properties.

Depending on what type of signal the encoder was attempting to code, several schemes could be used to keep the system producing nearly chaotic signals in the channel. The encoder could analyze the data it is outputing to the channel and add small changes to the states of the system when chaos is not present. If there were no modification made to the decoder, it would respond to the encoded channel as if it were corrupted. However, the decoder would sync up after the proper delay. If the signal being transmitted were audio or video, the receiver might only produce a momentary "pop". The human ear or eye would probably not even catch the mistake. This allows for the expense to only be spent at the encoder while keeping the decoder circuit simple and cheap.

The above scheme does not work if the data being encoded must be 100% correct when decoded. For example, if a data stream from a computer is being sent via modem, it is very important not to corrupt the data in the above format. However, there are other improvements that can be made if some expense is added to the decoder.

The CODEC is a working encoder/decoder system which operates in real time on signals and data. The encoder produces signals which are nearly chaotic in the channel. These signals are not easily decoded unless the nonlinear

function is known a priori. Also, it is necessary to know how many delays are associated with the system in order to even begin to guess a solution. However, because of the simple structure of the encoder, it is easy to change the nonlinear function. If the nonlinear function is being implemented via a look-up table, EPROM, then the function can be switched easily. The EPROM in the encoder and the decoder only need to be changed to one with a different look-up table.

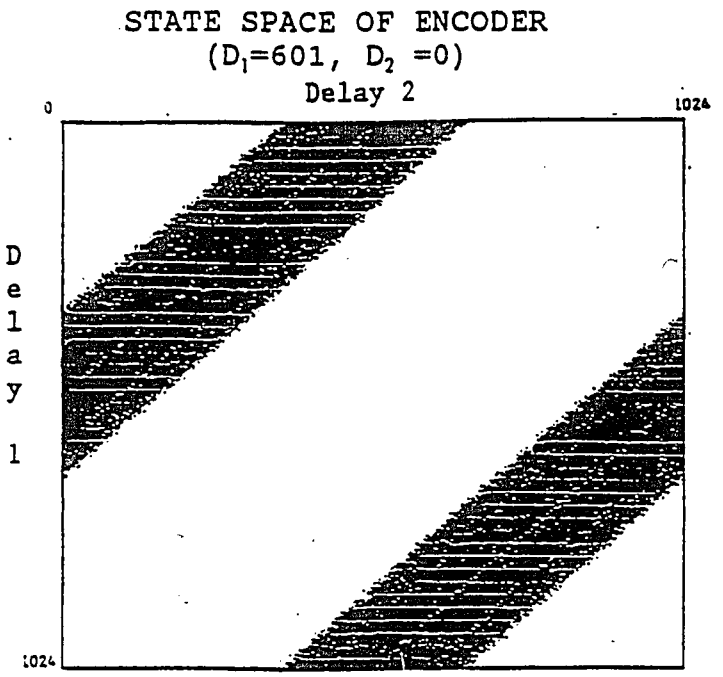
All linear codes can be broken in some finite time although this time might be significantly long. However, this code is based upon a nonlinear function in the feedback. This nonlinear function creates a problem in using the typical linear techniques.

To construct a decoding device to "break" the nonlinear code, there are several assumptions one must make. Therefore, the "code breaker" would not only have to guess the nonlinear function, but also the correct number of delays. Note that the nonlinear function, if implemented with a look-up table, could be nothing more than a set of random numbers. This forces the "code breaker" to build a look-up table of his own. In order to generate such a table, he needs to know the input before it is encoded. Furthermore, since the configuration is not known, the entire feedback will have to be a look-up table. This should complicate the scheme sufficiently to discourage

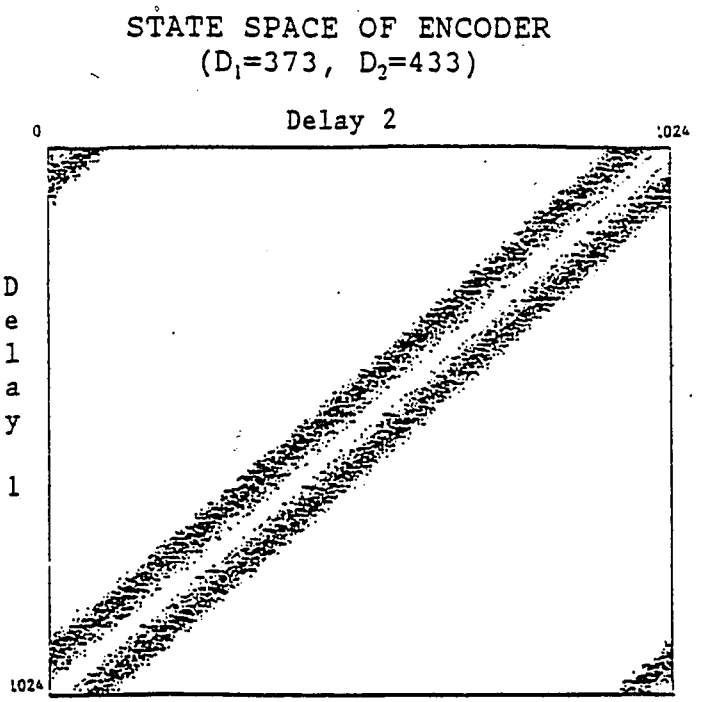
"code breakers."

Finally, the CODEC is a practical encoding/decoding system which works in real time and possesses chaotic coding characteristics in real time implementation. The system has been demonstrated with both 4 bit and 10 bit CODECs. Both systems were simulated with several types of input, and in all cases the encoder/decoder combination showed very promising results. The modification of the nonlinear function could produce some interesting results and could improve the quality of the channel.

## VI. APPENDIX

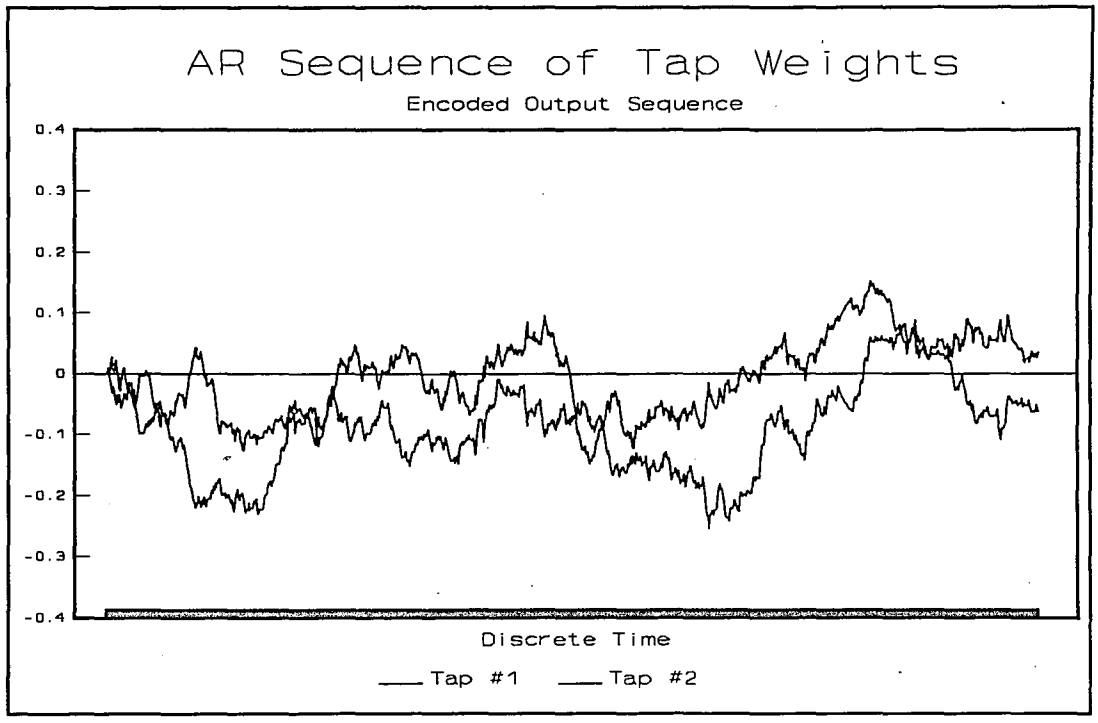



(figure 27)

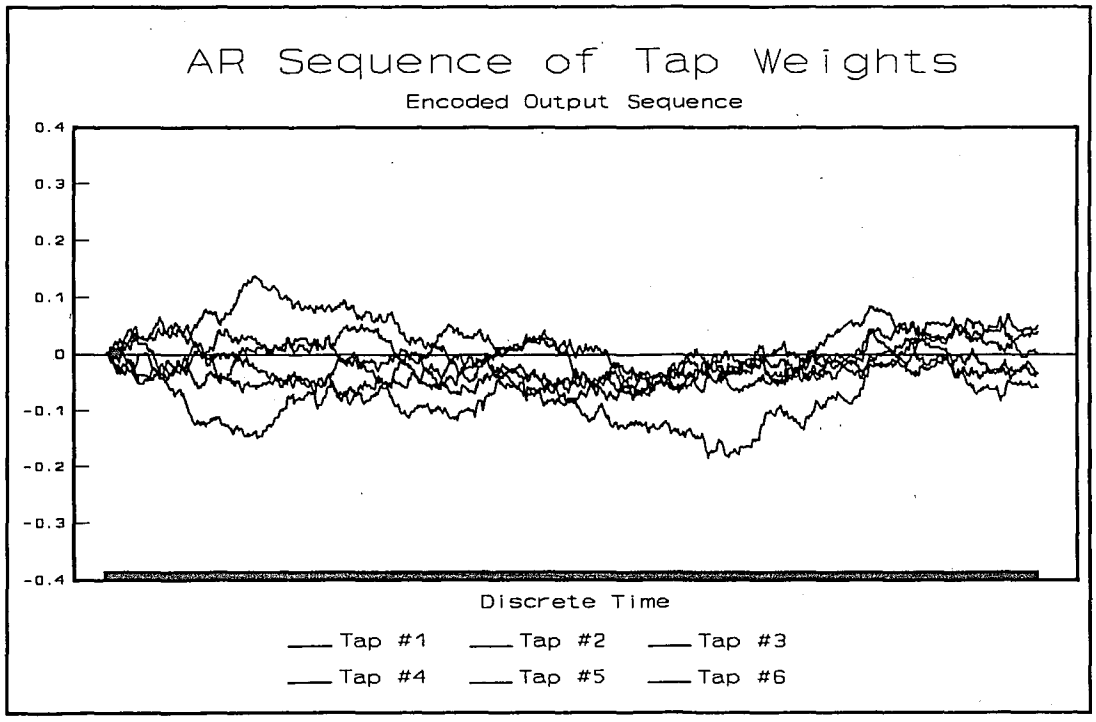



(figure 28)

Figure 27 and 28 show the attractors in the state space.




(figure 29)




(figure 30)

Figures 29 and 30 are the encoded output sequence of an general linear process with taps .75 and -.75.

## VII. REFERENCES

[1]Leon O. Chua and Rabinder N. Madan, "Sights and Sounds of Chaos," IEEE Transaction on Circuits and Systems, Vol Cas 27 No. 11, pp. 3-13 January 1988.

# END OF TITLE