Communications Research Centre

USER'S GUIDE FOR THE DRL SPREAD-SPECTRUM SIMULATION FACILITY

by

G.O. Venier

The Simulator was developed under a task sponsored by the Director of Maritime Combat Systems (DMCS-6) of the Department of National Defence.

Government of Canada
Department of Communications

Gouvernement du Canada
Ministère des Communications

CRC REPORT NO. 1403
OTTAWA, APRIL 1986

Canada

COMMUNICATIONS RESEARCH CENTRE

DEPARTMENT OF COMMUNICATIONS
CANADA

USER'S GUIDE FOR THE DRL SPREAD-SPECTRUM SIMULATION FACILITY

by

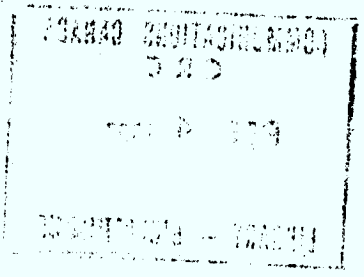
G.O. Venier

Directorate of Radio Propagation and Systems(DRL)

CRC REPORT NO. 1403

April 1986
OTTAWA

The Simulator was developed under a task sponsored by the Director of Maritime Combat Systems (DMCS-6) of the Department of National Defence.



CONTENTS

USER'S GUIDE FOR THE DRL SPREAD-SPECTRUM SIMULATION FACILITY

BY

G.O. Venier

ABSTRACT

This report describes, from a user's point of view, the spread-spectrum simulation facility developed in the Directorate of Radio Propagation and Systems (DRL) at the Communication Research Centre. This facility simulates, on a VAX-11/750 digital computer, the operation of complete spread-spectrum systems including transmitter, HF propagation path, interference, and receiver, and provides all the data and signal generation and analysis capabilities necessary to determine the performance of the simulated systems. Both direct-sequence and frequency-hopping systems may be simulated. The simulator was made as flexible as possible, permitting the user to select from a number of subsystems to simulate a wide range of existing and proposed communication systems. This report provides detailed information on the simulator that is essential for understanding it and operating it in both interactive and batch modes.

1 INTRODUCTION

The DRL spread-spectrum simulation facility simulates the operation of complete spread-spectrum systems including transmitter, HF propagation path, interference, and receiver, and provides all the data and signal generation and analysis capabilities necessary to determine the performance of the simulated systems. Both direct-sequence and frequency-hopping systems may be simulated. The simulation program was written in FORTRAN-77 and runs on a Digital Equipment Corporation VAX-11/750 computer under the VMS operating system.

Figure 1.1 indicates the communication system processes that are implemented in the simulator. The solid arrows show the normal flow of data and signals, while the dashed ones indicate alternative routes that may be used for testing of all or a part of the simulated system. The user decides which of the processes he wishes to include in the simulation and selects them and their parameters in an interactive process in which the program questions him on the desired values. Thus the user can specify the system at the terminal. The simulator was made as flexible as possible, permitting a wide selection of subsystems that should cover most actual and proposed systems. A batch mode of operation under control of a command file is offered as an alternative for long computer runs simulating complex systems. Because of the complexity of the simulator, a run to produce a single bit-error-rate value may take many hours when spread spectrum with high processing gain is being simulated.

A consequence of the flexibility of the simulator is the necessity for the user to understand, in detail, the processes being used; many of the design parameters are left to him. This document is intended as an aid to that understanding, but it is, of course, impossible to include here all the theory applying to all of the processes provided. The user should be familiar with the theory for the techniques he selects. References are provided in many cases to allow further study where this familiarity is lacking. The algorithms used are described in some detail. If further detail is required, reference should be made to the source routines. These

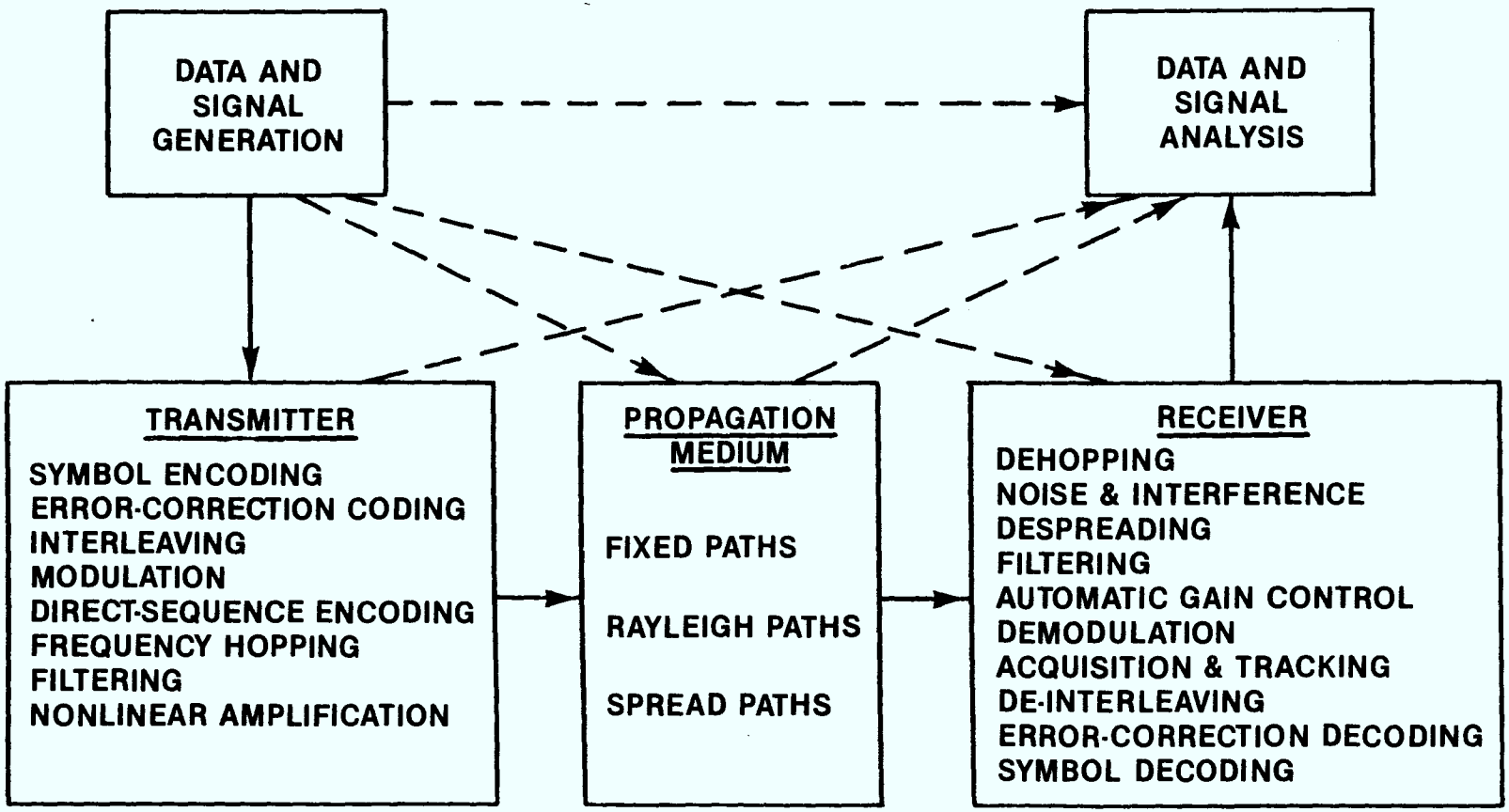


Figure 1.1 Simulation Processes

include many comments to aid in understanding the code.

2 STRUCTURE OF THE SIMULATOR

The main routine of the simulator is called MODEM (capital letters will be used to refer to routine names or commands as they are to be typed by the user); the simulation is started by typing RUN MODEM. MODEM is only a controlling program which leaves the simulation tasks to a number of subprograms as shown in Figure 2.1. Any of the eleven main subprograms shown may be entered by typing its name in response to the question "Main level command?". In the following, the terms "command", "subprogram" and "subroutine" are used to refer to the boxes in the program structure diagrams. These boxes represent subprograms or subroutines (we have used "subprograms" for the highest level below the main program), but their names are entered by the user as commands. A menu of the eleven commands with brief descriptions is displayed at the beginning of the program and at any time the command HELP is typed in the main routine. The main subprograms may in turn call a number of other subroutines which are not shown in Figure 2.1. In general, the command structure forms a tree without connections between branches, but there is some sharing of utility routines. The software controlled by the eleven main-level commands varies greatly in size. The upper row in Figure 2.1 contains relatively simple utility routines, and the bottom row contains more complex subprograms. PROCES contains, by far, the most software.

When the main program is first entered the user is asked two questions before being presented with the menu of subprograms. First he is asked whether he wishes to perform the simulation from the terminal or in batch mode; then he is asked whether he wants the output to go to the terminal or to a file. This latter decision may be changed during the operation of the simulation by the use of the FORM command which is described later. A distinction should be made here between the simulator output and the questions and menus printed on the terminal by the program. These latter data always go to the terminal or, in batch mode, to the specified batch log file. The simulator output contains summaries of the parameters used,

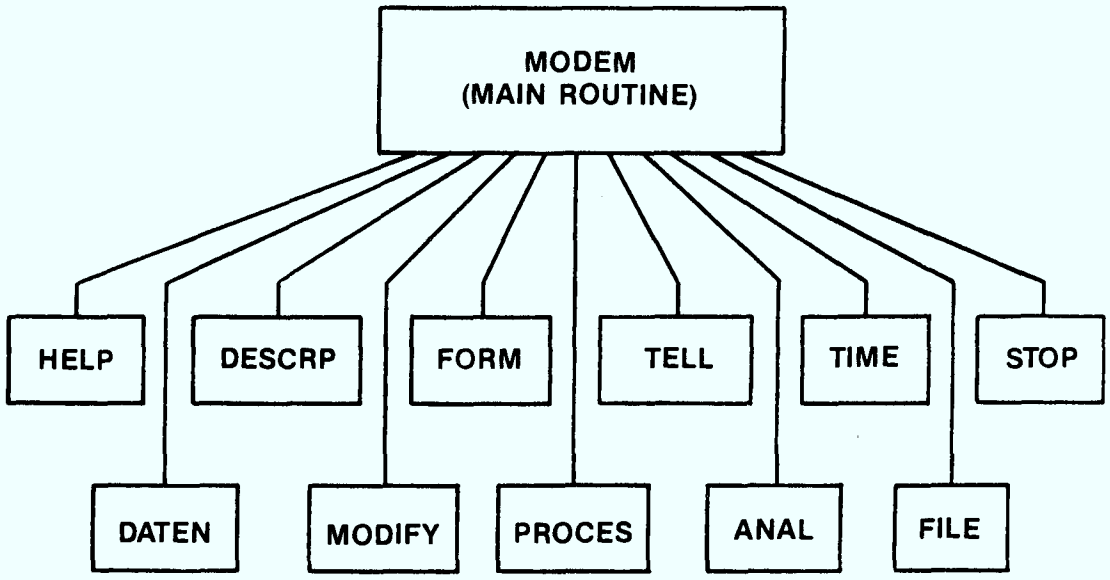


Figure 2.1 Simulator Program Structure

either as specified by the user or as computed from the user specifications, and the results of any analyses performed on the simulated signals and data. When batch mode is used the terminal is logically equivalent to the batch log file, and therefore, if the terminal is specified as the output device the output will go to the batch log file along with the questions and menus.

There are some differences in the running of the simulator between terminal and batch mode that have to do with the monitoring of the operation, and this is why the program must be informed of the mode. When batch mode is used the questions are answered by a command file which must be prepared in advance. More will be said about running in batch mode near the end of this document.

2.1 Brief Description of the Subprograms

The five subprograms in the bottom row of Figure 2.1 will be described in their own sections, but we will first mention their functions.

DATEN generates both binary data for input to the simulated system, and complex (i.e. comprising real and imaginary components) signals as test inputs for parts of the system.

MODIFY modifies existing data for convenient generation of other, possibly more complex, signals, and includes some other special-purpose functions.

PROCES performs the actual simulation of the communication system, including transmitter, propagation medium, and receiver.

ANAL is used to analyze the results of a simulation experiment with various display and processing routines.

FILE allows the saving of data at any point in the simulation process, and the later re-input of the saved data. The file of data saved may also be analyzed by off-line programs.

The subprograms in the upper row of Figure 2.1 are utility routines that are intended to aid in the running of the simulation. HELP repeats the menu of main-level commands displayed at the beginning of the program.

DESCRP displays a brief description of the simulation program.

FORM allows the user to change the form of the ASCII output. This output is the information provided by the program on the parameters selected, and the results of any analyses. It is intended as a record of the simulation experiment. The user may specify the terminal or a file as the output device. If a file is chosen the user supplies a name and a brief description (maximum of 32 characters) which will be included in the file header. The file name will automatically be given the extension .DAT. In batch mode the terminal is equivalent to the batch log file which can be specified by the user in the VMS operating system's SUBMIT command used to start batch operation.

TELL informs the user of the number of data currently in the original and final arrays and of the maximum allowed data array size. This latter quantity is the number of complex values permitted in each of the original and final arrays. The TELL command may be used from many of the other subprograms without a return to the main program.

TIME reports the percentage of time spent in each of the routines since the last TIME command. It is intended to help the user determine which routines are consuming the most time. It may also be useful in predicting CPU time required for a simulation when the CPU time for other runs with different characteristics has already been measured. TIME should not be considered to be of great accuracy since it takes its time from the system clock rather than measuring CPU time and it includes time used for parameter entry. Thus for reasonably meaningful timing results, large

quantities of data should be processed, and the simulation should be run in batch mode with the parameters entered from a command file.

STOP is used to bring the simulation program to an end in an orderly fashion.

2.2 Original and Final Data Arrays

The various processes that represent parts of the simulated system in PROCES operate on an array of input data and produce an array of output data. These arrays are referred to in the simulator as original (ORG) and final (FIN) data respectively. The abbreviations ORG and FIN are the form in which they are specified by the user when required. They will be used also in this report to refer to the data in these arrays, since the full names "original" and "final" could easily be misinterpreted as the more general meaning of these words. The ORG and FIN data may represent binary data or sampled signals. In either case they are floating-point complex numbers; when representing binary data their real part is either one or zero and their imaginary part is always zero. While this is not a very efficient way to store binary numbers it does have the advantage of simplicity (only one type of array is used), and, since only one ORG and one FIN array is kept at any one time, it does not increase the total storage required (enough space must be reserved for the generally larger sampled signal arrays in any case). Subprograms other than PROCES may generate or process either ORG or FIN data as specified by the user.

2.3 Frequencies and Data Rates

Frequencies are always specified as a fraction of the rate at which signals are sampled in the simulator. For example, a sine wave with a frequency of 0.1 would have ten samples per cycle. When input data is generated the user is asked for its rate in values per second. This number is kept track of and changed as necessary - for example when decimation is

used. In addition, this rate is used to convert all signal frequencies from the fractional sample-rate specification to the corresponding frequency in Hertz, which is displayed to the user in that form as well as in the fractional-sample-rate form. The value in Hz is not used by the simulation itself and is provided only for the user's convenience.

2.4 Voltage,Power and Energy

The floating-point number representing a sample of a waveform is considered to be the voltage, in volts, of the waveform at that point. Power is defined as the power that would exist in a one-ohm resistor at the point in question. Energy is defined on a time scale determined from the arbitrary user-defined data rate. That is, if the program had determined the sample rate, based on the user-specified data rate, to be f_s samples per second then the energy in N samples at a voltage V would be

$$V^2 N / f_s .$$

3 DATA AND SIGNAL GENERATION (DATEN)

The DATEN subprogram is used to generate both binary data and sampled complex waveforms as input for the simulation runs or as test inputs for parts of the simulation. All waveforms and data are represented as complex values; when they are real the imaginary part is simply set to zero. The various subroutines that can be called in DATEN are shown in Figure 3.1. Routine CANCEL simply permits the user to exit DATEN without generating any data. He may wish to do this if, for example, he has entered it by mistake. There are two ways of generating signals. In the first, the user enters them value by value to produce any desired sequence, and in the second he specifies a data or waveform type along with its parameters, and the routine generates the complete sequence for him. This second method is much less tedious but more limited in possible signals. It also permits the generation of signals with pseudo-random characteristics, such as

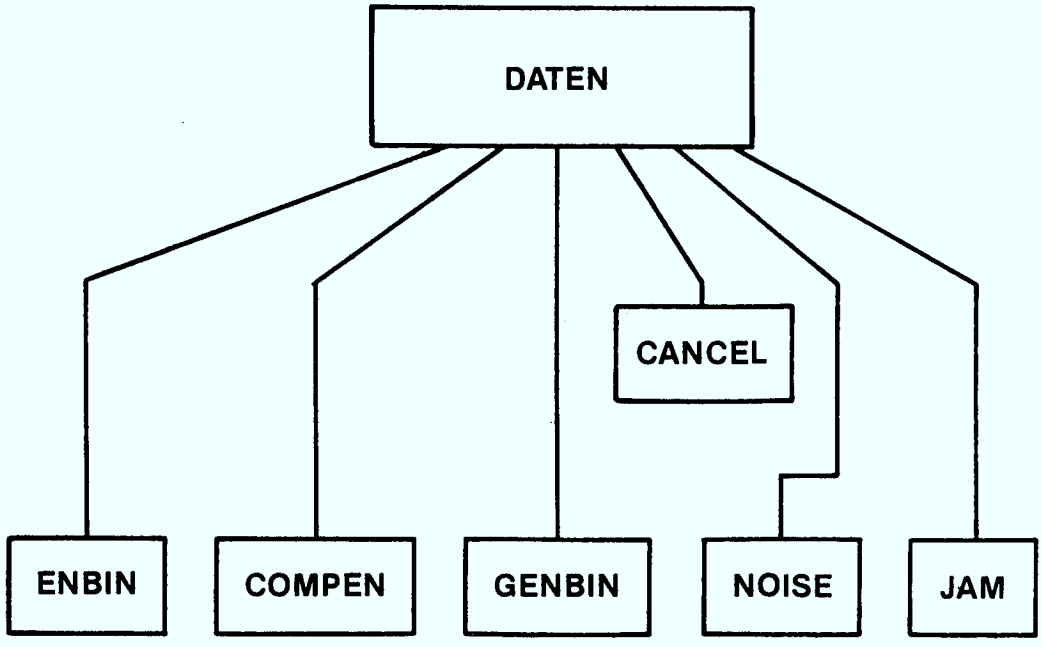


Figure 3.1 DATEN Subprogram Structure

noise. Two routines are available for the first method, ENBIN for binary data, and COMPEN for complex waveforms. Routines for the second method are grouped into two types called NOISE and JAM. These are also used in the receiver (RECVR) to introduce noise and jamming. When DATEN is entered a menu of the available routines with a description for each is displayed. One of the subroutines is then selected, and after the data is generated an exit is made automically from DATEN to the main program.

The user may enter the signals into either the ORG or FIN data arrays. Normally, any process takes data from the ORG array, and, after processing it, leaves the result in the FIN array. Thus, most of the time the signal will be input to the ORG array, but not always. As will become evident in Section 4 on the MODIFY subprogram, it is important to allow input to the FIN array.

On entry to the DATEN subprogram the user is asked whether the data is to be ORG or FIN data, the number of values to be generated, and the data rate. There is a limit on the number of data values that may be generated. This number is printed out at the start of the program and when the TELL command is used. It is set by a parameter called SIZVEC, and can be changed by a source program modification followed by re-compiling and re-linking. The procedure for doing this is described in the technical documentation that accompanies the software in the DRL VAX-11/750. This documentation can be found in file TECN.MEM. This parameter affects the program memory requirements, and may require a higher allocation from the system if it is increased.

After he enters the above values the user may select one of the above-mentioned routines to generate the data. Each of these routines will now be described in more detail.

3.1 Binary Data Entry (ENBIN)

ENBIN allows the user to enter binary data value by value. It prompts the user by displaying a series of dots indicating the number of

entries specified. The user enters a one or zero under each dot. If the number to be entered is more than 64, the data are entered 64 at a time, with each group being re-displayed by the routine to allow acceptence or rejection and re-entering by the user. The data are entered into the real parts of the selected array and the imaginary parts are set to zero.

3.2 Complex Data Entry (COMPEN)

Like ENBIN, COMPEN allows the user to enter values into the selected array, but in this case the values are floating point complex numbers representing an analogue waveform and they are entered four values at a time - four real and then four imaginary values. The routine prompts for the correct type and number and asks for verification of each group of four before continuing. Real waveforms may be entered by entering zeros for the imaginary parts.

3.3 Pseudorandom Binary Data Generation (GENBIN)

GENBIN is intended mainly as a source of data to be transmitted by the simulated system. It generates pseudo-random binary data by means of a binary feedback shift register of 31 stages, with the outputs of stages 28 and 31 added modulo-2 and fed back to the input. These connections result in the generation of a maximal-length sequence (length of 2^n-1, where n is the length of the shift register) [1]. This sequence will consist of $2^{31}-1$ or over $2x10^9$ bits before repeating. A 31-bit starting value or seed is required to start the generation. This seed determines where in the $2^{31}-1$ bit period the sequence will start. The user decides whether to enter the 31-bit seed or to accept a default seed instead. This default seed will be the same each time it is used.

3.4 Noise Generation (NOISE)

Three types of noise may be generated in NOISE. These are Gaussian, CCIR, and impulse noise. The user is asked if he wishes to generate each of the types in turn, and he may choose one or more of them.

If he chooses more than one the resulting samples will represent the sum of the chosen types. At the beginning he must decide whether to enter a seed for the random number generator or to accept the default one. The seed is a double-precision integer, and it is recommended that it be large and odd. All three noise generators make use of the FORTRAN uniform random number generator RAN supplied with the VMS operating system. This generator is of the multiplicative congruential type. According to the documentation this generator is "fast but prone to nonrandom sequences when considering triples of numbers generated by this method". This is not likely to cause any problem in the simulation, and speed is important. However, if any problems become evident the generator could easily be replaced by the user with a slower one having better properties.

For each type of noise selected the user is given the opportunity to specify a filter to filter the noise and change its spectrum from white to some other desired shape. Details on filter specification are given in Section 8.1.

Two methods of Gaussian noise generation are available. One method, referred to here as the sum method (SUM), makes use of the central limit theorem which states that when a number of samples of a random variable are summed the resulting sum is a sample from a distribution that approaches a Gaussian distribution as the number summed is increased [2]. It turns out that, when the original distribution is uniform, relatively few samples (generally less than ten) are needed in the sum to give a good approximation to the Gaussian distribution. The SUM routine simply sums the number of uniformly distributed samples specified by the user to produce each Gaussian sample and scales the result to give the root-mean-square (rms) value specified by the user.

The other method of Gaussian noise generation is the inverse method (INV) [3]. INV maps a pair of uniformly distributed samples into a pair of Gaussian samples by using the inverse of the Rayleigh amplitude probability distribution function on one sample to generate a Rayleigh sample, and scaling the second to the range 0 to 2π to generate a uniformly distributed

phase angle. Then a conversion from polar to Cartesian coordinates results in two independent Gaussian samples. The only information required from the user in this case is the desired rms voltage.

CCIR noise has a distribution taken from a set of semi-empirical curves published by the International Telecommunication Union [4]. These distributions apply to the magnitude, or envelope, of the noise and are believed to be typical of those of noise found at HF. A parameter, V_d, specifies the ratio of rms voltage to the mean envelope voltage, and is expressed in dB. The minimum V_d of 1.05 corresponds to the Rayleigh distribution, and increasing values of V_d correspond to increasingly more impulsive distributions (higher probability of very high values). The value of V_d actually found depends on the bandwidth of the system in which it is measured, being higher for higher bandwidths.

In the simulator, a method developed by Akima [5] is used to generate CCIR noise. In this method the CCIR amplitude probability distribution curves are approximated in a nonlinear coordinate system by two straight-line segments joined by an arc, and the method of inversion of the distribution function is used to generate the noise samples from uniformly-distributed samples. The user specifies the rms value of the noise and the value of V_d.

Impulse noise consists of single complex impulse samples among a larger number of zero samples with intervals between impulse samples that are random with an exponential distribution and a mean rate specified by the user; i.e. the impulse arrival times are generated by a Poisson process. The maximum rate is 0.1 times the sample rate. The amplitude of the impulse noise may be either fixed, or random with a Rayleigh distribution. In either case the user specifies the rms value of the impulse samples (not of the complete waveform). The phase of each impulse sample is always selected from a uniform distribution between zero and 2π. If a particular interval is less than one-half the sample interval, the second impulse will occur one sample interval from the first instead of being coincident with it.

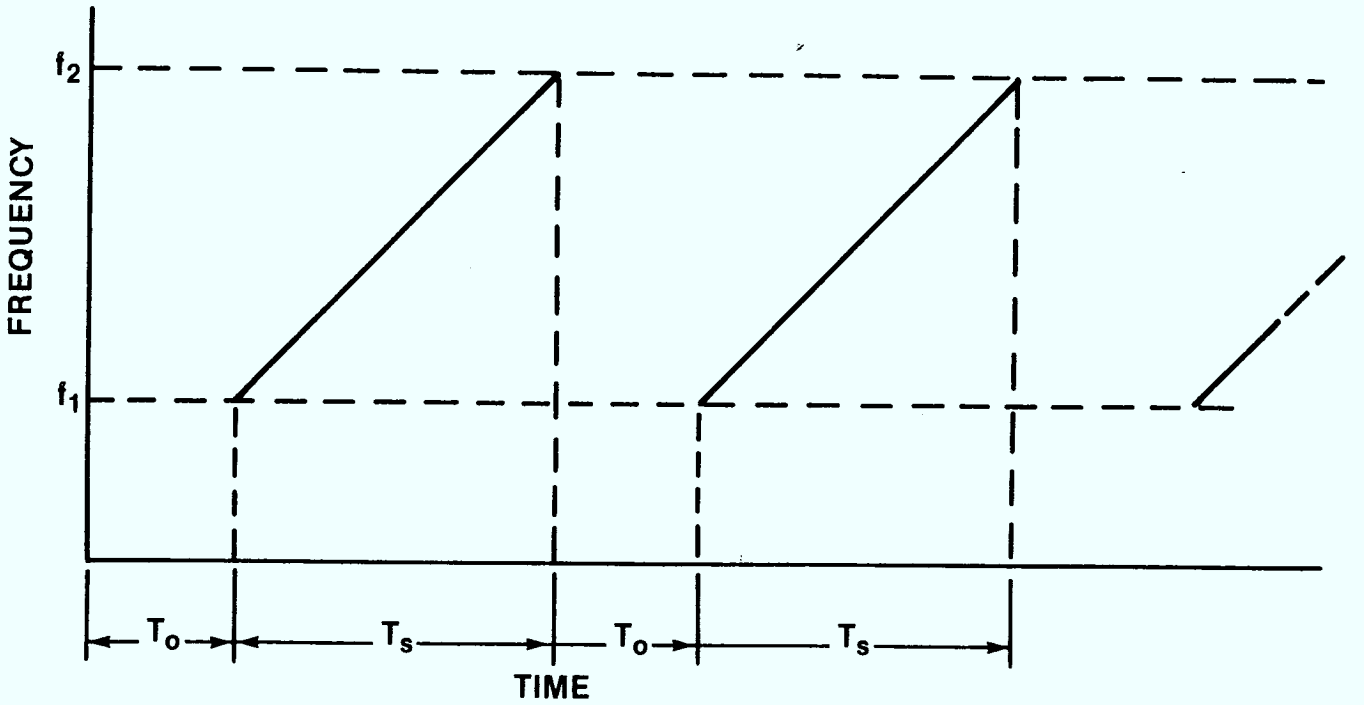
3.5 Jamming Signal Generation (JAM)

Four types of jamming waveforms are available in JAM. These are Gaussian, tone, pulse, and linear frequency-sweep. As in NOISE one or more may be selected with results summed. Before selecting waveforms the user decides whether to enter a random-number seed or to use the default seed. The Gaussian generator is identical to that in NOISE and is included in JAM for the convenience of the user.

Up to four complex tones may be generated. The user specifies the frequency, amplitude, and phase of each. The frequency must be in the range -0.5 to +0.5 in terms the sample frequency.

Pulses may be generated at random intervals in the same manner as the impulses in NOISE, with a user-specified mean rate. In addition, the user specifies the pulse width in samples, their amplitude (fixed), and their carrier frequency (-0.5 to +0.5). The phase of the carrier of each pulse is selected from the uniform random-number generator. The mean rate is limited to one-third the inverse of the pulse width (width is entered before rate). However, any particular random interval between pulses may be less than the pulse width. When this occurs, and the pulses overlap, they add linearly. But if more than three overlap, the fourth etc. are lost. This is not a significant deficiency since in reality so many overlaps would be an extremely unlikely event, even at the highest allowed rate. Filtering may be applied to the pulse jamming.

Figure 3.2 illustrates the specification of the linear frequency sweep jamming. The user must enter the start frequency, the stop frequency, the off period, and the sweep period. The direction of the sweep may be either positive or negative. The smallest period allowed is typed by the program as computed from a minimum time-bandwidth product of 5. In addition, the user specifies the initial phase and the amplitude. The program displays the selected parameters in Hz and seconds. It also gives the power in watts. This is the instantaneous power during the on portion of the waveform, not the average power over a full sweep cycle.



f_1 = START FREQUENCY	$-0.5 < f_1 < +0.5$
f_2 = STOP FREQUENCY	$-0.5 < f_2 < +0.5$
T_o = OFF PERIOD	$T_o \geq 0$
T_s = SWEEP PERIOD	$T_s > 5/\|f_2 - f_1\|$

Figure 3.2 Linear Frequency Sweep Specification

4 DATA MODIFICATION (MODIFY)

The MODIFY subprogram allows the user to modify data, whether it has been created by DATEN or is the output data from any of the processes. Routines included in MODIFY are shown in Figure 4.1. There is no automatic exit from MODIFY after calling of a routine. For this reason there is a RETURN command to return to the Main program. The HELP command gives a menu of available commands and their description, and the TELL command gives information about the ORG and FIN data; this is the same routine as used in the main level. The above routines have no effect on the data. The remaining ones, except for CONVRT, actually modify the data.

CONVRT changes FIN data into ORG data to allow it to be used as input when entering the PROCES command level. The change is accomplished by simply renaming the data; no time-consuming shifting of data is involved. The original ORG data is lost when this routine is used, and, as a safety measure, the routine reminds the user that the ORG data will be lost and asks for verification of the CONVRT command.

ADD adds ORG data to FIN data sample by sample. If one of the data sets is longer than the other, the shorter will be extended with zeros to the same length as the longer before the addition.

REPEAT duplicates either ORG or FIN data a user-specified number of times with the new values added to the end of the existing sequence. The duplicated data may be any block from the existing sequence; the user specifies the index values of the first and last samples of the block.

INSERT allows the user to insert a specified number of zeros before the existing ORG or FIN data. These are complex zeros; that is, for each zero requested, both real and imaginary components of zero will be generated. Insert may be used even if no data already exists in the specified array.

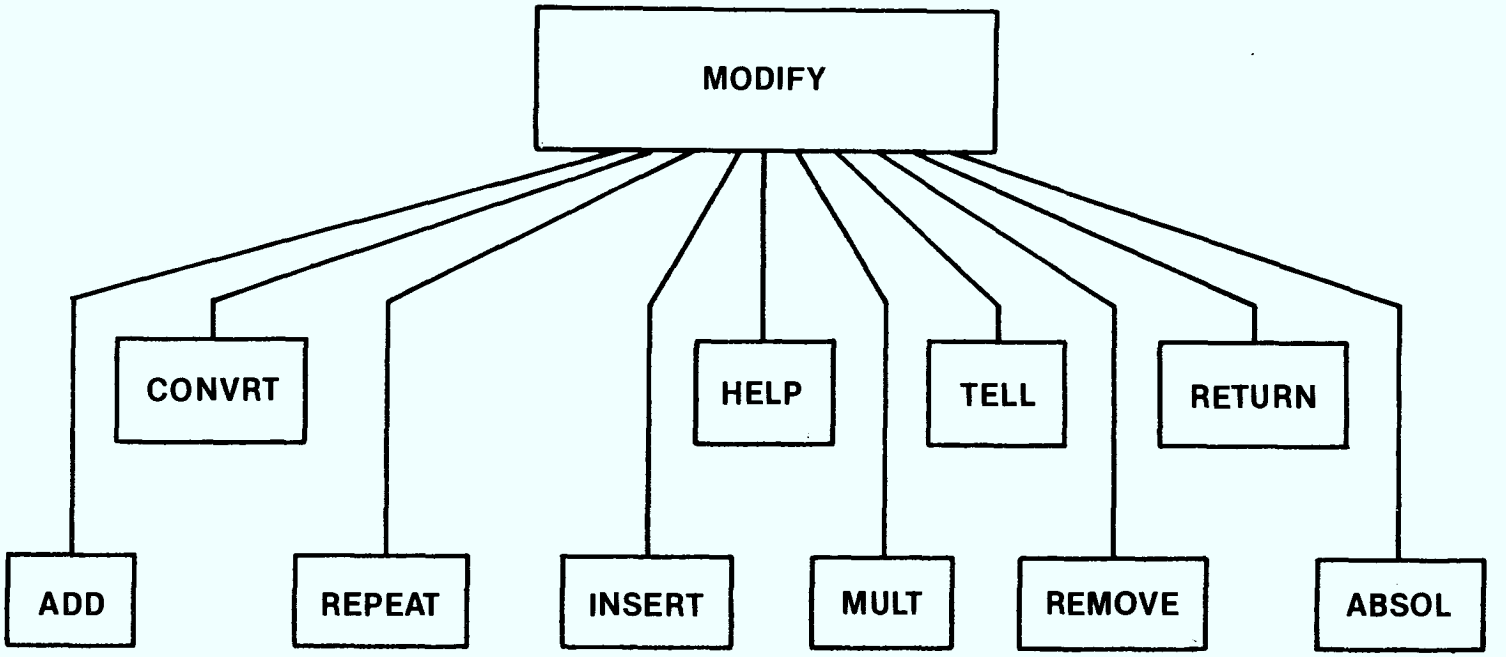


Figure 4.1 MODIFY Subgprogram Structure

MULT is used to multiply any block of ORG or FIN data by a real constant. The constant may be zero. The user specifies the index values of the first and last samples of the block.

REMOVE deletes a block of data from the end of ORG or FIN data. The user specifies the number of complex values to be removed.

ABSOL takes the absolute value of a block of ORG or FIN data. The user specifies the index values of the first and last samples of the block. The real part of each sample in the block is replaced by the absolute value and the imaginary part is replaced by zero. As an option, the absolute value may be squared. This modification is intended as an aid in the analysis of data.

5 ANALYSIS ROUTINES (ANAL)

A number of subroutines are available for the analysis of output signals and data. These include simple display of any component of the digital samples, computation of histograms and other statistical values, comparisons of input and output of signals and data, computation of bit-error statistics, and computation of Fourier transforms. The analysis command level is entered by using the ANAL main-level command. The available subroutines are shown in Figure 5.1. A RETURN command is provided since there is no automatic exit to the main program. The HELP command displays the menu for the ANAL commands, and TELL is the same as in the main program. Six different commands, described in detail below, are available to analyze the data. Three of the commands (COMPRB, ERRCOM, COMPRC), are comparison commands and require both FIN and ORG data to be in existence. For these utilities, the user enters the number of data to be compared and the ORG and FIN starting index values. Any block of contiguous ORG values can be compared with any block of the same number of contiguous FIN values. The other three commands (VIEW, FFT, and HISTO) require only one type of data, either ORG or FIN. Only one starting index value along with the amount of data to be analyzed is entered.

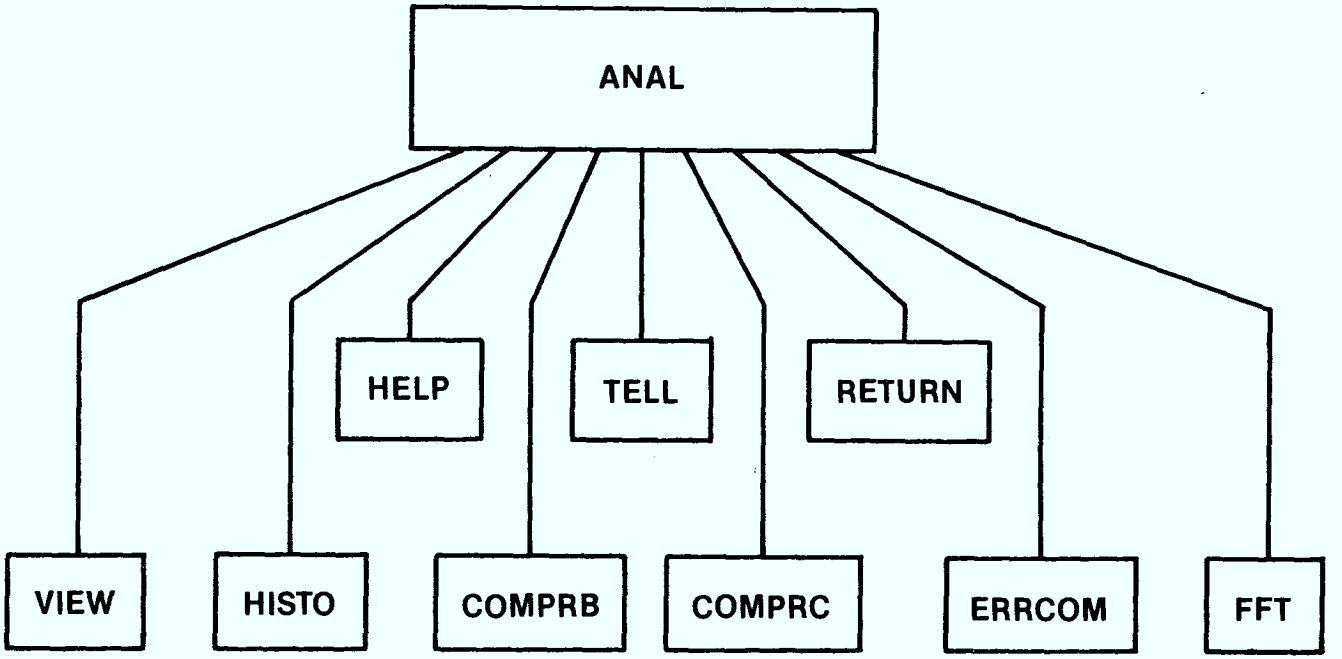


Figure 5.1 ANAL Subprogram Structure

5.1 Binary Analysis Utilities

5.1.1 Binary Data Analysis Command (COMPRB)

Two binary analysis utilities are available. COMPRB compares binary ORG and FIN data on a bit-by-bit basis. The utility was created primarily for the purpose of comparing original binary data before it is sent through the communication channel with final binary data coming out of the channel. It computes the bit error rate, the longest sequence of 1's and 0's, and the percentages of 1's and 0's.

5.1.2 Binary Error Analysis Command (ERRCOM)

ERRCOM analyzes bit errors on the basis of time as well as number. The data is broken up into windows whose length is specified by the user and the frequency of the periods between errors is computed for each window. The user can optionally have the index values of the bits in error output. At the end of the analysis, the total number of occurrences of each period for all the windows is presented. Note that the results can be affected quite drastically by the window size and the analysis starting positions that are selected. For example if all periods are greater than 25 but the window size is 20, no periods will show up in the analysis. The fractional bit-error-rate is also computed.

5.2 Complex-Component Display Command (VIEW)

Command VIEW displays a choice of five ORG or FIN data components: real, imaginary, magnitude, phase in degrees, and phase in radians. In addition, the mean, standard deviation, and mean of the squares of the displayed components are output. If the user wishes to see only these statistical quantities, an option is available to suppress the component display.

5.3 Complex Comparison Command (COMPRC)

COMPRC plays a role similar to COMPRB, except that it compares complex components rather than binary numbers. The standard components, real, imaginary, magnitude, degree phase, or radian phase can be compared. Instead of a bit error rate, the root-mean-square error is output, along with the mean, standard deviation, and mean of the squares of the ORG and FIN data components. Note that COMPRC provides some of the same information that VIEW does.

5.4 Fast Fourier Transform Command (FFT)

A fast Fourier transform, either forward or inverse, of either the ORG or FIN data can be performed by command FFT. The number of data to be analyzed is restricted to integer powers of two. The forward transform is defined as:

$$YF_m = \sum_{n=0}^{n-1} x_n \exp(-j2\pi nm/N)$$

while the inverse transform is defined as

$$YI_m = \frac{1}{N} \sum_{n=0}^{n-1} x_n \exp(j2\pi nm/N)$$

The significant point to note here is that the inverse transform is divided by the number of points, N, but the forward transform is not. This assures that a forward transform followed by an inverse one will leave the data as it was. When FFT is used on the ORG or FIN data array, the transformed data replaces the data in that array, and the data that was there is lost. As a safety measure, the user is informed of the impending loss and asked

to confirm the command. As an example, if there were 200 ORG data samples, the user could transform 128 of these by specifying the power of two of the number of samples as 7. He could specify the start of the transformed block to be 40, and this would result in samples 40 to 167 being transformed with the 128 samples of transformed data replacing the ORG data (there will be only 128 samples of ORG data after the transformation). The FFT algorithm is a radix-two-decimation-in-time algorithm taken from Reference [6]. The data is scrambled before the transform and comes out in the correct order. To save time the exponential factors are computed ahead of time.

5.5 Histogram Command (HISTO)

Histograms of components of ORG or FIN data can be produced by using the command HISTO. The choice of components is the same as for VIEW. After specifying the number of data to be processed, the user enters the histogram lower limit, the histogram cell size, and the number of histogram cells. The number of cells includes a first cell for all values falling below the lower limit, and a last cell for all values falling above the upper limit of the next-to-last cell. Therefore the minimum number of cells allowed is three. After the histogram is produced in tabular form the user is given the option of viewing a bar-graph version. Finally, he is given the option of having the data replaced with the histogram cell values, so that these can be written to a file. When this is done, the program words storing the number of data and the data rate are replaced by ones indicating the number of histogram cells, and the cell size, respectively. The real part of the data vector receives the cell values. The first imaginary location receives the histogram lower limit, while the second imaginary location receives the upper limit. The third position gets the code indicating which component was processed. All remaining imaginary locations are set to zero.

6 DATA STORAGE FILES (FILE)

Complex data storage files may be created by the FILE subprogram. These may be used to store intermediate data between runs or within a run. Such files can save considerable computer processing time when the same data is used a number of times, as, for example, when different values of some receiver parameter are tried with the same transmitted signals and the same propagation medium. Figure 6.1 indicates the two subroutines FILIN and FILOUT that are available, and the CANCEL command which simply allows an exit to the main program without any action. An automatic exit to the main program occurs after either FILIN or FILOUT.

The user may send a block of ORG or FIN data to a file using the FILOUT command. He specifies the number of data, the index value of the first datum, a description of the file with up to 32 characters, and a file name. A file header is generated containing the creation date, the program version number, the number of data, the data rate, and the user-entered file description.

Any file created in this way may be read back, in the same run or in a later one, as ORG or FIN data (regardless of where it came from) with the FILIN command. The user specifies the destination and the file name, and when the file has been read in, is informed of the number of values received, the sample rate, the date it was created and the file description. If data already exists at the destination, the user is warned that it will be lost and asked to verify the FILIN command. If the program version number in the header does not match that of the current program the user is warned of this fact but the input of data proceeds anyway. The warning is intended to alert the user to the possibility of errors caused by program changes that may make the data incompatible.

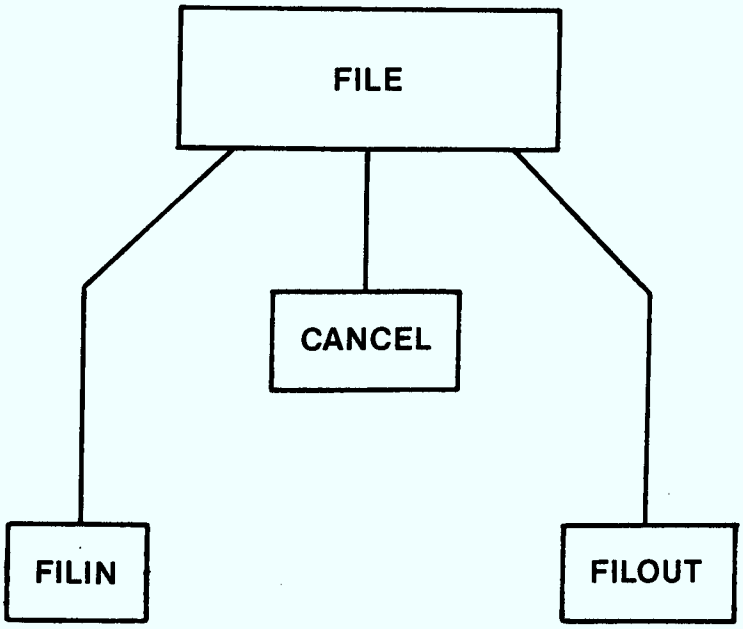


Figure 6.1 FILE Subprogram Structure

7 SIMULATION OF THE COMMUNICATION SYSTEM (PROCES)

7.1 Introduction

The PROCES subprogram performs the actual simulation of the components of the communication system including transmitter, propagation medium, and receiver. The structure is shown in Figure 7.1. The bottom row of subroutines perform the actual simulation. In addition to the usual utility commands shown in the upper row, a command called NULL is included. This simply copies ORG data to FIN data, provided no FIN data already exists. If a subroutine has already created FIN data, NULL makes no change.

Each of the six subroutines in the bottom row implements a portion of the communication system. BITSRC performs error-correction coding and interleaving. MODCOD forms integer symbols from the data bits, performs some symbol-to-symbol encoding such as differential encoding, adds direct-sequence chip symbols, and produces complex modulation samples. HOPPER performs frequency-hop encoding and includes some general-purpose operations such as signal filtering and decimation. MEDIUM simulates the propagation of the complex signal through a medium. RECVR performs the receiver functions down to demodulation into integer symbols. BITSNK performs symbol expansion into bits, and post-detection processing of the data bits. The above order of subroutines, or processes as we will now call these particular ones, is the order in which the communication would normally occur. Each process comprises components that we will call sub-processes.

Once in PROCES the user may call a number of processes in sequence. The FIN data from one process automatically becomes the input data for the next. To avoid loss of ORG data this input data is stored in an intermediate array that is used as the input to each sub-process. As a result, on entry to the PROCES subprogram, the ORG data must be copied into the intermediate array. Thus, when the PROCESS subprogram is initially

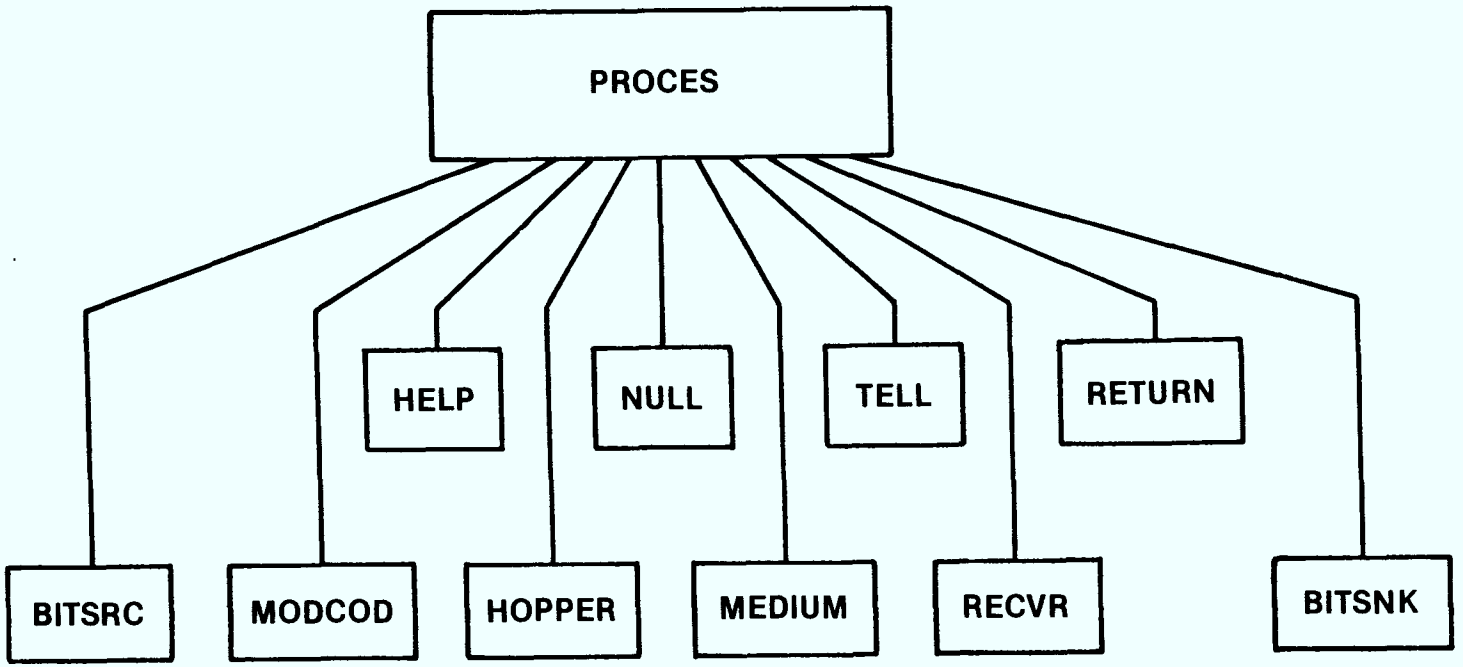


Figure 7.1 PROCES Subprogram Structure

entered with a large number of ORG data, the program will take some time to duplicate the ORG data. When PROCES is exited the ORG data is the ORG data that existed when PROCES was entered. If PROCES is later re-entered to continue operating on the data with other processes, the FIN data will first have to be converted to ORG data by using the data modification level command CONVRT. An example of where this may be required is where the data is to be saved at some intermediate point in the simulation, say after the propagation medium simulation. In this case PROCES must be exited after MEDIUM to allow the FILE command level operation FILOUT to be performed. Then, CONVRT in the data modification level would be called so that the output data from MEDIUM would be the input for the next process when PROCES is re-entered. It should be remembered that the ORG data existing before the CONVRT operation is lost, and, if important, should be saved using FILE.

It is important for the user to realize that it is his responsibility to assure that the correct sequence of processes with the correct parameters is chosen. For example, it makes no difference to the program whether the process RECVR is run before or after the process MODCOD, even though running RECVR before MODCOD is normally meaningless. The advantage to the user of this seeming weakness is that the simulator maintains maximum flexibility. Individual processes or parts of them can be tested or run through at different times. For example, MODCOD could be run one day and the output (FIN) data saved using FILE command FILOUT. The next day the old FIN data could be read in by FILE command FILIN as ORG data, and used in MEDIUM. Another example is the use of the RECVR process consecutively on different data sets (obtained from data files) to determine the effect of the process on data generated with different characteristics (say different propagation conditions). Also, some general-purpose devices such as a saturating amplifier, which exist in one process may be used at other places simply by calling the process that contains them after leaving the current process and before entering the next one (or re-entering the original one). Within a process particular sub-processes may be chosen by answering "no" to all other sub-processes.

Sub-processes inside a process are structured in a more rigid manner than processes; they are performed in a fixed order. However, the user is asked if the sub-process is to be used or not. Generally, sub-processes work on all of the data in a block fashion, one sub-process being started only when the last one has finished with all the data. Each block sub-process normally obtains information from the user as it needs it. The RECVR and MEDIUM process are the exceptions to this. Each of these operates in a sequential manner, with each sample being processed by each part of the process before the next sample is input (there may be some block processing, however; in this case the samples are accumulated and output delayed). This is not apparent to the user who still must set up a structure of sub-processes. The difference is that the sub-processes do not really exist. Rather, there is a single sequential process which comprises a number of operations in a loop which is followed for each sample.

7.2 Error Coding and Interleaving Process (BITSRC)

The error-coding routines have been structured to allow two levels of coding, an inner binary code and an outer symbol code. This would allow the implementation of concatenated codes. However, at this time only the inner coding routine has been implemented. The inner coding was to include both block and convolutional codes, but only a cyclic-block-coding routine is complete. The program asks the user if he wants to use any of the intended types, but if he chooses anything but inner block coding he will be warned that the chosen type has not yet been implemented, and that the input data has been passed to the output without change.

7.2.1 Binary Cyclic-Block-Code Generation

A general encoding algorithm for binary cyclic block codes is provided in BITSRC. The user enters the desired number of bits in a coded block, n, the number of information bits to be encoded in the block, k, and

the generator vector for the desired error correction code, or, if he prefers, its parity vector. (The parity vector method is not yet implemented, but is described here since it is partly implemented and could be completed with little effort.) The generator and parity vectors are simply the binary coefficients of the generator or parity polynomial. See Reference [7] for details on finding these polynomials for particular codes. The first coefficient of these polynomials, that of the zero-order term, is always one. The vector, however, is defined here to include this coefficient, and the user must enter it. If he enters a zero as the first element, the routine will reject the entry and ask that the vector be re-entered. Each output block will consist of the input block (information bits) followed by the parity bits. If the last block does not contain k information bits it is filled out to k bits by the addition of zeros before the coding is applied, and the user is so informed.

The encoding algorithm for the generator-vector method is represented by Figure 7.2 in conjunction with the following steps which are performed for each block to be encoded.

1 All registers are initially set to zero.
2 Switch S1 is closed and S2 is set to position 1.
3 k bits of input data are shifted into the circuit and simultaneously into the output stream through S2.
4 S1 is opened and S2 is set to position 2.
5 n-k shifts are performed. This will output n-k parity bits into the output stream.

The encoding algorithm for the parity vector method is represented by Figure 7.3 in conjunction with the following steps which are performed for each block to be encoded.

1 All registers are initially set to zero.
2 Switch S1 is set open and S2 is closed.
3 k information bits are shifted into the circuit and simultaneously into the output stream.

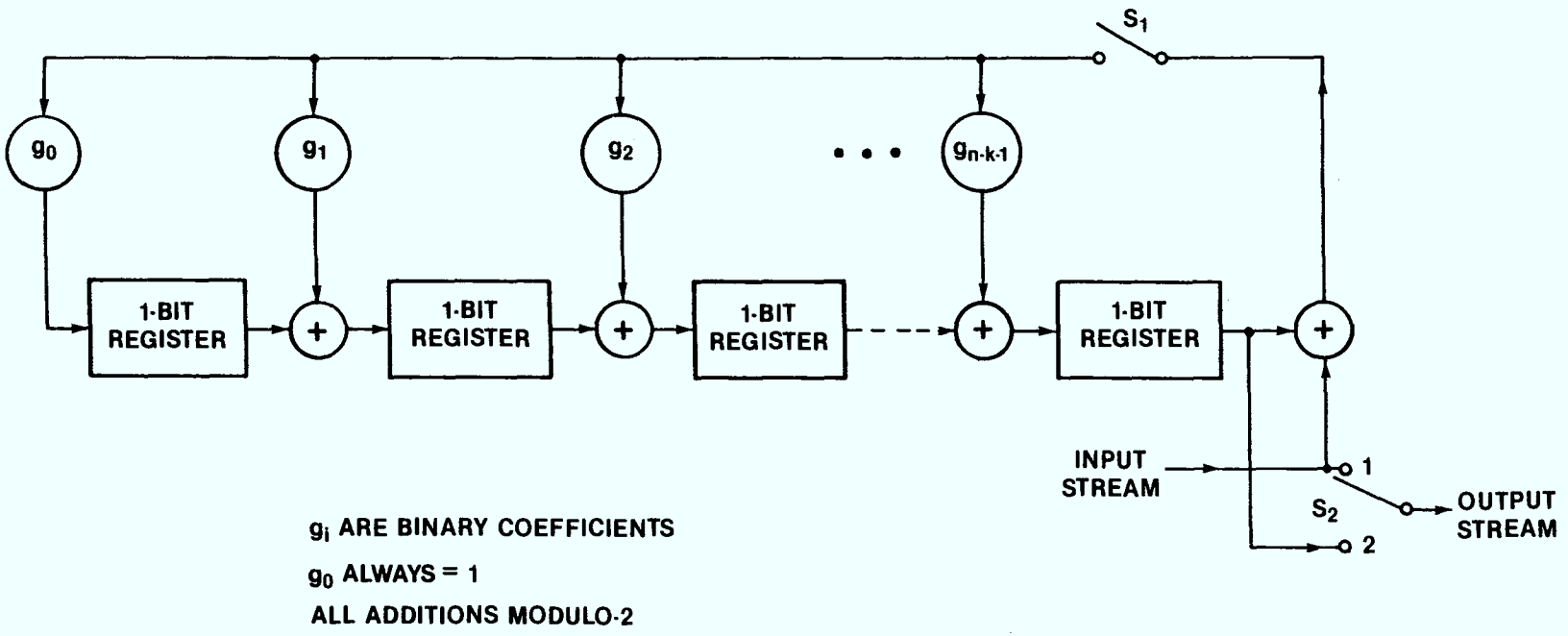


Figure 7.2 Generator Vector Encoder for Binary Cyclic Codes

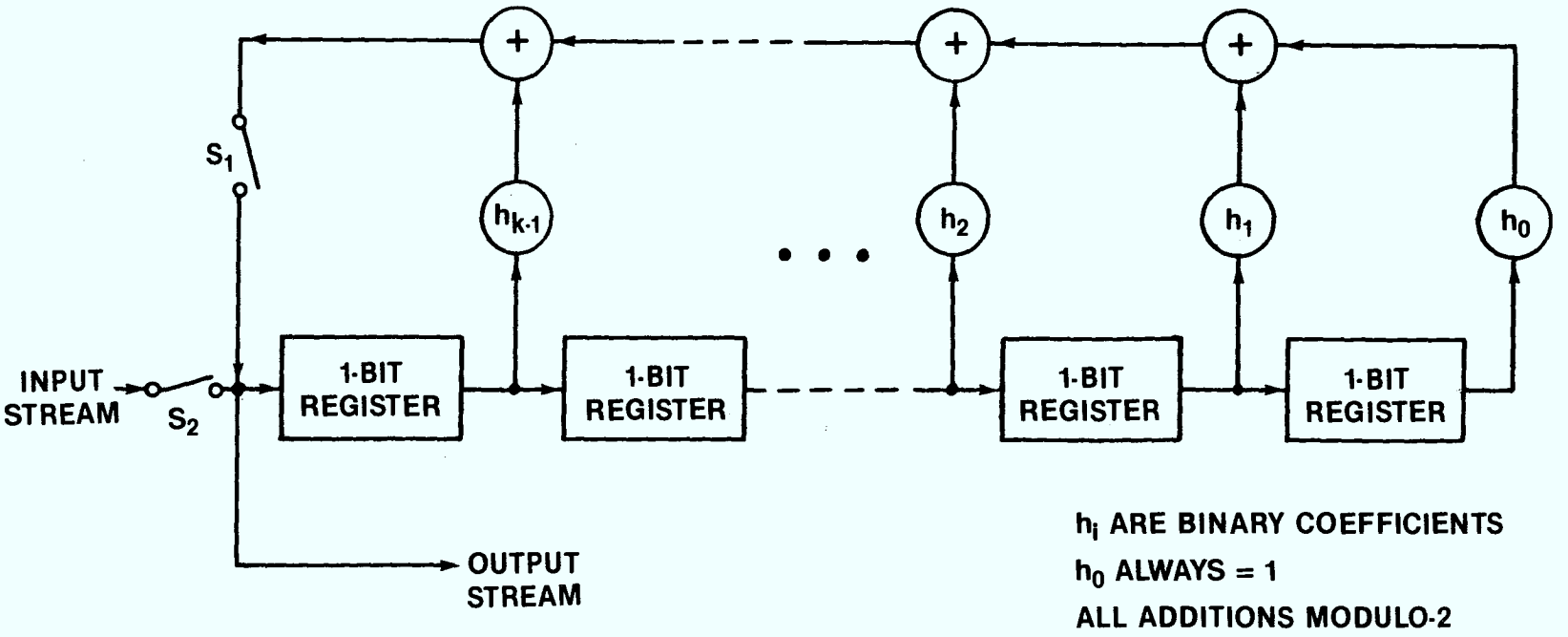


Figure 7.3 Parity Vector Encoder for Binary Cyclic Codes

4 S1 is closed and S2 is opened.

5 n-k shifts are performed. This will output n-k parity bits to the output stream.

It should be noted that error coding will always increase the reported data rate by the inverse of the code rate.

7.2.2 Interleaving

When errors are likely to occur in bursts, interleaving of the data after error-correction coding can be used to reduce the error rate by spreading the burst of errors over many coded blocks so that each block will have a small enough number of errors to permit the errors to be corrected. Of course, in the receiver the interleaving process must be reversed before error-correction decoding is performed.

A block interleaving routine is provided in the simulator. The method is to write the data into a matrix by rows, and then to read them out in columns. The user specifies the number of rows and columns. For example, if a ten-by-five matrix were specified (ten rows and five columns) then the bits would be output in the following order:
1,11,21,31,41;2,12,...,39,49;10,20,30,40,50. The semicolons indicate where a new column was started. In the receiver, an identical process (but called de-interleaving) is performed before error-correction decoding. For the data to be restored to its original order, the number of rows and columns in the receiver de-interleaver must be interchanged from those in the interleaver; that is, in the above example five rows and ten columns would be specified in the receiver de-interleaver. If the quantity of data is such that the last block (matrix) is not filled, the remainder is filled with zeros, and the user is informed of the addition.

7.3 Modulation and Coding Process (MODCOD)

The modulation process in the simulator is defined as the

conversion of data symbols into sampled representations of analogue waveforms for transmission through the medium. These waveforms form a set of distinguishable signals, one for each of the possible data symbols. The simulation represents these waveforms by complex samples of the signal at zero centre or carrier frequency. The complex representation permits the positive and negative parts of the waveform spectrum to be independent. As a result, the centre frequency is arbitrary and the simulation results apply just as well for any carrier frequency. The modulation waveforms available in the simulator are various forms of frequency-and phase-shift keying.

The coding referred to in this section deals with bits and symbols before modulation, but excludes error-correction coding. Types of coding provided are: coding of bits into integer symbols; conversion of symbols into new symbols for the purpose of minimizing the bit error rate for a given symbol error rate; conversion of symbols into new symbols to allow differential demodulation where a coherent form of modulation is used even though the propagation medium does not provide long-term coherence of the signal; coding for multiple-code-shift keying (MCSK); and addition of direct-sequence codes to the data symbols for spread-spectrum systems.

The word "symbol" is used here to describe two different things: an integer representing an element of data, and the corresponding waveform used to transmit the data element. There is normally a one-to-one correspondence between the two. Where the meaning is not clear from the context, the term "data symbol" or "integer symbol" is used for the former and "waveform symbol" is used for the latter.

Although coding, if performed in a real system, precedes modulation, for convenience modulation will be discussed first.

7.3.1 Modulation

The modulated signal is represented by samples of the analogue

waveform. The user-specified number of samples per symbol is an important parameter since it affects the processing time and memory requirements of the computer, and also the accuracy of the simulation. These requirements are conflicting, with the former demanding few samples per symbol and the latter demanding many. The program imposes a minimum on the number of samples per symbol, depending on the modulation type, which prevents a gross violation of the Nyquist-rate criterion, but the user should not depend on this to provide a good simulation. Since the symbols are of finite time duration, their spectrum will not be of finite extent, and the Nyquist-rate criterion can never be perfectly satisfied. It is up to the user to decide how many samples are necessary to provide the desired accuracy for a particular simulation. Since the samples are complex, the rate should be at least twice the frequency magnitude beyond which the energy is deemed insignificant. Since the modulation is applied to a carrier of zero frequency, this means that the sample rate should be at least equal to the signal bandwidth, where bandwidth is defined as the range within which almost all of the spectral energy falls.

The number of samples per symbol must be an integer. The symbol boundary is assumed to fall halfway between two samples; that is, the first sample of each symbol is one-half sample period after the leading edge of the symbol and the last is one-half period before the trailing edge.

Two basic types of modulation are available in the simulation, frequency-shift keying (FSK) and phase-shift keying (PSK). Two forms of PSK are multiple-code-shift keying (MCSK) and differential PSK (DPSK). Each of these uses normal PSK modulation but codes the symbols before the modulation. These types of coding are described in Section 7.3.2. Frequency-shift keying is divided into three types, single-tone FSK which we refer to here as FSK, multiple-tone FSK (MFSK), and minimum-shift keying (MSK).

7.3.1.1 Phase-shift Keying

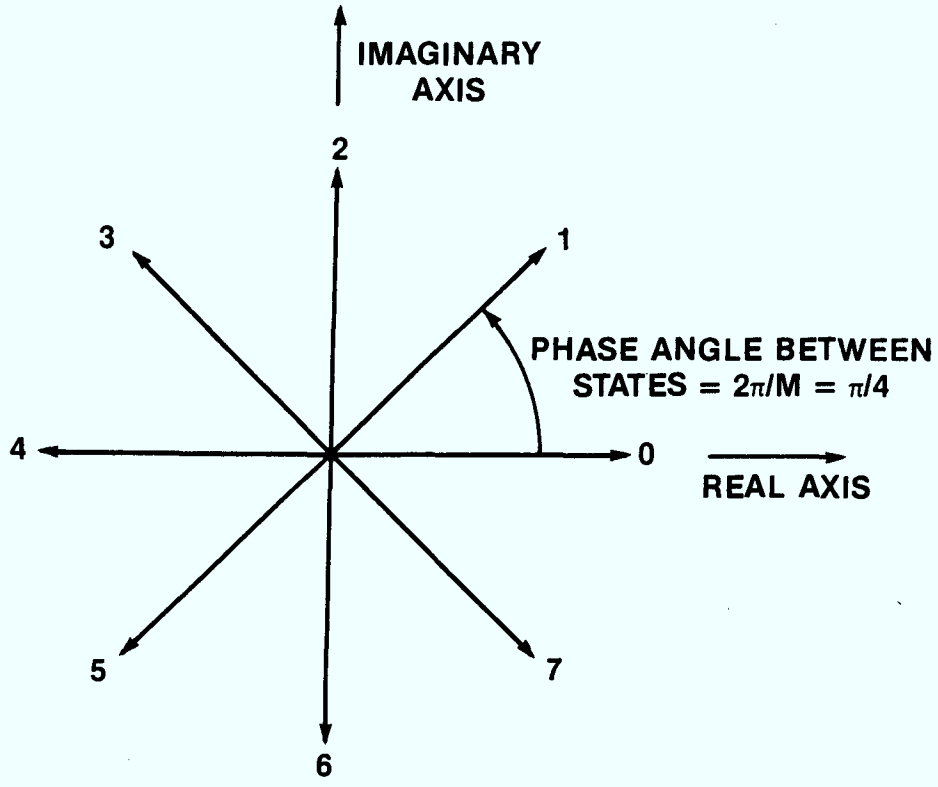
In PSK the waveform has zero carrier frequency and a constant phase selected from one of a set of possible phases corresponding to the set of symbols to be sent. If the data has been encoded into symbols of N_b bits each then the 2^{N_b} phases are chosen by dividing 2π radians into 2^{N_b} equal parts as shown in the example of Figure 7.4. For an integer symbol, I, $(I=0,1,2,\ldots,2^{N_b}-1)$, the phase is simply $2\pi I/2^{N_b}$ radians. The magnitude is equal to the rms voltage specified by the user, since the samples are complex. The number of samples generated for each symbol is specified by the user. As a result of the zero carrier frequency, all samples representing one symbol are identical.

7.3.1.2 Frequency-Shift Keying

In single-tone FSK a symbol of N_b bits produces one tone from a set of $M=2^{N_b}$ possible tones. The tones are symmetrically positioned about zero frequency with equal spacing as shown in Figure 7.5. The user specifies the frequency spacing between adjacent tones. The tones will have frequencies of

$$-(M/2-1/2),\ -(M/2-1/2-1),\ \ldots\ -1/2,\ 1/2,\ \ldots\ M/2-1/2-1,\ M/2-1/2$$

times the specified separation, with the first corresponding to symbol 0, the second to symbol 1, etc. Each new symbol waveform begins with a complex sample having a magnitude equal to the user-specified rms voltage and a random phase selected from a uniform distribution between 0 and 2π radians. The remainder of the N_s samples (N_s = number of samples per symbol) in that symbol waveform have the same magnitude as the first, but their phase increases by $2\pi f_t$ radians for each succeeding sample, where f_t is the tone frequency as a fraction of the sample frequency. The tone separation is usually made equal to the symbol rate, since this makes the symbols orthogonal (a filter matched to one will have zero response to any other one). A closer spacing will require less overall bandwidth but will

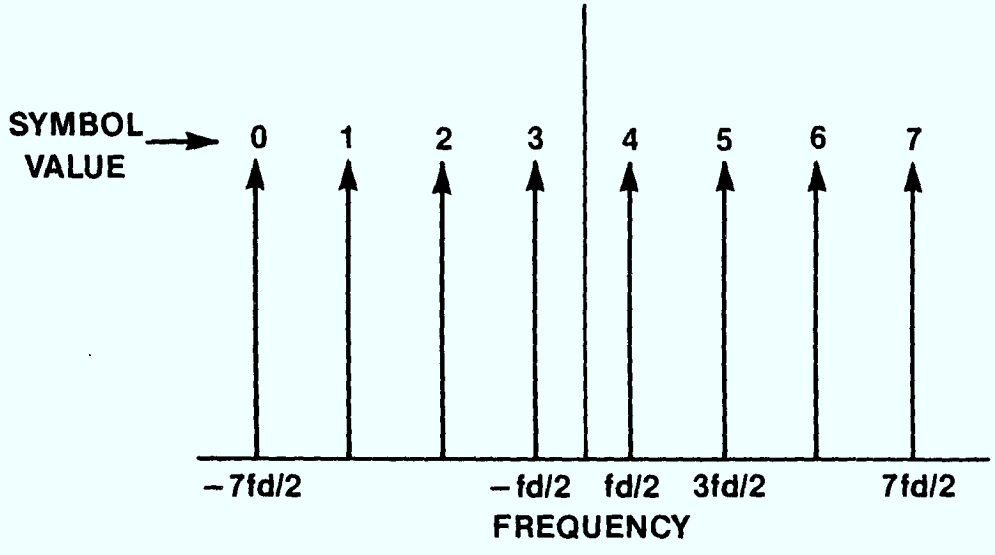


N_b = NUMBER OF BITS PER SYMBOL = 3

M = NUMBER OF PHASES = 2^{N_b} = 8

DATA SYMBOLS = 0, 1, 2, 3, 4, 5, 6, 7

Figure 7.4 Example of PSK Modulation



fd = USER-SPECIFIED TONE SEPARATION

NOTE: EACH UPWARD ARROW REPRESENTS A TONE AT THE FREQUENCY GIVEN BY THE HORIZONTAL AXIS. THE ACTUAL SPECTRUM FOR EACH TONE WILL HAVE SOME WIDTH AS A RESULT OF THE FINITE SYMBOL DURATION. ONLY ONE TONE IS PRESENT AT ANY TIME.

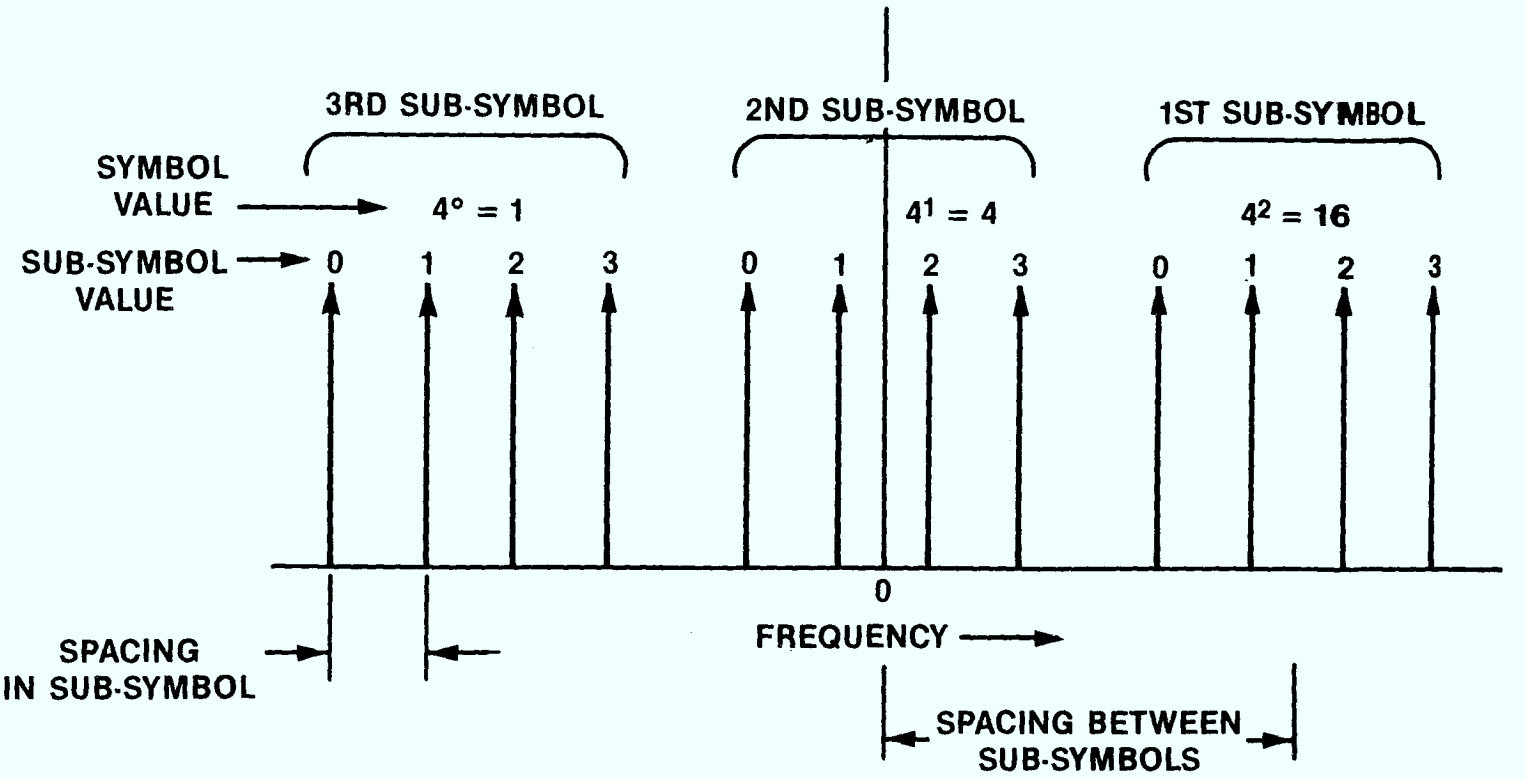
Figure 7.5 Example of Single-Tone FSK

result in reduced bit-error-rate performance. A wider spacing will not result in reduced performance if the spacing is a multiple of the symbol rate, and in only slightly reduced performance otherwise. If symbol envelope shaping is applied (to be described later) the best performance will generally be obtained for spacings of between one and two times the symbol rate, depending on the type of shaping selected. The program will reject any spacing that puts the highest tone above half the sample rate.

7.3.1.3 Multi-Tone Frequency-Shift Keying

In multi-tone FSK the data symbol is divided into sub-symbols and each sub-symbol generates a single tone, with the tones from all sub-symbols added to form the output waveform. The arrangement of the tones is illustrated by the example in Figure 7.6. In this example the user has specified six bits per symbol, and three tone sets or sub-symbols. Note that the number of sets must be a factor of the number of bits per symbol. The user is informed of this requirement, and is asked to enter a new value if he does not satisfy it. When the number of sets is even, they are arranged symmetrically on either side of zero frequency. The user must specify the separation between sets as well as the separation of the tones within each set. Only one value is allowed for each of these parameters and applies to all tone sets. Normally the frequency separation between sets is chosen to make the separation between the highest tone in one set and the lowest in the next set equal to the separation between tones in a set, but the user is not restricted to this value. The program will reject any pair of separations that causes sets to overlap or causes the highest one to exceed half the sample rate.

In the example of Figure 7.6, if the input symbol were 54 (110110 binary), then the first sub-symbol would be 3 (11), the second 1 (01), and the third 2 (10). The corresponding tones, one from each sub-symbol would be generated and added. The generation of any one tone is identical to that in single-tone FSK.



NOTE: EACH UPWARD ARROW REPRESENTS A TONE AT THE FREQUENCY GIVEN BY THE HORIZONTAL AXIS. THE ACTUAL SPECTRUM FOR EACH TONE WILL HAVE SOME WIDTH AS A RESULT OF THE FINITE SYMBOL DURATION. ONLY ONE TONE FROM EACH SET IS PRESENT AT ANY TIME.

Figure 7.6 Example of Multi-tone FSK Modulation

7.3.1.4 Minimum-Shift Keying

Minimum-shift keying (MSK) is a form of FSK in which the tone separation is very small (one-half the symbol rate) and the phase is chosen to be continuous across a symbol boundary. This is done to minimize the modulation bandwidth. Instead of a random phase being chosen at the beginning of each new symbol, the phase at the end of the last symbol is used. This is the phase at the symbol boundary, not that of the last sample of the the last symbol; this sample is a half sample period before the boundary. Only single-tone MSK has been implemented in the simulator at this time.

7.3.2 Envelope Shaping and Smoothed Transitions

After modulation has been performed the user may select some form of symbol waveform modification that will control the shape of the spectrum. With the exception of MSK, all modulation types provided have abrupt transitions from symbol to symbol, and this results in a spectrum which has "sidelobes" which decrease slowly with frequency away from the main lobe as shown in Figure 7.7. This is the spectrum of a single waveform symbol of PSK, which is simply a rectangular pulse. The spectrum for FSK is the same except shifted by the tone frequency. This spectrum was plotted from a simulated symbol with 20 samples per symbol, and is shown out to one-half the sample frequency on each end. The part of the spectrum near the edges is affected by aliasing, and does not correspond exactly to the theoretical unsampled function spectrum. The sidelobes may be reduced at the cost of a slight widening of the main lobe of the spectrum, by either of two methods: envelope shaping, or smoothed transitions.

7.3.2.1 Envelope Shaping

Envelope shaping multiplies the modulated symbol by a weighting

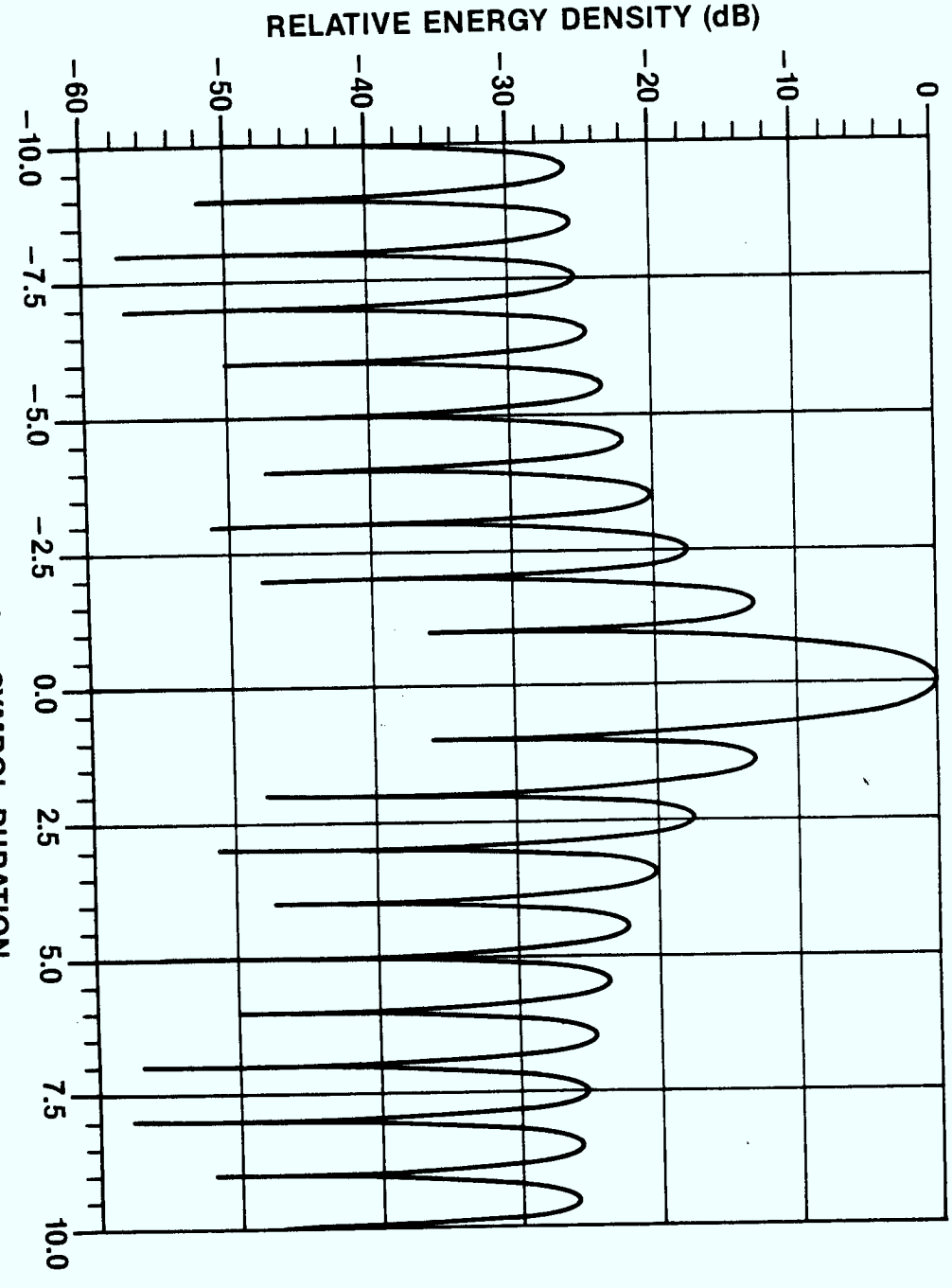


Figure 7.7 Spectrum of Rectangular Symbol (No Shaping)

function that is maximum at the centre of the symbol and minimum at each end. Four weighting functions are provided. These are: Hanning, modified Hanning, Hamming, and sine of sine of sine. They are specified by the equations below, and are graphed in Figures 7.8 to 7.11. The spectra of symbols shaped by these functions are plotted in Figures 7.12 to 7.15.

For Hanning shaping (shown in Figure 7.8) the weight is:

$$W(t) = \{\cos[\pi(\frac{t}{T} - 0.5)]\}^2, \qquad \text{for } 0 < t < T$$
$$= 0, \qquad \text{otherwise,}$$

where t is time from the start of the symbol and T is the symbol duration. This function may also be written as follows:

$$W(t) = \frac{1}{2} - \frac{1}{2}\cos(2\pi t/T) \qquad \text{for } 0 < t < T$$
$$= 0, \qquad \text{otherwise.}$$

In modified-Hanning shaping the cosine-squared weighting is applied to only a part of the symbol on each end, and the middle of the symbol is left unweighted. A parameter m must be specified to determine the portion of the symbol that is weighted. The parameter m is the ratio of the entire symbol duration to the portion that is weighted. Thus the weighting occupies a time T/2m on each end. An example of the modified-Hanning function with m = 2 is shown in Figure 7.9. The weight for the general modified-Hanning shaping is:

$$W(t) = \cos^2[\pi(mt/T - 0.5)], \qquad \text{for } 0 \leq t < T/2m$$
$$= 1, \qquad \text{for } T/2m \leq t \leq (1-1/2m)T$$
$$= \cos^2[\pi(mt/T - m + 0.5)], \qquad \text{for } (1-1/2m)T < t \leq T.$$

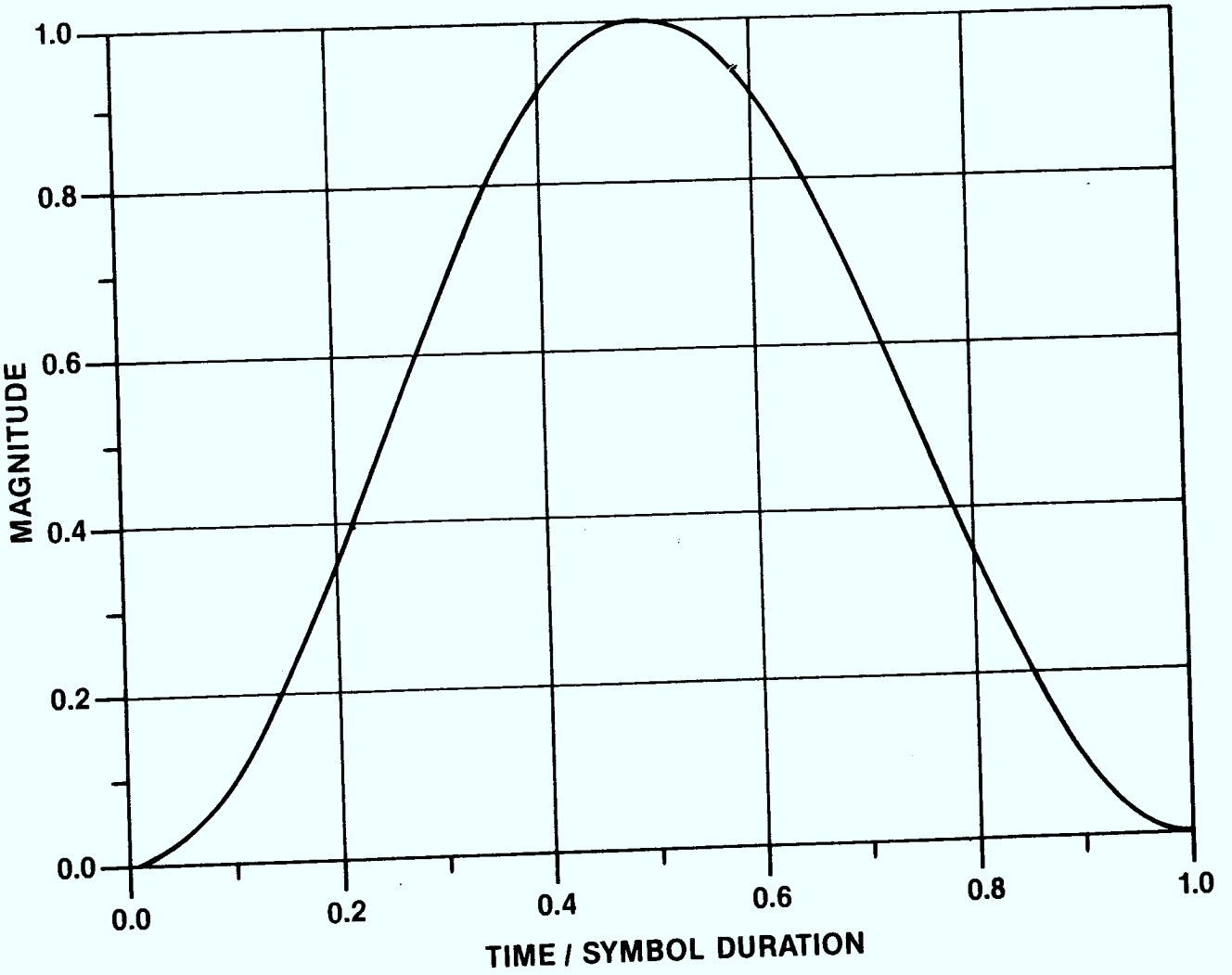


Figure 7.8 Hanning Shaping Function

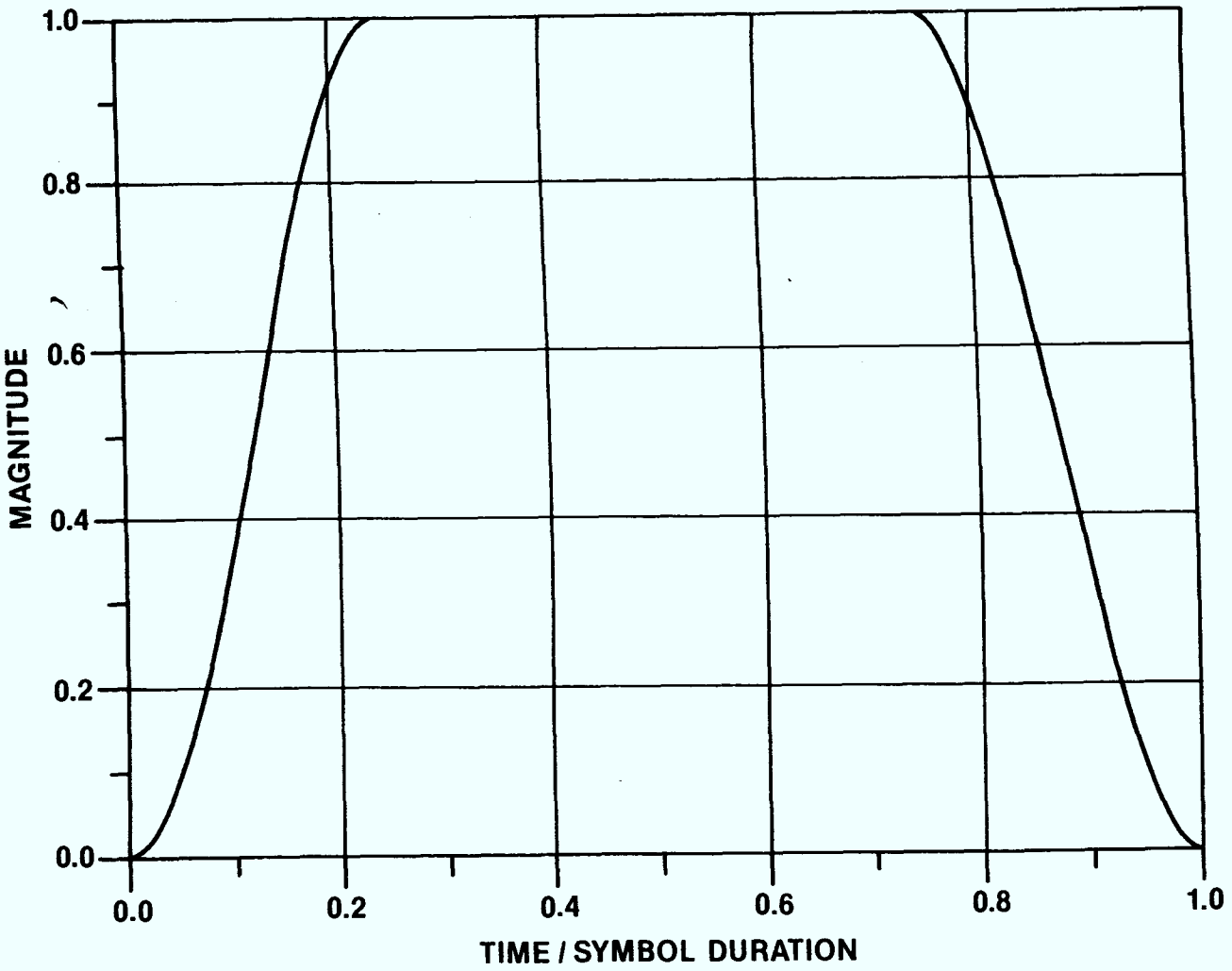


Figure 7.9 Modified-Hanning (m=2) Shaping Function

The Hamming function, shown in Figure 7.10, is similar to the Hanning function but is raised slightly so that it does not go to zero at the ends. The weight is given by:

$W(t) = 0.54 - 0.46 \cos(2\pi t/T)$, for $0 \leq t \leq T$

$= 0$, otherwise.

The sine of sine of sine function is shown in Figure 7.11. Its weight is:

$W(t) = \sin\{\pi/2 \sin[\pi/2 \sin(\pi t/T)]\}$, for $0 \leq t \leq T$

$= 0$, otherwise.

The spectra for the above functions, shown in Figures 7.12 to 7.15, were generated from sampled waveforms with 20 samples per symbol. Thus, the half-sample-rate points are at plus and minus ten times the symbol rate, and the spectra near the ends are affected to some extent by aliasing.

Shaping reduces the energy in the modulated symbol. The reduction factors are .375 for Hanning, 1-.625/m for modified Hanning, .3974 for Hamming, and .769 for sine of sine of sine. The energy per symbol before and after shaping is computed and displayed for the user.

The best shaping function for any system will depend on the particular requirements of that system since there will be tradeoffs among the various characteristics including: energy for a given peak power, width of the main lobe of the spectrum, level of the near-in sidelobes, and level of the farther-out sidelobes. A detailed look at the information and plots provided above can help in the choice of a function for a particular situation.

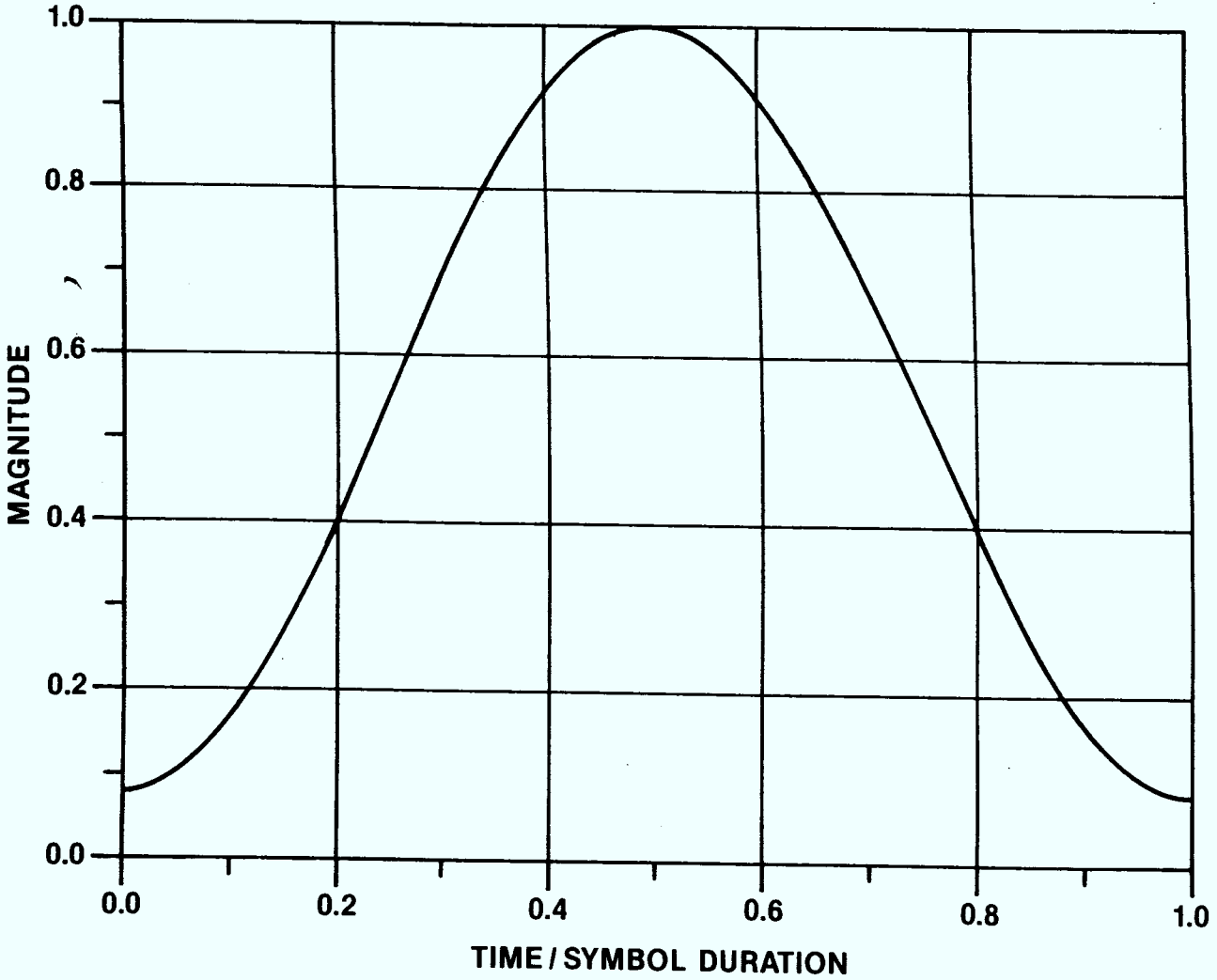


Figure 7.10 Hamming Shaping Function

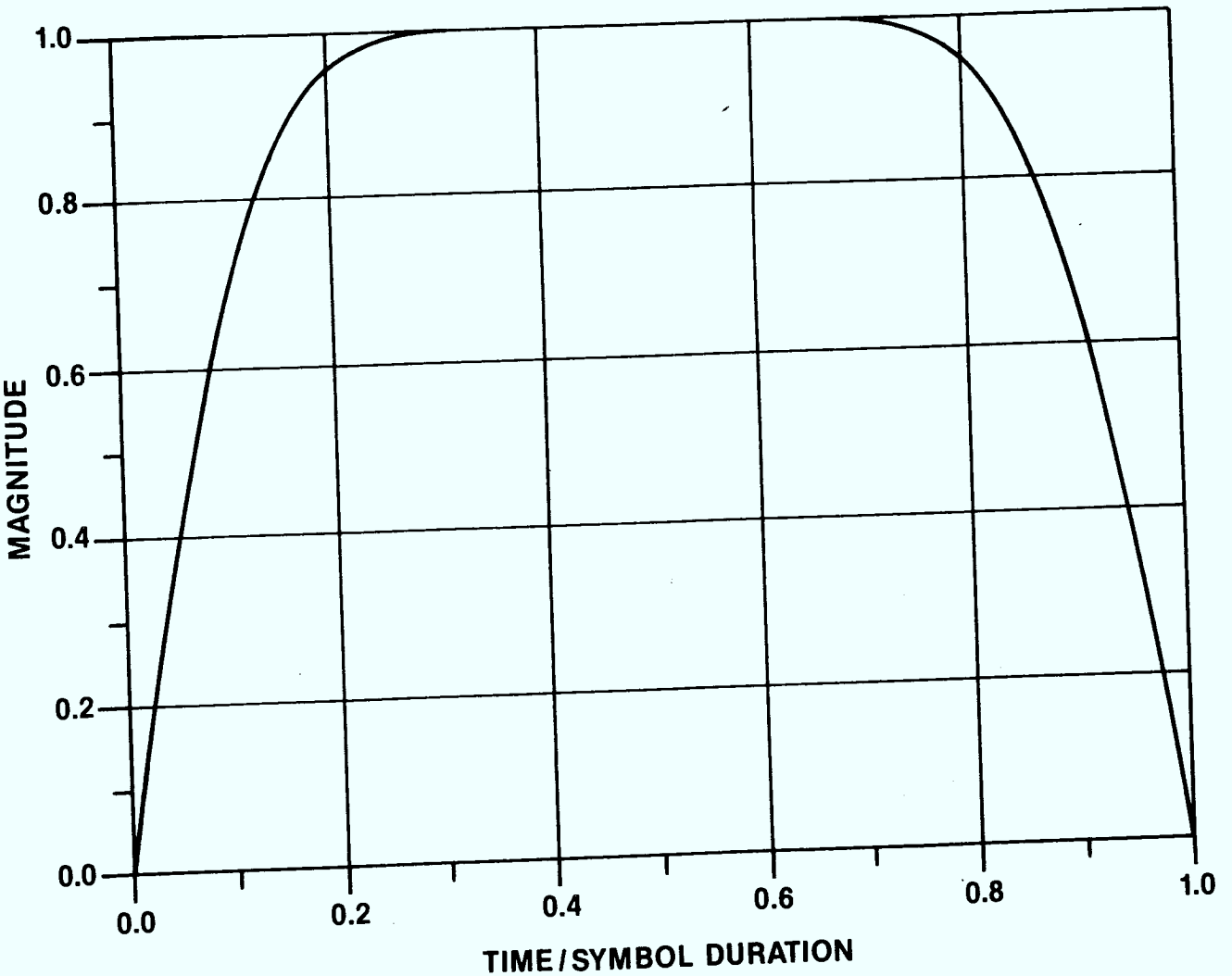


Figure 7.11 sine of sine of sine

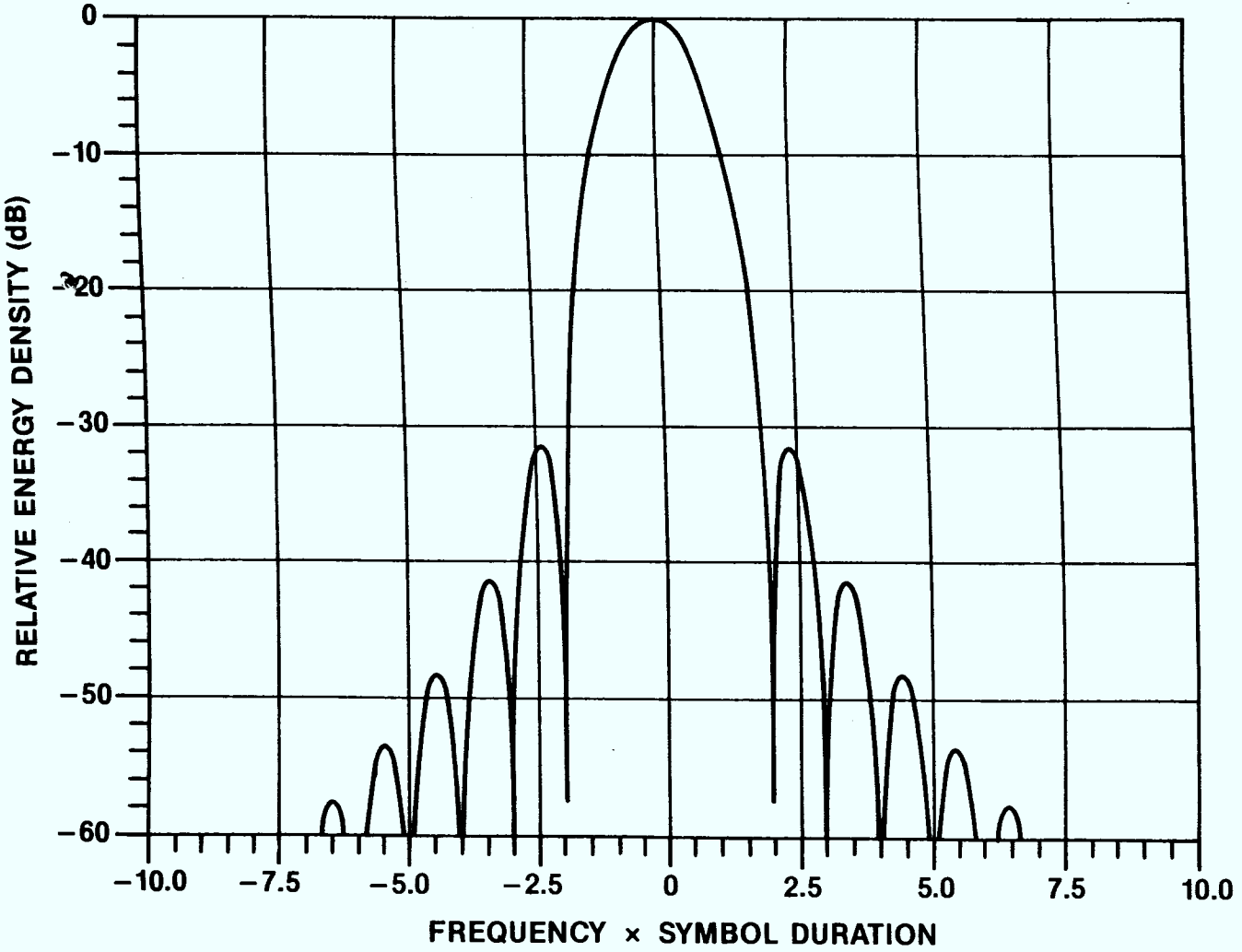


Figure 7.12 Spectrum of Hanning-Shaped Symbol

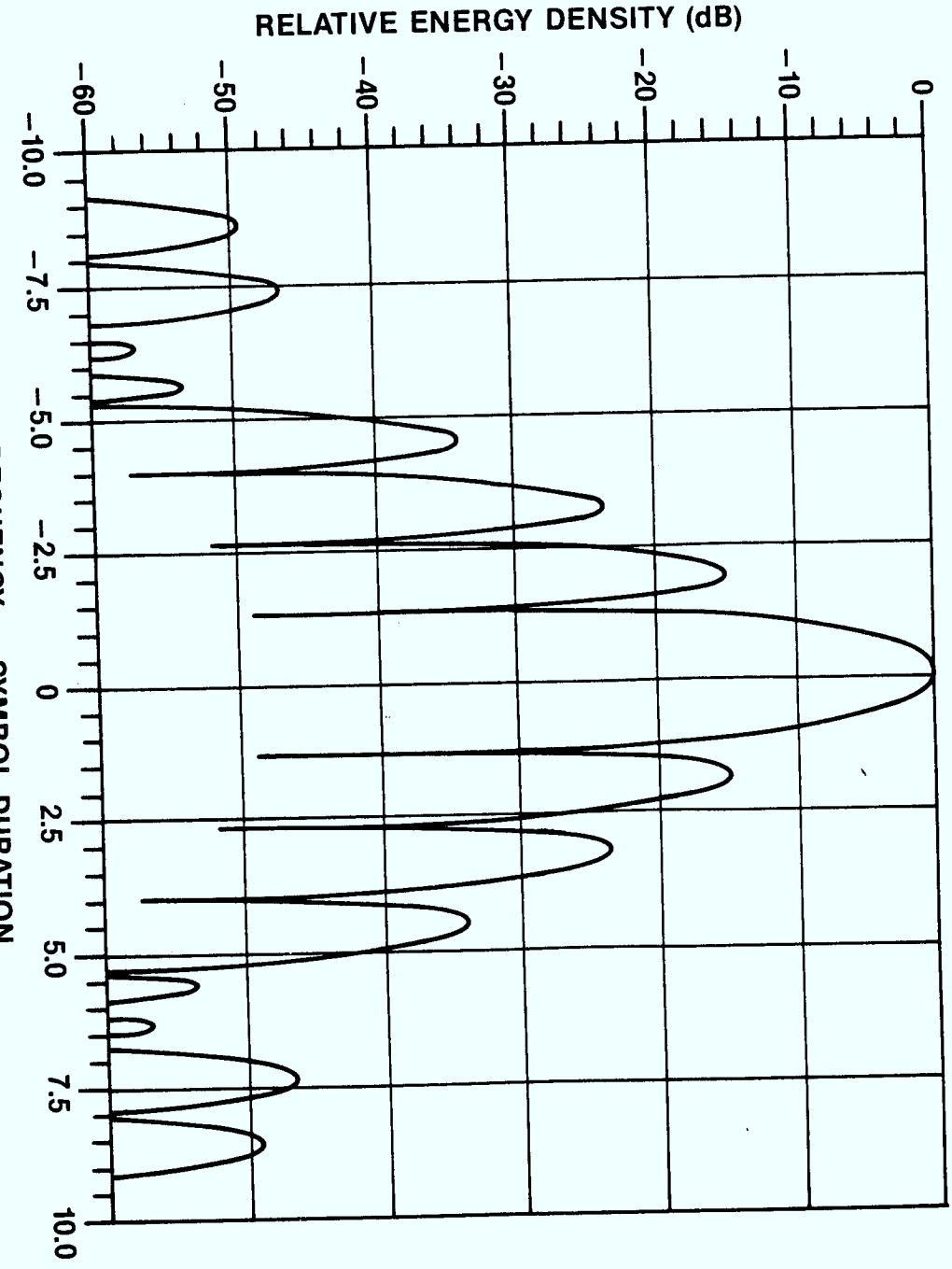


Figure 7.13 Spectrum of Modified-Hanning-Shaped Symbol

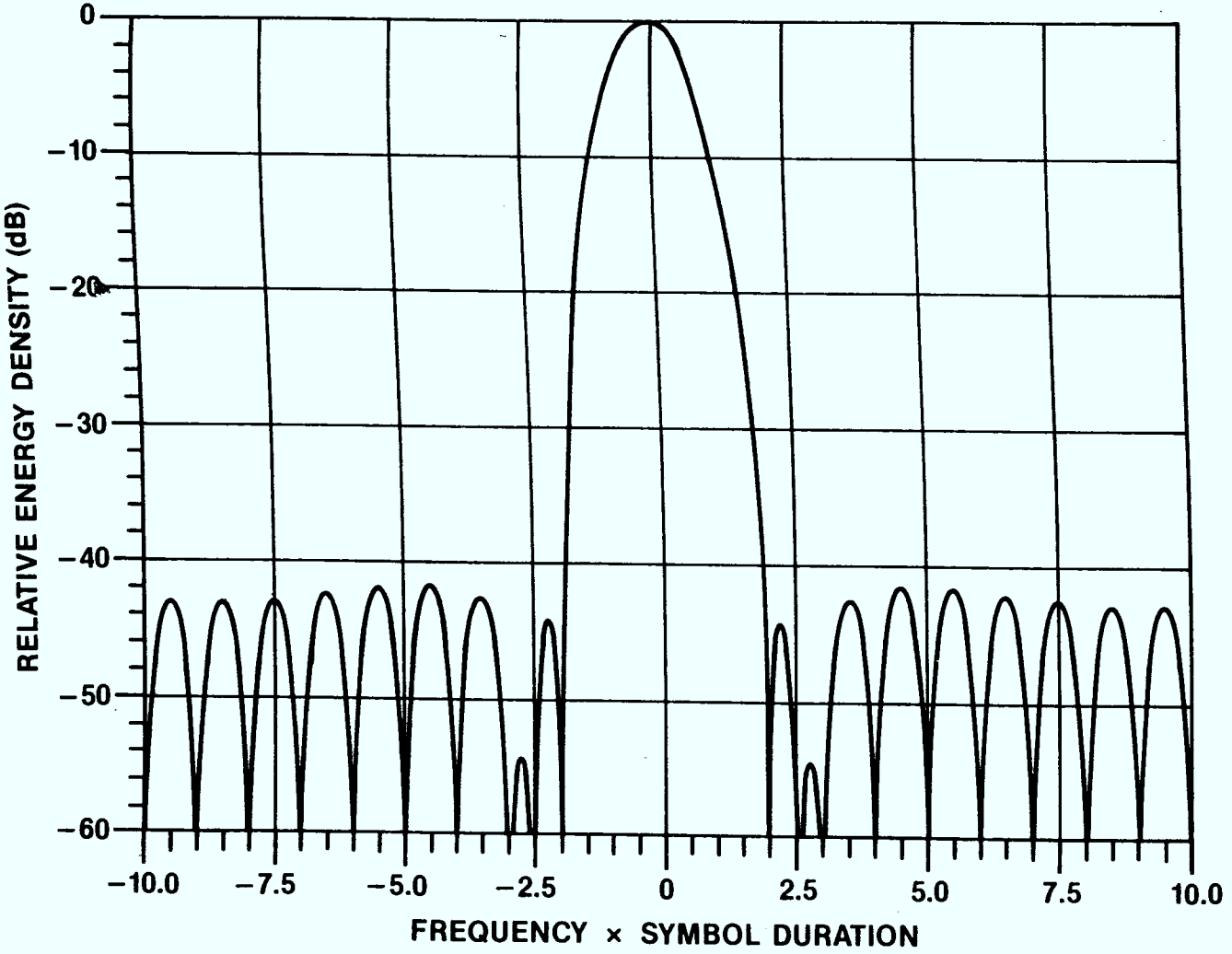


Figure 7.14 Spectrum of Hamming-Shaped Symbol

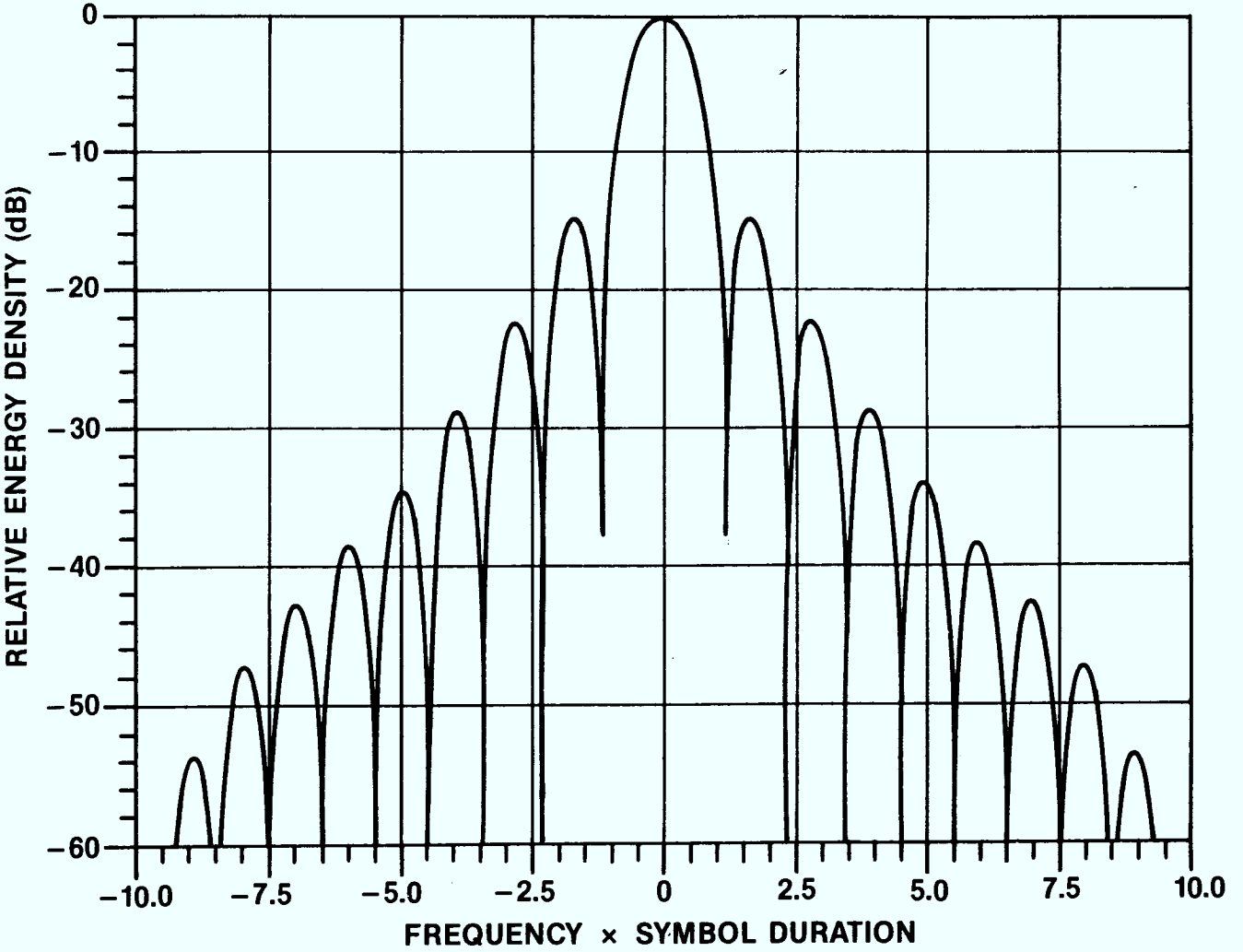


Figure 7.15 Spectrum of sine-of-sine-of-sine-shaped Symbol

7.3.2.2 Smoothed Transitions

The other method of spectrum control, smoothed transitions, may be applied only to PSK-modulated symbols. It modifies a part of adjacent symbols near their boundary to provide a smooth transition from one to the other. Phase, or magnitude, or both may be affected. A choice among three types of transitions is provided. A user-specified parameter that applies to all three is the transition ratio, which is defined in Figure 7.16 as the ratio of the transition period to the total symbol period. Half of the transition period occurs in each symbol.

The first type of transition is a transition of both the real and imaginary components in a sinusoidal fashion over half a period of the sinusoid. Let x_1 and x_2 be the values of the real or imaginary component of symbols 1 and 2 respectively. Then the value of that component in the transition region, with the time origin taken at the symbol boundary, will be:

$$x = (x_2 + x_1)/2 + (x_2 - x_1)/2 \sin(\pi t/T), \qquad -T/2 < t < T/2$$

This transition type affects both amplitude and phase except when the symbols are binary, in which case only the amplitude is affected.

The second type of transition is a linear transition of phase from one symbol to the next, with amplitude held constant. This has the advantage of maintaining the energy in the symbol, but the disadvantage of modifying the information-carrying part of the symbol.

The third type of transition is one in which the phase is held constant, with the amplitude made to change sinusoidally through zero. However, if two adjacent symbols are the same there is no change. If the symbol amplitude is A, the amplitude in the transition region $-T/2 < t < T/2$ is given by:

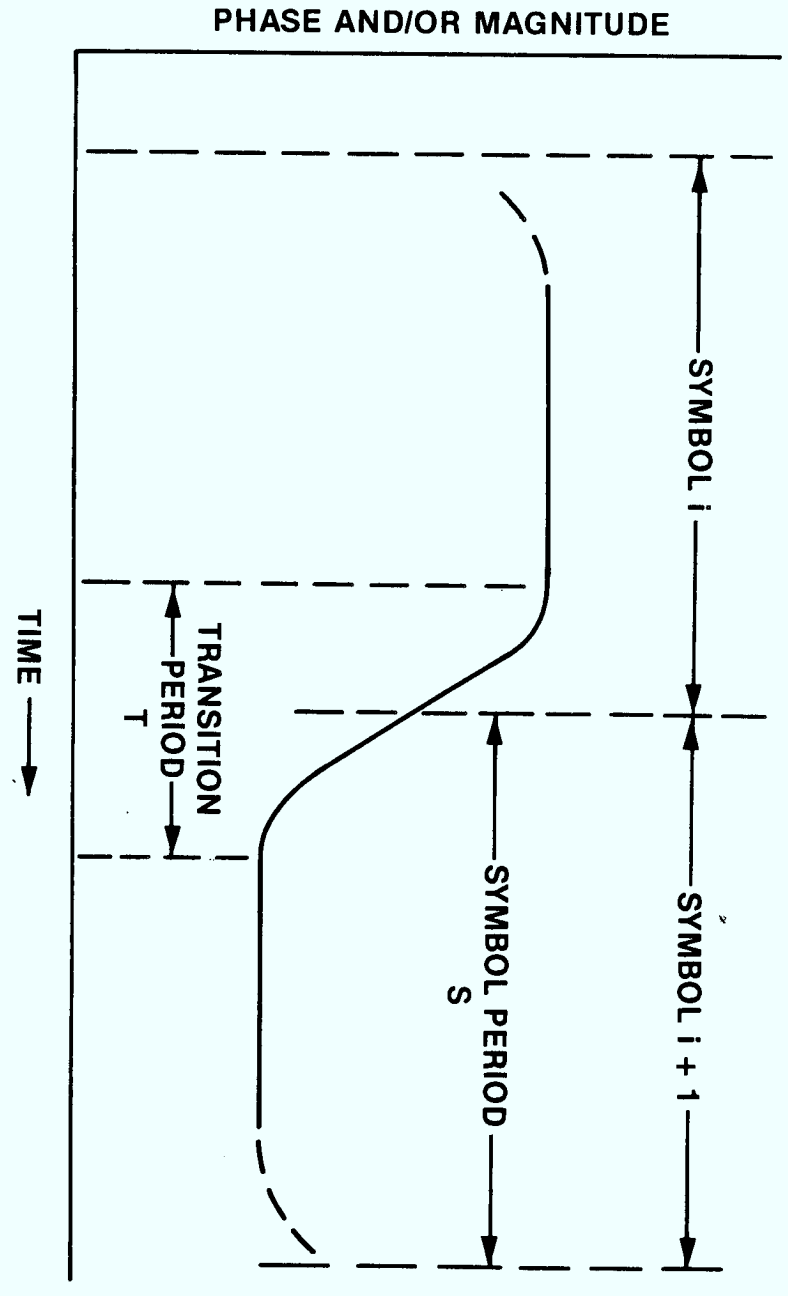


Figure 7.16 Transition Ratio Definition

$$a = \left| A \sin(\pi t/T) \right|, \qquad \text{when the symbols are different,}$$

$$= A, \qquad \text{when the symbols are the same.}$$

This method could be considered a type of shaping. The difference is that the shaping is not applied when it is not needed, that is, when adjacent symbols are the same. This increases the average energy, but the increase becomes less as the number of bits per symbol increases, since the likelihood of identical adjacent symbols becomes less.

The detailed spectra for these transitions depend on the data that have been modulated. It should be possible to compute averaged spectra for very long pseudo-random data sequences, but that has not been done at this time. We can say that transitions will significantly reduce the spreading of the spectrum that results from the abrupt changes between symbols, but a quantitative analysis is left to the user.

7.3.2.3 Effect on Aliasing

The reduction in spectrum spreading realized from shaping or transitions can help reduce the number of samples per symbol required to prevent serious aliasing. Since the modulation is performed before the shaping or transitions, it might at first appear that more samples would be required before the spectrum width is reduced. But this turns out to be unnecessary. Although the samples may not well represent the unshaped waveform, they do represent the values at the points sampled exactly. Since the shaping only multiplies these points, the aliasing in the original spectrum has no effect, and as long as there are enough samples to prevent aliasing in the shaped waveform, the waveform will be well represented.

7.3.3 Coding

Data is originally generated in binary form (bits), but since some modulation schemes use more than two waveforms, there is a need to combine bits into higher-level symbols to provide the source for this modulation. These symbols are simply the integer values represented by a group of bits where the first bit of an N-bit symbol has value 2^{N-1} the second 2^{N-2}, etc. Only an integer number of bits may be coded into a symbol; that is, the symbol alphabet must consist of a number of elements equal to a power of two with a maximum number of elements of 2^{24}. When bits are formed into symbols the data rate reported by the program changes to show the number of symbols per second instead of the number of bits per second.

The first thing the MODCOD process does is ask for the number of bits per modulation symbol. This is the number that specifies the number of modulation symbols as indicated above. There is the possibility of some confusion here if multiple-code-shift keying (MCSK) is selected since MCSK, which will be described later, may be considered both a type of code and a type of modulation. Therefore, one may think of the MCSK symbols as modulation symbols. One should not think of them as such when entering the desired number of bits per symbol (a message warning about this is displayed when the question is asked). The symbols referred to there are the modulation symbols that are the elements of MCSK symbols when they are used.

Another fact worth noting here is that entering a value for the number of bits per modulation symbol does not cause the symbols to be formed. To accomplish this the user must answer "yes" to the next question on whether symbols are to be formed. He must also answer "yes" if he wishes to include any of the types of coding other than direct-sequence encoding. There may be occasions when the user may elect not to form symbols even when he enters a number greater than one for the number of bits per modulation symbol. One of these is when he wants only one data bit to be encoded in each symbol but wishes to add direct-sequence encoding with M-ary (with $M > 2$) phase modulation (when direct-sequence coding is

used a modulation symbol refers to an element of the direct-sequence modulation). While it is not clear that such a scheme has any advantages, it is allowed.

7.3.3.1 Inverse Gray Encoding

When there are more than two different modulation waveforms, they may not be equally distant from one another in the signal space, i.e. there may be a higher likelihood that noise will cause a waveform to be interpreted as one wrong waveform than a different wrong waveform. Under such circumstances it is important to insure that the most likely symbol errors will result in only a single bit error after symbol decoding, in order to minimize the bit error rate. If the waveforms are ordered so that adjacent ones, in terms of symbol number, have minimum distance from each other, then an inverse Gray encoding of the symbols will accomplish this goal. This is so because the original symbols (and those after decoding in the receiver) will have a Gray encoding relative to waveform symbols, which means that adjacent symbols differ in only one bit. Inverse Gray encoding is available in MODCOD for this purpose.

7.3.3.2 Differential Encoding

Differential encoding may be performed on symbols as well as bits. For symbols of N bits, differential encoding consists of adding, modulo-2^N, the present symbol with the previous encoded symbol. Since when encoding the first symbol there is no previous encoded one, a zero is arbitrarily assigned to that encoded symbol.

7.3.3.3 Multiple-Code-Shift Encoding

In MCSK, a number of binary-sequence codes are used to represent the various data symbols. PSK modulation of the code elements results in

the MCSK waveforms. The user specifies the length of the codes (all must be the same length), and for data symbols of N_b bits per symbol he must enter 2^{N_b} different binary sequences, one for each data symbol. These should be chosen to be orthogonal to each other (they should have zero crosscorrelation, at least for zero relative delay), to minimize the probability of error in the receiver. If the codes are orthogonal the bit-error-rate performance is identical to that for orthogonal FSK, since both are forms of non-coherent orthogonal signaling. (When we say "non-coherent" we mean non-coherent from symbol to symbol; both FSK and MCSK signals must be coherent over a symbol duration.)

In a special binary differential mode of MCSK, only a single code is entered, and the same code is used for the two possible binary symbols, but with a phase inversion for one of them. This is equivalent to DPSK with the addition of a spreading waveform (the MCSK code), and is really a form of spread-spectrum modulation. However, the spreading code repeats from symbol to symbol, and this may be unacceptable in an ECCM system. For this reason an alternative mode using a non-repeating code is available for the binary differential MCSK case. In this mode the user does not enter a code at all, but only a seed (or accepts a default seed) for the 31-bit maximal-length sequence (M-sequence) generator. For each data symbol (binary) a new segment of the very long M-sequence is used as the MCSK code, with its phase determined by the data symbol. The result is identical to that for DPSK with direct-sequence spread-spectrum modulation. However, this particular form of MCSK was designed to allow the use of an MCSK matched filter in the receiver rather than the correlation de-spreading usually used in direct-sequence systems. This has certain advantages as will be discussed under the receiver section. While the modulation could have been produced by the normal direct-sequence coder, a separate MCSK coder was used for consistency with the receiver matched-filter demodulator.

The differential MCSK does not use the differential encoder already described. Differential encoding is included in the MCSK routine.

MCSK elements may be given weight greater than one for use with multi-phase direct-sequence modulation. But MCSK coding always uses binary elements to produce phase changes of zero or π radians. For example, if the direct-sequence modulation has four phases, the MCSK elements will be given weight 2 so that it will affect the phase by zero or 2 times the phase increment set by the direct-sequence modulation or $2\pi/2 = \pi$ radians. This weight is automatically computed by the program as one-half the number of phase states (determined by the number of bits per data symbol).

MCSK encoding will change the data rate reported by the program. This rate is the rate of MCSK elements per second. These elements are referred to as symbols by the program. They should not be confused with MCSK symbols that comprise a number of these elements.

7.3.3.4 Direct-sequence Encoding

In a direct-sequence spread-spectrum system the phase-modulated data symbols are, in effect, multiplied by a phase-modulated pseudo-random symbol sequence of much higher rate to produce a new signal with a much higher bandwidth. In practice this is accomplished by a modulo-2^N addition of the two sequences before phase modulation, where N is the number of bits per symbol (multiplication of complex waveforms results in addition of their phases). This spreading of the bandwidth of the transmitted signal provides advantages in combatting the effects of jamming and in reducing the probability of intercept. The processing gain against a jammer of limited total power is equal to the ratio of spread bandwidth to data bandwidth. This ratio is also equal to the factor by which the signal spectral power density can be reduced without increasing the error rate, and hence is a measure of gain against an interceptor. Information on direct-sequence systems may be found in References [8] and [9].

In the simulation, maximal-length sequences (M-sequences) generated by a feedback shift register are used for spreading and de-spreading. These have a length of $2^{31}-1$ before repeating. Feedback taps are at stages

31,30,28, and 16. In real ECCM systems such codes are rarely used because it is too easy to determine the full sequence from a few elements. Instead, modified codes that are much harder to determine will be used. However, whatever codes are used, they are not likely to have statistics significantly different from those of M-sequences, and only M-sequence generators have been implemented in the simulator at this time.

The spreading symbols are formed from the M-sequence bits by grouping N_b bits (the number of bits per symbol) into a symbol in the same way that the data bits are formed into symbols. The same parameter N_b controls the generation of symbols for both the data and the spreading code, and these will therefore normally both have the same number of bits per symbol. However, if the user has not asked for the data to be formed into symbols (he must still specify N_b) it is possible that the data will have symbols of fewer bits than the spreading code.

The user specifies the number of spreading symbols per data symbol, N_c. This causes N_c spreading symbols to be added modulo-2^{N_b} separately to each data symbol to produce N_c output symbols for each data symbol. The data rate is thus increased by a factor of N_c and the new data rate is computed and displayed. The spread data is still in the form of integer data at this point; it is converted to a spread waveform when PSK modulation is performed later.

7.4 Frequency-Hop Encoding Process (HOPPER)

When frequency hopping is performed, the carrier frequency of the signal is changed at regular intervals. Each new frequency is selected from a set of possible frequencies in a pseudo-random fashion. Frequency-hopping systems are discussed in more detail in Reference 9. In the simulator, a 31-bit feedback shift register with feedback from stages 31, 30, 28, and 24 (giving a maximal-length sequence of $2^{31}-1$ bits) is used to generate the pseudo-random numbers for frequency selection. The user specifies the number of bits, N_f, to be used for the frequency number.

This determines the size of the frequency set as 2^{N_f}. Bits from the shift register are taken N_f at a time to form N_f-bit words defining the particular frequency number (frequencies are numbered 0 to $2^{N_f}-1$) from the set. These are multiplied by a user-specified multiplier and added to a user-specified offset to determine the actual frequency, in khZ, to be transmitted. For example, if N_f were chosen as 5, the multiplier as 10, and the offset as 8,000 (these last two need not be integer), then there would be 32 frequencies in the set, with frequencies 8000 khZ, 8010 khZ, ..., 8310 khZ. If the bit sequence 01101 were generated, this would represent frequency number 13 or 8000 + 13x10 = 8130 khZ. The value of N_f is limited to a maximum of 28 (over 268 million frequencies).

The user sets the hop rate by specifing the number of samples (complex values) per frequency word. This is limited to a maximum of 1024.

Since the simulation is carried out entirely at baseband, the simulated signal is not actually hopped in frequency but the samples at a given frequency are grouped into a block with the first two words of the block (one complex word) being used to indicate the frequency of that block and the number of samples in the block. The propagation medium routines make use of the frequency words to determine which propagation characteristics to apply to the samples in that block. The front end of the receiver also looks at the frequency words to determine whether to accept the samples (a frequency-hop generator in the receiver determines receiver tuning).

The spacing between possible frequencies (the multiplier) normally should not be less than the modulation bandwidth, or the hop rate if that is greater.

7.5 Propagation Medium Process (MEDIUM)

As an HF propagation medium the ionosphere is characterized by multipath, Doppler shifts and spreads, and fading. Fading is a natural

result of the addition of multipath components with different Doppler frequencies, but fading of individual components is also common as a result of unresolvable components in the apparently single paths. A frequent cause of this latter phenomenon is the presence of both ordinary and extraordinary rays with only slightly different paths. Also, in some cases, instead of discrete multipath components a time-delay spread exists, resulting a time smearing of the received signal. These characteristics are a function of frequency and this complicates the design of frequency-hop systems.

The simulator attempts to model all these conditions. Four types of path are available, and these may be combined (outputs added). The types are: perfect transmission (output = input), no transmission (output amplitude = 0), multiple delays with user-specified fixed gain and Doppler, and multiple delays with independent Rayleigh fading for each (mean gain, mean Doppler, and fading rate specified by the user). In the last two types the initial delay and incremental delay are specified by the user. When the incremental delay is made much less than the time resolution (inverse of bandwidth) of the communication system the propagation path will appear spread in delay to the system rather than comprising discrete components. These path types will be described in more detail later.

After answering some preliminary questions relating to general conditions, the user specifies a number of "paths". Each "path" may actually comprise multiple paths but the term is used in the simulator to identify a propagation mode, which is then specified over a number of frequency ranges. For each path, the user is asked the number of frequency ranges over which it will be specified. The total of these ranges always covers the the entire frequency range from zero to infinity; this question only determines how many different ranges the frequency will be divided into, and allows a maximum of 64 ranges. The user then specifies the boundaries between the ranges, and the propagation characteristics for each range. The first range always has a lowest frequency of zero. After specifying the type and parameters required for that type, the user enters the lowest frequency of the second range (which also defines the upper

boundary for the first range), and then the type and parameters for that range. This process is continued for the number of frequency ranges specified. The upper boundary of the last range is automatically taken as infinity. When all of the ranges have been specified for path one, the user is asked if he desires any more paths. If he does, the above process is repeated for path two, and for as many paths as he wishes. The boundaries and the number of frequency ranges need not be the same for the various paths. A number of paths of the same or different type may exist in any frequency range. This allows not only a number of different modes to be simulated, but also a modification of the type; for example, Ricean fading may be simulated by combining a Rayleigh and a fixed path.

The first two path types are simple and need no further explanation. The last two, the fixed and Rayleigh-fading multiple-path types, require a more detailed description. This will be provided with the aid of Figure 7.17 which illustrates the algorithm used. A tapped delay line provides the multiple delays. It comprises an initial delay and a number of equal incremental delays separating the individual paths. Each of the delay taps is multiplied by a complex multiplier which is different for each tap. For the fixed path these multipliers have fixed amplitude but may have a phase that changes at a uniform rate to simulate a Doppler frequency. In the fixed-path case, switch S1 is in the upper position and its input is unity with imaginary part equal to zero. Thus the multipliers, G_{ik}, are equal to $A_k \exp(ji\phi_k)$. A_k is the user-specified amplitude multiplier, and ϕ_k is the angle by which the phase increases for each sample. This value, which is also specified by the user, determines the Doppler frequency of that path (the Doppler is ϕ_k multiplied by the sample rate).

In the Rayleigh fading mode, switch S_1 is in the lower position and its input is a complex Gaussian random variable whose time correlation is controlled by the filter. This filter, specified by the user, uses the general filter routine which is described in Section 8.1. The filter bandwidth is usually quite small to provide typical fading rates (the

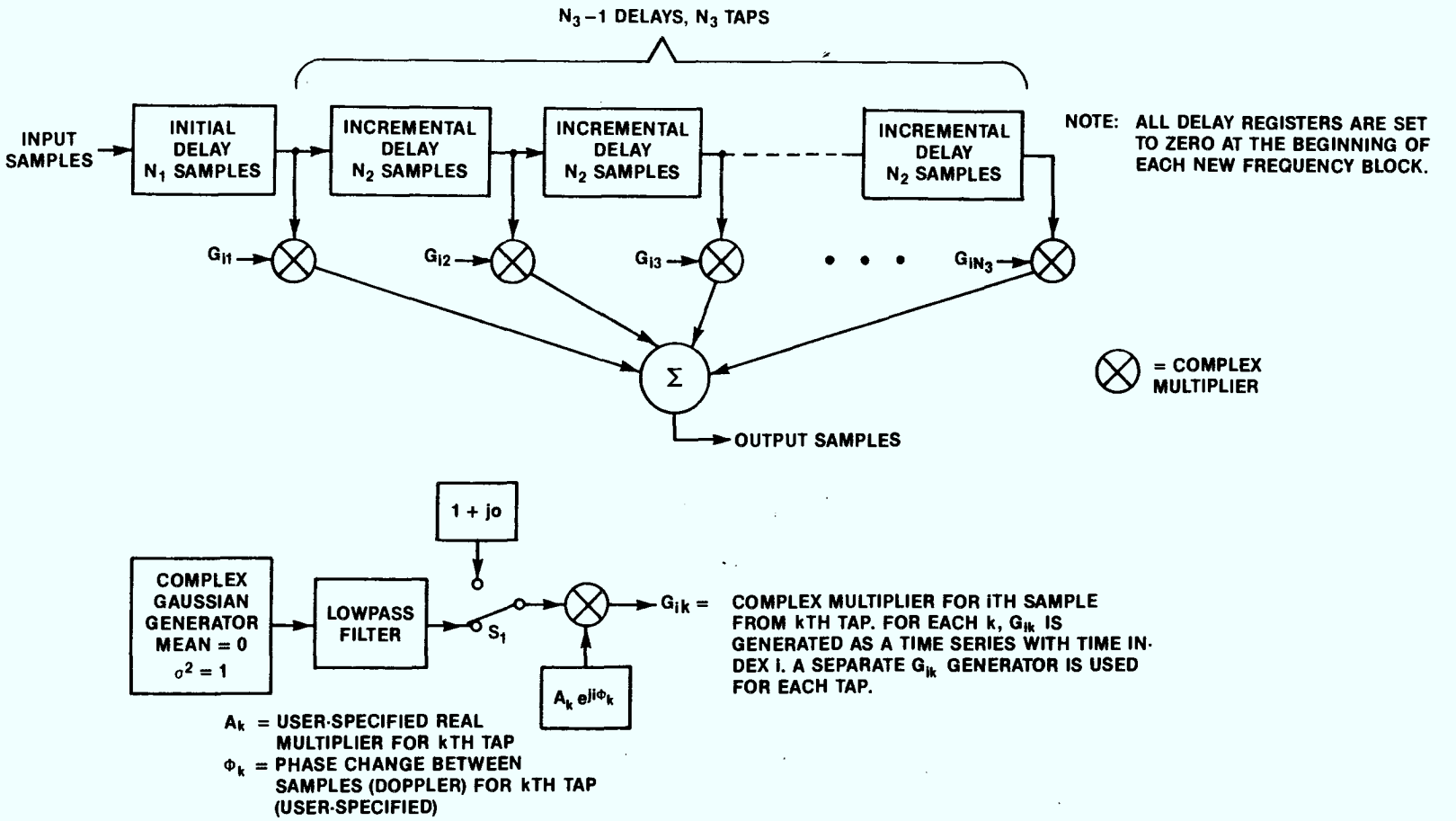


Figure 7.17 Multipath Generation

bandwidth is approximately equal to the fading rate). Section 8.1 on filtering discusses a type of narrow low-pass filter suitable for use in this application. The filter should have an integrated-noise-power gain of unity (as defined in the section on filtering). The term "Rayleigh" refers to the probability distribution of the magnitude of the multipliers, G_{ik}. This is the distribution of the magnitude of a complex variable whose real and imaginary components are independent Gaussian variables of equal variance. In Figure 7.17 the variance of the Gaussian generator is given as unity. This is the variance of the complex variable; the variance of each component is one-half.

The index i is the time or sample-number index and the index k refers to the tap number. A separate generator for the G_{ik} values is required for each value of k. Since the Gaussian generator provides independent samples, the same Gaussian generator is shared among the various G_{ik} generators.

In the Rayleigh mode the amplitude multiplier, specified by the user, determines the rms value of the tap output for unit input. The Gaussian process generating the G_{ik} will provide an rms value of $|G_{ik}|$ of unity if a filter with unity integrated-noise-power gain is used. The Doppler phase increment, also specified by the user sets the mean Doppler. For both fixed and Rayleigh modes, the amplitude multiplier has a minimum of zero and a maximum of one.

Since the propagation-medium simulation allows more than one delay, the output will contain more samples than the input. For example, if the input contains 100 samples and the medium has two discrete paths with delays of zero and 10 samples, then the simulation will produce 100 samples from the first path and 110 from the second (the first 10 will be zero). These will of course be combined, with zeros added to the end of the samples from the first path, to give 110 samples. In a frequency-hop system each block of samples at a different frequency must be treated separately. The frequency word in each block of samples is examined to determine which set of medium characteristics is to be applied to that

block. Now we have extra samples for each frequency-hop block, the ones at the end of one block overlapping in time index with those at the beginning of the next block. These cannot be combined at this time since the receiver must treat the different frequencies differently. Therefore it is necessary to carry separate blocks of samples of different lengths and overlapping time index values. The program keeps track of the total number of samples and informs the user of this value.

7.5.1 Medium Parameter Files

To assist the user in entering medium parameters, a medium parameter file feature has been created. The user may enter the medium parameters either from the terminal or from a file. To create a file, the user must process the data with the parameters entered from the terminal. The medium parameter files are similar to the output parameter files in that they are in ASCII format and can be looked at either by typing them or editing them. The file header takes the same form as in the output files. The user should be careful about changing the file format when editing, otherwise the program may not accept the edited file. Only one ASCII medium parameter file can be used each time the medium process is used. In other words, a medium parameter file contains the parameters for all the paths, not just one.

7.6 Receiver Process (RECVR)

7.6.1 Introduction

As mentioned earlier, the receiver process is a sequential one and the data is passed, one sample at a time, through the entire process. This is required because there is feedback in the synchronization systems; computations on the data at one point affect the parameters at an earlier point. Thus, the entire receiver must be specified before processing is

begun. This does not affect the way the user enters the specifications, but means that instead of having to wait, while each sub-process completes its computations, before he enters specifications for the next sub-process, he enters a much larger number of parameters, and then must wait a much longer time for the computations to be completed. Of course, under batch processing there is no difference.

A feature present only in the receiver is the Monitor feature which provides a screen output of various receiver quantities updated in place at a rate specified by the user. Its purpose is to allow the user to keep track of the progress of the simulation and to determine the performance of various parts of the receiver as an aid in deciding whether to stop the simulation and change parameters. It is useful when the simulation is first being set up to speed up any trial-and-error tests that may be required. Of course, the Monitor feature may be used only in the interactive mode.

Another feature unique to the receiver is the Display feature which permits the user to select one of a number of internal values of the receiver (for example AGC gain or demodulator tracking voltage), and have this recorded so that it can be examined when the run is finished. The run can also be stopped at any time and the values examined. Some statistical parameters, including histograms may be computed during the examination. The Display feature may be used in the batch mode. The Monitor, Display, and run control features will be described in more detail in a later section.

A general block diagram for the receiver process is shown in Figure 7.18. This is intended to indicate the general configuration. Where a number of functions are included in a block, those functions are indicated in brackets under the name of the block. Each of the blocks, will be described in more detail later. Any of the blocks (and even portions of a block) may be omitted in a run. The central line of blocks indicates the functions that may be applied in sequence to the signal. Below these are the blocks that perform the synchronization functions by controlling delays

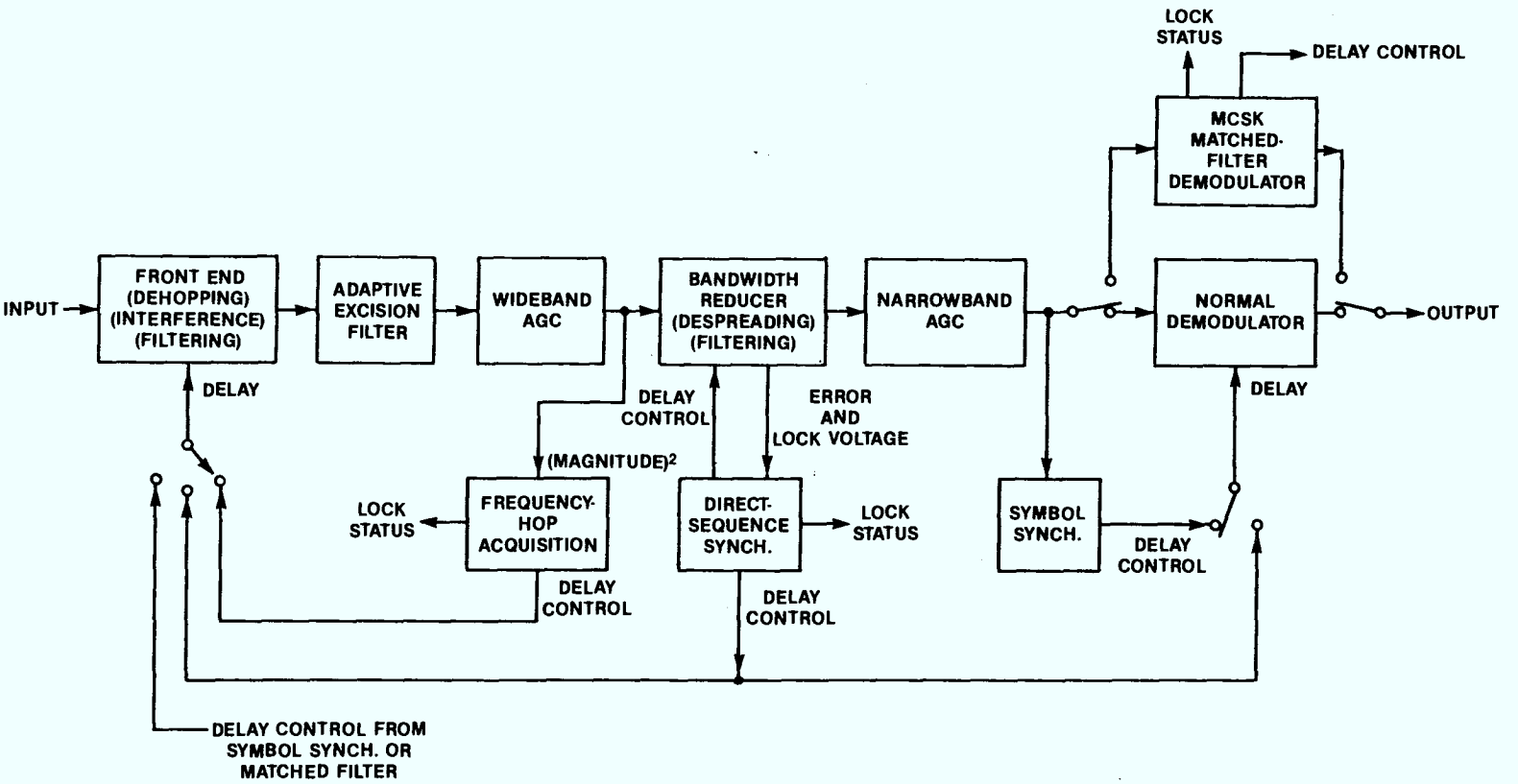


Figure 7.18 Receiver General Block Diagram

of the various reference signals. The switching is intended to indicate which references may be controlled by which synchronization systems. In general, any synchronization system may be used to control the delay in a device of the same or lower time-delay resolution. Not shown are the connections indicating how acquisition systems may pass delay estimates to others, as, for example, when a frequency-hop acquisition is performed and its delay estimate used to set the initial delay for the direct-sequence acquisition. The upper block, the MCSK matched-filter demodulator, is an alternative to the normal demodulator when MCSK modulation is used. It allows demodulation at a number of different reference delays, and combining to provide a type of diversity under multipath conditions.

The individual blocks of Figure 7.18 will now be described in more detail. After that the interactions between blocks will be discussed. The various synchronization systems involve such interactions.

7.6.2 Front End

The functions carried out in the receiver front end are shown in Figure 7.19. Any of the functions may be selected or omitted by the user. The first function is de-hopping of the carrier frequency. The de-hopper reads the frequency word from each block of input samples and compares it with the frequency word from the receiver frequency-hop generator, which indicates the frequency to which the receiver is tuned. If the two words are the same the samples in that block are passed, and if not they are rejected (set to zero). A number of different blocks may contribute to the output at a given instant, since, as explained in the section on the propagation medium, different blocks may overlap in time. The samples passed by the de-hopper must be recombined to provide a single sequence of samples uniformly spaced in time. The number of samples is therefore reduced by the de-hopper to essentially the number that existed before the propagation medium (there may be a few more as a result of the stretching of the signal at the end by propagation delays).

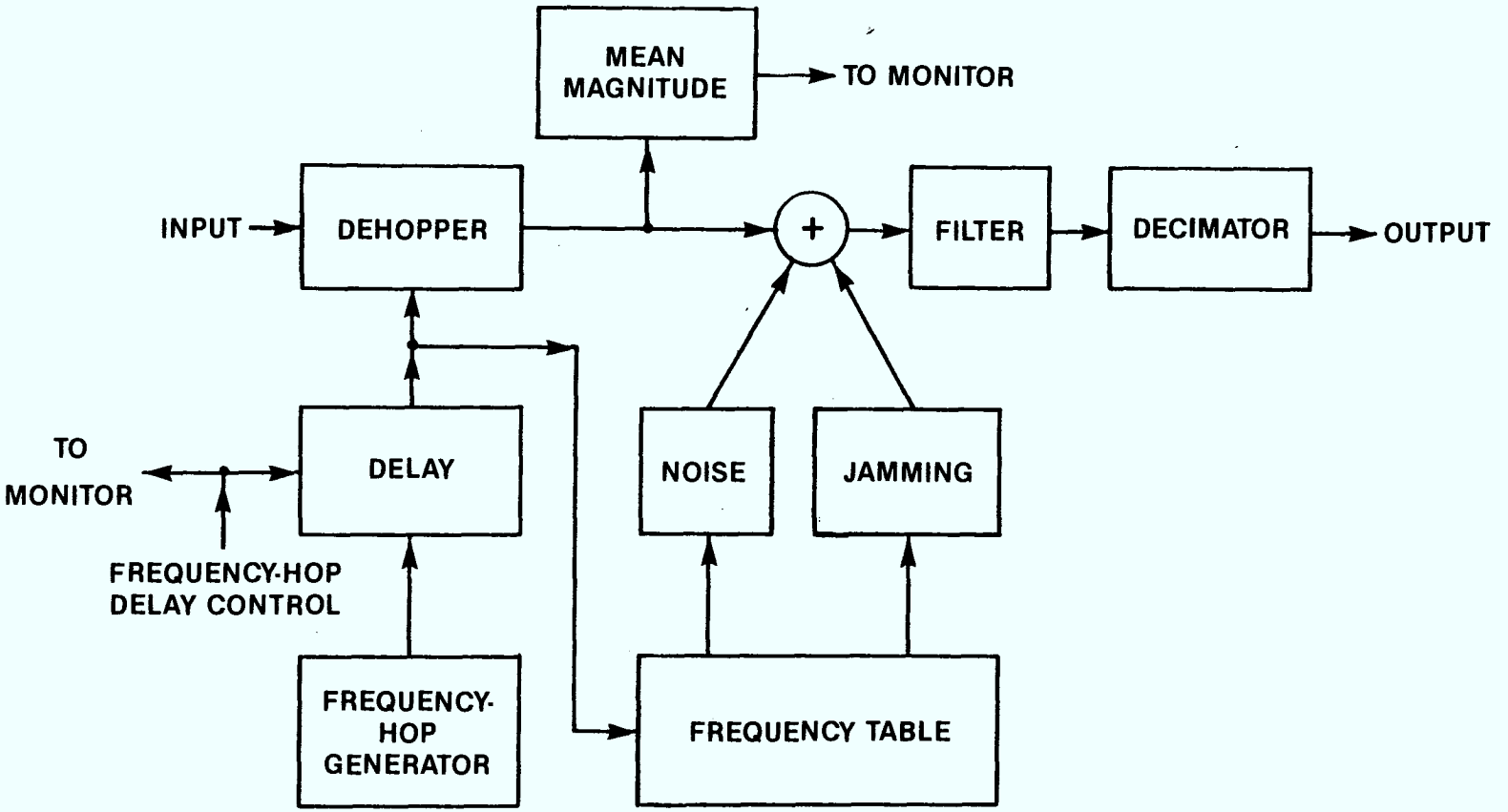


Figure 7.19 Receiver Front End

The frequency-hop words from the generator are delayed before entering the de-hopper. This is to allow synchronization with delayed input samples. The receiver frequency-hop generator should be started with the same seed used for the generator in the transmitter (HOPPER). This can be accomplished with the default seed or identical user-specified seeds. The delay may be set at the start of the run by the user, or dynamically controlled by one of the synchronization systems as will be described later. Delays are specified in number of samples.

Following de-hopping, noise and jamming may be added. The noise and jamming waveforms available have been described earlier in Sections 3.4 and 3.5 respectively. The noise and jamming parameters may be set independently for a number of different frequency ranges to accommodate frequency hopping. It is important to understand that the parameters are controlled by the frequency-hop words generated in the receiver rather than those accompanying the signal. This is because the noise and jamming depend on the frequency to which the receiver is tuned, and not on the frequency of the signal. This control is shown in Figure 7.19 by the line from the delayed hop words to the frequency table, which contains the information on parameters as a function of frequency.

For a given signal voltage, the noise voltage required to provide a given value of bit-energy-to-noise-power-density ratio (E_b/N_o) depends on the number of samples per data bit and the type of shaping used on the modulation symbols. An equation for calculating the required noise voltage is derived in Appendix A.

A filter and decimator are included in the front end. These are general-purpose devices which can be programmed for the desired response. They are described in Section 8.

A Monitor signal point is provided in the front end just after the de-hopper. Its output is referred to as "average de-hopped sample voltage between updates". This is the mean value of the magnitude averaged over the user-specified interval between updates.

7.6.3 Bandwidth Reducer

The functions of the bandwidth reducer are to remove the direct-sequence spreading, and to provide inputs to the direct-sequence synchronization system. These are shown in Figure 7.20. The de-spreading takes place in the top channel. The input signal is multiplied by the complex conjugate of the delayed direct-sequence reference signal. This reference is produced in the bottom row of components by modulating the binary sequence from a generator, identical to the one in the transmitter, in a PSK modulator and delaying it by an amount intended to equal any delays experienced by the received signal. The user should use the same seed for the reference generator as he used in the generator in the transmitter, either by entering identical seeds, or by accepting the default seed in both places. The de-spread waveform is filtered and decimated to reduce its bandwidth to a value corresponding to the expected bandwidth of the de-spread communications signal. The user must choose a decimation rate equal to the spreading ratio (the number of spreading elements per data symbol). The decimation rate selected provides the information for the set-up of the modulation and sampling of the reference signal.

Although correct de-spreading of the communications signal should reduce its bandwidth to the desired value, it will not reduce the bandwidth of interference or of signal components at different delays. Therefore, it is necessary to precede the decimation with filtering to prevent aliasing. The filter specification is left to the user. He should insure that no significant energy is passed at frequencies beyond half the after-decimation sample rate. If direct-sequence acquisition and tracking are to be used, the filter bandwidth should be made as small as possible without seriously affecting the spectrum of the data symbol, and the filter should have reasonably linear phase. The matched filters in the demodulators assume that the symbol spectrum has not been distorted. As a minimum, all frequencies from minus the data symbol rate to plus the data symbol rate should be passed without significant change. A useful modification to the simulator would be a separation of the specifications

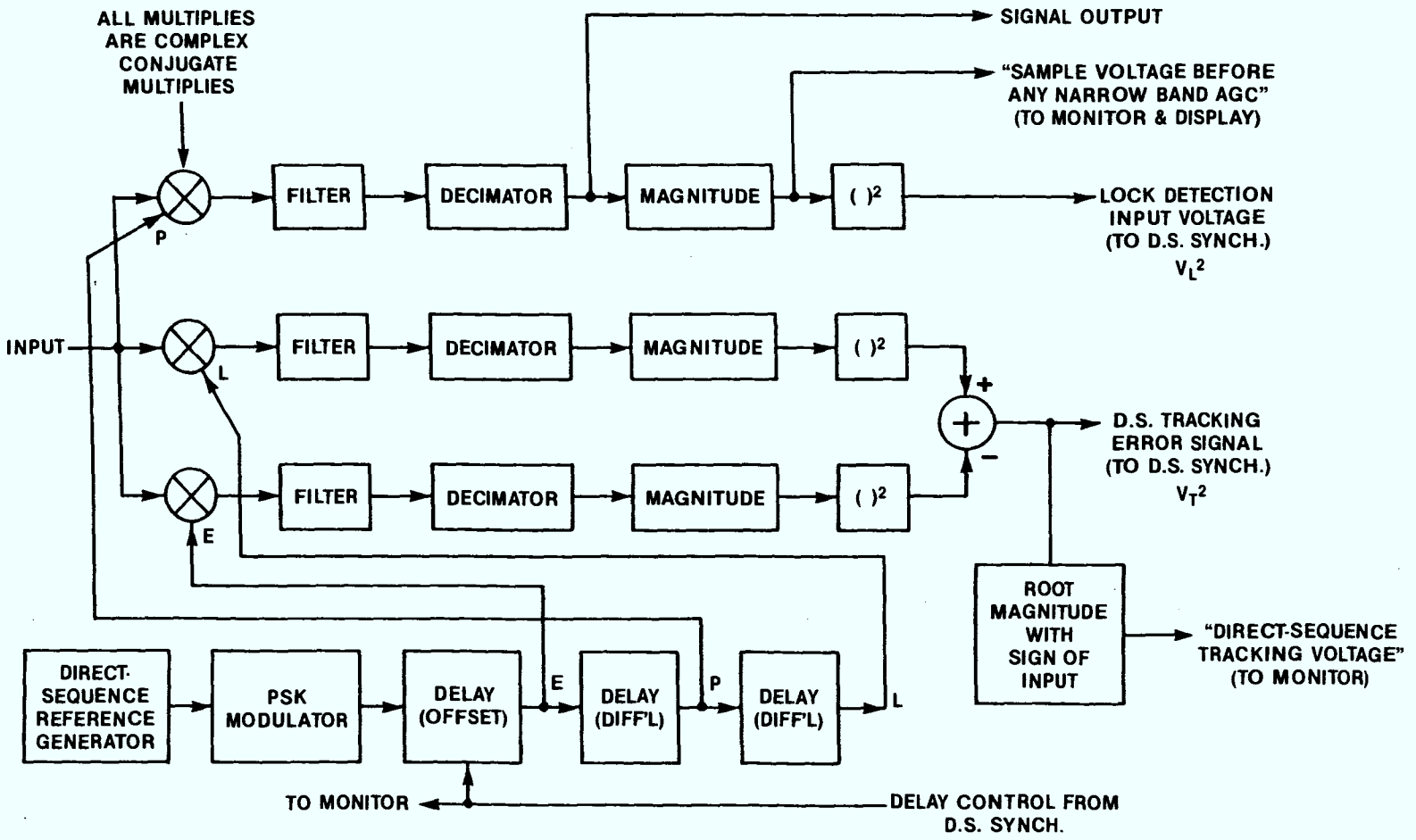


Figure 7.20 Receiver Bandwidth Reducer

for the main signal channel (which should have a wide filter for anti-aliasing) and the other channels used for acquisition and tracking (which should have filters matched to the data symbol spectrum as explained in the section on direct-sequence acquisition and tracking). The use of the filter specification and design routines is covered in Section 8.1.

The magnitude of the de-spread and decimated signal is provided to the Monitor and to the Display. It is referred to in these facilities as "voltage before any narrowband AGC". The square of the magnitude is fed to the direct-sequence synchronization system where it is used to determine if the tracking is in lock. It is also used in one of the two available tracking systems to generate an error signal, as will be discussed later.

Two other channels, identical to the top one, are provided for use in a delay-lock-loop tracking system. These use early and late references in their de-spreading, spaced by a user-specified differential delay on either side of the main-channel delay. The letters P, E, and L in Figure 7.20 refer to present, early, and late respectively. The early and late references remain at fixed delays relative to the offset delay which may be set by the user or controlled by the synchronization system, depending on the receiver mode. The squared magnitudes of the signals from the early and late channels are subtracted to provide an error signal for use by the direct-sequence tracking system. The tracking system is described in Section 7.6.7.4. The square-root of the magnitude of this error signal with the sign set to that of the error signal is provided to the Monitor. This voltage is referred to as "direct sequence tracking voltage".

An alternative to the delay-lock loop tracking is the Tau-dither system. This generates an error signal by alternately advancing and retarding the reference delay in the main channel and providing the resulting squared-magnitude voltages to the synchronization system, which uses them to generate an error voltage. This method has the advantage of requiring only one channel instead of the three of the delay-lock loop method, but has the disadvantages that the error signal is generated at only half the rate, and that the signal voltage to be used in demodulation

will have some loss since the reference is always slightly early or slightly late.

7.6.4 Demodulator

7.6.4.1 General

Demodulation in the simulator is based on correlation of the received signal with the various expected un-degraded symbol waveforms. Integration is over exactly one symbol duration. The reference waveform generating the greatest correlation indicates which symbol has the highest likelihood of being the one sent. One type of demodulator, that for DPSK, does not quite fit the above description since it uses as a reference the delayed input waveform from the previous symbol interval rather than an un-degraded reference waveform.

A special demodulator is available for MCSK signals. We call this a matched-filter demodulator because it computes the correlation at a number of delays, to provide an output waveform as from a filter. The samples of this output waveform are then combined to exploit multipath conditions as a kind of diversity.

Figure 7.21 defines the correlator that is a common component of all the demodulators. Since the signals are complex, the complex conjugate of the reference is taken before the complex multiplication is performed. And since the signals are sampled ones, the integration takes the form of a summation over the N_S samples contained in each symbol. The integration is always coherent in the sense that amplitude and phase are used in the summation. In certain cases the demodulation is referred to as non-coherent because the phase of the resulting correlation is ignored in determining which symbol is most likely. This may be necessary where the propagation medium causes rapid phase changes in the received signal and these cannot be tracked. It is important to understand, however, that the

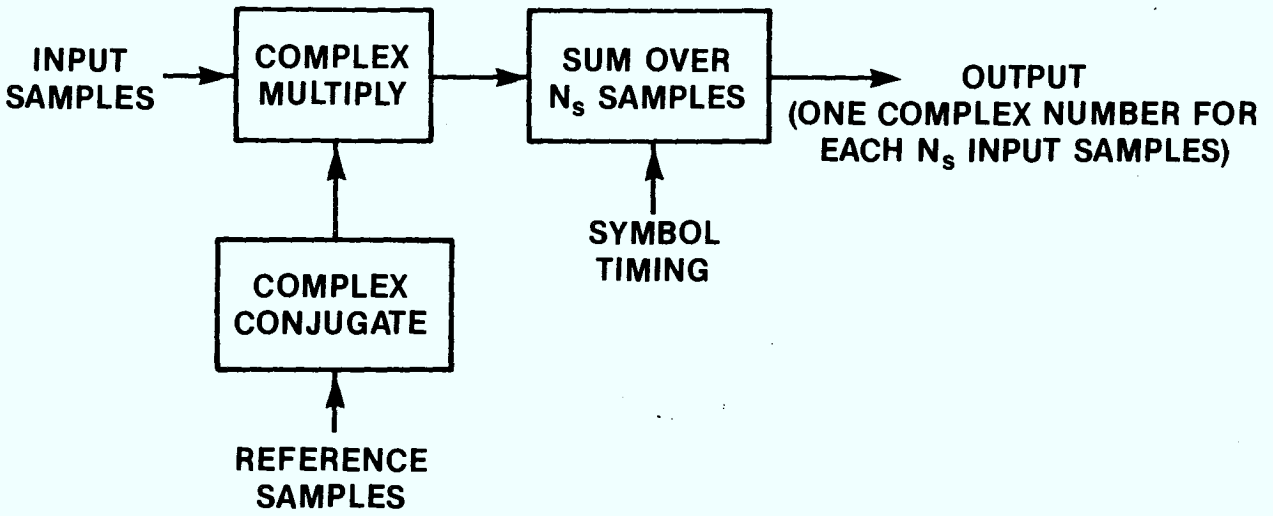


Figure 7.21 Correlator for Demodulator

phase must be stable over a symbol interval for the demodulator to perform satisfactorily, whether it is coherent or non-coherent.

7.6.4.2 FSK Demodulator

The FSK demodulator will be discussed first because it is the most general, that is, it does not have any simplifications made possible by the nature of the modulation as some others do. Thus, it best illustrates the general principles of the demodulation scheme used in the simulator. The operations performed by the FSK demodulator are indicated in Figure 7.22. If there are N_b bits encoded in each symbol, the integer symbols will be 0, 1, 2, ..., $2^{N_b}-1$. The demodulator must test the input against the waveforms corresponding to each of these symbols. In the lower left of the figure the counter holds an integer representing the symbol to be tested . It is first reset to zero (the first symbol) and the modulator generates the corresponding FSK waveform which is then correlated with the incoming samples (S_1 is in position 1). Since the correlator integrates exactly N_s samples, it must be told when to start. The start is synchronized with the change of symbol in the counter, which is controlled by a signal from one of the synchronization systems. Synchronization is discussed in Section 7.6.7; in the present description correct synchronization will be assumed. The output of the correlator after N_s samples is a single complex number. In the FSK demodulator non-coherent demodulation is performed; therefore, only the magnitude of this complex number is passed to the first register. After N_s samples, the counter is incremented and the process is repeated for reference symbol 1. But since this reference symbol must be correlated with the same input symbol, the input symbol is recirculated with S_1 in position 2 (there is no loss of input data in this process since the simulation is not in real time and the input data simply waits while the remainder of the demodulation of the symbol proceeds). The result of the second correlation is stored in register 2, and the process is repeated until all 2^{N_b} symbols have been tested. The decision on the output symbol is simply one of determining which register contains the largest value, and choosing the symbol that was used

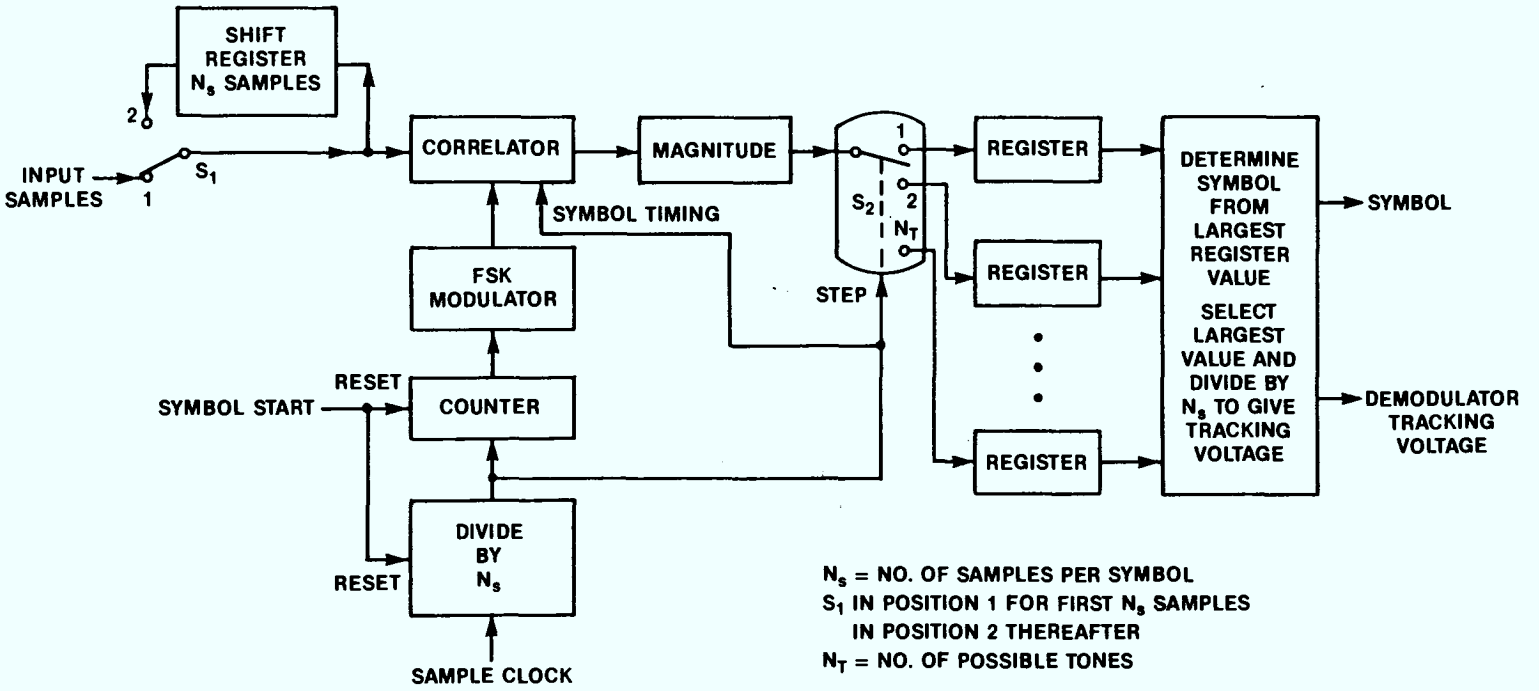


Figure 7.22 FSK Demodulator

as a reference in producing that value. The largest register value, divided by the number of samples per symbol, N_s, is output as the "demodulator tracking voltage" for the Monitor and Display.

7.6.4.3 MFSK Demodulator

The MFSK demodulator is identical to the FSK one except that the symbol is divided into sub-symbols each of which is represented by the transmission of one tone out of a possible 2^{N_a} tones, where N_a is the number of bits encoded in each sub-symbol. The number of sub-symbol tones to be tested becomes the number of sub-symbols per symbol times 2^{N_a}. After correlation, the register values are grouped into groups of 2^{N_a}, and the sub-symbols are determined from the largest value in each group; then the symbol is formed from the sub-symbols. To generate the "demodulator tracking voltage" the largest register value in each group is selected, and the mean of these is computed and divided by the number of samples per symbol, N_s.

7.6.4.4 MSK Demodulator

Coherent MSK demodulation has not been implemented at this time. If MSK demodulation is selected the noncoherent FSK demodulator is used and the tone spacing is automatically set to the correct MSK value according to the symbol length.

7.6.4.5 PSK Demodulator

The PSK demodulator is simpler than the FSK one because the PSK waveforms differ from each other only in their phase. The correlation between two different symbol waveforms will have unit magnitude and a phase that is equal to the difference between their phases. Thus, it is only necessary to correlate the input signal with one reference symbol

waveform. The decision on the received symbol is based on the phase of the correlation value. The process is shown in Figure 7.23. Symbol 0 is used to form the modulated symbol waveform, and the phase of the correlation is examined to determine the symbol. The symbol whose phase is closest to this value is taken as the received symbol. As an example, consider the case of a PSK modulation with two bits per symbol. The integer symbols will be 0, 1, 2, and 3, and the phases of the corresponding modulated symbols will be 0, $\pi/2$, π, and $-\pi/2$. Then if the phase after correlation is between $-\pi/4$ and $\pi/4$ the received symbol is taken as 0; if it is between $\pi/4$ and $3\pi/4$, it is taken as 1; if it is between $3\pi/4$ and $-3\pi/4$ it is taken as 2; and if it is between $-3\pi/4$ and $-\pi/4$ it is taken as 3. The magnitude of the correlation, divided by the number of samples per symbol, is taken as the "demodulator tracking voltage".

7.6.4.6 DPSK Demodulator

The DPSK demodulator, shown in Figure 7.24, is similar to the PSK demodulator but it uses as a reference signal the previous input symbol, obtained by delaying the input by one symbol duration. The integration for the correlation is performed before the multiplication in this case in order to minimize the noise in the reference. The integration over a symbol may be considered a matched-filtering operation on the symbol, and this results in only a single pair of samples to be multiplied. If the integration were performed after the multiplication, additional noise would be generated by the product of noise components outside the signal bandwidth. The uniform integration used is equivalent to a matched filter for a rectangular symbol. If shaping has been used on the modulated symbols this filter will not be matched to the symbol and some loss will occur. In this case the samples integrated should be weighted in accordance with the symbol shaping function, but such a scheme has not been implemented in the simulator at this time. The same effect can be acheived if the demodulator is preceded by a filter that, when combined with the integration filter (which has a sin x/x response), will give an overall

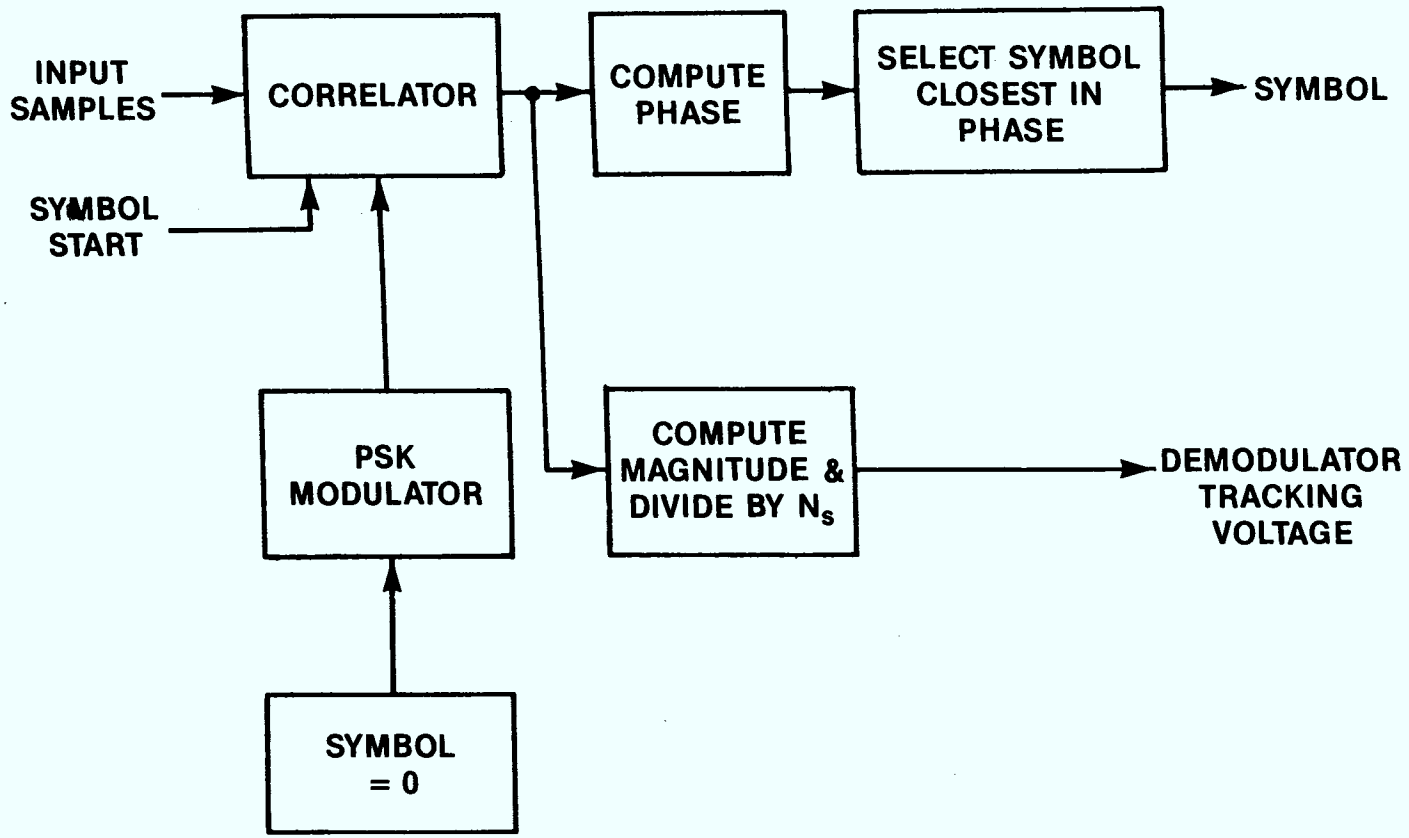


Figure 7.23 FSK Demodulator

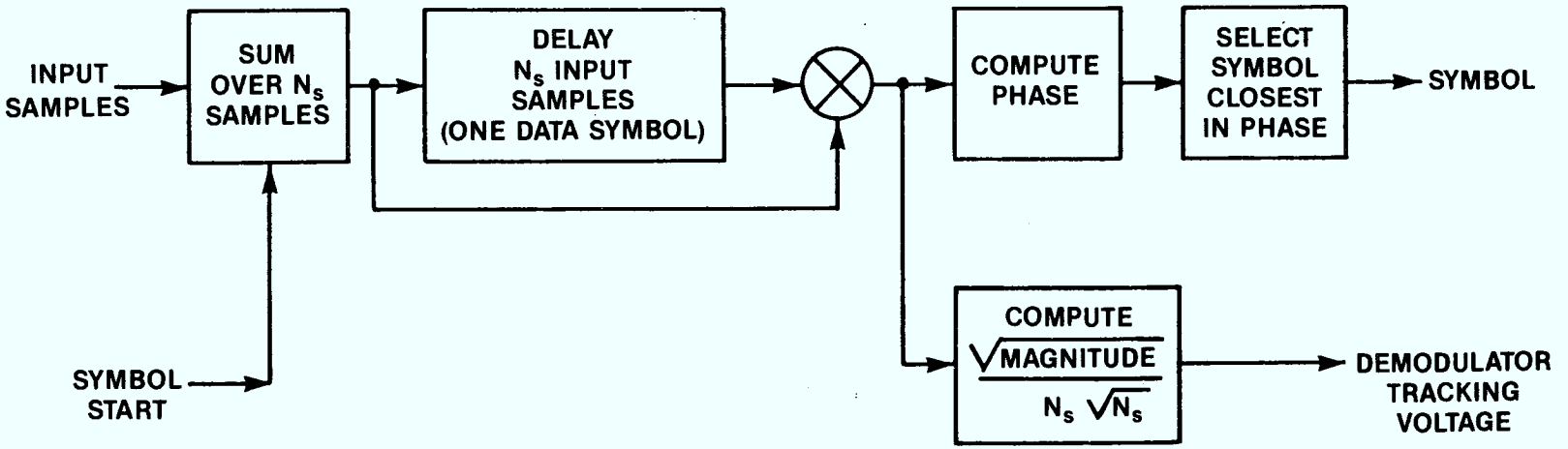


Figure 7.24 DPSK Demodulator

response matched to the shaping function spectrum. The filters available in the front end and in the bandwidth reducer can be used for this purpose.

Since in DPSK the data symbol sent differs from the previous one by the symbol intended (the symbol before differential encoding), the correlation phase, which is the phase difference between the input and the reference symbols, indicates the intended symbol, and the symbol determination is the same as for PSK. The square root of the magnitude of the correlation, divided by the number of samples per symbol, is used as the "demodulator tracking voltage" in this case, since both inputs to the correlation product are received signals.

An important difference between the PSK and DPSK demodulators is that in the latter the reference has suffered degradation by noise and interference. This results in poorer performance than for coherent PSK. The bit-error-rate curves [10] show a loss of about 1 dB relative to PSK for error rates around 10^{-4}.

7.6.4.7 MCSK Demodulator

The MCSK demodulator, shown in Figure 7.25, is similar to the FSK demodulator. Instead of an FSK modulator, it uses a PSK modulator driven by an MCSK code generator. The user must specify PSK when asked the modulation type before he is asked about the MCSK parameters. The correlation is performed on the entire MCSK waveform comprising a number of PSK modulation symbols. The input shift register and the divider use the number of samples per MCSK symbol, N_C, rather than the number of samples per modulation symbol, for both length and divisor. Each of the MCSK waveforms is generated in turn as the correlator reference and the largest correlation value is used to determine the received symbol. The largest value, divided by the number of samples per symbol, is also output as the "demodulator tracking voltage". A special version of the MCSK demodulator is available as an alternative. This is referred to as the "MCSK matched

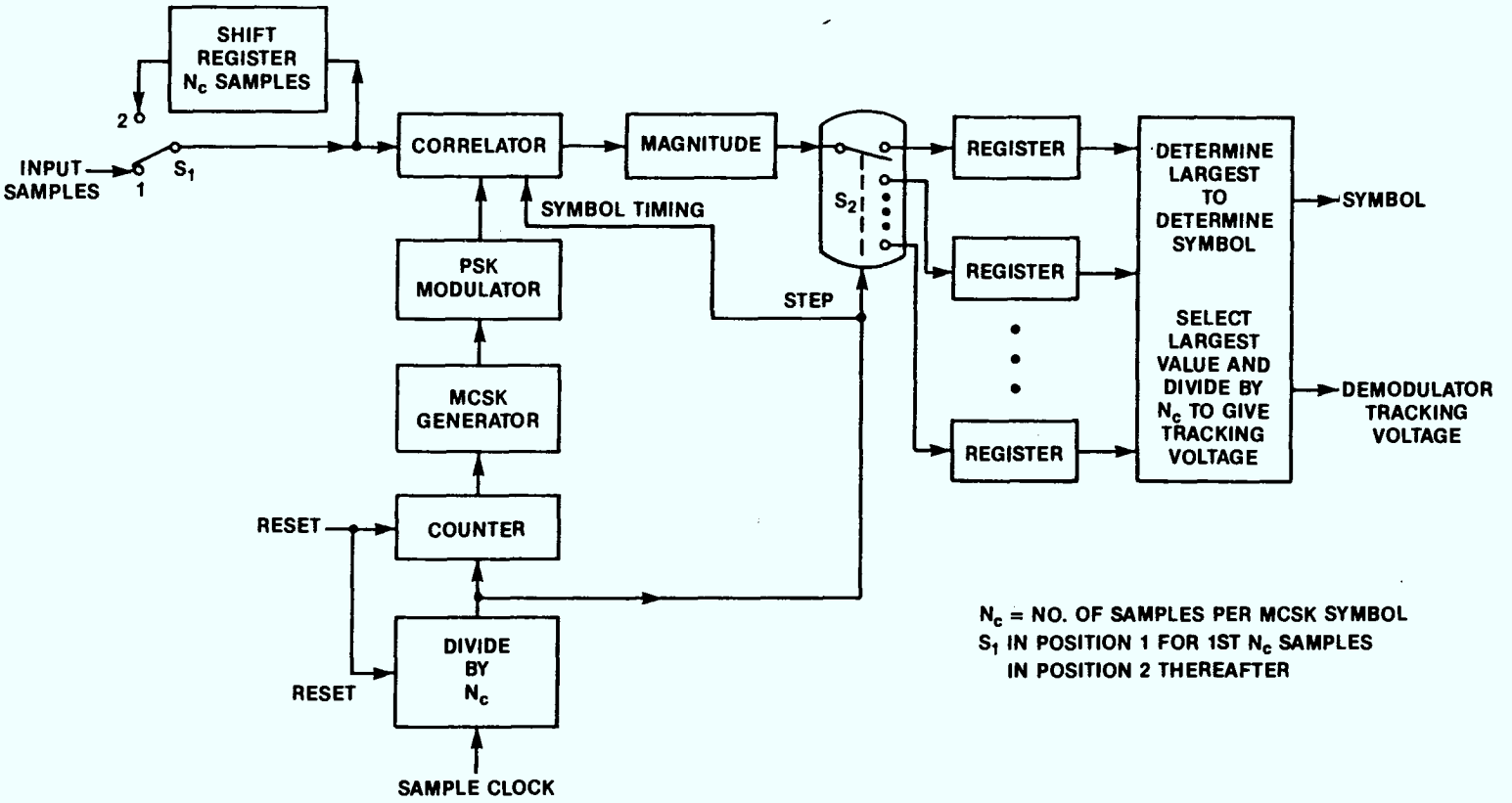


Figure 7.25 MCSK Demodulator

filter demodulator", and will be described in Section 7.6.4.9.

7.6.4.8 FEK Demodulator

Frequency-Exchange Keying (FEK) is a special form of demodulation for FSK signals that permits improved performance when selective fading exists. It treats a binary FSK signal as a pair of on-off keyed signals and combines the results as diversity signals. An estimate of signal and noise in each of the two channels is used to adjust the combining ratio.

The FEK demodulator is identical to the MFSK demodulator of Figure 7.22 to the left of switch S_2. The part to the right is replaced by the process shown in Figure 7.26. The algorithm performed by the assessors of Figure 7.26 is shown in Figure 7.27 in flow-diagram form. Multiple tones can be accommodated as a number of binary sub-symbols. Each of the binary sub-symbols is demodulated by a pair of assessors, one for the mark and one for the space (marks and spaces are used here to refer to the two symbols of the binary alphabet, instead of ones and zeros). These assessors are identical except for the sign of the output. A brief description of the algorithm of Figure 7.27 follows.

An estimate of the signal-plus-noise level and of the noise level is updated for each new input sample. These estimates are Max and Min and are based on the assumption that when no signal is present in that channel Min will be updated, while when signal is present Max will be updated. Max and Min are made to decay toward Mean, the mean value of Max and Min, with a user-specified time constant, T_2. When a new sample is above Max or below Min, Max or Min is updated by adding a fraction of the difference between the sample and Max or Min. This fraction is a function of a user-specified attack time-constant T_1. The output for each new input sample is taken as the difference between the input sample and Mean if the assessor is in the mark channel and the negative of this if it is in the space channel.

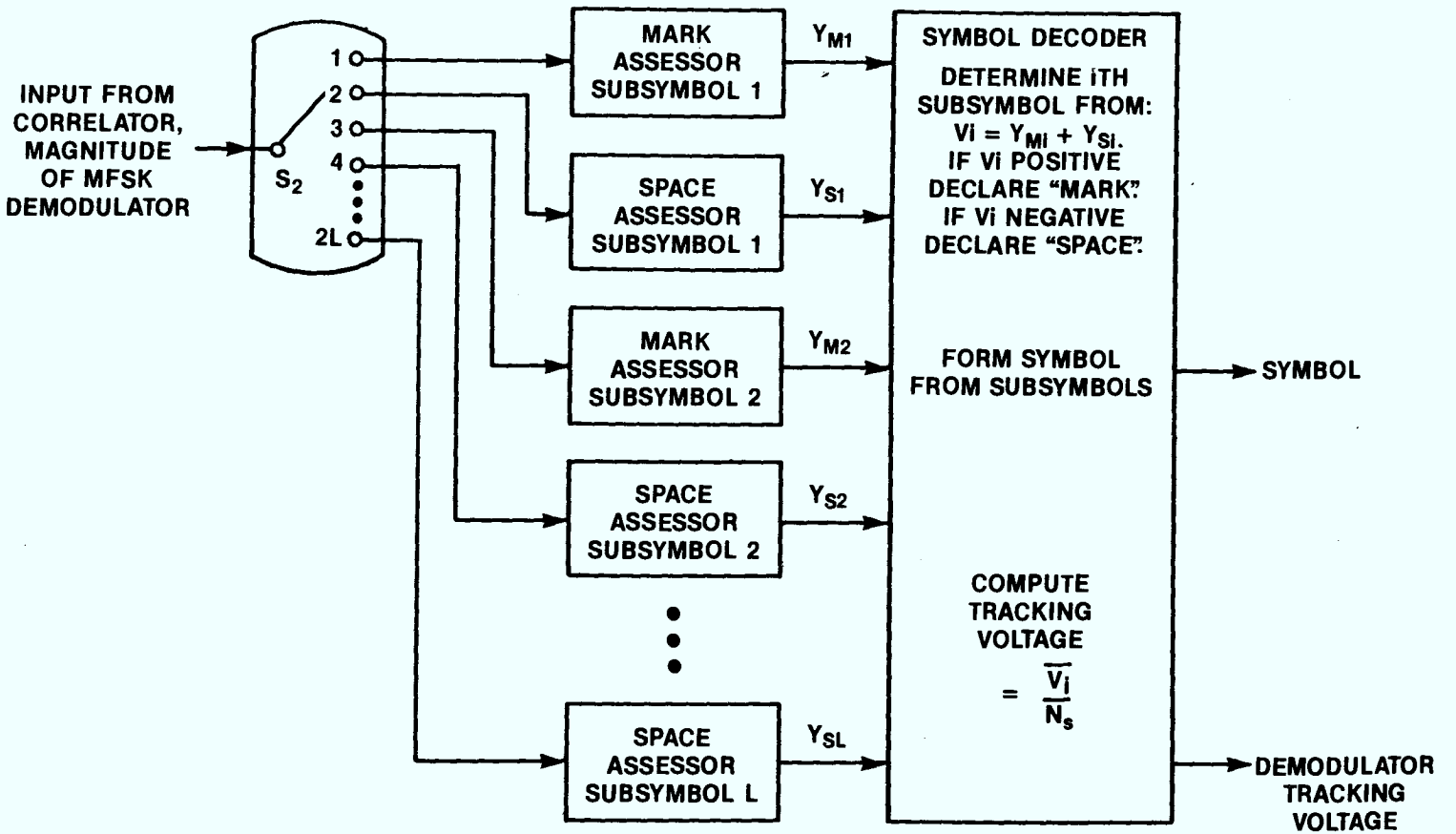
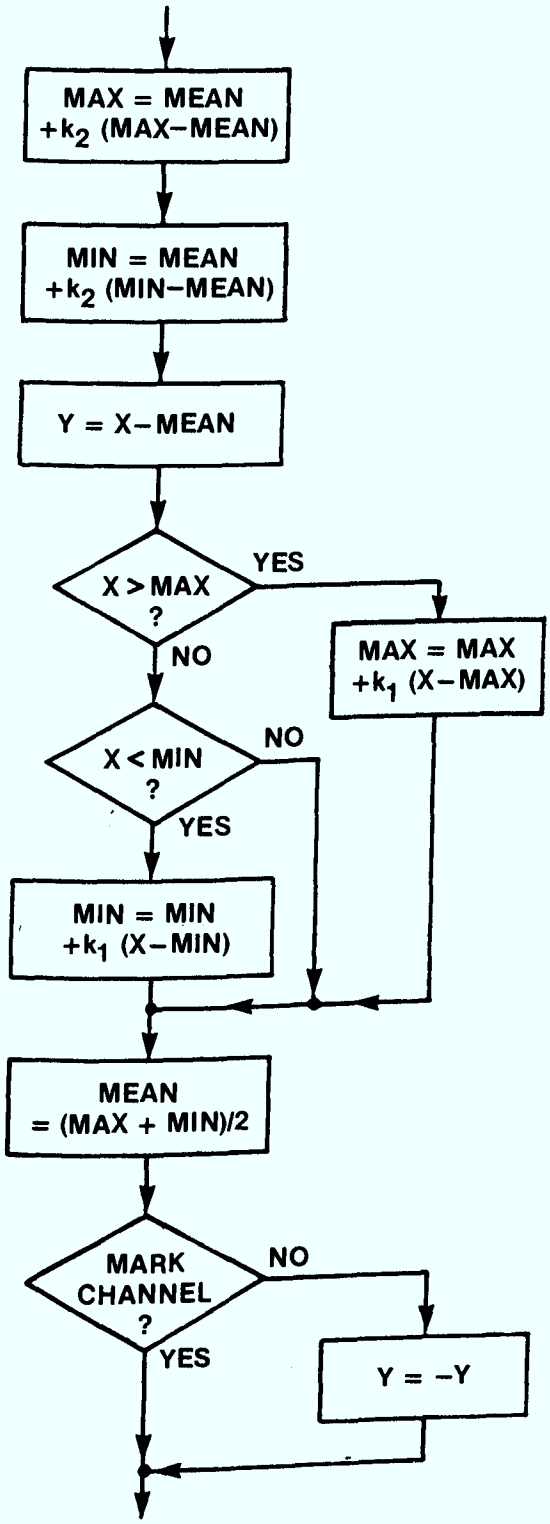


Figure 7.26 Changes to MFSK Demodulator for FEK



X = INPUT SAMPLE
Y = OUTPUT SAMPLE

MAX, MIN, AND MEAN SET TO ZERO AT BEGINNING OF A RUN

k_1 = ATTACK TIME-CONSTANT COEFFICIENT
$= 1-EXP(-1/T_1)$
WHERE T1 = ATTACK TIME CONSTANT

k_2 = DECAY TIME-CONSTANT COEFFICIENT
$= EXP(-1/T_2)$
WHERE T2 = DECAY TIME CONSTANT

T_1 AND T_2 SPECIFIED IN NUMBER OF SYMBOLS (MAY BE NON-INTEGER)
LIMITS: $1 \leq T_2 \leq 4000$
$0 < T_1 \leq T_2$

ONE PASS IS MADE THROUGH THIS DIAGRAM FOR EACH NEW INPUT SAMPLE

THERE ARE SEPARATE ASSESSORS FOR MARK AND SPACE

Figure 7.27 Flow Diagram of FEK Assessor Algorithm

In the symbol decoder the outputs of the mark and space assessors are added and the decision for mark or space (one or zero) is based on the sign of the result, with positive indicating mark. If there are multiple tones, the sub-symbols from each pair of assessors are then used to form the decoded symbol by combining them as bits of a multi-bit number and outputting the symbol corresponding to that number. The "demodulator tracking voltage" is computed from the mean of the sums of the outputs of all the mark and space assessor pairs.

The user must specify the attack and decay time-constants T_1 and T_2. T_2 should be large enough to hold the estimate over any long periods between marks or spaces. On the other hand, if it is too long it will not be able to follow the fades. Therefore, it should correspond roughly to the shortest fade periods expected. T_1 should be low to allow the assessor to follow the fading changes even when either marks or spaces are infrequent (Max can be updated in the mark channel only when marks are present and Min can be updated only when they are not). This means it should be a small fraction of the shortest expected fade period. However, if T_1 is too low, noise or interference spikes will degrade the estimate.

7.6.4.9 MCSK Matched-Filter Demodulator

The matched-filter demodulator performs time-domain correlation by means of frequency-domain processing. This results in the computation of the entire cyclic correlation function for the number of input samples the user specifies, rather than at just the single delay used in the other demodulators. The output correlation function in the time domain is windowed to select only a portion of it to be used for symbol determination and delay tracking.

The process is depicted in Figure 7.28. In this figure broad lines are used to represent block transfers and narrow lines are used to represent serial ones. In the simulator, all processes in the receiver are in reality serial; that is, for each new sample each of the operations in

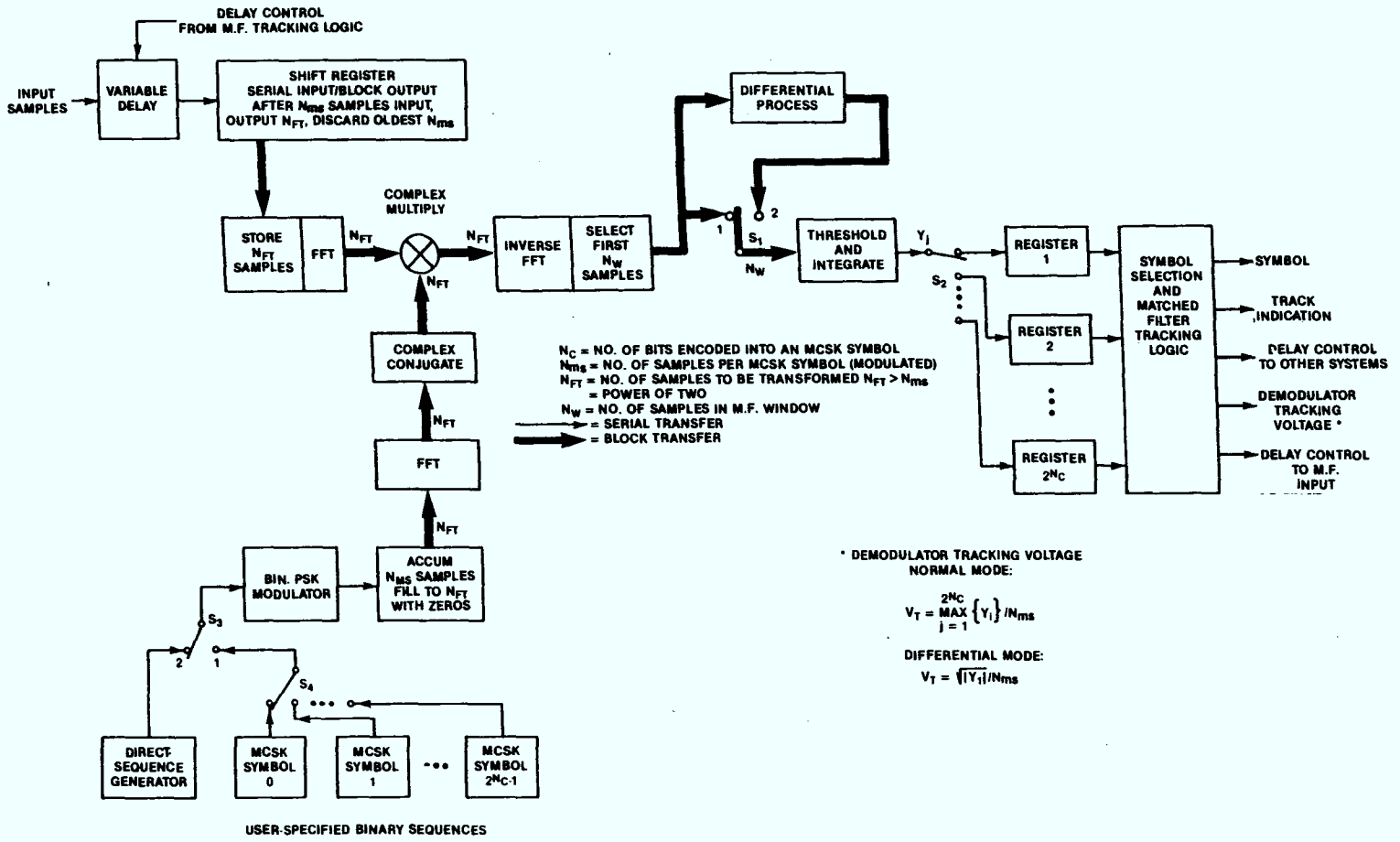


Figure 7.28 MCSK Matched-Filter Demodulator

the receiver is updated in turn, but some of these operations consist of accumulating samples until a certain number have been obtained and then performing a block process on them before passing the resulting samples, one at a time, to the next operation. Functionally, however, this processing may be considered to be done in blocks, with a parallel transfer of the output to the next operation.

Correlation is performed by converting both the input signal and the MCSK symbol reference to the frequency domain by means of a fast-Fourier-transform routine, and multiplying sample by sample. An inverse transform converts the result back to the time domain. The number of samples transformed, N_{FT}, which must be an integer power of two, is specified by the user. The resulting N_{FT} complex values, after return to the time domain, represent the cross-correlation function at sample points separated in time delay by the time between input samples. Only the first N_w (specified by the user) of these correlation samples are retained in the subsequent windowing operation. The correlation is performed once per MCSK symbol, but the block of samples used in each transform, N_{FT}, must be larger than the number of samples per MCSK symbol, N_{ms} (N_{ms} is equal to the number of samples per modulation symbol times the number of elements - modulation symbols - in each MCSK code). The shift register at the upper left of Figure 7.28 forms the input samples into blocks for the transform. This process is clarified in Figure 7.29, by an example in which $N_{ms} = 5$ and $N_{FT} = 8$ (these numbers are much too low to be realistic, but it is easier to illustrate the process with small numbers). The upper boxes represent the shift register in some arbitrary nth iteration. Samples S_i to S_{i+7} are in the register (note that the samples advance to the right in the register, and, therefore, time increases to the left). These 8 samples are the next to be transformed. After 5 new samples have entered the register the situation is as shown in the lower boxes. S_{i+5} to S_{i+7} are still in the register and take part again in the next transform. Thus the correlation functions for succeeding MCSK symbols involve overlapping input blocks, each larger than the MCSK symbol, and with the start of each block advancing by exactly one MCSK symbol duration.

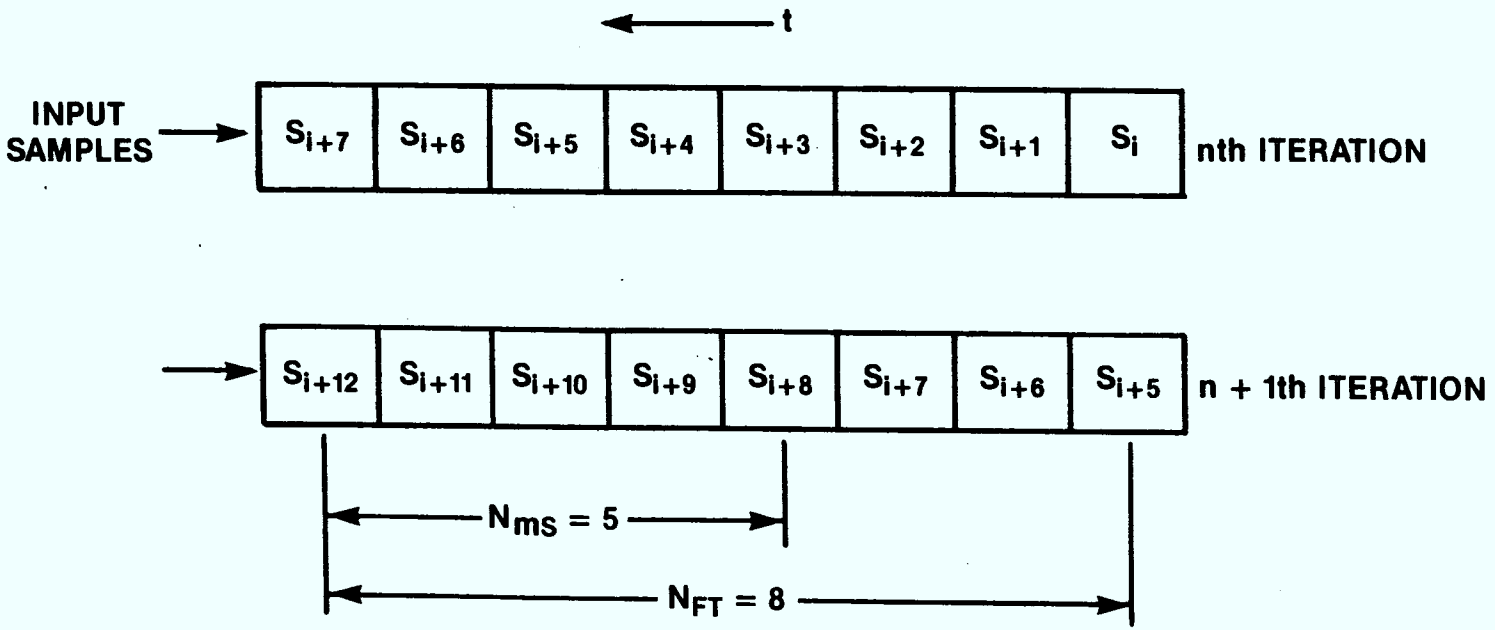


Figure 7.29 Data Blocking for FFT

Returning to Figure 7.28, we see that the reference waveform is an MCSK symbol and will therefore have a length of N_{ms} samples. Zeros are added to the end of this to fill it to the transform length of N_{FT}. This, along with the windowing of the correlation output, prevents problems caused by the cyclic nature of the FFT. There are two modes of MCSK symbol generation, repeating and non-repeating. In the repeating mode S_3 is in position 1 and the user-entered codes stored in the MCSK symbol 0 to MCSK symbol $2^{N_c}-1$ are switched in turn by S_4 to the modulator. Each reference waveform from the modulator is multiplied in turn by the same input block stored in the box preceding the FFT to produce a correlation function. After windowing, it is processed by the threshold-and-integrate box and stored in one of the registers (S_2 changes in unison with S_4). When all of the MCSK reference symbols have been tried, the values in the registers are used to determine the symbol and to update the tracking if it is used. Then, the next N_{sm} input samples are entered into the shift register and the process repeated for the next symbol.

If differential demodulation has been chosen, S_1 will be in position 2, and the differential-process box is inserted. This process, shown in Figure 7.30, multiplies the correlation function of the present symbol by the complex conjugate of that of the previous symbol to extract the phase difference from which the intended symbol is derived in the subsequent boxes of Figure 7.28. When differential demodulation is used, only binary coding is allowed (one MCSK symbol and the symbol reversed in phase). Then, S_2 and S_4 are fixed and only one register is used.

If non-repeating codes are selected the binary differential mode must be used. In this case S_3 of Figure 7.28 is in position 2 and the direct-sequence generator is used as the symbol generator. Successive segments of the binary sequence, of length equal to that specified for the MCSK symbol are used to generate the reference. The same seed should be used for the sequence generator for both the transmitter and the receiver.

Figure 7.31 describes the algorithm used in the threshold-and-integrate box. When normal (not differential) demodulation is used, the

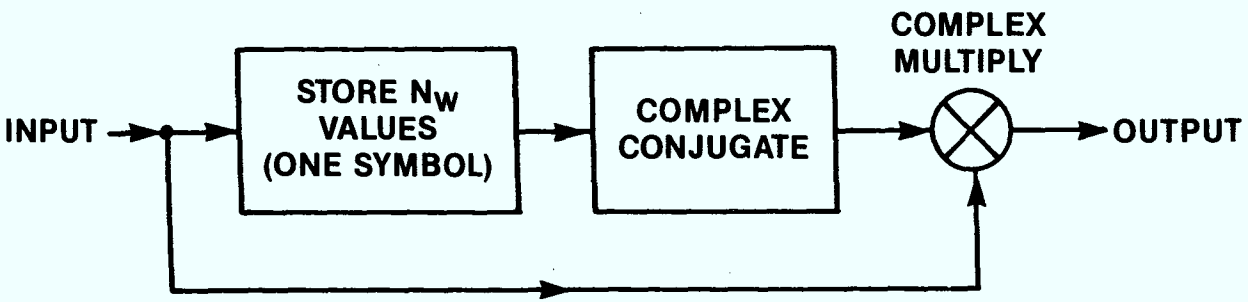


Figure 7.30 Differential Process in MCSK Matched Filter

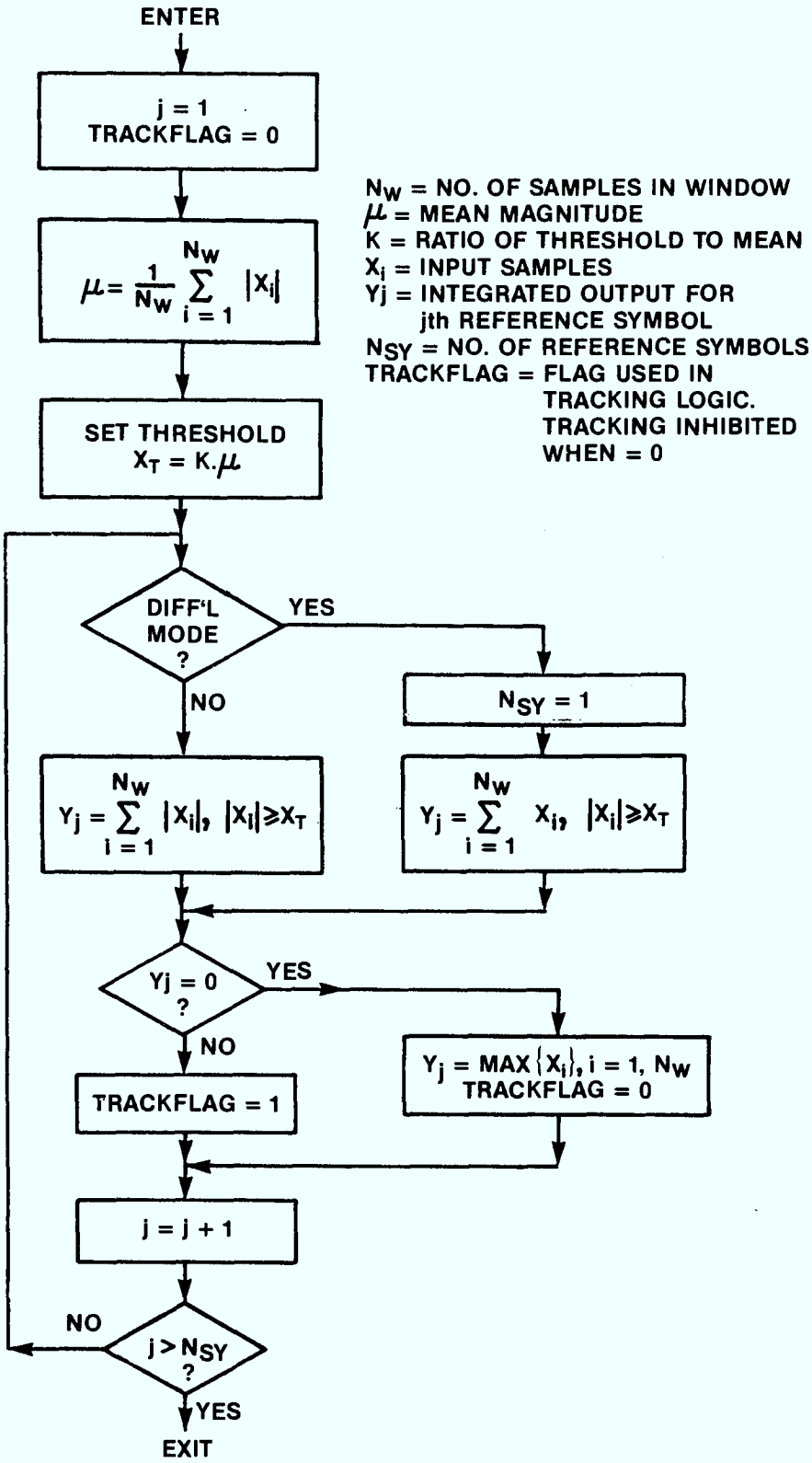


Figure 7.31 Threshold-and-Integrate Algorithm

magnitude of the windowed correlation function from each reference symbol is integrated to give a magnitude for that symbol. A threshold based on an estimate of the noise is used to remove from the integration those samples for which the signal-to-noise ratio is estimated to be small. If none of the samples exceeds the threshold the largest sample is taken as the output, and if this occurs for all reference symbols a flag is set to inhibit delay tracking (performed by the symbol-tracking logic to be described later). When differential demodulation is selected only one reference symbol is used in the correlation and the important information is in the phase of the integrated value. Consequently, the integration is performed on the full complex values rather than on the magnitudes. The result is a complex value used by the symbol-selection logic.

The threshold is based on the mean magnitude, μ, of the correlation of the input values with the first reference symbol. The mean is intended to be an estimate of the noise and interference level and it is multiplied by a user-specified multiplier, k, to determine a threshold that has a low probability of being exceeded by noise alone, but a high probability of being exceeded by the desired signal plus noise. If the probability distribution of the noise magnitude is known, e.g. Rayleigh, then the value of k for a given probability of exceeding the threshold may easily be computed. If the signal is present in a significant percentage of the delay elements in the window, it may affect the estimate of the noise, and should be taken into account. From experience, it has been found that a value of k between 1.5 and 3 is usually satisfactory. For simplicity, the mean is computed for only one reference symbol. The correlation outputs for all reference symbols have the same noise power (since the same noise is correlated) but the signal is autocorrelated in only one of the correlations. Some improvement would be possible if the simulator were changed to compute the mean over all reference symbols, or to average over a number of input symbols.

The purpose of the threshold is to remove from the integration those samples for which the signal-to-noise ratio is low. The best performance would be obtained if the samples integrated were weighted in proportion to

the estimated signal strength in each. For large signal-to-noise ratio the signal-plus-noise level is a reasonable estimate of the signal level, but when the signal voltage is comparable to the noise level it is not. Therefore the threshold may be used to remove the low signal-to-noise samples and thus give them zero weight. When the threshold is exceeded the signal is, in effect, given the proper weight since, as a result of the differential demodulation, the signal voltage after the differential process is proportional to the square of the input signal voltage.

In the normal mode of the matched-filter demodulator, the symbol selection logic outputs the symbol that is the same as the reference symbol that was used in the correlation that produced the largest integrated magnitude. In the differential mode the output is zero if the complex integrated value is in the right-half complex plane and is one if it is in the left-half plane.

In the normal mode the "demodulator tracking voltage" for the Display and Monitor facilities is computed by dividing the largest integrated voltage from the correlations by the number of samples per MCSK symbol. In the differential mode the square root of the magnitude of the integrated voltage is taken and the same division is performed.

The matched-filter tracking logic, when selected, attempts to keep the "centre of gravity" of the correlation sample magnitudes corresponding to the selected symbol in the centre of the matched-filter window. The delay correction computed by the tracking logic, after each symbol, is applied to the variable delay at the input to the matched filter. The "centre of gravity" relative to the centre of the window is computed as:

$$D_{cg} = \sum_{i=1}^{N_w} [i-(N_w+1)/2] \left|X_i\right| / \sum_{i=1}^{N_w} \left|X_i\right|, \qquad \left|x_i\right| \geq k\,\mu$$

where x_i is the ith sample in the correlation window, and the other values are as defined above. It can be seen from this definition that when the window size is even, the centre is taken as the sample just early of the centre. Since the "centre of gravity" is measured from the centre of the window, it is a measure of the tracking error. Some smoothing is applied by multiplying this error by the inverse of a user-specified tracking time-constant, before it is used to correct the delay. If the track-inhibit flag is set, indicating that none of the correlation samples exceeded the threshold in the threshold-and-integrate box, this computation is inhibited, and zero tracking error is generated. The tracking logic also outputs a track-lock indication which is available to the Monitor and Display. It is also used to control changes in the synchronization systems (e.g. return to acquisition when track lock is lost). Whenever the track-inhibit flag is not set, the lock indication is "lock". If tracking is inhibited for a user-specified number of consecutive symbols the indication changes to "unlock". The action taken when this happens depends on the synchronization mode. These modes will be described in detail in Section 7.6.7 on synchronization. In one mode the matched-filter tracking error is also used to control the reference delay in the frequency de-hopper.

7.6.5 Automatic Gain Control (AGC)

AGC is available in two places in the receiver as indicated in Figure 7.18. They are referred to as wideband and narrowband AGC, but these terms apply to their place in the receiver; the same routine is used in both places with the parameters specified by the user for each independently. Either, both, or neither of the two AGCs may be used in a simulation run. The user specifies the threshold below which the gain is fixed, the small-signal gain below this threshold, and the attack and decay time-constants.

The algorithm is described in Figure 7.32. Consider first the block diagram on the left. The magnitude of the input samples is taken, and

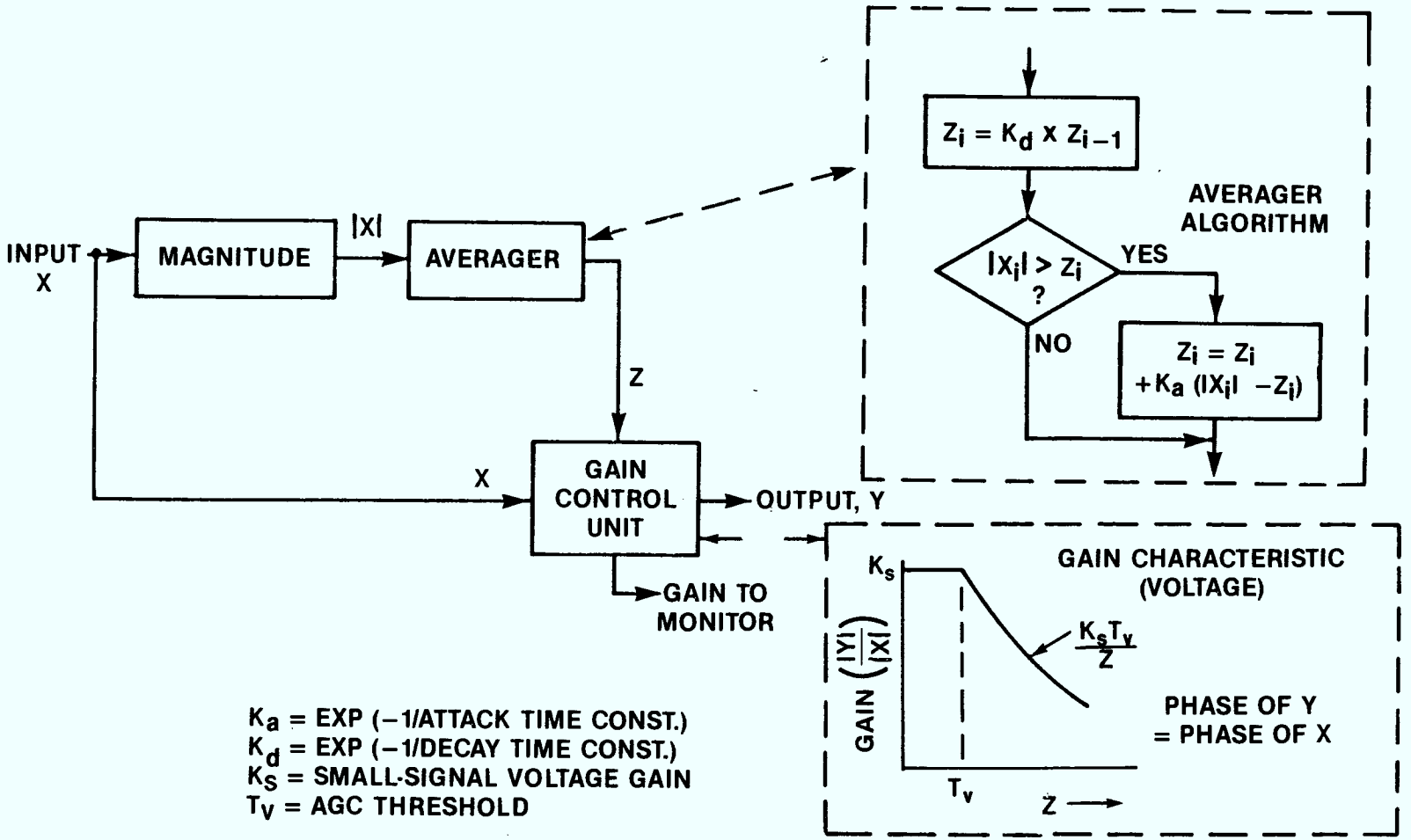


Figure 7.32 Automatic Gain Control

these values are averaged to provide a voltage that controls the gain in the signal channel. Details of the averager algorithm and of the gain control characteristics are provided in the dashed boxes. The first box in the averager algorithm provides a decay of the control signal by multiplying the previous value by a constant, K_d, which is related to the user-specified decay time-constant by the equation shown in the notes at the lower left of Figure 7.32. Next the input sample magnitude is tested to see if it is above the control signal. If it is not, no further action is taken for that sample. If it is, the control voltage is increased by the difference between the input and the control voltage, multiplied by a constant, K_a, which is related to the user-specified attack time-constant as shown in the notes. All time constants are specified in samples.

The gain characteristic is shown in the lower dashed box of Figure 7.32. The gain is constant up to a user-specified value of the control voltage, T_V, and above this value the gain is reduced at the rate shown, to provide a constant output level of $K_S.T_V$ for any fixed input signal level (after settling).

7.6.6 Adaptive Excision Filter

The function of the adaptive excision filter is to remove narrowband interference in a direct-sequence spread-spectrum system before the de-spreading operation. While the de-spreading will provide good discrimination against narrowband intereference, even further improvement can be obtained by exploiting the narrowband characteristic. The adaptive excision filter attempts to whiten the signal-plus-interference spectrum, and, since the direct-sequence signal itself is essentially white, the effect is to attenuate the interference, which has an impulse-like spectrum, while leaving the direct-sequence signal nearly unchanged. The excision filter takes the form of a complex FIR filter. The coefficients are computed from an algorithm operating on the most recent part of the input signal, and are updated at regular intervals.

The algorithm used to compute the coefficients is based on the Wiener method of linear prediction. The idea is to predict the interference signal (a narrowband signal is amenable to prediction because of its narrow bandwidth) and subtract the predicted value from the input signal. The direct-sequence signal, having large bandwidth, is not predictable and so is not much affected, even though it is present in the composite signal from which the predictions are made. Prediction is based on an estimate of the autocorrelation function of the signal using the most recent samples, and depends on the solution of a matrix equation, but as a result of some special properties of the autocorrelation matrix involved, a simplified iterative solution known as the Levinson-Durbin algorithm is possible, and is used here. Details of the method, and references can be found in Appendix B. The results of tests, and a discussion of the operation are also provided there, and it is hoped that these will provide some insight, and aid in the selection of parameters.

When the excision filter is selected the user must specify the number of coefficients in the filter (this is the number of computed coefficients; the actual number is one more than this since the first coefficient is fixed at unity), the spacing of the coefficients, the number of samples used in the autocorrelation function estimate, the desired signal delay at the filter input and the interval between updates of the coefficients. The maximum number of coefficients is 128, and their maximum spacing is 32 samples.

The coefficient spacing applies to the spacing of samples used in the autocorrelation estimate as well as to the taps of the FIR filter. A coefficient spacing of more than one sample is required when the sample rate is greater than one sample per element of the direct-sequence signal in order to remove any dependency between the samples. Such dependency would provide some predictability to the direct-sequence signal and result in degradation of the filtered signal. Coefficient spacings greater than one sample will cause the excision filter to have a frequency response that is periodic with a period of 1/N times the sample rate, where N is the coefficient spacing in samples. This may cause some degration under certain

conditions. This problem is discussed in some detail in Appendix B.

The number of samples used in the correlation estimate must be greater than the number of coefficients, but not greater than 4097. The signal delay is intended to allow the signal to enter the filter after that same signal has been used to compute the coefficients. This would require a delay equal to the number of samples in the correlation estimate (computation time does not have to be included in the simulator since it does not work in real time). This is the maximum delay allowed. The interval between updates must be at least as large as the number of samples in the correlation estimate.

7.6.7 Synchronization

7.6.7.1 Introduction

Receiver systems generally require synchronization in both time and carrier frequency or phase. Time synchronization is included in the simulator but frequency or phase synchronization is not included at this time. It was felt that phase synchronization was not too important since our main interest is in non-coherent and differentially coherent systems, which do not require phase synchronization. At HF the likelihood of medium-induced Doppler shifts large enough to cause significant frequency errors was considered negligible, and the frequencies generated in the simulation are precise and do not drift. Coherent PSK modulation is provided in the simulator as a reference, and when used it requires that the user determine the phase of the received signal, either by calculation or by test, and set the phase reference accordingly. As long as no large phase variations are produced by the medium (including unknown mean Doppler, even if small) this should permit satisfactory operation. Other modulations require only that the frequency error be small. Any short-term phase variations that would degrade these signals could not be compensated for by a phase synchronization system in any case. On the other hand

frequency or phase synchronization systems could be useful when airborne manoeuvering platforms, drifting frequency generators, or coherent demodulators are to be considered, and such systems may be worth adding in the future.

If propagation delays were fixed, time synchronization could be accomplished by user-specified delays. However, some of the propagations conditions of interest at HF involve delay variations. In addition, it was felt that time synchronization was of such importance in spread-spectrum systems that it was well worth including these synchronization systems to allow experiments in this area. Consequently, the simulator was given a significant time-synchronization capability.

Reference generators for frequency de-hopping and direct-sequence de-spreading must be brought into time synchronization with the sequences in the received signal by synchronization acquisition systems, and, under varying propagation conditions or with drifting clocks, this time alignment must be maintained by a tracking system. In addition, demodulation requires an accurate determination of the time of the start of each modulated symbol by a symbol synchronization system. The simulation provides frequency-hop and direct-sequence acquisition systems, direct-sequence tracking systems and symbol synchronization systems. No frequency-hop tracking is provided since more precise tracking such as direct-sequence or symbol tracking is assumed to be used after frequency-hop acquisition and is assumed to feed back information to the frequency-hop system for correction. Timing for frequency-hop operation is usually less critical than that for the other systems and it is less capable of providing accurate timing information. A possible exception is the case of fast frequency hopping in which the hop rate is higher than the symbol rate, but the ratio is not expected to be very high, and symbol synchronization information will likely be adequate to control the frequency-hop reference delay.

The time synchronization is performed at various places in the receiver. In Figure 7.18 the boxes marked "frequency-hop acquisition",

"direct-sequence synch.", and "symbol synch." contain time synchronization systems. In addition, the "MCSK matched-filter demodulator" has its own symbol synchronization system, which is described along with that demodulator in Section 7.6.4.9.

These different synchronization systems must interact with each other in many cases, and a number of modes are provided in the simulation to reflect the way in which they do. For example, when both frequency-hop and direct-sequence spreading are used the initial acquisition is usually performed by the frequency-hop system, since it will have the coarsest time requirements and can search very quickly. Once acquisition has been accomplished the time information is passed to the direct-sequence system. This system requires much more accurate time information, and so must search over the area of uncertainty left by the frequency-hop system. This combination permits much faster acquisition than direct sequence alone because the latter is much slower as a result of its need to search in smaller steps. When direct-sequence acquisition is successful the tracking mode is entered and changes in timing are detected and corrected. The more accurate timing information from the direct-sequence acquisition and tracking may be used by the frequency-hop system to improve and update its timing. This information may also be used for symbol synchronization if the relationship between the direct sequence and symbol edge is known. If not, a conventional symbol synchronization system may be used independently of the direct-sequence synchronization.

Nine different modes of synchronization are provided in the simulation, ranging from complex ones like that just described to the very simplest in which no automatic synchronization is used and all timing information is provided by the user. While this simple method would not be used in real systems it is a useful simulation mode because it allows the performance of a system to be examined independently of the effects of the synchronization system. These modes will be described in detail after the individual acquisition and tracking algorithms have been discussed.

There are three delays that may be set in the receiver by the user to

compensate for various delays that may occur in the propagation path or in the receiver. They represent the delays set into the receiver "clocks" to align them with the "clock" used to time operations in the transmitter. These delays may be used to provide manual synchronization when no automatic synchronization systems are used, or to provide a starting point for the acquisition systems. The first is the propagation delay. This is not the actual delay determined by the medium, but the user's estimate of it. He may choose to enter a delay that he knows is not the true delay (for example, to use as a starting point for an acquisition search when he wants to test the acquisition algorithm). The second delay is the front-end delay, which is intended to compensate for delays in the receiver front-end caused by filtering at that point. The third delay is the bandwidth-reducer delay which is intended to compensate for delays in the bandwidth reducer, caused by filtering at that point. The reason for the separate delays is that they affect different processes. A look at Figures 7.18, 7.19, and 7.20 will indicate which processes are affected. The propagation delay will affect all receiver processes since it precedes them all. The other delays are associated with the filters in the front end and bandwidth reducer, and only affect those processes that follow. The delays are specified in samples at the points where they apply. Therefore, the user need not be concerned with any decimation operations used in the receiver. The user enters these delays in response to questions near the end of the receiver set-up procedure. Negative delays may not be entered, but these delays may become negative when changed by the synchronization system.

7.6.7.2 Frequency-Hop Acquisition

The process of acquisition is one of changing the delay of the frequency-hop reference (the output of the receiver frequency-hop generator used for de-hopping) in increments smaller than the duration of a hop interval and testing the energy integrated from the squared magnitude of a given number of samples of the de-hopped signal for each such delay. When that energy exceeds some threshold the acquisition system decides that

acquisition has been successful and the search is stopped. In fact this decision of "in lock" is usually based on a more sophisticated algorithm than the crossing of a single threshold, but always involves the crossing of thresholds by integrated power. Three different lock-indication algorithms are available for the user to select from and each of these will be described in turn. But first let us explain how the search in time delay is carried out.

7.6.7.2.1 Search Strategy

A "zigzag" search has been implemented, in which the delay is changed in increasing sweeps in alternate directions as shown in the example of Figure 7.33. The strategy is based on the assumption that the starting delay of the search is the most likely value of the actual signal delay and that the probability of finding the signal diminishes as we move away from this value. Thus we spend more time searching where the probability of success is highest. With the correct choice of parameters this search can be made into a linear search in one direction.

The user specifies the delay increment, the initial turn-around excursion (the term excursion is used here to mean the delay change from the initial delay, and turn-around excursion is the excursion allowed before the search direction is reversed), and the maximum excursion. The initial delay is derived from the propagation delay entered by the user as mentioned earlier. At each reversal of direction the turn-around excursion is doubled. When the maximum excursion is reached the direction of search is reversed, and when it is reached a second time (in the opposite direction) the search is restarted from the initial delay. Figure 7.33 illustrates what happens when the delay reaches the turn-around excursion. When the delay increment causes the excursion to equal or exceed the turn-around value the incremented value of delay is used, but the search direction is reversed for the next increment. The same is true for the maximum excursion except for the second time the maximum is reached; the delay is reset to the initial delay immediately at this point without the

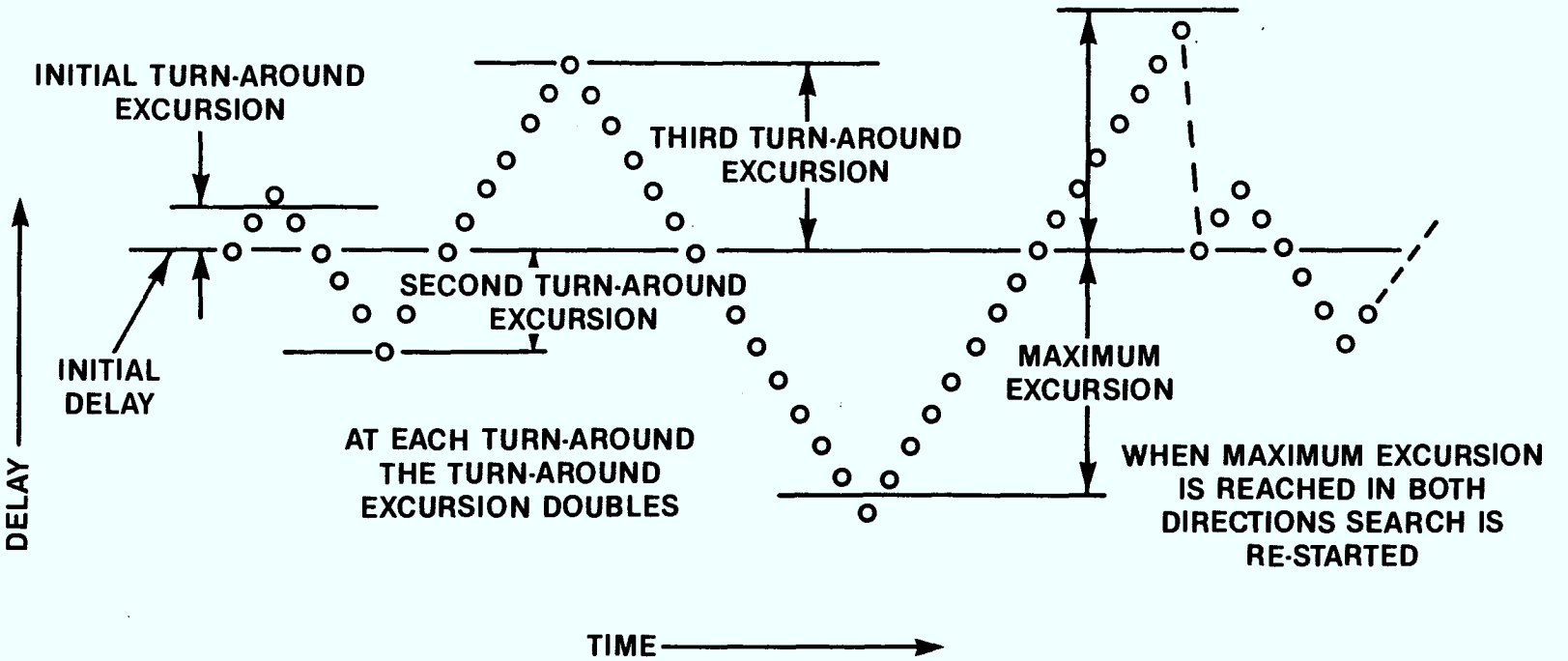


Figure 7.33 Search Strategy

incremented value being used.

7.6.7.2.2 Double-Integration Method

In the double-integration method signal power is integrated for a relatively short interval in the first stage, and for a longer interval in the second stage only when the first stage threshold is crossed. The idea is to keep the search time small by accepting the high false-alarm rate resulting from the short integration interval of the first stage (assuming the probability of a miss must be kept small), and discarding the false alarms that do occur by testing in the second stage. There, the longer integration time will allow both a low probability of false alarm and a low probability of a miss, while the cost in time will not be so high since it is used relatively infrequently. The double-integration method can also serve as a single integration one if the second integration threshold is set low enough that it is always exceeded.

The algorithm for this method is shown in Figure 7.34. The user specifies the first integration time, T_1, the first integration threshold, δ_1, the second integration time, T_2, and the second integration threshold, δ_2. Integration times are specified in terms of samples. The input is the squared magnitude of the signal taken after the wideband AGC as shown in Figure 7.18, but it is the square root of the sum which is compared with the threshold. Thus, the thresholds are specified in volts rather than as energy. (Actually, the units should include the square root of time, but since our time is defined in terms of number of samples we have chosen to leave out a reference to time here, and consider the sum to be over a number of values rather than time.) Appendix C presents equations that will be useful in determining integration times and thresholds for desired false-alarm and miss probabilities when the noise is Gaussian.

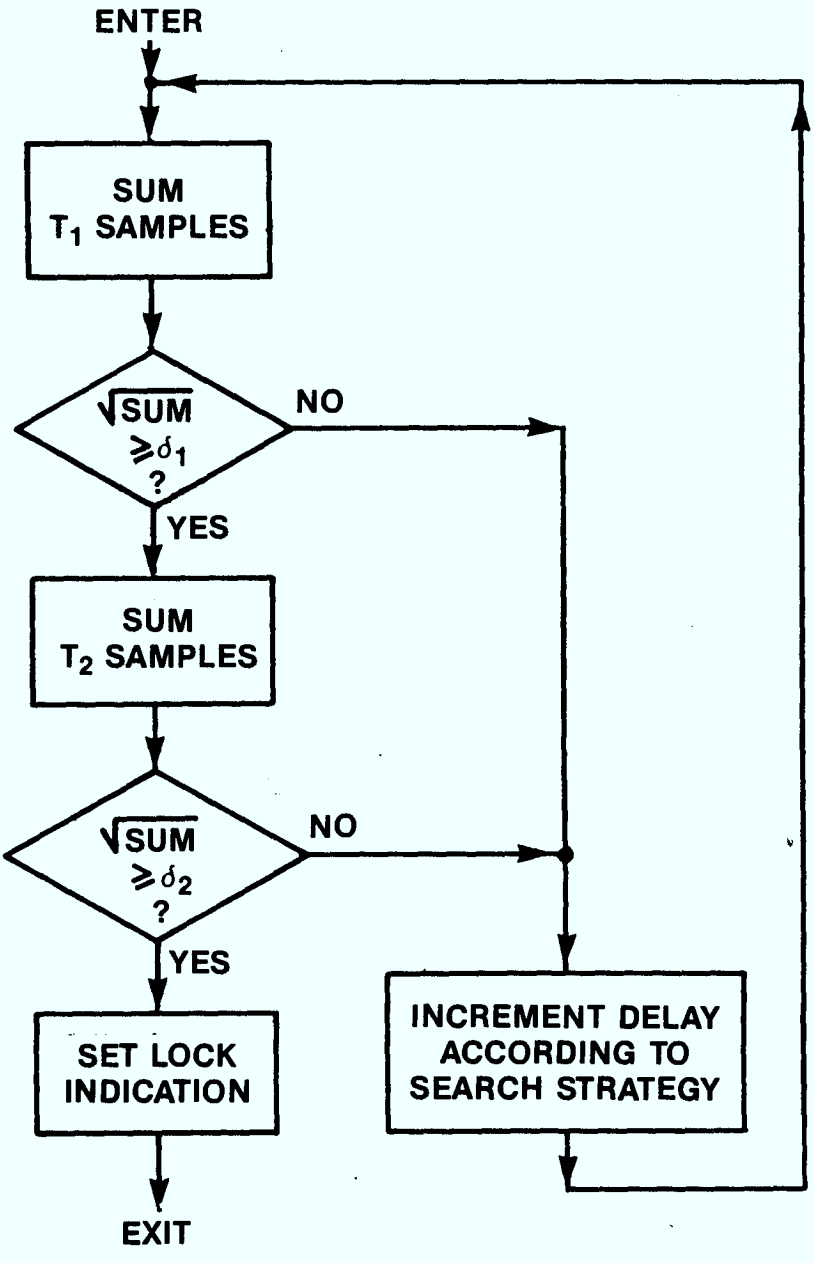


Figure 7.34 Double Integration Lock Detection

7.6.7.2.3 Sequential Detection Method

In the sequential detection scheme a decision on "signal present" or "signal not present" is made only after enough data has been collected to reach a high level of confidence in the decision. This results in a variable time spent at each delay. A limit is put on this time to prevent the possibility of a very long wait. When this limit is reached without a decision the decision is made in favor of "signal not present". In the algorithm used in the simulator, and shown in Figure 7.35, the sequential detection is based on a count of the number of times a threshold is exceeded by the integrated power in a fixed integration interval. The count is increased by one each time the threshold is exceeded, and decreased by one each time it is not exceeded. When this count reaches a particular positive value the decision is "signal present". When it reaches a particular negative value the decision is "signal not present". A more optimal sequential detection in white Gaussian noise would continue to integrate samples in a single sum and compare that sum to two thresholds. However, the method used here has the advantage of reducing the effect of strong intermittent interference since a short burst, no matter how strong, will only affect one count, and have only a small effect on the decision. Such interference is expected to be common in a frequency-hopping system as the receiver hops into bands with different interference levels.

When the sequential method is selected the user specifies the integration time (number of samples), T_1, the integration threshold, δ_1, the count threshold for detection, C_1, the count threshold for dismissal, C_2, and the iteration threshold for dismissal, C_3. The integration threshold is in volts since it is compared with the square root of the integration sum. Appendix C presents equations that should be helpful in determining the integration time and the integraton threshold.

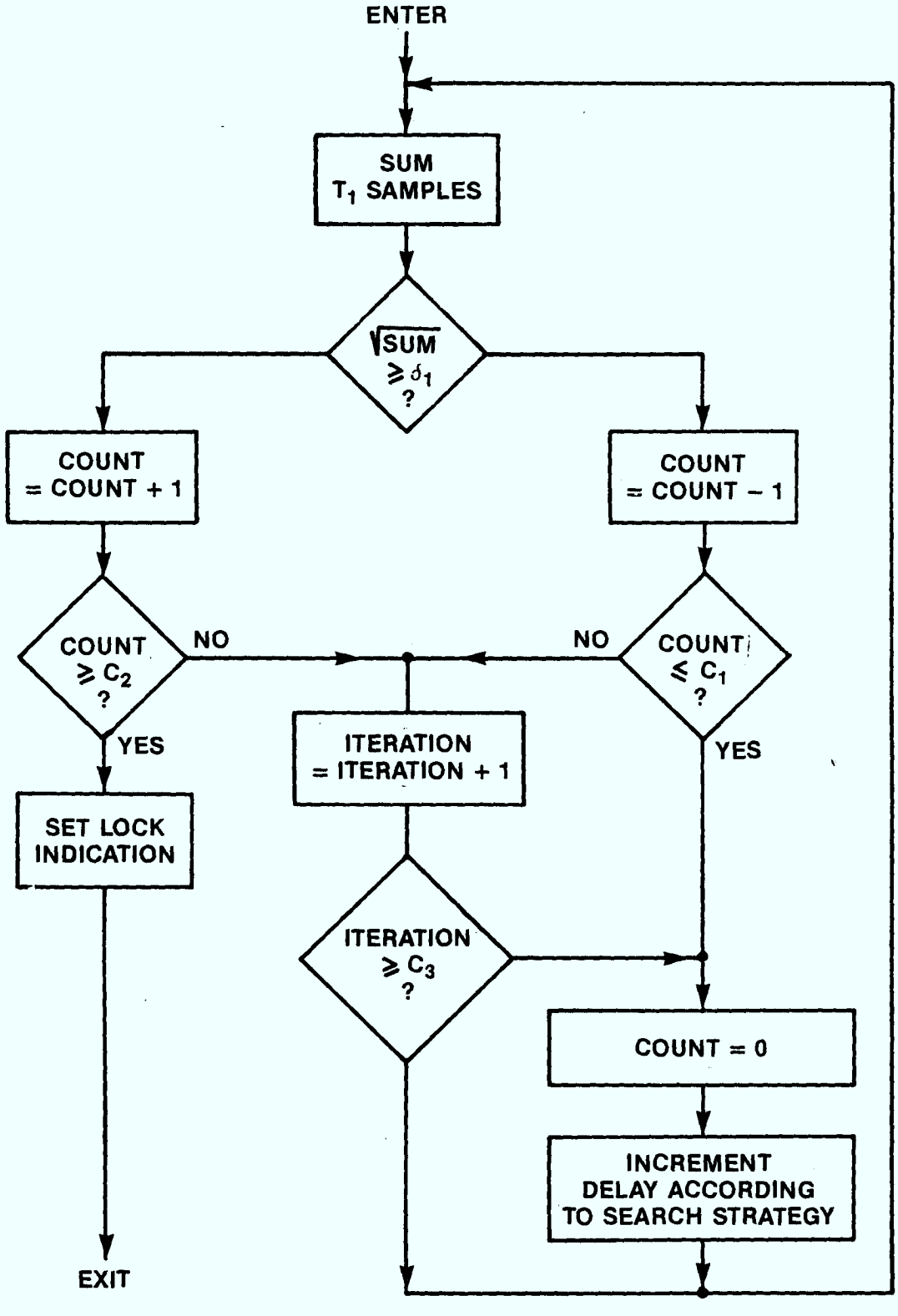


Figure 7.35 Sequential Detection

7.6.7.2.4 Mean-Delay Method

The mean delay method is intended to provide a more accurate estimate of the true delay by computing a weighted mean of the delay based on the integrated power at each delay for which a threshold is exceeded. When the threshold is exceeded a specified number of times in succession a detection is assumed but the search is not stopped. Instead, it is continued in the same direction until the threshold is not exceeded a specified number of times. Then the mean of the delay is computed with each delay weighted by the integrated power measured at that delay. This routine is intended to be used for frequency-hop acquisition when no direct-sequence acquisition is available to refine the delay estimate. The other two acquisition methods are not accurate enough to determine delay within one symbol duration, as is necessary for refinement by the symbol synchronization routine.

The algorithm for this method is shown in state-diagram form in Figure 7.36. The square root of the integration sum over a block of T_1 samples is compared with the threshold, δ_1, and the action taken depends on the state that the algorithm is in. After the particular action is taken a new sum is taken over the next T_1 samples, the delay is incremented, and the process is repeated. Before acquisition, the algorithm is in state 1. As long as the threshold is not exceeded it remains in this state. When the threshold is exceeded state 2 is entered. In this state a count is kept of the number of consecutive "hits" (sums above the threshold). When N_1 such hits have occurred state 3 is entered. A "miss" occuring before the threshold N_1 is reached will cause state 1 to be re-entered and the hit count to be reset to zero. The intention is to prevent false alarms from intermittent bursts of interference, and to define the start of solid acquisition more precisely. Once state 3 is entered a detection is assured; all that remains is to determine the delay at which loss of signal occurs. State 3 is left only when a miss occurs. Then state 4 is entered and a count is kept of the number of consecutive misses. After N_2 consecutive misses, lock is established (state 5) and the mean delay is computed from the weighted mean of the delays in states 2 and 3 (when the

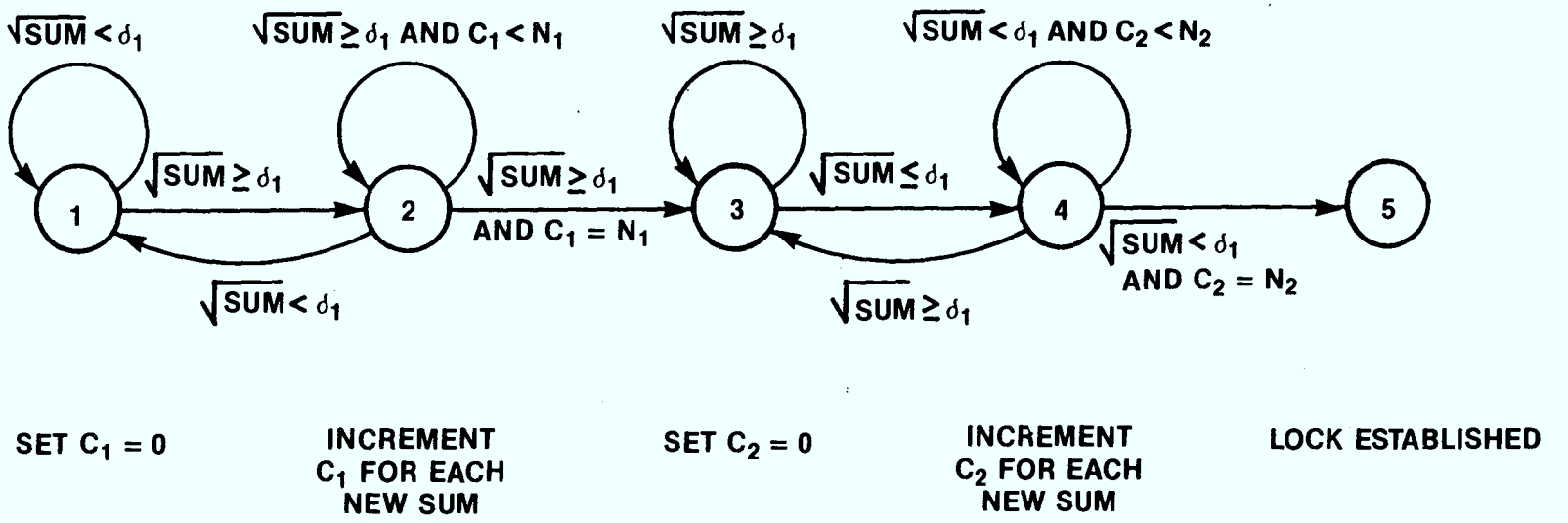


Figure 7.36 State Diagram of Mean Delay Algorithm

threshold was exceeded), weighted by the integrated power at each delay. The reference delay is then set to this mean value. If the threshold is exceeded in state 4 before N_2 consecutive misses state 3 is re-entered and C2 is reset to zero. This is to reduce the probability that a fade will signal a false end to the acquisition. Not shown in Figure 7.36 is a test to end the acquisition if the threshold is exceeded for a very long time as a result of either sustained very strong interference or a bad threshold setting. A maximum of 1024 delays in states 3 and 4 are permitted before state 5 is entered. While this is not an ideal solution, it at least limits the number of integration values that must be stored.

For the mean delay method the user, in addition to specifying the integration time and threshold, must specify the number of successive detections for acquisition, N_1, and the number of successive misses for completion, N_2. Appendix C should be helpful in determining the integration time and threshold.

7.6.7.3 Direct-Sequence Acquisition

The algorithms for direct-sequence acquisition are the same as those for frequency-hop acquisition; the only difference is the input signal. For direct-sequence acquisition the input is the squared magnitude of the de-spread signal shown in Figure 7.20 as "lock detection input voltage". Noncoherent integration is performed in the acquisition algorithm since the squared magnitude of the signal is summed, and coherent integration is performed by the filter of Figure 7.20. Coherent integration is more effective than noncoherent, but it may be performed only over one data symbol since the phase of the next symbol, relative to the current one, is unknown. Thus, for acquisition, the filter should match the spectrum of the data symbol as well as possible. However, in the present simulator the filters in all channels of Figure 7.20 must be identical, and the filter in the main signal channel should be a little wider since the demodulator that follows will perform a matched-filter operation. The loss in the acquisition channel for the wider filter recommended for the signal channel

should be quite small.

When direct-sequence acquisition is in operation the search controls the direct-sequence reference signal delay (although it may control the frequency-hop delay as well if frequency-hopping is used). This is the "offset" delay in Figure 7.20. The delay increment should be smaller than the width of the autocorrelation peak of the spreading signal for high probability of detection. This peak will have a width of two direct-sequence elements at the base, and of one element at the half-voltage point. Since a direct-sequence element corresponds to a modulation symbol, the element length in samples will equal the number of samples per modulation symbol, a value that the user has previously specified. The initial delay for the search will be the delay determined by the frequency-hop acquisition if it has been used; if it has not, it will be determined by the program from the user-specified propagation delay and front-end delay.

7.6.7.4 Direct-Sequence Tracking

Direct-sequence tracking is performed by a comparison of the energy in the correlation of the received signal and the direct-sequence reference for delays of the reference of slightly less than, and slightly greater than, that of the received sequence. Since the correlation peak has a triangular shape the difference between these two correlation values, which straddle the peak, provides both direction and magnitude information required for correction of the reference delay to allow it to track the delay of the received signal. Coherent integration for the correlation is performed by the filters in Figure 7.20. For best performance these filters should be matched to the data symbol spectrum. However, as stated above, the main signal channel demands a slightly wider bandwidth, and this may cause some degradation in the tracking performance. But the degradation should be quite small, since further noncoherent integration is performed in the tracking algorithm.

Two alternative routines are provided in the simulator to perform the tracking task. These are the delay-locked loop and the tau-dither loop.

7.6.7.4.1 Delay-Locked Loop Tracking

The input to the delay-locked loop tracking algorithm is taken from the output of the difference circuit of Figure 7.20, while lock status is determined from the squared magnitude of the main signal channel voltage. The two channels feeding the difference circuit generate the correlation of the signal with the early and late versions of the direct-sequence reference. The delay difference between the early and late references is twice the user-specified differential delay (see Figure 7.20). The difference of the squared magnitudes of the two channels is used to determine the magnitude and direction of the correction. A third correlator with reference delay midway between those of the other two is used for the de-spreading of the signal to be demodulated. When the tracking is accurate this delay should be the actual delay of the received signal.

The tracking algorithm is described in Figure 7.37. Both the tracking error signal and the lock detection voltage are summed over a user-specified number of samples, T_l. If the square root of the sum of the lock detection voltage, S_L equals or exceeds a user-specified threshold δ_L, a new value for the direct-sequence reference delay is computed; if not, the delay is left unchanged and the miss counter is incremented and tested against a user-specified threshold, C_m. If C_m consecutive misses have occurred, tracking is assumed to have failed and "unlock" is declared, causing acquisition to be re-entered starting from the last delay determined by the tracking system.

When the lock threshold is exceeded the tracking sum, S_T, is mapped into a new voltage, M_T, according to one of two types of mapping selected by the user. The first is a linear mapping (no change), and the second is a limiter mapping where the output, M_T, is equal to the input, S_T, when

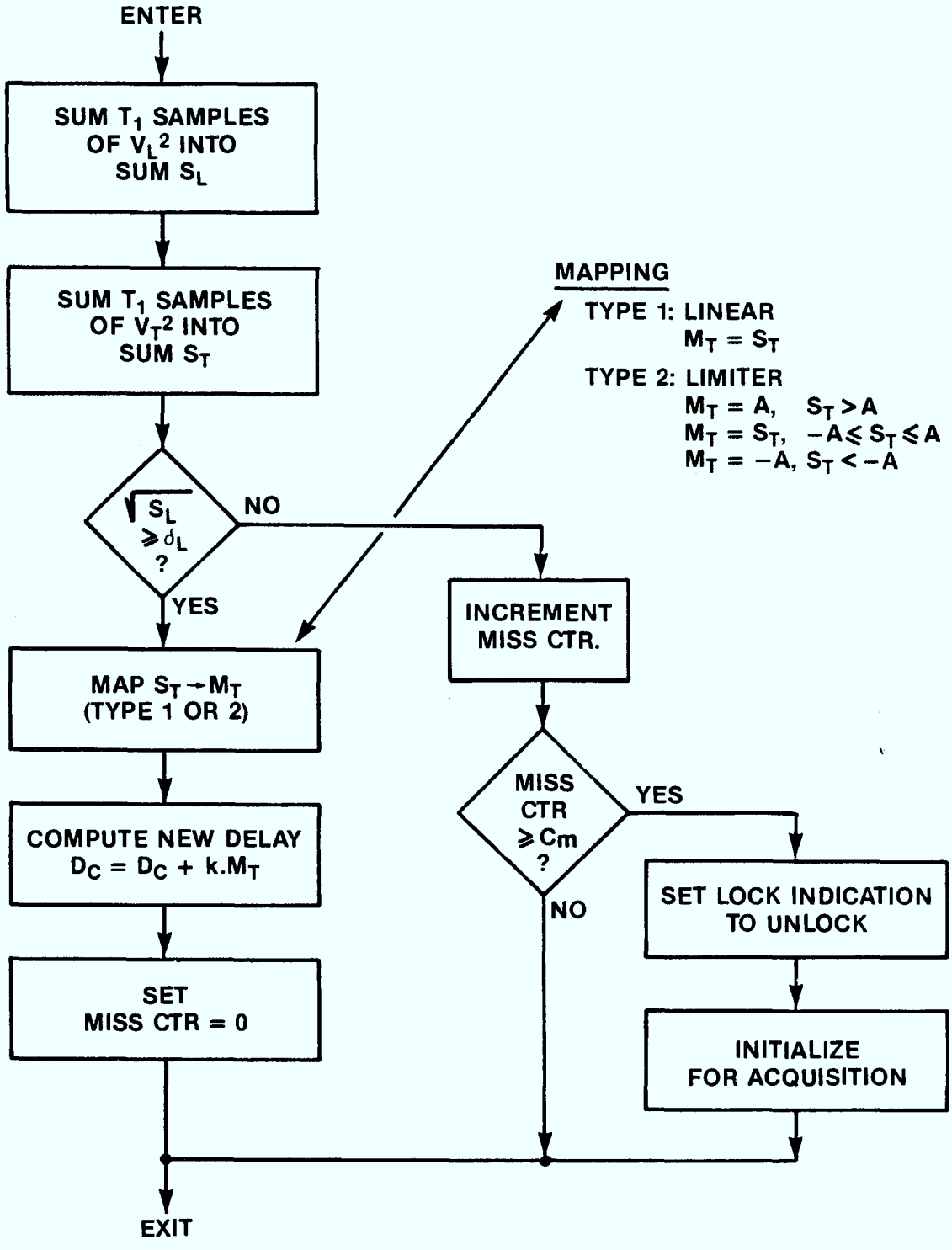


Figure 7.37 Delay-Locked Loop Algorithm

the magnitude is less than or equal to a user-specified threshold, A, and limited to plus or minus A when the threshold is exceeded. A user-specified multiplier, k, is used to multiply the value M_T to provide the delay correction that is added to the old value of direct-sequence reference delay to determine the new delay. The units of k are samples per volt. Larger values of k will cause faster correction, but will allow greater errors from strong interference bursts, and may lead to instability if the correction generated is greater than twice the error. The use of the limiter-type mapping makes it possible to set good values of k when input signal levels are unknown or widely varying. A maximum change, A.k, of not greater than one sample interval is recommended. It should be noted that although the actual delay of the reference cannot change by less than one sample, the delay value, D_C, is computed as a floating-point number that is rounded to the nearest integer only when controlling the reference delay. Thus, a series of computed corrections, each much less than one sample interval, will eventually lead to a reference delay change.

7.6.7.4.2 Tau-Dither Tracking

In the Tau-dither system the two correlations are performed sequentially in a single correlator. The reference delay is alternated between the early and late values, remaining at each for the specified integration time. The tau-dither system has the advantage of requiring only one correlator, but it takes twice as long to compute a correction. Another disadvantage is that the de-spread signal going to the demodulator is taken from the same correlator as is used in tracking, and is therefore always offset a little from the correct delay, causing a small loss in demodulator performance.

The input to the Tau-dither algorithm is the squared magnitude of the voltage in the main signal channel, V_L^2 (see Figure 7.20). This is used for both lock detection and correction. The parameters specified by the user are the same as for the delay-locked loop. Figure 7.38 describes the algorithm. A flag called the early/late (E/L) flag is toggled each time a

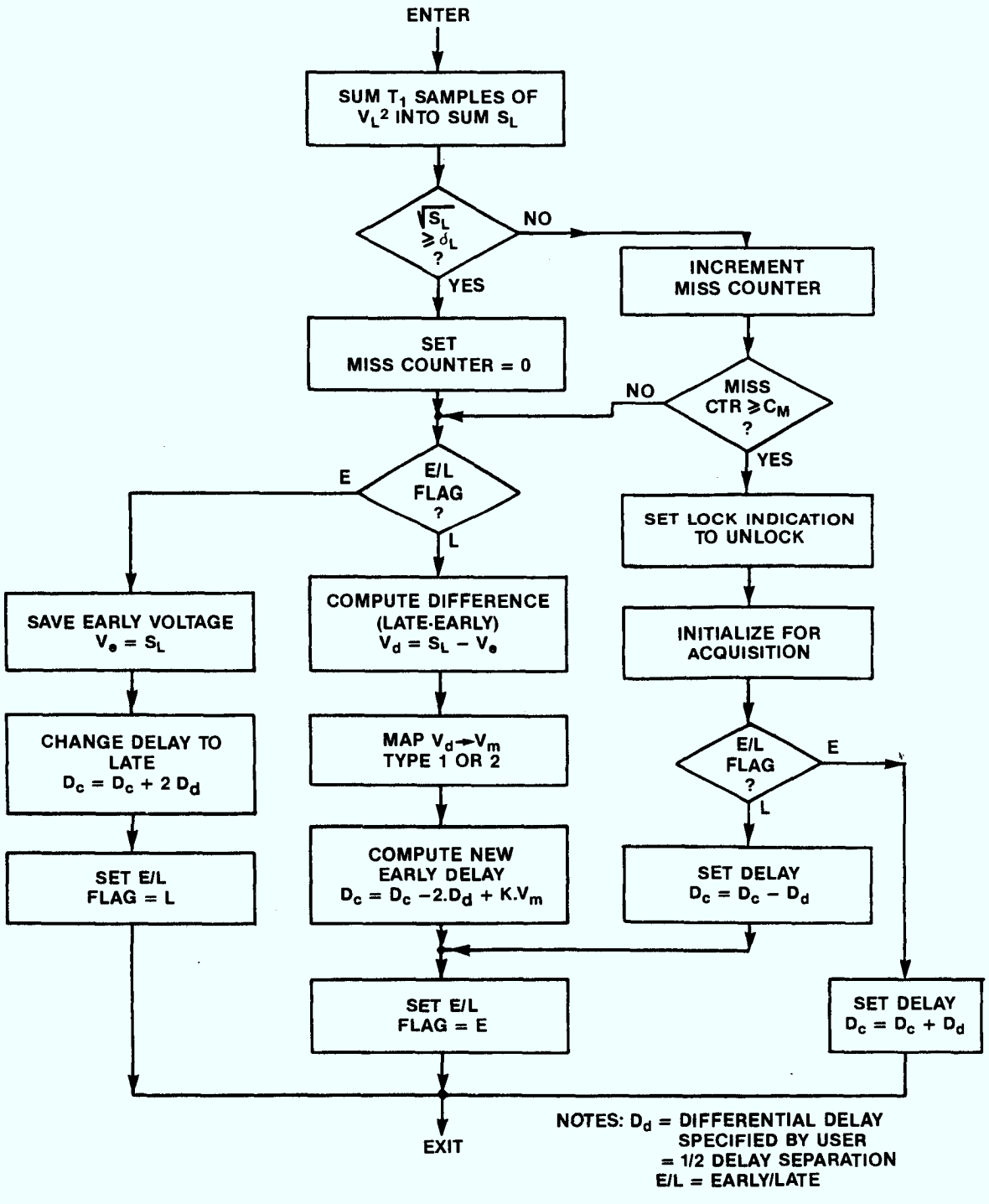


Figure 7.38 Tau-dither Loop Algorithm

sum is computed, and is used to alternate between early and late reference delays that are separated by twice the user-specified differential delay, D_d, just as they are in the delay-locked loop system. The early integration is performed first, and when the corresponding late value is computed the early sum is subtracted from it, and the delay correction computed as for the delay-locked loop. Only one correction is computed for each early/late pair of integrations. One difference from the delay-locked loop algorithm is that correction is not inhibited when the integration threshold is not exceeded. When lock is lost (C_m consecutive misses) the delay is set to the point midway between the last early and late values, in preparation for a return to the acquisition mode, and the E/L flag is set to the early position in preparation for the next entry into the tracking mode.

7.6.7.5 Symbol Synchronization

A symbol-synchronization capability is included in the simulator to allow realistic operation under varying multipath conditions. The "normal" synchronization for PSK signals is fairly standard, and the routine gives recommended values for some of the parameters; however, it is left to the user to set them, giving him the option of experimenting. A modification to this standard system for use with FSK signals is included. This modified system, as far as is known, has not been used in any real systems, and is offered here as an experimental one that has not been well tested. Finally, a synchronization system for any modulation that uses amplitude-shaping of the modulated symbols is included. This makes use of the shaping to simplify the algorithm.

The symbol-synchronization system, with all its modifications, is shown in Figure 7.39. For the normal system, ignore the input mixers and adder to the left of S_{1a} and assume the switch is in position 1. The timing information is extracted from the phase changes between symbols. Unfortunately, the spectrum of the unshaped PSK signal contains no power at the symbol rate, $1/T_s$. A squaring circuit is used to move some of it

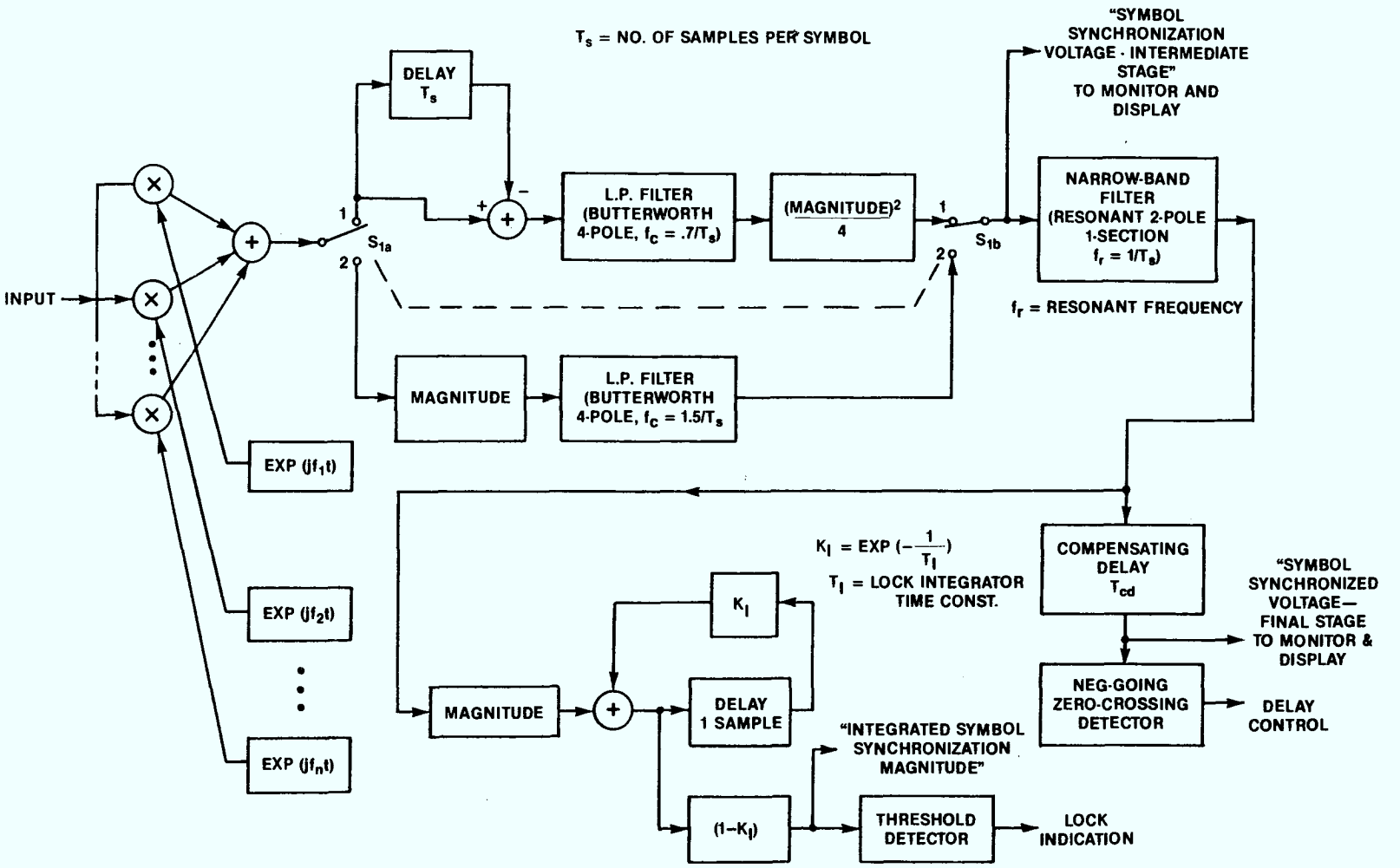


Figure 7.39 Symbol Synchronization

there from the energy at $1/2T_s$, but it is necessary to use a "prefilter" before this to remove some of the spectrum. The delay and difference circuit, in combination with the low-pass filter, forms this prefilter. The prefilter should provide a response with a broad peak in the frequency domain at one-half the symbol rate. The delay and difference circuit is a simple filter with a half-cycle-sine-wave frequency response between zero and the symbol rate ($1/T_s$). It has a maximum voltage gain of 2 at one-half the symbol rate. This response is repeated with a period $1/T_s$, and it is the purpose of the low-pass filter to remove these repetitions leaving only the part from zero to $1/T_s$ (and its image at negative frequencies). A four-pole Butterworth filter (2 sections) with cutoff frequency $0.7/T_s$ has been found to work well in this role. This recommendation and the computed value of the cutoff frequency are displayed for the benefit of the user.

After squaring, the energy at $1/T_s$ is extracted by a narrow-bandwidth filter. The resonant filter (resonator) from the general filter routines (see Section 8.1.2.2) is useful for this purpose. Each section of this filter has a complex-conjugate pair of poles defining the peak of the response, and zeros at one and minus one in the Z-plane. The user is advised to use a one-section resonant filter and is given the computed resonant frequency $1/T_s$. If the user chooses to use the resonant filter he is asked to specify its time constant. This time constant is the inverse of π times the filter bandwidth. There is a tradeoff in the selection of this time constant. A long time-constant will result in the system holding its timing over longer periods of fade or of repeated symbols (timing information is only generated when there is a change of symbol), and will also result in less output noise. On the other hand, the high inertia of a long time-constant will make acquisition of track slower, and restrict the rate of track correction.

The symbol timing information is given by the zero crossings of the resonant filter output. The negative-going crossings are chosen arbitrarily to mark the start of the symbols, but as a result of delays in the circuit, a compensating delay, specified by the user, must be inserted

to bring the crossings into coincidence with the actual start of the symbols. This delay, although fixed for any particular set of parameters, is difficult to predict accurately, and it is recommended that it be found by a test of the selected system without noise, and with the compensating delay set to zero. The Display feature can be used to record the "symbol synchronization voltage - final stage" (see Figure 7.39), and the negative-going zero crossings compared with the symbol edges (these can be computed from the sample number and the number of samples per symbol if no delay of the symbol has occurred). The compensating delay delays the zero crossings, so the required delay will be the difference between the symbol edge and the nearest earlier negative-going zero crossing.

The lower part of Figure 7.39 generates a lock indication by taking the envelope of the synchronization voltage and comparing it with a user-specified threshold. The user-specified "lock indicator time constant" should be a few symbols in length to provide small decay between cycles of the narrowband signal, but less than the resonant filter time-constant so that it may follow amplitude changes in the signal. The threshold should be set so that when there is only noise there are very few false lock indications. The noise at the threshold detector should be Rayleigh, and for the normal mode its mean value is approximately equal to 0.4 times the input rms noise voltage divided by the square root of $T_s.T_r$, where T_r is the time constant of the narrowband filter, and T_s is the number of samples per symbol. A threshold of at least three times this should be used. It would probably be worthwhile testing the system with noise only and adjusting the threshold as necessary.

When FSK is used the different symbols have spectra centered at different frequencies across the modulation bandwidth. The simulation sets these in a symmetrical manner about zero frequency, and there is never one centred at zero. Thus, the prefilter will not provide the required signal to the squarer. This can be rectified by shifting the received spectrum enough to put the lowest tone at $1/T_s$. For binary FSK with separation of $1/T_s$, this requires an offset of $3/2T_s$. This will produce an amplitude modulation, as the tone frequency changes from symbol to symbol, that can

provide the desired timing information from the narrowband filter. The complex tone generators and multipliers are included for this purpose. Multiple generators (up to 16) are available so that, if there are many tones, more than one may be shifted to $1/T_s$ to increase the frequency of responses from the prefilter. While this technique has been found to work it has not been analyzed in any detail, and it is left to the user to determine the optimum setup for a particular system.

When the modulated symbols have had amplitude shaping applied, the shaping can be used to provide the necessary input to the narrowband filter. It is only necessary to take the magnitude of the shaped symbol (it does not matter whether it is PSK or FSK). The shaped-symbol mode of symbol synchronization provides this feature. When it is selected, S_{1a} and S_{1b} of Figure 7.39 are in position 2. A four-pole Butterworth filter with cutoff frequency of $1.5/T_s$ is shown following the magnitude computer. This filter is not really necessary since it is followed by a narrowband filter, but it provides some filtering for the Monitor signal at this point. The values are only recommended ones and the user may choose any filter he desires. When the shaped-symbol mode is selected the noise level at the lock threshold detector should be Rayleigh with a mean voltage of about $0.34/\sqrt{T_r}$ times the input rms noise level.

7.6.7.6 Synchronization Modes

There are nine different synchronization modes available. These are not selected directly by the user, however; the program determines the mode from the answers to questions regarding the use of the various synchronization systems. Each of these modes will be described, and the interaction between the different synchronization systems involved will be explained.

7.6.7.6.1 Mode 1 - No Acquisition and no Tracking

In this mode all reference signal delays and the start of the symbols are derived from the propagation, front-end, and bandwidth-reducer delays specified by the user near the end of the receiver set-up. The user may choose to enter incorrect delays there to allow him to determine the effect of delay errors.

7.6.7.6.2 Mode 2 - Symbol Synchronization Only

The only type of synchronization used in this mode is symbol synchronization. If frequency hopping is used the frequency-hop delay is derived from the symbol synchronization system. Direct-sequence spreading may not be used in this mode. Since the symbol-synchronization system will only determine the nearest symbol edge, the initial delay set by the user (the combination of propagation, front-end and bandwidth-reducer delays) must be accurate to within half a symbol if the output data bits are to have the same time-index values as those input to the transmitter. A difference in index value will not be a problem if frequency hopping is not used, since bit comparisons can be made with different index values, but when frequency hopping is used a difference in index value will lead to an error in the timing of the frequency-hop (FH) reference.

7.6.7.6.3 Mode 3 - Frequency-Hop Acquisition and Symbol Synchronization

Direct-sequence spreading may not be used in this mode. Frequency-hop acquisition starts from the propagation delay specified by the user. After sucessful FH acquisition, symbol synchronization is performed, and, when successful, feeds corrections to the FH system to refine its delay. An error of more than half a symbol in FH acquisition cannot be corrected by the symbol-synchronization system; it will lead to a FH reference delay that is in error by one or possibly more symbols. For this reason it is recommended that the mean-delay acquisition algorithm be selected in this

mode. It provides a higher accuracy than the other methods. If symbol synchronization is lost, FH acquisition is re-entered with the previously used initial conditions but with the search starting from the current delay. An indication of "lock" to the Monitor and Display occurs only when the symbol synchronization system is locked.

7.6.7.6.4 Mode 4 - Direct-Sequence (DS) Acquisition and DS Tracking - No Symbol Synchronization

DS acquisition starts from the combination of the propagation and front-end delays specified by the user. When it is successful DS tracking is initiated. Symbol delay is derived from the DS system. If frequency hopping is used its delay is also derived from the DS system. If DS tracking lock is lost acquisition is re-entered with the previously used initial conditions but with the search starting from the current delay.

7.6.7.6.5 Mode 5 - DS Acquisition and DS Tracking - Independent Symbol Synchronization

This mode is similar to mode 4 except that the symbol delay is obtained from an independent symbol synchronization operation rather than from the DS system. If either symbol synchronization or DS tracking lock is lost the overall lock indication becomes "unlock".

7.6.7.6.6 Mode 6 - FH Acquisition and DS Acquisition and DS Tracking - No Symbol Synchronization

FH acquisition starts at the user-specified propagation delay. When it is successful DS acquisition starts from the delay determined by the FH acquisition, taking into account any front-end delay. From this point on mode 6 is the same as mode 4. Loss of DS lock will result in reentry into the DS acquisition as in mode 4. FH acquisition is not reentered.

7.6.7.6.7 Mode 7 - FH Acquisition and DS Acquisition and DS Tracking and Independent Symbol Synchronization

This mode is the same as mode 6 except that the symbol delay is derived independently from the symbol synchronization system rather than from the DS system. Loss of either symbol synchronization lock or DS tracking lock will cause an overall "unlock" indication.

7.6.7.6.8 Mode 8 - MCSK Matched-Filter Symbol Synchronization Only

Symbol tracking occurs within the matched filter only. Since the MCSK matched filter replaces the normal demodulator, normal symbol synchronization is not possible. If frequency hopping is used its delay is derived from the matched-filter timing. DS spreading may not be used in this mode. The initial delay must be accurate to about half the matched-filter window width to assure correct synchronization. If it is not, correlation "sidelobes" may bring the window toward the peak, or random drift may result in eventual synchronization, but this cannot be assured. As well, synchronization may be obtained with the wrong symbol as in mode 2, with the same result.

7.6.7.6.9 Mode 9 - FH Acquisition and MCSK Matched-Filter Symbol Synchronization

FH acquisition starts at the user-specified propagation delay. Once acquisition is successful this mode becomes identical to mode 8 except that if symbol lock is lost FH acquisition is re-entered with the previously used initial values, but with the search starting from the current delay. A "lock" indication exists only when the symbol lock is on.

7.6.8 Monitoring and Control Facilities

In this section we will expand on the Monitor, Display, and program control features mentioned in the introduction to the RECVR process.

7.6.8.1 Monitor Feature

When the simulation is running in the interactive mode (running from the terminal) the Monitor displays a number of different variables updated at a rate specified by the user in terms of the update interval measured in number of input samples. Three of these variables are displayed regardless of the receiver configuration; the others are selected on the basis of the particular receiver configuration selected. The three that are always displayed are: the index value of the input sample being processed, the number of final (FIN) values generated up to this point in the run, and the mean of the magnitude of the voltage at the output of the de-hopper, averaged over the update interval. The point from which the voltage to be averaged is taken is shown in Figure 7.19. If de-hopping is not performed the voltage is the input voltage to the front end.

The other variables that may be displayed are shown in Table 7.1 along with an indication of the Figure from which the point of measurement or the definition can be found. If the particular variable has significance for the receiver configuration selected it will be displayed by the Monitor.

When the program is operating in batch mode the Monitor is not used and the question relating to it (update interval) is not asked.

Table 7.1

Variable	Units	Pertinent Figures	Notes
Frequency-hop reference delay	samples	7.19	1
Direct-sequence reference delay	samples	7.20	1
Demodulator Sample delay	samples	7.18	2
Demodulator "tracking" voltage	volts	7.22-7.28	3
AGC gain	volts/volt	7.18,7.32	4
Direct-sequence lock status	on/off	7.20	5
Direct-sequence tracking voltage	volts	7.20	6
DS tracking integrated lock voltage	volts	7.20,7.38	7
Symbol synch lock status	on/off	7.39	
Symbol synch voltage - intermed. stage	volts	7.39	
Symbol synch voltage - final stage	volts	7.39	
Integrated symbol synch magnitude	volts	7.39	
Frequency-hop acquisition lock status	on/off	7.18	8

Notes:

1 For the source of the delay control see Section 7.6.7.6 on synchronization modes.

2 This is the delay applied to the symbol integration start time in the demodulator (either ordinary or MCSK matched-filter type). For the source of the delay control see Section 7.6.7.6 on synchronization modes.

3 This is the integrated voltage corresponding to the largest symbol response in the demodulator (among the set of possible symbols). The term "tracking" may be somewhat misleading here.

4 This may include both wideband and narrowband AGC. The gain is the ratio of voltage magnitudes.

5 This is derived from V_L^2 of Figure 7.20 by the acquisition or tracking algorithms of Figures 7.34 to 7.38, depending on type.

6 This is meaningful for delay-locked loop tracking only.

7 This is derived from V_L^2 of Figure 7.20 by the summing in the first block of Figures 7.37 and 7.38.

8 This is derived from the input to the frequency-hop acquisition system (Figure 7.18) by the acquisition algorithms of Figures 7.34, 7.35, and 7.36.

7.6.8.2 Display Feature

The Display feature permits the storing of a sequence of samples of one of the variables listed under the Monitor feature so that it can be examined after a run. Any of the listed variables may be designated as the Display variable, but only one may be selected for each run. In addition to the Monitor variables, two others, not available in Monitor, are available in Display. These are the transmission hop frequency and the reference hop frequency. All of these are listed in a menu when Display is selected. The menu indicates the narrowband and wideband AGC separately, and the matched-filter lock indication is distinguished from the symbol synchronization lock indication.

At the end of a run, while still in RECVR, a menu for display or analysis of the stored samples is presented. The samples may be looked at directly, or a histogram produced, in much the same way as the ORG or FIN data may be analysed with VIEW and HISTO in the ANAL subprogram. In addition, the output samples (the values in the FIN data array) from the receiver run may be analysed in the same way. A final option in the menu allows a return to the processing in RECVR if the processing is not complete. If it is, there is a return to PROCES. Processing will not be complete if the user has specified that only a part of the data be processed or if he has decided to interrupt the processing.

7.6.8.3 RECVR Run Control

Just before the user is asked if the Display feature is wanted, he is asked if he wants to process all the remaining samples. If he does not, he is asked to specify the number of samples to be processed. In either case, he may choose to interrupt the processing at any time by typing a CONTROL/X (control key and x key held down at the same time). This, or the completion of processing of the specified number of values, will result in the suspension of the processing and the presentation of the Display analysis menu. The user may then examine the data as explained above.

When he exits the Display routine with a BYE, and all the samples have not been processed, he is asked if more samples are to be processed. If he answers "yes", the program returns to the point where the question on whether all the remaining samples are to be processed is asked. The receiver remains in the state that it was left in by the previous run, and the processing may continue from that point. The variable selected for storage under Display must be reentered, and need not be the same as on the previous run. All of the Display values from the previous run are lost at this point, but all of the output samples in the FIN data array generated over different runs are maintained.

7.7 Post-Detection Operations Process (BITSNK)

BITSNK performs the inverse of the error-correction coding and interleaving of BITSRC. In addition, it performs inverses of the data-symbol encoding and inverse Gray-encoding functions of MODCOD. Of course these functions must be carried out in the reverse order to the order used in BITSRC and MODCOD. BITSNK takes care of this, but the user is responsible for entering the correct parameters; they are not remembered from BITSRC and MODCOD. Some of the coding functions in MODCOD are not inverted in BITSNK. These include differential encoding and MCSK coding. The decoding for these is part of the demodulation process carried out in RECVR. The inverse functions in BITSNK will be described in the order in which they would be performed.

7.7.1 Data-Symbol Decoding

Any demodulator that uses symbols of more than one bit will have multi-bit symbols as its output. To allow comparison with the bit stream used as the data source for the transmitter, and to allow other bit-oriented processes to be performed when necessary, each symbol must be decoded or expanded into a series of bits. The user must specify the number of bits per symbol. The process is the inverse of the

bit-to-data-symbol encoding described in MODCOD. If encoding was performed in the transmitter, zeros may have been added to the end of the bit sequence to make the last symbol have the required number of bits. If this is the case the user may wish to remove these bits after decoding. The routine asks the user if this is desired. If Gray encoding is to be performed, the removal is delayed until that process is completed. (The question on whether Gray encoding is to be done is asked before the question on the removal of zeros.)

7.7.2 Gray Encoding

The corresponding process in MODCOD was a transformation from Gray to binary coding; we referred to this as inverse Gray encoding. The inverse process here is therefore Gray encoding, a transformation from binary to Gray code that restores the bits to their original values (providing there have been no errors). The bits are grouped into blocks of size equal to the already-specified number of bits per symbol for this process.

7.7.3 De-Interleaving

De-interleaving is identical to interleaving; the bits are read into an array in rows and read out in columns. To restore the data to its original form the user must interchange the specifications of the numbers of rows and columns that he used in BITSRC.

7.7.4 Decoding of Error-Correction Codes

As explained under Error Coding, the only type of error-correction coding that is available at this time is cyclic block coding, but the structure is in place for other types, and the program will offer the user corresponding choices. Warnings are given, however, that these are not implemented, and when chosen, they simply pass the input data to the

output.

7.7.4.1 Decoding of Binary Cyclic Block Codes.

7.7.4.1.1 Error-Trapping Decoding

This decoding algorithm is based on the error-trapping method described in Reference [7]. It can correct all single and double errors, and all triple errors if the number of parity bits in the block is greater than one-third the total number of bits in the coded block. If this criterion is not satisfied some triple errors may not be corrected. Of course the above corrections are possible only if the code used has the required error-correction capability.

The user is asked to specify the number of bits in a coded block, n, The number of information bits in a block, k, the syndrome register feedback vector (same as the generator vector described under BITSRC), the number of errors to be corrected (1,2, or 3, depending on the capability of the code), and a block delay. This last parameter delays the starting point of a block to compensate for any delays that have not already been compensated. No automatic synchronization is provided for the error decoding.

The syndrome computer is shown in Figure 7.40. The block of n bits is entered with the switch in the closed position and the switch is then opened. The outputs of the taps form the syndrome of n-k bits. The weight of the syndrome (the number of ones in it) is tested against the number of bits to be corrected. If it is greater than this number the registers are shifted and a new syndrome is formed. The test and shift are repeated until the syndrome weight is equal to or less than the number of bits to be corrected. At this point the block may be corrected by a modulo-two addition of the syndrome with a cyclic shift of the received data bits where the number of bits shifted is determined from the number of shifts

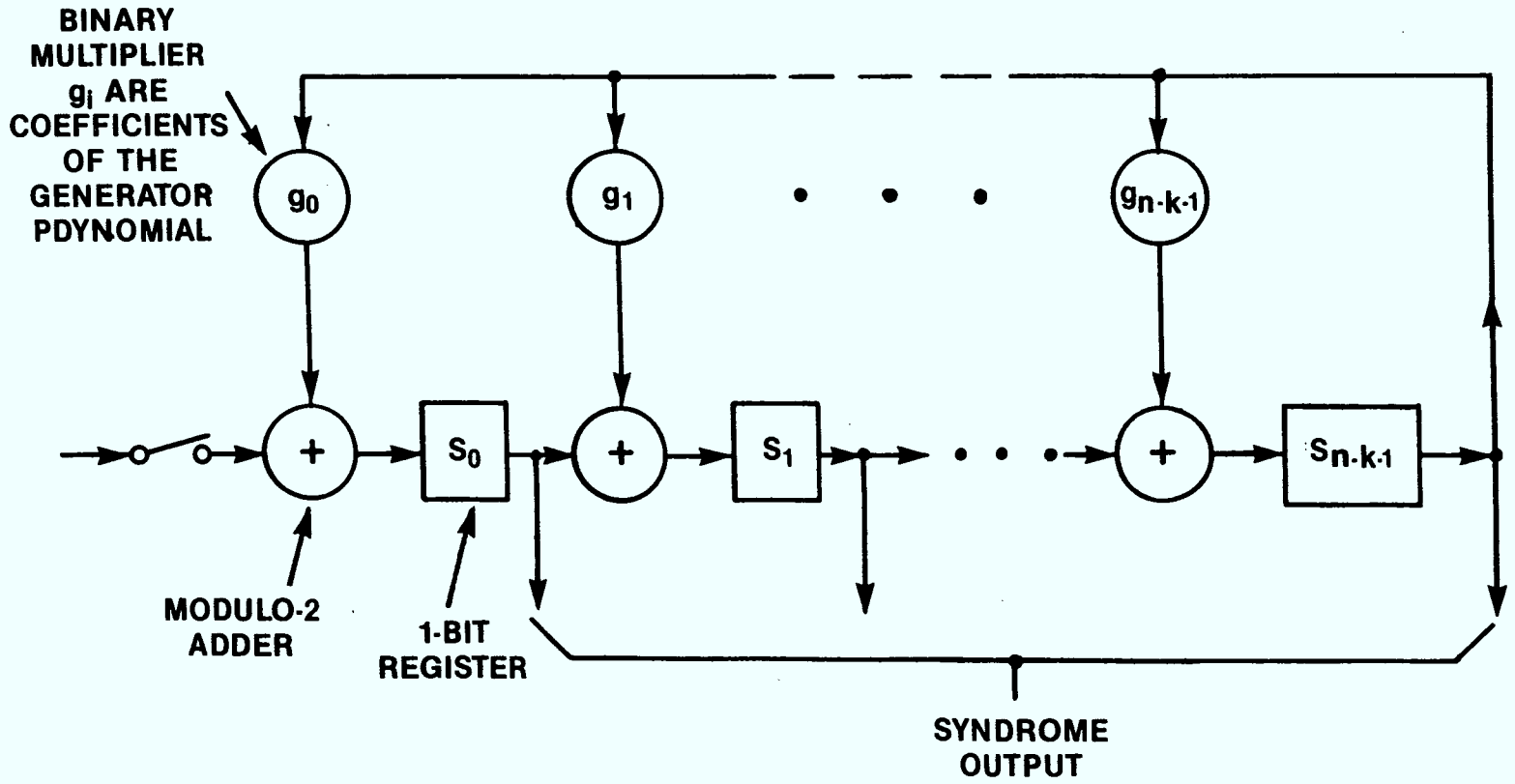


Figure 7.40 Syndrome Computer

performed in the syndrome register. If the weight criterion is not satisfied after k shifts and more than one bit is to be corrected, each bit of the data block is inverted in turn and the above procedure repeated until a correction is possible or all bits have been tested by inversion. If the latter occurs, correction is not possible and the data bits are left unaltered. After each test the tested bit is re-inverted before the next bit is inverted.

8 MISCELLANEOUS DEVICES

8.1 Filters

Finite impulse-response (FIR) and infinite impulse-response (IIR) filters are available as general-purpose devices for use in various parts of the simulation. For each type the user may enter the coefficients into the standard configuration provided to generate the desired response, or he may let the routine compute the coefficients for particular filter types such as Butterworth or Chebyshev from the specifications he enters in response to questions by the routine. While most applications will use real coefficients, the coefficients are actually complex numbers. This provides complete generality when used with the simulator's complex signal representation. When real coefficients are desired the imaginary parts are simply set to zero.

The complex signal representation may result in some ambiguity in the definition of filter types. A filter with a pass band symmetrical about zero frequency may be considered either a band-pass or a low-pass filter depending on point of view. Where a low-pass design technique has been used the design routine will ask for a cutoff frequency. The bandwidth of the filter when used as a band-pass filter centered on zero frequency will be twice this cutoff frequency.

Single-precision floating-point numbers (24-bit mantissa and 8-bit exponent) are used for all coefficients. This limited precision can lead to inaccuracies under certain conditions in IIR filters, and for this reason it is recommended that the response be tested before the filter is used. Methods for accomplishing this will be discussed later.

Normally, a filter is designed to have unity gain at its centre frequency. However, when it is used to filter a random variable that controls a gain factor, such as in the simulation of Rayleigh fading, it may be desirable to design a filter whose output power is equal to its

input power when its input is white noise. In this case it has a centre-fequency gain that depends on its bandwidth. In the filter designs provided this capability is provided as an alternative.

8.1.1 FIR Filters

The configuration of the FIR or transversal filter is shown in Figure 8.1. The blocks marked Z^{-1} are one-sample delays. Up to 256 complex coefficients (C_i) may be specified. The impulse response is given in sampled form by the series of coefficients, and the frequency response will be the Fourier transform of this series.

8.1.1.1 Simple Low-Pass Design

Two design methods are available to generate the coefficients automatically for the user. The first is a simple low-pass design that provides an exponential voltage rolloff with frequency beyond the cutoff frequency. This results in a linear response when plotted in dB on a linear frequency scale. Figure 8.2 shows the target response. The coefficients are real, giving a symmetrical amplitude response about zero frequency. This filter is referred to as a low-pass one in the simulation but as mentioned above can be considered a band-pass filter for complex signals. The user specifies the number of coefficients, the cutoff frequency, and the desired attenuation at half the sample rate. In this case the cutoff frequency is defined as the breakpoint in Figure 8.2 rather than as the 3-dB point.

The routine uses an inverse Fourier transform to approximate the desired shape. Of course this will only be an approximation, and the accuracy will depend on the number of coefficients and the rolloff rate. The greater this rate, the greater will be the number of coefficients needed to give a reasonable approximation. The routine warns the user if the values entered will give a poor approximation. As an example, Figure

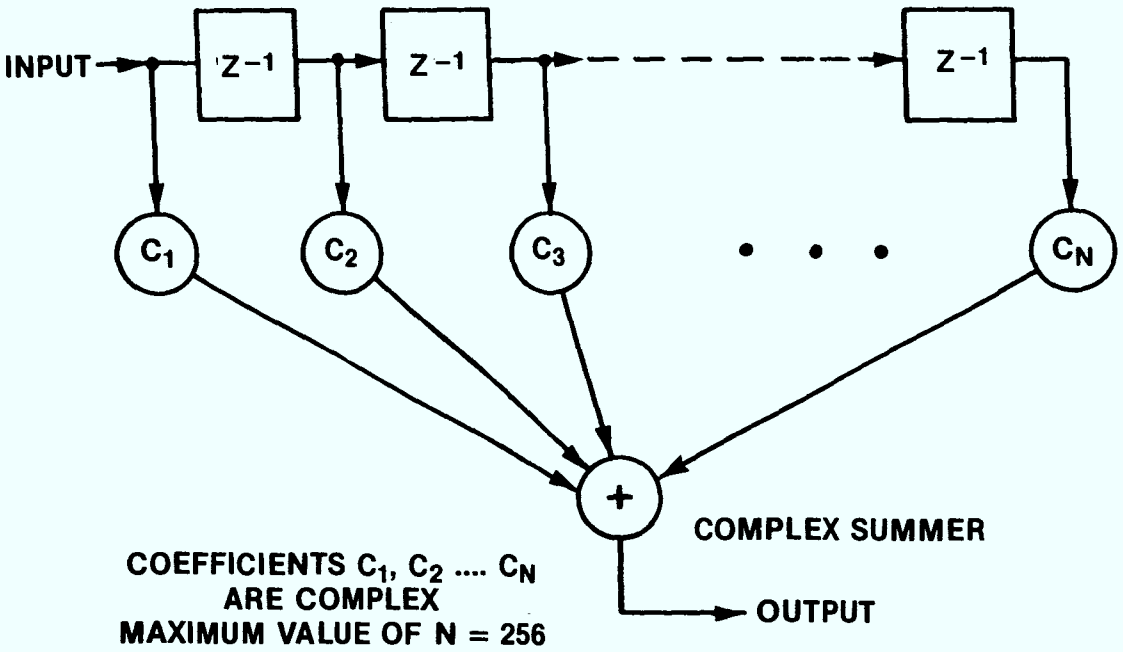
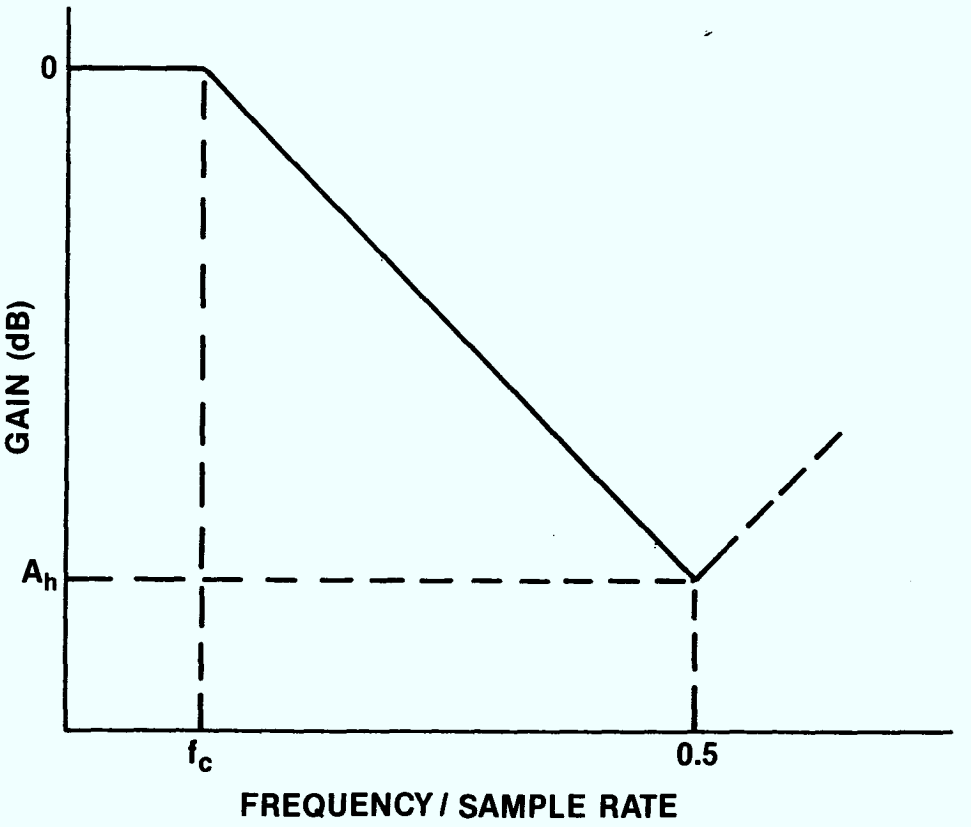


Figure 8.1 Finite Impulse-Response Filter



f_c = CUT-OFF FREQUENCY (0 dB)

A_h = GAIN AT HALF SAMPLE RATE

Figure 8.2 Target Frequency Response of Simple FIR Lowpass Filter

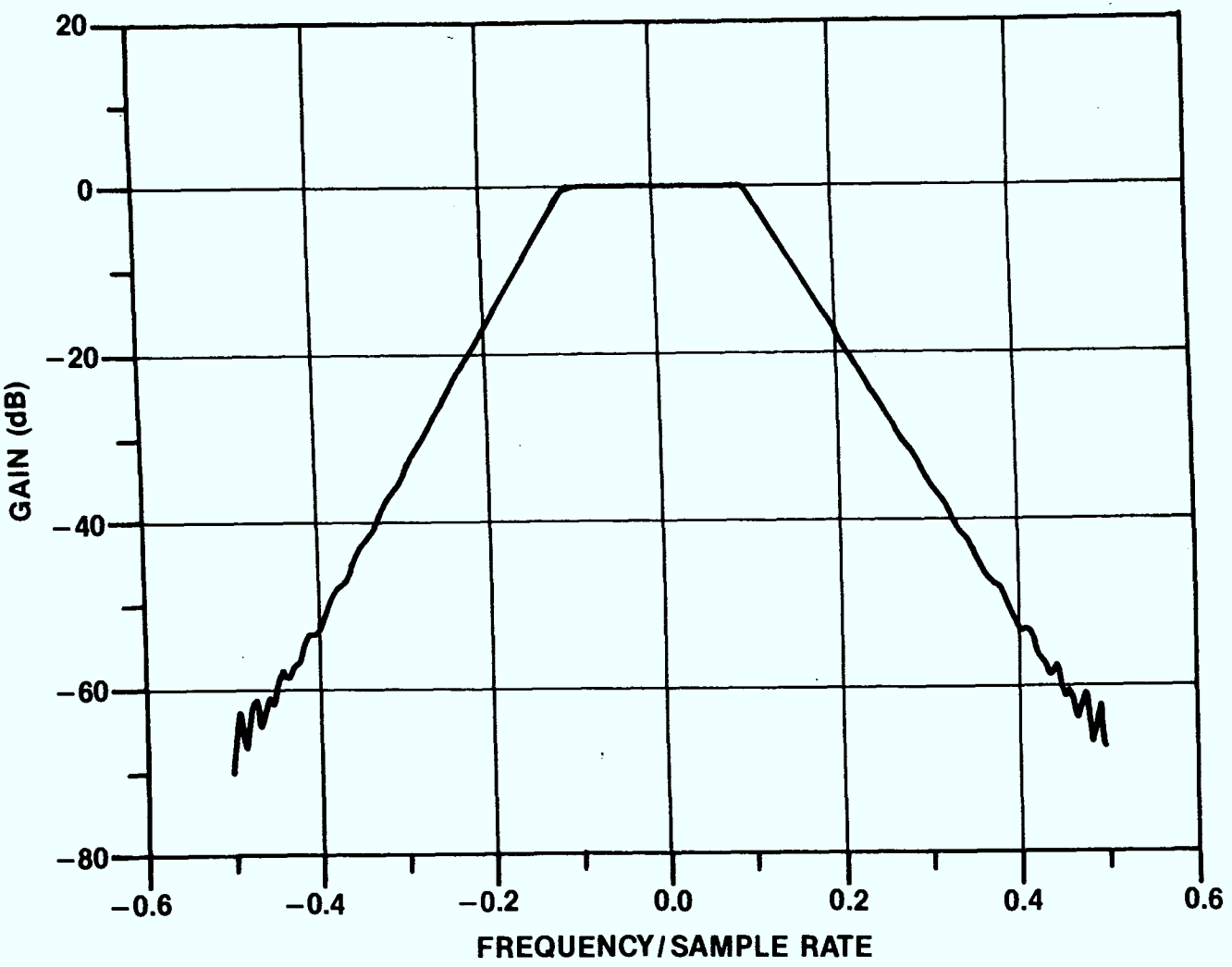


Figure 8.3 Response of Simple FIR Lowpass Filter when Rolloff Rate too high for No. of Coefficients (64 Coefficients)

8.3 shows the measured response for a 64-coefficient filter with cut off of 0.1 times the sample rate and an attenuation of 70 dB at the half-sample-rate point. The routine warned that the maximum attenuation for an accurate filter with this cutoff and number of coefficients was 51.2 dB. The ripples near the half sample rate are the result of the too-high attenuation. Since these may be quite acceptable in some circumstances the routine allows the user the choice of using the original specification or of changing it.

The impulse response is adjusted to put the peak at the middle of the response. This gives a signal delay of one-half the number of coefficients. When the number is odd the delay is half of one less than the number of coefficients. The routine informs the user of the actual delay.

8.1.1.2 Complex Band-Pass Design

The second FIR design method used is the window method in which the we start with the impulse response corresponding to a perfect rectangular band-pass shape and modify it with a window or weighting function that tapers the impulse-response sidelobes away from the peak. This tapering allows the impulse response to be represented with a reasonable number of coefficients without truncating it severely. Such truncation would lead to large ripples in the frequency domain. The tapering results in a decrease in the rolloff rate. A number of weighting functions are available for the user to choose from. These are: rectangular, triangular, Hanning, modified Hanning, Hamming, Kaiser, Blackman, and sine of sine of sine. It is recommended that the actual band-pass shape be examined by a test in the simulator before any operational simulation run.

8.1.1.3 User-Specified Coefficients

For special designs the user may compute the coefficients and enter them through the keyboard himself. Complex coefficients may be specified.

8.1.2 IIR Filters

IIR filters can be configured from up to 8 cascaded second-order sections of the form shown in Figure 8.4. The transfer function of a section is:

$$H(Z) = \frac{(1+C_3Z^{-1}+C_4Z^{-2})C_5}{1-C_1Z^{-1}-C_2Z^{-2}} \quad .$$

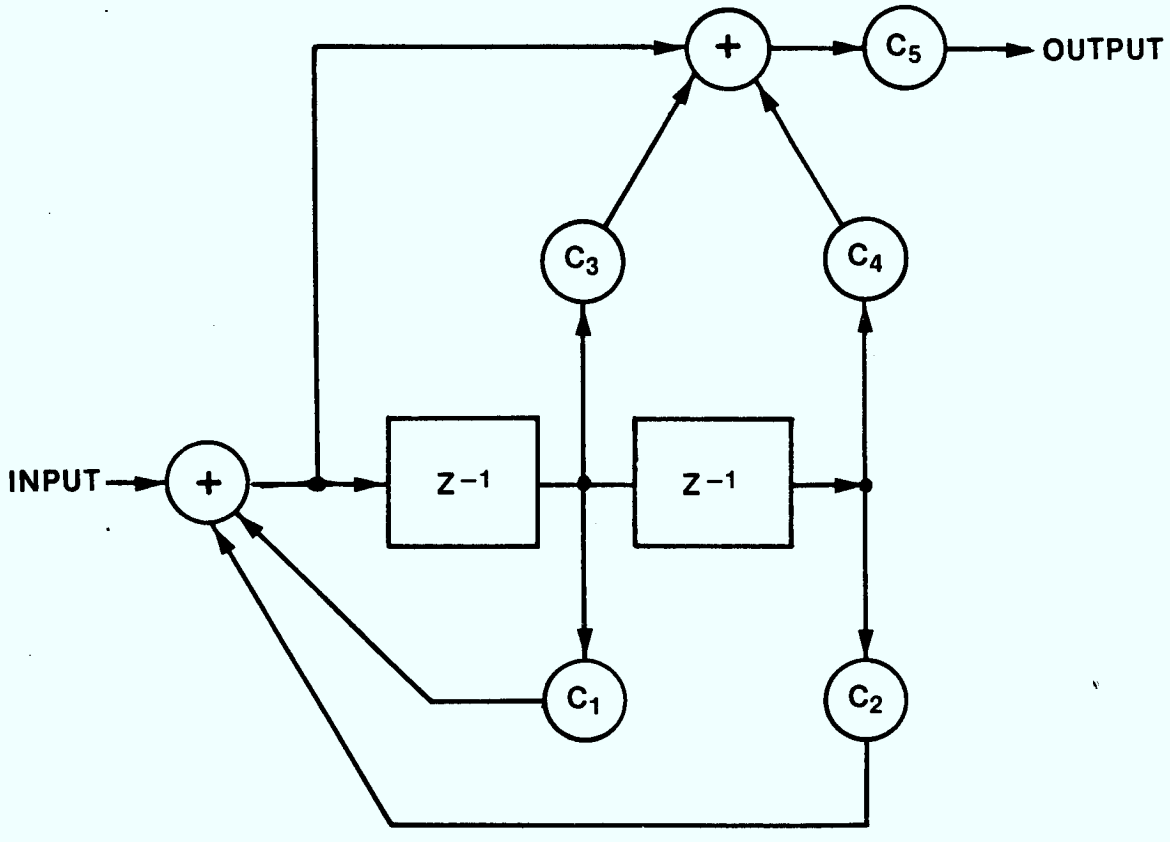
8.1.2.1 Butterworth and Chebyshev Designs

Design routines are available to design Butterworth and Chebyshev low-pass filters, with the coefficients automatically entered into the IIR filter. The inputs requested from the user are the bandwidth, the number of sections, whether the last section is a full or half section, and, in the case of the Chebyshev design, the pass-band ripple. In addition, the user may choose either unity centre-frequency gain or unity integrated-noise-power gain. The warning about the effect of limited precision in the number representation, mentioned earlier, is worth repeating here.

8.1.2.2 Resonator Design

A design routine is also provided for a simple resonator using a pair of complex-conjugate poles inside the unit circle at the resonant frequency. Zeros at 1 and -1 are also included. This gives a transfer function of:

$$H(Z) = \frac{1-Z^2}{1-2aZ+(a^2+b^2)Z^2}$$



COEFFICIENTS C_1, C_2, C_3, C_4 AND C_5 MAY BE COMPLEX

Figure 8.4 Second - Order Section of IIR Filter
(up to 8 Sections May Be Cascaded)

(up to 8 sections may be cascaded) for poles at a+jb and a-jb. The user enters the number of these sections desired, the resonant frequency, and the time constant of one section. This time constant will be equal to $1/(\pi W)$, where W is the 3-db bandwidth of one section. The gain coefficient C_5 is set to give unity gain at the resonant frequency. This is a real filter and therefore also has a resonant peak at minus the specified resonant frequency. The resonant frequency should be many bandwidths away from zero frequency to avoid distortion of the pass-band by the tail of the negative peak (folding about zero).

8.1.2.3 Narrowband Low-Pass Filter Design

There is a need for a very narrowband low-pass filter for the generation of Rayleigh fading with sufficiently low fading rates. This can be accomplished with a pair of poles on the real axis just inside the unit circle. A design routine has not yet been added to the simulator for this filter, but the coefficients can easily be determined from the following manual procedure.

For a 6-dB cutoff frequency of f_c, determine the distance of the poles from the unit circle as $e = 2\pi f_c$. The coefficients may then be computed, for small e, from:

$$C_1 = 2(1-e)$$

$$C_2 = (1-e)^2$$

$$C_3 = 0$$

$$C_4 = 0$$

$$C_5 = e^2 \qquad \text{for unity gain at zero frequency,}$$

$$= 2e^{3/2} \qquad \text{for unity integrated-noise-power gain.}$$

Identical sections may be cascaded to increase the cutoff rate. This will reduce the bandwidth, and e should be adjusted to compensate.

Also, in the case of the unity integrated-power-gain design, the value of C5 will have to be adjusted to give the correct overall power gain.

When e is less than about 0.001 the precision of the number representation is not sufficient to provide an accurate frequency response, and the peak can split into two. Considerably lower values can be used if they are selected so that the coefficient representation is close to the desired value; that is, they should be selected so that the difference between a single-precision and a double-precision representation is small relative to the resolution of the single-precision number. The value of e can be varied around the desired value, and the above test carried out in an off-line program to find suitable values close to the desired one. The frequency response should then be determined by a test on the resulting filter to verify that it is satisfactory. A useful value of e near 0.0005, found this way, is 0.00048846.

8.1.3 Testing the Frequency Response of a Filter

The frequency response of a filter can be determined by taking the Fourier Transform of its impulse response. An off-line routine has been developed to compute the transform and plot the result of a 512-point complex record of the impulse response. The impulse response is generated by using a data file consisting of a one followed by 511 zeros generated by DATEN and MODIFY as input to the filter in the HOPPER process. The resulting output is the impulse response and is written to a file using FILE. An off-line routine called FILSPTL can then be used to read the file, perform the transform, and plot the frequency response on a Tektronix 4014 graphics terminal.

8.2 Decimation

When the signal bandwidth is reduced, as when direct-sequence de-spreading is performed, it is desirable to reduce the sample rate to decrease the computing load. The sample rate need only be higher than twice the highest frequency magnitude in the signal spectrum. In practice "the highest frequency magnitude" may be interpreted as the frequency magnitude beyond which there is no significant energy. The definition of "signal" must include noise and interference, since under-sampled noise will be aliased into the desired signal band. The simulator provides a decimation routine for reducing the sample rate. The user is asked if he wishes to use decimation at various points in the receiver after filtering has been performed, but he may also elect to use it at any time between processes by calling the HOPPER process which contains some general-purpose utilities such as filtering and decimation. This process may be entered for the purpose of filtering and decimation without performing the hopping function. Decimation should only be performed after filtering to avoid under-sampling.

The decimation algorithm is simple. The user is asked for the desired decimation rate. This is the ratio of the sample rate before decimation to that after. For a decimation rate of N, only every Nth sample is retained with all others discarded. The first sample retained is number (N+1)/2 (truncated to an integer if necessary). As an example, if the decimation rate is specified as 10, then the samples retained are numbers 5,15,25, etc.

8.3 Saturating Amplifier or Limiter

In some cases amplifiers may be driven into saturation and become nonlinear. This is particularly true in the case of a power amplifier used to transmit multiple tones simultaneously. Such tones can add in phase at times to generate high peak voltages relative to the average voltage. For efficient use of the amplifier it is usually required to operate with high average power, not too far below the saturation level. In this case the

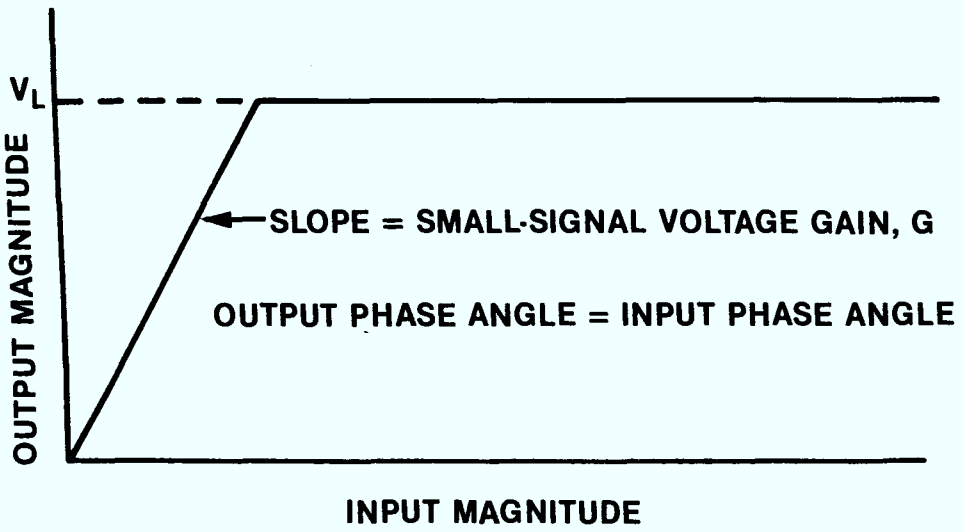
peaks can exceed the saturation level and be limited.

It may also be desired to limit large interfering peaks or to control signal levels in the receiver and so a limiting amplifier may be used there. To simulate these conditions a saturating amplifier is provided in the simulator. It consists of an amplifier with a user-specified gain, which is linear up to a user-specified threshold and has a fixed output magnitude for all input signals above this threshold. The phase of the complex signal is maintained by the amplifier. The characteristics of this amplifier are shown graphically in Figure 8.5.

The saturating amplifier is a general-purpose device, but has been included, for convenience, in the HOPPER process. If it is desired to use it as a limiter in the receiver it is necessary to perform in the RECVR process, on the first entry, only those functions that precede the limiter (usually at least noise and jamming addition), then enter the HOPPER process to do the limiting, and finally re-enter the RECVR process to perform the remaining functions.

9 BATCH-MODE OPERATION

If the batch mode of operation is selected a command file must be prepared in advance with the answers to all of the questions that the program will ask. This is difficult to do without first running the simulation from the terminal to find out what the questions are. We cannot easily make a list of the questions as a guide since the route taken through the simulation, and hence future questions, depends on the answers to the earlier questions. The best method of finding the questions is to perform the desired simulation with a reduced quantity of input data on a hard-copy terminal. This will record the questions along with your answers, and if the number of bits of input data is kept small, long waits during processing of the data will be avoided. A video terminal may be used, but in this case it will be necessary to write down the questions (or the chosen answers) by hand. One difference between the interactive and



G AND V_L SPECIFIED BY USER

LIMITS: $0 < G \leq 10^6$

$0 < V_L \leq 10^6$

Figure 8.5 Characteristics of Saturating Amplifier

batch mode that should be kept in mind is that the question on whether the Monitor facility is desired in the receiver is asked only in the former and not in the latter.

The user is advised to make the first batch run with a reduced number of points since this will minimize the waste of CPU time if an error has occurred in the batch command file. It can also allow the performance to be checked with the Display facility to determine if changes need to be made before committing a large amount of CPU time. Even crude estimates of bit error rates may be obtained sometimes with few samples to determine if there are gross errors in the simulated system. Another reason for using short runs at the beginning is to obtain an estimate of the running time of a longer run. Such estimates should take into account that a significant amount of time will be used in starting the program and in stopping it, regardless of the size of the simulation. This is about 40 seconds of CPU time with the current simulator and operating system (VMS 4.2).

When operating in the batch mode it is usually best to specify a file as the output device. If an output file is specified the batch log file will contain only the questions and menus from the program along with any error messages. If the terminal is specified as the output device the output will also go to the log file. The log file is specified in the SUBMIT command that starts the batch operation.

An example of a batch command file is included in Appendix D. This is for the simulation of a binary DPSK system through a medium consisting of two Rayleigh fading paths with small delay difference, and with CCIR type noise of V_d equal to 4.2 dB. The command file is written for the DEC VMS 4.2 operating system. The lines beginning with a "$" are commands. A "!" in a command line indicates that the text following it in quotation marks is a comment which is included to help in interpreting the command file. The lines not beginning with a "$" are the user responses to the questions asked by the program which begins after the RUN command. The WRITE commands cause the subsequent text to be copied to the output device, which, in the case of batch operation, is the batch log file. The SET

DEFAULT command tells the system which directory to run the program in. The parameter, P1, which specifies the directory, should be included in the VMS SUBMIT command that is used to submit this command file. More will be said about this later.

The RUN command starts the simulation program, MODEM, which is in directory VENIER.FREYSENG.MODEM. At this point the program takes control and begins asking the user for information. The first question relates to the mode of operation and is answered "BAT" to enter the batch mode. After this the question on output is answered "FILE" and the file heading and name are specified. The questions corresponding to the remaining responses may be viewed on the terminal by running the simulator in interactive mode, or they may be found in the batch log file generated when this command file was run in batch mode.

This log file is included in Appendix E. The two lines of information written to the log file by the commands in the command file can be found at the beginning. These are used to identify the particular experiment. They are followed by an indication of the simulation program name and version number, written by the simulation program, along with the date on which the run was performed and the maximum amount of data the current version can handle. After this we see the questions and information such as menus intended to help the user answer the questions. At the end of the log file there is a summary of accounting information which includes the amount of CPU time used.

The output of the simulation, was directed to file DPSK0123.DAT by one of the responses in the command file. This output is reproduced in Appendix F. It contains all of the information on the specification of the simulated system and the output of any analyses requested.

Appendix G lists a general-purpose command file for submitting a batch command file. It can be run by typing @BATCH. It asks for the name of the batch command file, the default directory (the directory in which the batch command file can be found), and the priority desired. This priority is the

priority in the batch queue, not the priority while running. It may be 1, 2, or 3, with 3 being highest priority.

10 ACKNOWLEDGEMENTS

The very difficult task of organizing the structure of this simulation program and generating an enormous amount of code was performed by Mr. Peter Freyseng of Datacap Ltd. The somewhat incomplete specifications provided to him at the beginning added to the difficulty, but did not deter him from his dedication to the development of well-structured and well-documented software. I am grateful for that dedication and for the results achieved.

The simulator was developed under a task sponsored by the Director of Maritime Combat Systems (DMCS-6) of the Department of National Defence.

11 REFERENCES

[1]. Davies, N. G., "Some Properties of Linear Recursive Sequences", Defence Research Telecommunications Establishment Report No. 1031, December, 1959.

[2]. Davenport, Wilbur B. Jr., and William L. Root, "An Introduction to the Theory of Random Signals and Noise", McGraw-Hill, 1958, pp 81-84.

[3]. Box, G. E. P., and M. E. Muller, "A note on the generation of random normal deviates", Annals of Mathematical Statistics, Vol. 29, pp 610-611.

[4]. CCIR (International Radio Consultative Committee), "World Distribution and Characteristics of Atmospheric Radio Noise",CCIR Report 322, Figure 27, International Telecommunications Union, Geneva, Switzerland, 1964.

[5]. Akima, Hiroshi, "A Method of Numerical Representation for the Amplitude-Probability Distribution of Atmospheric Radio Noise", Telecommunications Research and Engineering Report 27, Institute for Telecommunication Sciences, Office of Telecommunications, U.S. Department of Commerce, 1972.

[6]. Brigham, E. Oran, "The Fast Fourier Transform", Prentice-Hall, 1974.

[7]. Lin, Shu, and Daniel J. Costello, Jr., "Error Control Coding: Fundamentals and Applications", Prentice-Hall, 1983.

[8]. Holmes, Jack K., "Coherent Spread Spectrum Systems", Wiley-Interscience, John Wiley & Sons, 1982.

[9]. Simon, Marvin K., Jim K. Omura, Robert A. Scholtz, and Barry K. Levitt, "Spread Spectrum Communications", Volumes I, II, and III, Computer Science Press, 1985.

[10]. Proakis, John G., "Digital Communications", McGraw-Hill, 1983, Figure 4.2.20, p 176.

APPENDIX A

CALCULATION OF NOISE VOLTAGE REQUIRED FOR A GIVEN VALUE OF E_b/N_o

APPENDIX A

CALCULATION OF NOISE VOLTAGE REQUIRED FOR A GIVEN VALUE OF E_b/N_o

Let N_s = the number of samples per modulation symbol,
(In the case of direct-sequence or MCSK modulation the modulation symbol is an element of the code.)

N_c = the number of spreading code elements (modulation symbols) per data symbol,

f_d = the data rate in bits per second,

N_b = the number of bits per data symbol,

f_s = the sample rate ($= N_s N_c f_d$),

V_s = the peak signal voltage at the receiver input,

A_s = the energy gain factor for shaping or transitions,

V_n = the rms noise voltage at the receiver input,

E_b = the energy per bit of the signal in joules,

E_c = the energy per modulation symbol,

N_o = the noise power density in watts per Hz.

Then $$E_b = \frac{V_s^2 \, A_s}{N_b \, f_d} \qquad \text{(A1)}$$

$$= \frac{V_s^2 A_s N_s N_c}{N_b \, f_s} \; . \qquad \text{(A2)}$$

$$N_o = \frac{V_n^2}{f_s} \qquad \text{(A3)}$$

and $$\frac{E_b}{N_o} = \frac{V_s^2 A_s N_s N_c}{V_n^2 \, N_b} \; . \qquad \text{(A4)}$$

From (A4) we get:

$$\frac{V_n}{V_s} = \sqrt{\frac{A_s N_s N_c N_o}{N_b E_b}} \; . \qquad \text{(A5)}$$

(A5) gives the ratio of noise voltage to peak signal voltage required to give a specified ratio of bit energy to noise power density.

The program displays the computed values of modulation symbol energy and noise power density. This allows the user to determine the ratio to check that it is correct. However, he must first determine the bit energy from the modulation symbol energy from:

$$E_b = \frac{E_c N_c}{N_b} \qquad \text{(A6)}$$

APPENDIX B

EXCISION TECHNIQUES FOR DIRECT-SEOUENCE SPREAD-SPECTRUM SYSTEMS

Note: This paper first appeared as Communications Research Centre Technical Memorandum MC/TM030/84, 14 May 1984.

TABLE OF CONTENTS

EXCISION TECHNIQUES FOR DIRECT SEQUENCE SPREAD SPECTRUM SYSTEMS

1. INTRODUCTION

The purpose of the investigation outlined in this Memorandum is to permit specifications to be written for an excision routine which will be used in the DMC Spread Spectrum Simulation Facility. It is not intended as an intensive study of the problem. The routine will have user-selectable parameters which will allow further study of the excision technique when completed. As well as providing the general design information, the present investigation should provide the basis for an intelligent use of the routine in such a study.

A direct-sequence spread-spectrum system provides good discrimination against a narrowband interfering signal. The despreading process in the receiver, which reduces the bandwidth of the communication signal by removing the spreading modulation, will cause the narrowband interference to be spread in bandwidth, and therefore only a small part of its energy will fall into the filter matched to the data modulation. This will be true whenever the bandwidth of the interference is less than the spread-spectrum bandwidth. When the interference has a bandwidth less than or equal to the data bandwidth, the spread-spectrum system will provide an improvement approximately equal to the processing gain, that is to the ratio of spread bandwidth to data bandwidth. However, even greater improvement is possible if the narrowband nature of the interference is properly exploited.

This can be accomplished by removing the narrowband interference with a notch filter before despreading. If the interfering signal is much less in bandwidth than the direct-sequence signal, only a little of the desired signal is lost in the process. Some estimate of the spectrum is necessary to determine where to place the notch. One method of doing this is to perform a spectral analysis or Fourier transform of the incoming signal

and place a notch where the spectrum exceeds some threshold level. It is desired that this be done automatically and some problems will arise in determining the proper threshold and notch width to use. The notch of a digital filter can easily be set to the desired frequency by setting the filter coefficients.

A solution which avoids some of these problems is to use a whitening filter which automatically adapts to the input signal in such a way as to try to whiten the output spectrum, that is, to suppress any narrow peaks in the spectrum. This can be accomplished by the technique of linear prediction which is more easily described in the time domain.

A narrowband signal will have an autocorrelation function whose peak is very broad. That is, there is strong correlation between values well separated in time. This makes the signal predictable from its history. A broadband spectrum such as the direct-sequence signal has a narrow autocorrelation peak and is therefore unpredictable over a time greater than a chip duration. Thus linear prediction will predict the narrowband interference but not the desired direct-sequence signal. If the value predicted for the next sample (in the future) is subtracted from the next sample when it arrives, then if the prediction is accurate the narrowband signal will be cancelled while the communication signal will be essentially unaffected.

Some of the details of this technique are presented in a paper by Hsu and Giordano [1]. Some of the results will be summarized here for the Wiener algorithm. Another algorithm, the maximum-entropy or Burg algorithm is also described in the above paper but will not be discussed here. This may be the subject of a later Technical Memorandum.

2. PREDICTION FILTER THEORY

The Wiener prediction filter takes the form of a finite impulse response (FIR) or transversal filter. The estimate of the k^{th} sample is computed from N previous input samples X_{k-N} to X_{k-1} by,

$$X_k = \sum_{n=1}^{N} b_n X_{k-n} \qquad (1)$$

The problem is to determine the coefficients $\{b_n\}$ of this FIR filter. These are determined from a least-mean-square criterion which leads to the set of equations [2]

$$\sum_{n=1}^{N} b_n r_{k-n} = r_k \ , \quad k = 1, 2, \ldots .N \qquad (2)$$

where $\{r_k\}$ are the autocorrelation coefficients of the input signal. These can be estimated in the standard manner from the signal, but it must be remembered that they are only estimates based on a limited number of samples.

The equations in (2) above can be written in matrix form as:

$$R \cdot B = R_v \qquad (3)$$

where is the NxN autocorrelation matrix,

B is the vector of filter coefficients, and

R_v is the vector of autocorrelation coefficients r_1, r_2, $\ldots . r_N$

The autocorrelation matrix as determined from (2) will have the form

$$R_N = \begin{bmatrix} r_o & r_{-1} & r_{-2} & \cdots & r_{-N+1} \\ r_1 & r_o & r_{-1} & \cdots & r_{-N+2} \\ r_2 & r_1 & r_o & \cdots & r_{-N+3} \\ \vdots & \vdots & & & \vdots \\ r_{N-1} & r_{N-2} & \cdots & & r_o \end{bmatrix}$$

For an autocorrelation function $r_{-i} = r^*_i$ (r^*_i = complex conjugate of r_i). The matrix R_N is therefore Hermitian ($a_{ij} = a^*_{ji}$) and Toeplitz (equal values on any diagonal). This makes possible an iterative solution known as the Levinson-Durbin Algorithm which is described in [3]. This algorithm is reproduced below with different notation and a sign difference to account for the different application (we are computing the coefficients of the prediction filter whose output will be subtracted from the next input). The notation $b_{i,j}$ refers to the i^{th} coefficient in the j^{th} iteration.

$b_{1,1}$ is computed first from

$$b_{1,1} = \frac{r_1}{r_o}$$

Each additional coefficient is computed in turn and previous coefficients are updated on each interation.

On the N^{th} iteration we compute,

$$A = r_N - \sum_{k=1}^{N-1} b_{k,N-1}\, r_{N-k}$$

$$B = r_o \prod_{k=1}^{N-1} \left(1 - \left| b_{k,k} \right|^2\right)$$

From these we compute the N^{th} coefficient,

$$b_{N,N} = A/B$$

and then update all previous coefficients by,

$$b_{n,N} = b_{n,N-1} - b_{N,N}\, b^*_{N-n,N-1}, \text{ for } n = 1,2 \ldots N-1$$

After N iterations we compute N coefficients. This requires only order N^2 operations as opposed to order N^3 operations required for a general matrix inversion.

When the coefficients $\{b_n\}$ have been computed they are used in a transversal filter indicated by (1) to predict the next value of the input, and this value is then subtracted from the next input sample. The process is illustrated in Figure 1*.

In this configuration the filter output is subtracted from the next sample simply by reversing the signs of all the $\{b_n\}$ and adding in the next sample with unit multiplier. The extra delay at the beginning takes care of the fact that the prediction is made for the next sample time after the time when the prediction is made.

---

* All figures in this appendix have been given the prefix "B". However in the text they are referred to without the prefix.

3. DESIGN CONSIDERATIONS

It should be recalled that the excision with minimum distortion to the desired signal is based on the assumption that the desired signal is uncorrelated from sample to sample. This is only true if there is a maximum of one sample per chip. Normally we would like to represent the signal with more than one sample per chip to prevent aliasing. This can be accommodated by computing the autocorrelation function only at delays separated by M samples, where M is the number of samples per chip, and spacing the filter taps every M samples. This means that the delays are made equal to M sample periods. This maintains the good representation of the signal at the output - that is, it does not introduce aliasing of the signal - but it does cause a periodic repetition of the filter frequency response at intervals of the inverse of the tap spacing. For example if there are 4 samples per chip we must choose a tap spacing of four samples. This will give an effective sample rate for the filter of 0.25 times the true sample rate. Therefore an interfering signal at 0.1 times the sample rate will produce a notch not only at 0.1 but at 0.25 + 0.1 = 0.35, at 0.5 + 0.1 = 0.6 or -0.4 and at 0.75 + 0.1 = 0.85 or -0.15.

The notches at higher frequencies will cause no problem since little signal energy exists there (above the rate of the taps) but the one at -0.15 will fall on the signal spectrum and will notch out some of it. Thus an interferer can have the effect of two interferers. Even worse, if an interferer falls above the tap sample rate say at 0.3 in the above example, it would cause very little problem by itself, but the filter will generate a notch at 0.3 - 0.25 = 0.05 as well as at -0.25 + 0.05 = -0.20. Thus the use of the adaptive filter may actually degrade the performance when the interference is outside the normal signal band. To minimize this effect we should low-pass filter to as low a frequency as possible, without seriously distorting the signal, before the adaptive filter.

The autocorrelation function can be estimated over a block of M input samples from

$$r_j = \frac{1}{M-j} \sum_{i=1}^{M-j} x_i x_{i+j}^* \quad , \; j = 0,1....N$$

Coefficients of higher delay will not be as precise since they will have fewer values in the sum. This can be avoided by making the upper limit of summation M-J, where J is the index of the highest delay coefficient desired. In this case coefficients for smaller values of j will not make use of all the in-formation available, but all summations will contain the same number of products. But normally N will be much less than M and this effect will be small. If N is not much smaller than M then a more efficient means of computing the autocorrelation function is to take the inverse Fourier Transform of the Power Spectrum. This involves two Fourier Transforms but if the number of coefficients to be found is a significant fraction of the block size the efficiency of the Fast Fourier Transform routine makes this the preferred method. However, care must be taken to avoid the cyclic nature of the autocorrelation function when generated in this manner. One method of accomplishing this is to add zeros to the end of the sample block before performing the Fourier Transform. The warning about the precision of the higher delay coefficients is even more important in this case since this method would only be used when the number of coefficients to be used is a significant fraction of the block length M.

One question of importance in the autocorrelation estimation problem is the size of the block over which the estimate is performed. The larger the block the better will be the estimate, providing the data are stationary which is rarely the case. Therefore some compromise is required so that the filter can adapt as the conditions change but still make use of a large enough block to permit a good estimate. Another factor is the requirement for storage of data. Ideally the data to be filtered should be stored until the filter has been computed, then put through the filter. That is, there should be a storage capability of at least the size of the block used. If no storage is provided the filter will have to operate on data following the data to which it has adapted,

making it even more vulnerable to non-stationarity. In the routine for the Spread-Spectrum Simulation Facility the block length and the storage capability should be made variable - to be set by the user - so that these problems can be further investigated.

It seems reasonable that the greater the number of coefficients in the filter, the better the filter will be able to remove the interference, particularly when the interferer has significant bandwidth or comprises a number of separate narrowband signals. Intuitive reasoning indicates that the notch width for a single interferer will be proportional to the inverse of the number of samples spanned by the filter, that is, to the inverse of the number of coefficients times the number of samples between taps. Thus to minimize the distortion to the desired signal we should use as many coefficients as possible. We would also expect that the greater the number of narrowband interferers the more coefficients we will need. But coefficients are expensive since the number of operations required in computing the coefficients is proportional to the square of the number of coefficients. For this reason it is important to determine how many coefficients are required for the desired performance. This will depend strongly on the characteristics of the interference and therefore a good estimate of the threat is important. In the Simulation routine it is essential that the number of coefficients be a user-selectable parameter. A maximum value of 64 should cover any feasible practical system.

4. SIMULATION EXPERIMENTS

The algorithm for computing the excision filter coefficients was implemented in Fortran on a VAX-11/750 computer. This routine computes the autocorrelation function estimate for a block of up to 512 complex input samples and uses this to compute from one to 16 coefficients with tap spacings of from one to 16 samples. All these parameters are user-selectable at run time.

Input samples are generated from the data-generation routines of the partially completed Spread-Spectrum Simulation Facility. The wideband signal representing the direct-sequence signal was simulated by various signals including white Gaussian noise, impulses and maximal-length sequences, depending on the purpose of the test. Narrowband interference was simulated by one or more complex sine waves or in some cases by a slowly swept complex sine wave.

The resulting coefficients were output to a file from which they were read by a general-purpose FIR filter routine which performed the actual excision process on the simulated input signal, or on an impulse when the impulse response was desired. The output of the FIR filter routine could be taken as a time series or converted to the frequency domain by a Fast Fourier Transform routine. The output domain, number of coefficients and tap spacing are user-selectable parameters. The input sample block is fixed at 512 samples, but a number of samples equal to the impuse response duration are removed from the beginning of the output time series, and only the last 256 output samples are used in the FFT routine. This is to avoid the effects of the start-up transient. This transient would not be a problem in practice since the filter state would be retained from one input block to the next. Only the first block would generate a transient from the zero state. These routines were used to test the adaptive algorithm and to provide some understanding of how the parameters affect the result.

4.1 Excision Filter Demonstration

The first experiment was intended to demonstrate the operation of the excision process under the simplest conditions. The direct-sequence signal was simulated by white Gaussian noise and a single complex sine wave was added to it to simulate the interference. The Gaussian samples were generated by the Box and Muller method [4]. The amplitude of the sine wave was five times that of the rms noise voltage and its frequency was 0.14 times the sample rate. (Henceforth we shall specify all frequencies as a number without units representing the ratio of frequency

to sample rate). The input signal is shown in Figure 2 and its spectrum in Figure 3. The spectrum is generated from a 512-point FFT of the input signal and the magnitude of the complex spectrum is plotted. The magnitude is the Fourier transform result without division by the number of points. For an N-point transform the sine-wave component will have an amplitude of N times its time-domain amplitude, while the noise will have an rms value of only $\sqrt{N}$ times its time-domain component. Thus the 5:1 time-domain ratio of sine amplitude to noise amplitude becomes a 113:1 ratio in the frequency domain. It should be noted that all output spectra are from 256-point transforms and this will result in a $\sqrt{2}$ reduction of the sine wave component relative to the 512-point input spectrum.

Eight excision filter coefficients with one-sample tap spacing were computed from the 512-sample input block and these coefficients were used to filter the same input block. The output of the FIR filter is shown in the time and frequency domains in Figures 4 and 5 respectively. It is clear that the sine-wave component has been effectively removed while the noise remains. A notch can be seen in the noise spectrum where the sine-wave component was.

4.2 Effect of the Number of Coefficients on Notch Characteristics

To investigate the characteristics of this notch and its dependence on the number of coefficients the following experiment was performed. A single impulse of unit amplitude at sample number 257 was used to simulate the wideband signal. This has the advantage of having zero autocorrelation for all non-zero delays and so does not affect the coefficient computation as the noise signal does. This permits the coefficients to be based entirely on the sine-wave component which was again at a frequency of 0.14 but had an amplitude of 0.1. This gives it an energy of about 5 times that of the impulse in the 512-sample block. FIR filter coefficients were computed using a 16-coefficient and a 4-coefficient calculation. The impulse response of the resulting filters were generated by using the impulse signal, without the sine-wave, as

input. The frequency-domain results (the frequency response of the filter) are plotted in Figures 6 and 8 and the time-domain impulse responses are plotted in Figures 7 and 9 for the 16- and 4-coefficient cases respectively. The ripples in the frequency response are characteristic of finite impulse responses. It is the notch itself which is of main interest. We see that the notch width is much greater for the 4-coefficient case as expected. It was speculated earlier that the notch width would be proportional to the inverse of the number of samples spanned by the filter (in this case the number of coefficients) or 0.0625 and 0.25 for 16 and 4 coefficients respectively. The width of the notch depends on how it is defined but if we take the point where the attenuation is 3 dB we find from Figures 6 and 8 that the width is just under one-half the inverse of the number of samples spanned.

The impulse responses in Figures 7 and 9 show that the impulse is well preserved (as it must be with a unity coefficient), but that the filter "rings" for a time equal to the filter length after the impulse. Since it is a finite impulse response filter it could not have an impulse response outside this region. The magnitude of the "ringing" is greater for the shorter filter but lasts a shorter time. This distortion will cause some degradation in the signal. The wider the frequency-domain notch the more degradation there will be since the greater will be the loss of signal spectrum. On the other hand when the interfering signal is not very stable in frequency and the coefficients are not computed very often, the wider notch (fewer coefficients) may actually provide better performance since the interference may change frequency by a greater amount without going outside a given attenuation level.

4.3 Autocorrelation Estimate

Another parameter of importance is the number of samples used in the autocorrelation estimate. Although the wideband signal may have zero autocorrelation for delays greater than the tap spacing, the estimate of that autocorrelation will not be zero when based on a finite number of

samples. For a noise-like signal we would expect the estimate to decrease in proportion to $1/\sqrt{N}$ where N is the number of products used in the correlation estimate. The number of products possible is equal to N-k where k is the delay in samples for that coefficient. To make this number independent of delay we took k_m, the largest k required (equal to the number of filter coefficients), and used $N-k_m$ products for all delays. That is, for shorter delays we did not use some of the later samples in the block.

Since it is only the wideband signal that takes part in the correlation estimate error, only a noise signal was used as input for the filter coefficient computation. Coefficients were generated for two different input block lengths and the resulting FIR filters were tested with an impulse signal. The results for 16 coefficients are shown in Figures 10, 11, 12 and 13.

Figure 10 shows the frequency response for 496 correlation products (block of 512 samples). If the correlation were perfect the magnitude should be unity for all frequencies. The fluctuations shown are a result of the estimation errors. We see that the rms variations are about 10 percent. In Figure 12 where only 124 correlation products were used the variation is about double this. Since variations from the all-pass situation will distort the wideband signal it is evident that a larger number of samples gives a better filter. Figures 11 and 13 show the time-domain versions of the outputs for 496 and 124 samples respectively. They verify that the latter case produces greater signal distortion.

Figure 14 indicates the interaction between the number of correlation products and the number of coefficients. It is the frequency response for the 4-coefficient filter generated from an autocorrelation estimate using 496 products. We see that the variations have been reduced from those of Figure 10. This indicates that the greater the number of coefficients, the greater must be the block size for a given mean squared error in the frequency response. The following analysis confirms this.

Suppose there are N products in the correlation estimate and M coefficients. If the input signal is noise-like we would expect each coefficient to have an rms error of about $1/\sqrt{N}$. Now the frequency response is simply the Fourier Transform of the coefficients and if the coefficients are random and zero mean (as we expect for a noise-like input) then each element in the frequency domain will result from an incoherent addition of M random values, each with standard deviation $1/\sqrt{N}$ (we will ignore for the moment the additional unity coefficient which is always present - see Figure 1). Thus the standard deviation in the frequency domain should be $\sqrt{M}/\sqrt{N}$, for the unnormalized definition of the Fourier Transform. Now the unity coefficient that we have ignored is a unit impulse and results in unity for each element in the frequency domain. The other M coefficients will produce a random fluctuation of $\sqrt{M}/\sqrt{N}$ about this unit mean value. Thus the fractional error should be approximately

$$E_f = \sqrt{M}/\sqrt{N}$$

That is, the error is proportional to the square root of the number of coefficients and inversely proportional to the square root of the number of products in the correlation estimate. We can check this against Figures 10, 12 and 14. For Figure 10, M = 16, N = 496 giving an rms error estimate of $\sqrt{16}/\sqrt{496} = 0.18$. For Figure 12 it should be twice this (since N reduced by a factor of 4) or 0.36. Finally for Figure 14 the rms error should be about $\sqrt{4}/\sqrt{496} = 0.09$. We see that these values agree reasonably well with a simple "eyeball" estimate of the rms values from the above figures. It does not appear justified to do a more quantitative estimate from these figures since the number of independent samples is not sufficient to give an accurate estimate.

4.4 Multiple Narrowband Signal Interference

The performance of the excision filter for multiple narrowband signals was investigated by combining four sine-wave signals with Gaussian noise and using this to generate the coefficients. The resulting filter

was tested for frequency response as well as for response to the above signal. Two cases were tried: in one the tones were spread widely across the sampling bandwidth while in the other they were closely spaced over a very small percentage of this band.

The spectrum of the widely-separated-tone-input case is shown in Figure 15. The tone frequencies were -0.3, -0.1, 0.05 and 0.15. All tones had unit amplitude in the time domain as did the noise. The differences in amplitude of the different tones in Figure 15 are caused mainly by the sampled nature of the plotted spectrum. The plot routine connects points by straight lines and the peaks on the plot will depend on where the samples fall relative to the actual peaks. In addition the noise and interaction of the finite duration tones will play a smaller part. Figure 16 shows the frequency response of the resulting filter with 16 coefficients. The full 512-sample block (496 products) was used in the coefficient computation. Four notches can be seen at the tone frequencies. Other fluctuations in the response are a combination of the finite-duration response and the finite estimation time for the noise autocorrelation function. The spectrum of the output of this filter with the signal of Figure 15 as input is shown in Figure 17. It is evident that the 16 coefficients have been effective in suppressing the tones.

When the number of coefficients was reduced to six the frequency response of the filter changed to that seen in Figure 18. The notches have widened and the two for the highest two tones have almost merged. This means much greater degradation of the wideband signal as seen in Figure 19 which is the spectrum of the output with the signal of Figure 15 as input. Also, the tones are not well suppressed. In this case we have only 1.5 coefficients per interfering tone and since these coefficients represent degrees of freedom we might expect relatively poor performance. Since the notch width is on the order of the inverse of the number of coefficients, then for widely separated tones, one coefficient per tone would be expected to eliminate a major part of the wideband signal spectrum. A minimum of two coefficients per interfering tone would seem to be a reasonable requirement under these circumstances.

The spectrum of the input signal for the closely spaced case is shown in Figure 20. The tone frequencies are 0.07, 0.08, 0.09 and 0.10 and all time-domain amplitudes including the noise rms are unity. The frequency response for the 16-coefficient filter generated from the input of Figure 20 is displayed in Figure 21. In this case all the notches have merged to give a single wider notch encompassing all of the tones. In Figure 22 we see the spectrum of the filter output when the input is the signal of Figure 20. The four tones are reduced to the level of the noise and the noise spectrum is not much affected. The frequency response when only four coefficients are used is shown in Figure 23, and the spectrum of the output when the input is the signal of Figure 20 is shown in Figure 24. We see that although the tone suppression is not quite as good as in the 16-coefficient case, the noise spectrum is not badly distorted, even though there is only one coefficient per tone. The reason is that the coefficients produce a single notch at the frequency of the grouped tones, rather than producing four separate notches as in the earlier case. Thus fewer coefficients are required than in the widely-spaced case for the same performance.

4.5 Swept Frequency Interference

A similar situation occurs when the interference is a single tone but is swept slowly in frequency. This gives it a wider bandwidth, roughly equal to the range of frequencies swept through, providing this range is significantly greater than the inverse of the sweep time. The spectrum of such an input signal is shown in Figure 25. This signal was swept linearly from 0.10 to 0.13 starting at the 21st sample and ending at the 495th sample. The signal had unit amplitude in this period and had zero amplitude for all other samples. Added to this was white Gaussian noise of rms amplitude equal to 0.2 to represent the desired wideband signal. The swept bandwidth of the interference was 0.03 which was much greater than the inverse of the time duration which was 1/475. Thus it satisfies the criterion mentioned above for slow sweep, and this is verified by Figure 25 which indicates the rectangular spectrum of the swept signal over the expected range.

In Figure 26 we see that the frequency response of the resulting 16-coefficient filter has a notch matching the interference spectrum in position and width. The spectrum of the output of the filter for the input of Figure 25 is seen in Figure 27. Some of the interference is still present but is reduced to about the level of the noise signal. The wideband signal spectrum is not seriously degraded for this case of 16 coefficients. The impulse response of Figure 28 verifies this although some "ringing" of the filter does occur.

When only four coefficients are used the frequency response of Figure 29 results and the frequency-domain response to the input of Figure 25 seen in Figure 30 shows a higher leak-through of the interference, and greater distortion (attenuation of the wideband signal near the interference) of the wideband signal spectrum. However, the impulse response in Figure 31 indicates only moderate distortion. Thus it appears that even as few as four coefficients are reasonably effective against a swept signal occupying three percent of the bandwidth.

In the above the full block of input signal was used to generate the coefficients. When only part of the input is used we would not expect such good results since the interference is not stationary and the filter can only be adapted to the conditions existing during the coefficient computation. This was illustrated by using only the first half of the input block for the coefficient generator. The frequency response of the resulting 16-coefficient filter is shown in Figure 32. The notch does not extend to as high a frequency as it does in Figure 26 since the higher frequencies do not occur until the second half of the input signal. In the frequency-domain response to the signal shown in Figure 33 we see serious leak-through of the interference at the higher frequencies. The time-domain response to the signal as seen in Figure 34 illustrates the problem more clearly. The interference is well suppressed over the first half of the output where the coefficients were computed, but as the frequency increases in the second half it moves out of the notch and we see the build-up in amplitude as the frequency increases. This illustrates the advantage of using the same block of input for coefficient

generation and as filter input. This requires delay or storage of the signal since the first sample cannot be applied to the filter until the entire block has been used for the computation. While this would be a serious problem in an analogue system, it would be much less so in a digital one. However, since the excision process must be carried out in the wideband part of the receiver before despreading, the use of digital techniques may be quite expensive.

4.6 Multi-Sample Tap Spacing

Another question that deserves investigation is the effect of multi-sample tap spacing as required when there is more than one sample per direct-sequence chip. The first test was intended to demonstrate the effect described earlier in the section on Design Considerations. The parameters used in the examples there were tested using an impulse as the desired signal component and sine waves at frequencies of 0.1 and 0.3 (two tests, one with each frequency) for the coefficient generation. A tap spacing of four samples was used as in the example. In the filter coefficient generation 16 coefficients were computed. The frequency responses generated by an impulse input are shown in Figures 35 and 36 for interfering frequencies of 0.1 and 0.3 respectively. In the first case the notch is repeated at frequency intervals of 0.25 (inverse of the tap spacing) from the desired response at 0.1 as expected. The negative frequency values can be computed either by subtracting intervals of 0.25 or by adding intervals of 0.25 and subtracting 1.0 when the value is above 0.5.

In the second case, the desired notch is above the tap sampling frequency of 0.25, leading to a positive frequency notch below the desired one at 0.05 as seen in Figure 36. The number of notches will always equal the tap spacing. The impulse response of the filter of Figure 36 is shown in Figure 37. The ringing now extends for a longer period than for the single-sample tap spacing case since now the impulse response has a length equal to the number of taps times the tap spacing - in this case 16 x 4 = 64.

4.7 Demonstration of Excision Followed by Direct Sequence Matched Filter

The effect of the excision filter on an actual direct-sequence signal with multiple sample chips was investigated using a 31-element maximal-length sequence (M-sequence). This sequence used eight samples per chip and was repeated to produce 62 elements comprising 496 samples. The unit amplitude sequence was added to a complex sine wave at a frequency of 0.02 and white Gaussian noise with unit variance. In this test the noise was intended to represent undesired noise, and not the wideband signal. Figure 38 shows the spectrum of the combined signal. The M-sequence is barely discernible in the noise over the range of -0.1 to +0.1 and the narrowband interference is seen near the centre of this spectrum. Figure 39 shows the time-domain magnitude of the output of a filter matched to the M-sequence (no excision filter was used). The triangular peak is the desired matched-filter response (the M-sequence was delayed to put the peak near the centre). The remainder results from the combination of noise and sine-wave interference (since the magnitude is plotted, the complex sinewave would appear as a constant level). The processing gain of the matched filter brings the desired peak above the noise and stronger interference. Note that the vertical scale starts at 100 rather than zero.

The input signal of Figure 38 was then used to generate excision filter coefficients and the original input signal was applied to this filter. Eight coefficients were generated using a tap spacing of 8 samples and 448 products in the correlation estimate (the largest number possible for a block of 512 with these parameters). The output spectrum is shown in Figure 40. The sine-wave interference has been removed but the noise remains since it is wideband. The output of the excision filter was then applied to the matched filter which has a processing gain against the noise. The result is seen in Figure 41 which shows the magnitude of the time-domain output. This output is clearly superior to that in Figure 39. The reason is that the excision filter has provided the extra gain against the sine-wave interference in addition to the gain of the matched

filter. The well-defined triangular peak rising well above the residual level indicates that the excision filter has not seriously degraded the M-sequence signal. Some loss is evident when comparing the peak levels of Figures 39 and 41, but the end result is a significant improvement in signal-to-interference ratio. The M-sequence used is a relatively short one, with spreading ratio of only 31 allowing only modest processing gain. In more typical systems with higher spreading ratios the improvement should be even more marked.

4.8 Effect of High Interference-to-Signal Ratio

In one of the experiments using noise to represent the wideband signal and a single tone to represent the interference it was found that, although the resulting filter suppressed the interference, it badly distorted the wideband signal. The frequency response of the filter had a large peak in it at a frequency near the notch frequency rather than being relatively flat as it should have been. The only difference between this and some similar experiments was the higher ratio of narrowband-to-wideband signal amplitudes. In this case this ratio was 20:1. When a ratio of 10:1 was used under identical conditions (512-sample block to generate 16 coefficients) this problem did not arise. This led to the conclusion that for this high ratio and for the precision of the computations (32-bit floating point numbers with 24-bit mantissa) the autocorrelation matrix was ill-conditioned. Since the autocorrelation of the narrowband signal has unit magnitude for all delays, this matrix would be singular if there were no wideband signal at all. If the wideband signal is small relative to the narrowband one, the matrix elements will all have nearly unit magnitude, with the wideband signal providing small perturbations about this value. Thus the determinant will be computed from a number of values which would add up to zero except for these perturbations, and the sum will be very small relative to the values summed. Under these conditions, large errors can occur when the precision of the computation is not sufficient. While the determinant is not calculated directly in the coefficient computation, the result is identical to that which a straightforward matrix inversion would give and

the same errors occur.

To rule out the possibility that the poor result was caused by an unlucky choice of the noise signal resulting in a near-singularity, the experiment was repeated with another noise sample. The frequency response was different, but the distortion was of about the same magnitude, indicating that the problem was not caused by an unlucky choice.

Two solutions to this problem come to mind. One is the use of higher precision in the computations; but this can be an expensive one. The other is to add white noise to the input signal or to the autocorrelation matrix when the coefficients are being computed to reduce the ratio of narrowband-to-wideband amplitudes. This will cause some degradation in performance when the interference is not very strong, but will provide an improvement when the interference is very strong. Thus it may be advisable to add the noise only when some test indicates that it would be useful. A determination of the ratio of the zero-delay autocorrelation coefficient to the mean magnitude of the other coefficients could be used for such a test. A value near unity for this ratio would indicate strong narrowband interference and the need to add noise.

Time does not permit a more thorough and quantitative analysis of this problem at this time.

5. CONCLUSIONS

An excision filter to remove narrowband interfering signals from a direct-sequence spread-spectrum signal can be implemented by the use of linear prediction to predict the next value of the narrowband signal and subtract it from the original signal. The direct sequence signal is not subtracted because it is uncorrelated over the sample interval and is therefore not predictable. This requires that the effective sampling rate not be greater than one sample per chip. Simulation experiments have verified the effectiveness of this technique. The effects of changing the number of coefficients used in the excision filter and of the size of the

input signal block used for the determination of the coefficients have been examined. In general, the more coefficients in the filter the better the result, but since the computing requirement increases faster than linearly with the number of coefficients it is important to keep their number down to that necessary to counter the interference threat.

Larger block sizes in the autocorrelation estimation also provide better performance providing there are sufficient coefficients, and it appears that a larger number of coefficients calls for a larger block size.

Delay of the input signal to allow computation of the coefficients before the signal enters the filter will provide better performance when the interference is varying, as will a more rapid updating of the filter coefficients.

Very large ratios of narrowband interference level to wideband signal (including noise) level can cause problems in the computation of the coefficients. This appears to be the result of an ill-conditioned autocorrelation matrix and the limited precision of the computations when the interfering signal is far above the signal and noise levels. The addition of white noise to the signal used for the coefficient computation is a possible means of overcoming this problem. The determination of more precise quantitative results would seem to be an area deserving of further investigation.

REFERENCES

1. Hsu, Frank M., and Arthur A. Giordano, "Digital Whitening Techniques for Improving Spread Spectrum Communications Performance in the Presence of Narrowband Jamming and Interference", IEEE Trans. on Comm. Vol. COM-26, No. 2, Feb 1978, pp 209-216.

2. Yule, G.U., "On a Method of Investigating Periodicities in Disturbed Series, with Special References to Wolfer's Sunspot Numbers", Phil. Trans. Roy. Soc., 1927, A.226, pp 267-298.

3. Kay, Steven M., and Stanley Lawrence Marple, Jr., "Spectrum Analysis - A Modern Perspective", Proc. IEEE, Vol. 69, No. 11, Nov 1981, p 1398.

4. Schmeiser, Bruce W., "Random Variate Generation: A Survey", Research Memorandom No. 80-06, School of Industrial Engineering, Purdue University, June, 1980.

LIST OF FIGURES

Figure B15 Spectrum of Input Signal - Four Widely-Spaced Tones plus Noise.

Figure B16 Frequency Response of Filter Adapted to Signal of Figure 15 - 16 Coefficients.

Figure B17 Spectrum of Output of Filter of Figure 16 when Signal of Figure 15 is Input.

Figure B18 Frequency Response of Filter Adapted to Signal of Figure 15 - 6 Coefficients.

Figure B19 Spectrum of Output of Filter of Figure 18 when Signal of Figure 15 is Input.

Figure B20 Spectrum of Input Signal - Four Closely Spaced Tones plus Noise.

Figure B21 Frequency Response of Filter Adapted to Signal of Figure 20 - 16 Coefficients.

Figure B22 Spectrum of Output of Filter of Figure 21 when Signal of Figure 20 is Input.

Figure B23 Frequency Response of Filter Adapted to Signal of Figure 20 - 4 Coefficients.

Figure B24 Spectrum of Output of Filter of Figure 23 when Signal of Figure 20 is Input.

Figure B25 Spectrum of Input Signal - Swept-Frequency Interference plus Noise Signal.

Figure B26 Frequency Response of Filter Adapted to Signal of Figure 25 - 16 Coefficients, 496 Products in Correlation Estimate.

Figure B27 Spectrum of Output of Filter of Figure 26 for Input of Figure 25.

Figure B28 Impulse Response of Filter of Figure 26.

Figure B29 Frequency Response of Filter Adapted to Signal of Figure 25 - 4 Coefficients, 496 Products in Correlation Estimate.

Figure B30 Spectrum of Output of Filter of Figure 29 for Input of Figure 25.

Figure B31 Impulse Response of Filter of Figure 29.

Figure B32 Frequency Response of Filter Adapted to Signal of Figure 25 - 16 Coefficients, 248 Products in Correlation Estimate.

Figure B33 Spectrum of Output of Filter of Figure 32 for Input of Figure 25.

Figure B34 Time-Domain Output of Filter of Figure 32 for Input of Figure 25.

Figure B35 Frequency Response of Filter Adapted to Noise plus Sine Wave at Frequency of 0.1 - 16 Coefficients, Tap Spacing = 4.

Figure B36 Frequency Response of Filter Adapted to Noise Plus Sine Wave at Frequency of 0.3 - 16 Coefficients, Tap Spacing = 4.

Figure B37 Impulse Response of Filter of Figure 36.

Figure B38 Spectrum of Input Signal M-Sequence + Sine wave + Noise.

Figure B39 Output of Matched Filter (no excision) for Input of Figure 38.

Figure B40 Spectrum of Output of Excision Filter Adapted to Signal of Figure 38 - 8 Coefficients, Tap Spacing = 8.

Figure B41 Output of Excision Filter followed by Matched Filter for Input of Figure 38.

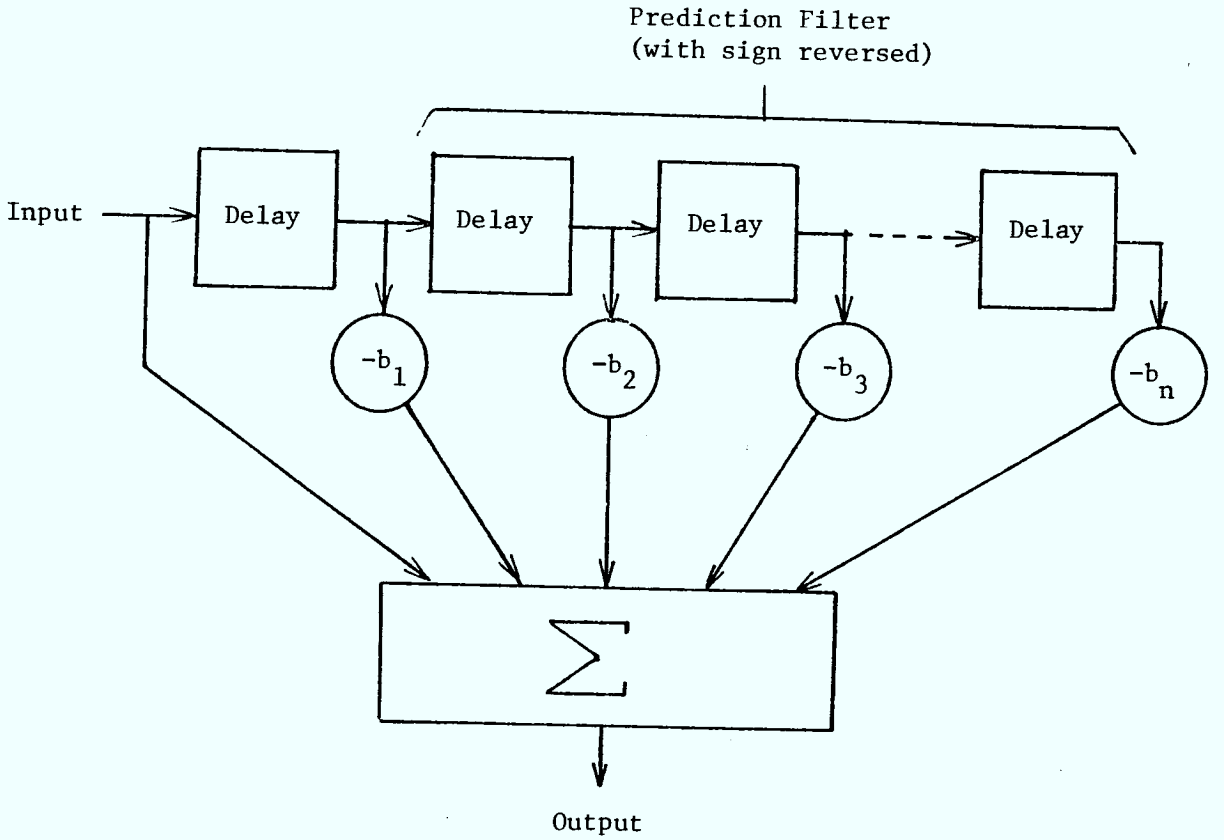


FIGURE B1 FIR Excision Filter

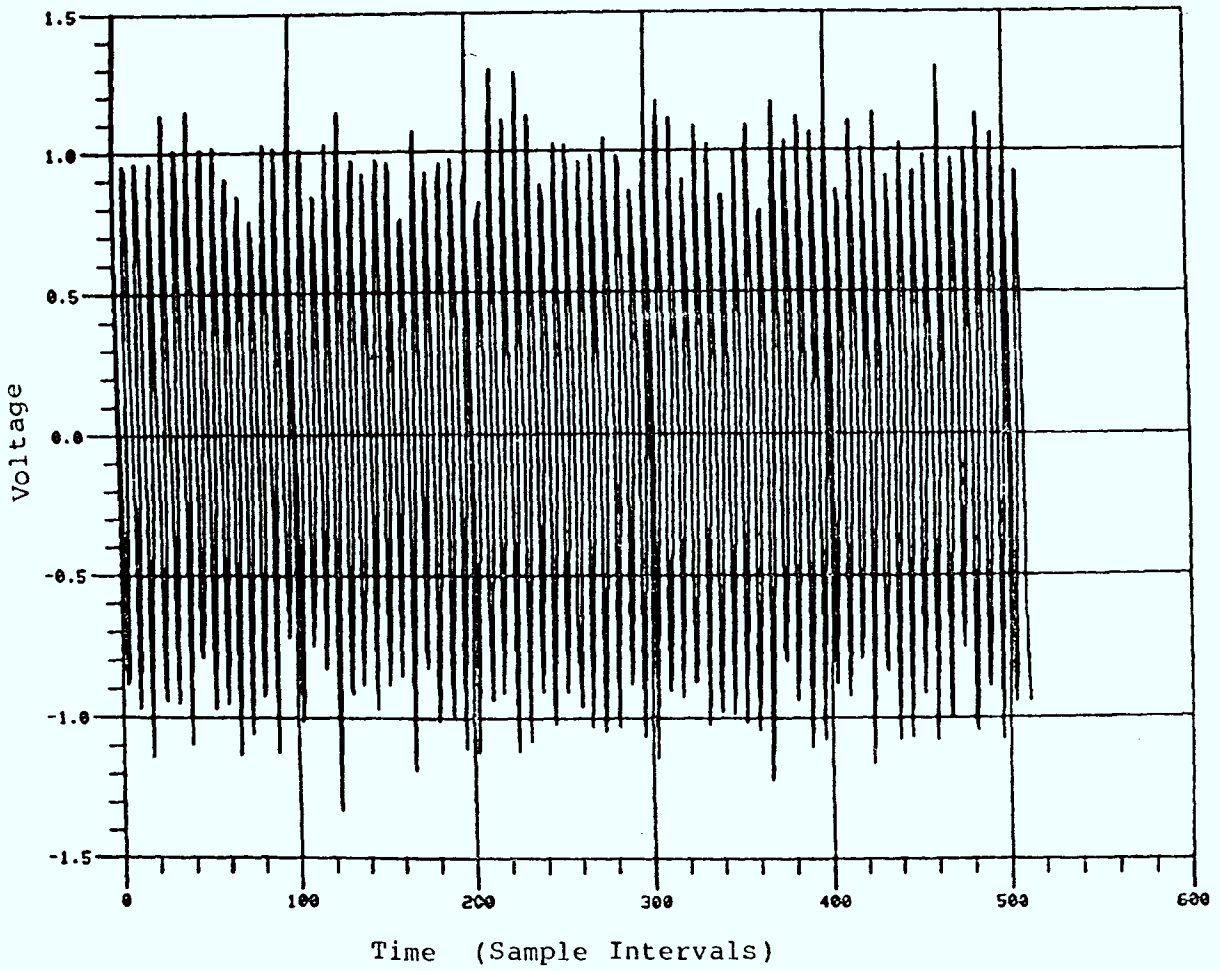


Figure B2 Input Signal - Noise plus Sine Wave at 0.14 x Sample Rate. Voltage Ratio sine/noise = 5.

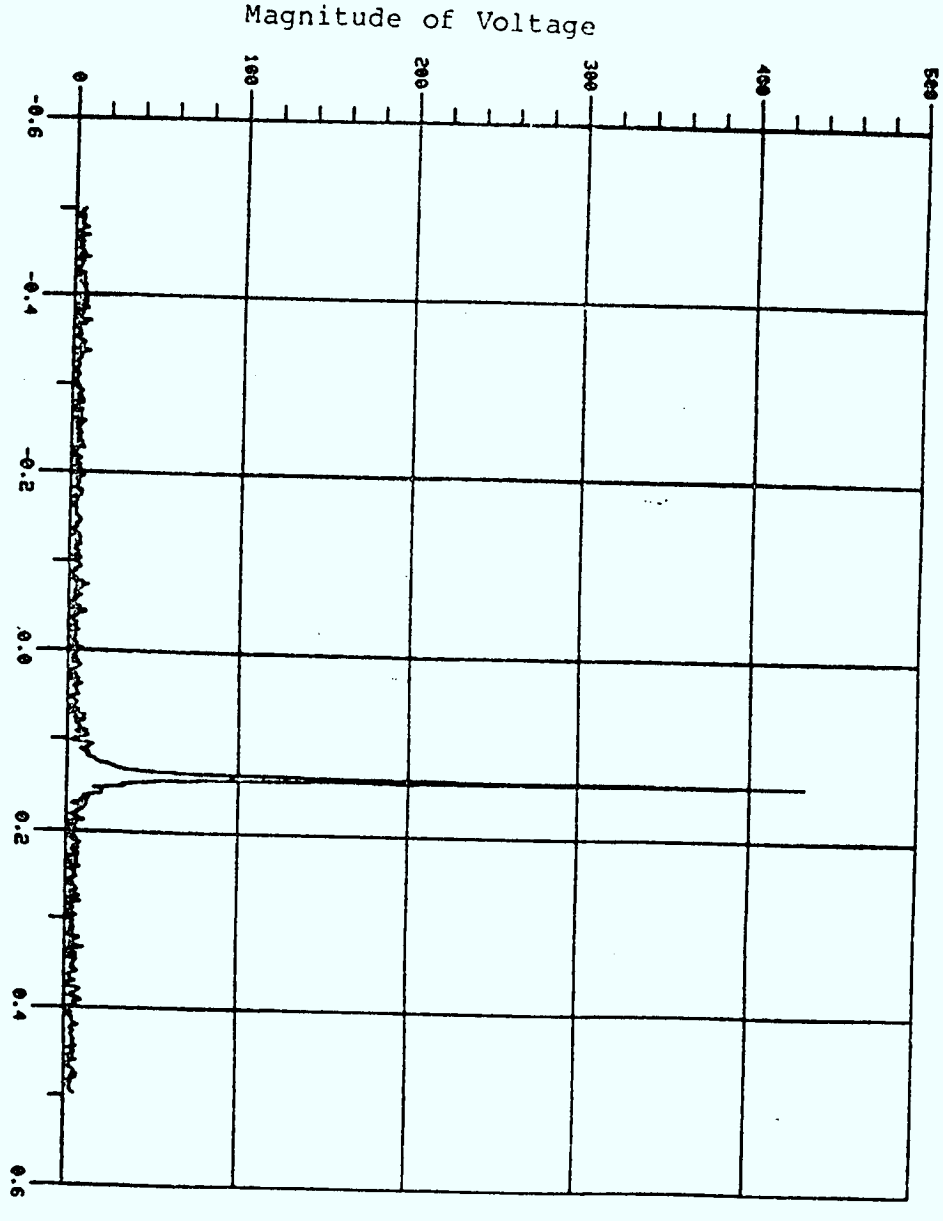


Figure B3 Spectrum of Input Signal of Figure 2.

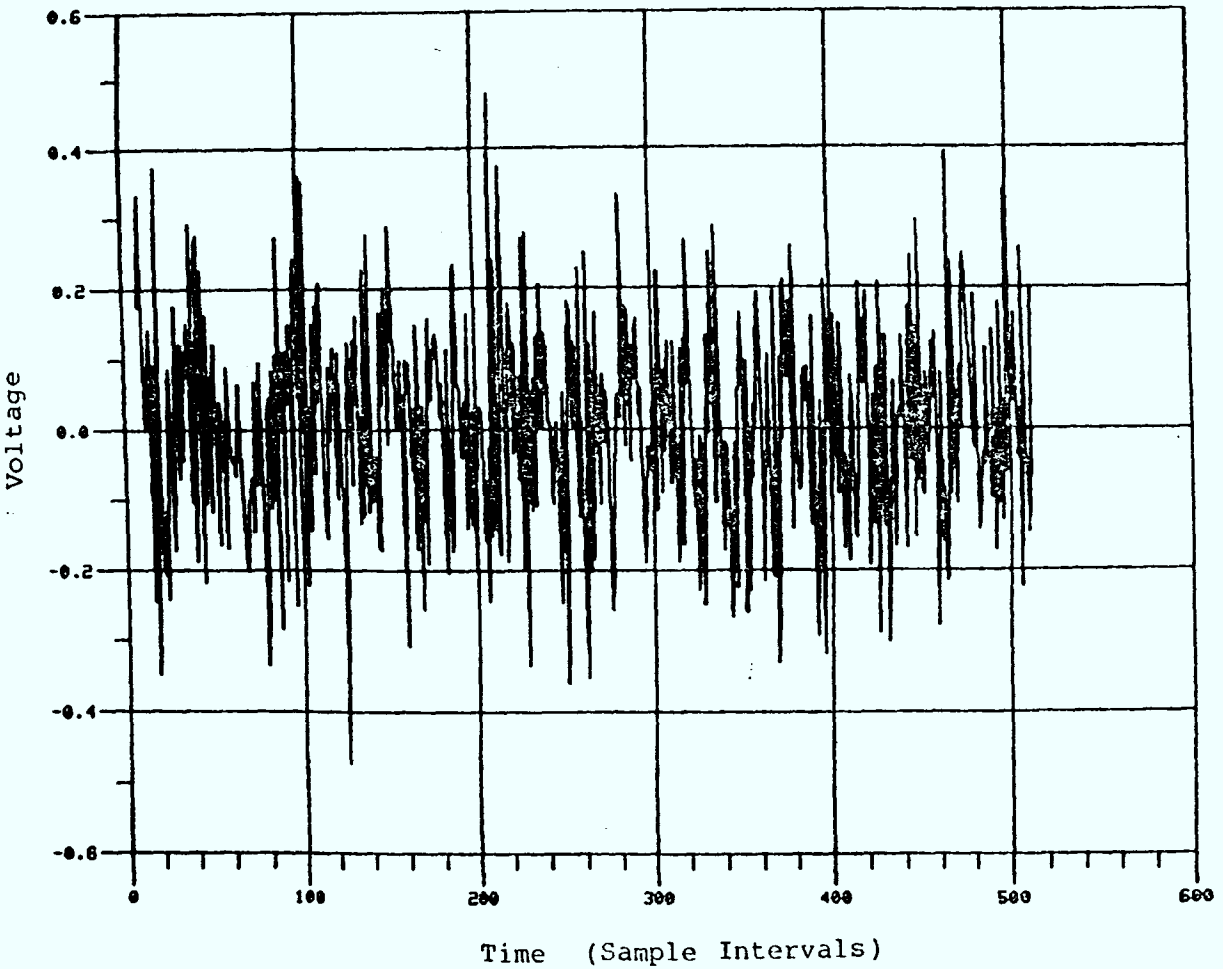


Figure B4 Output of Filter for Input of Figure 3 - 8 Coefficients.

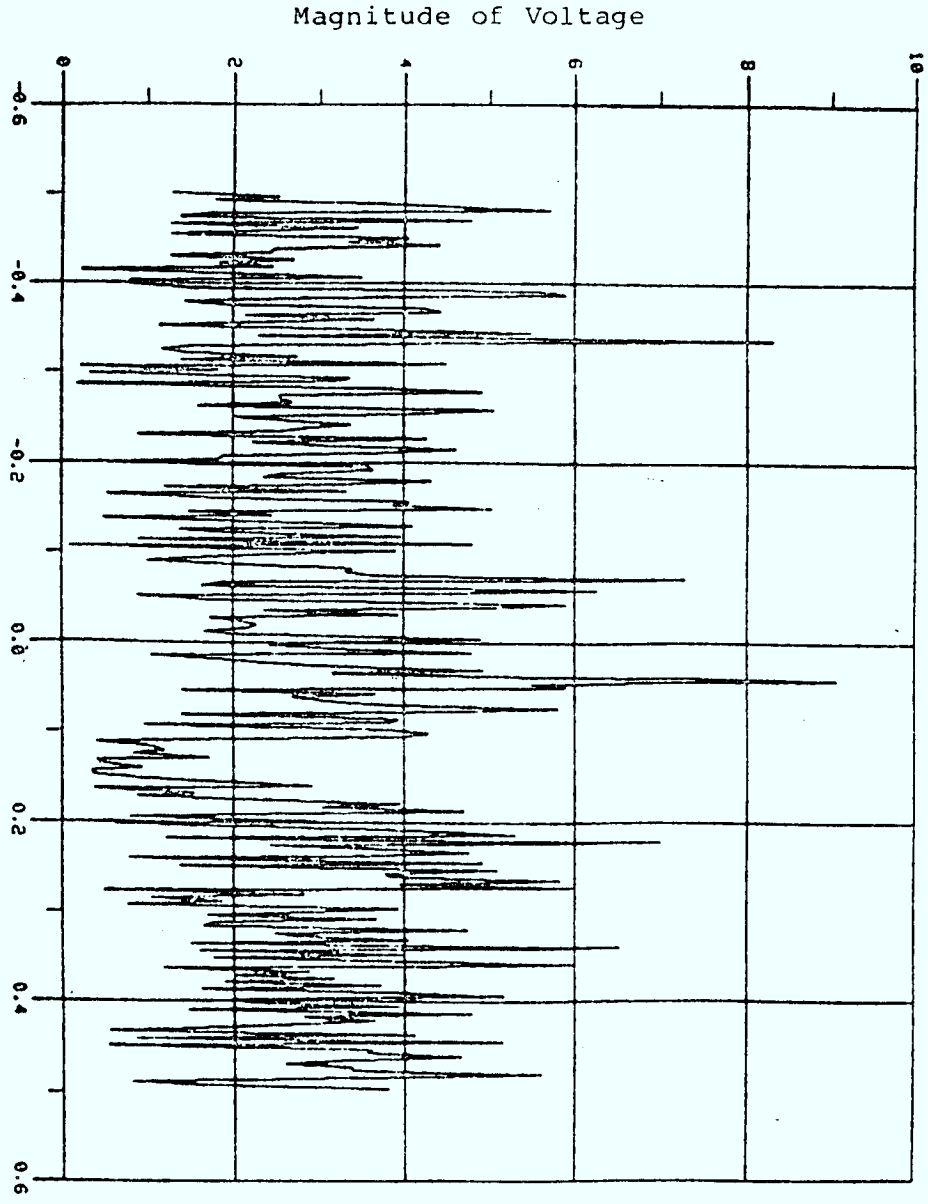


Figure B5 Spectrum of Output Signal of Figure 4.

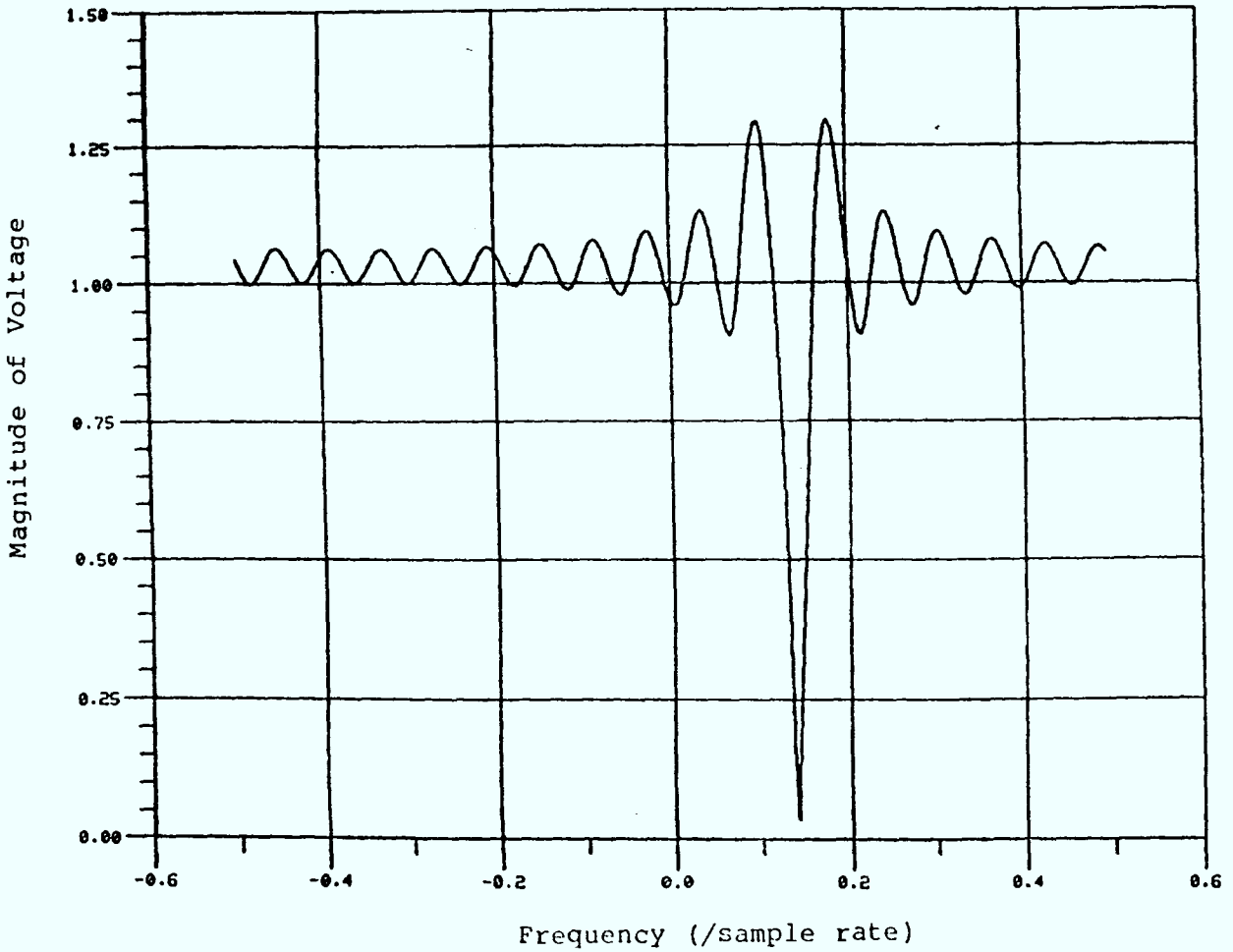


Figure B6 Frequency Response of Filter Adapted to Sine Wave plus Impulse - 16 Coefficients.

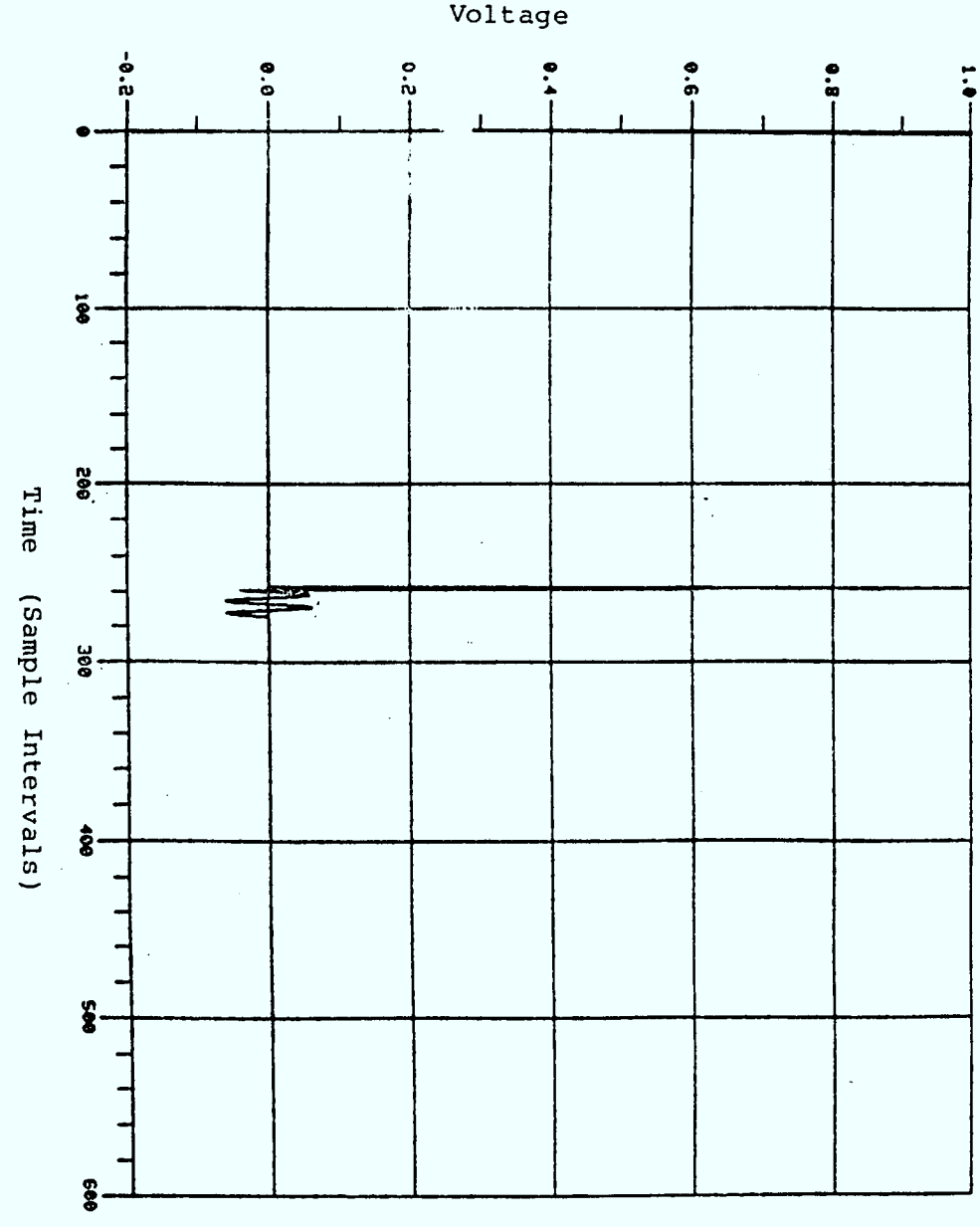


Figure B7 Impulse Response of Filter of Figure 6.

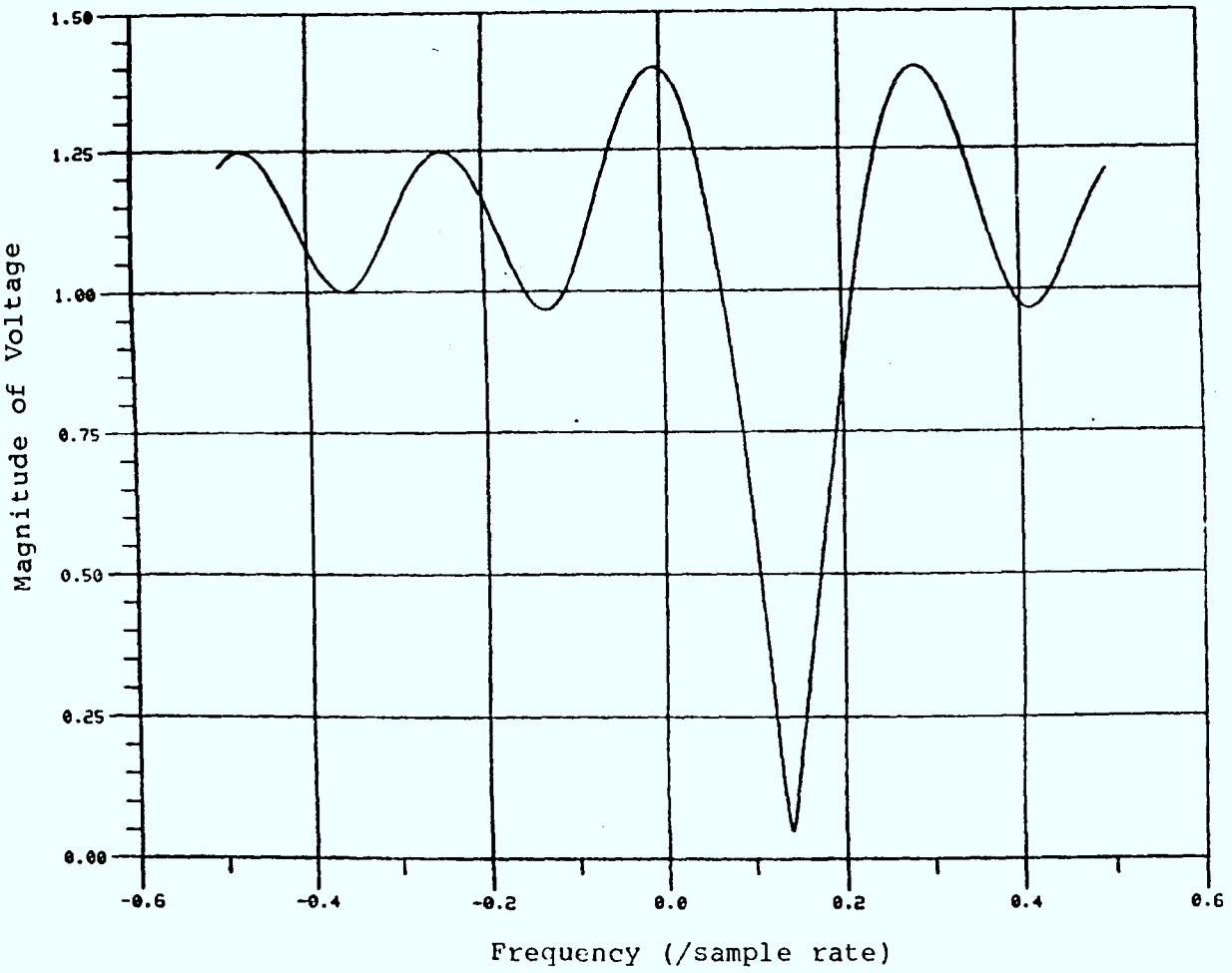


Figure B8 Frequency Response of Filter Adapted to Sine Wave plus Impulse - 4 Coefficients.

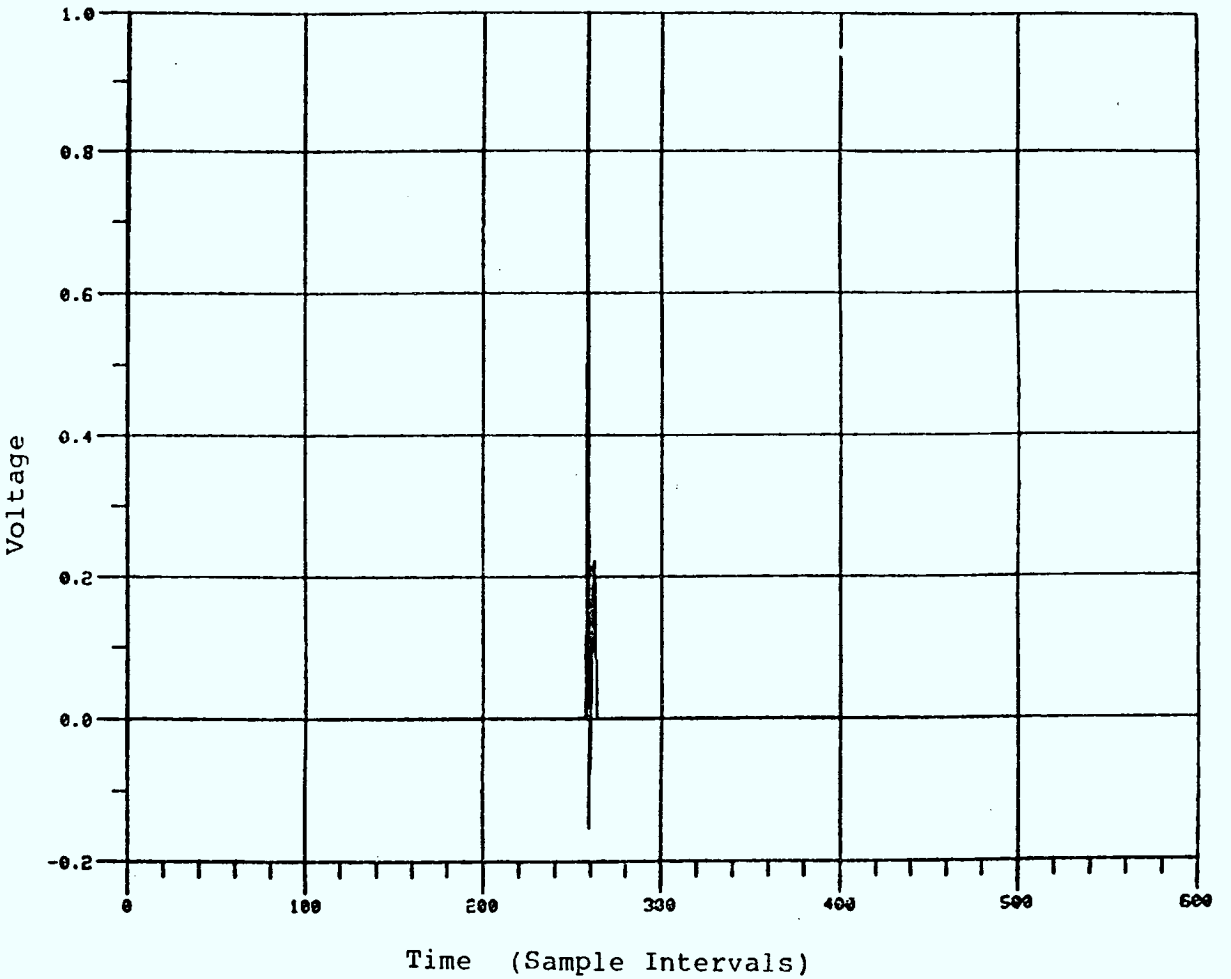


Figure B9 Impulse Response of Filter of Figure 8.

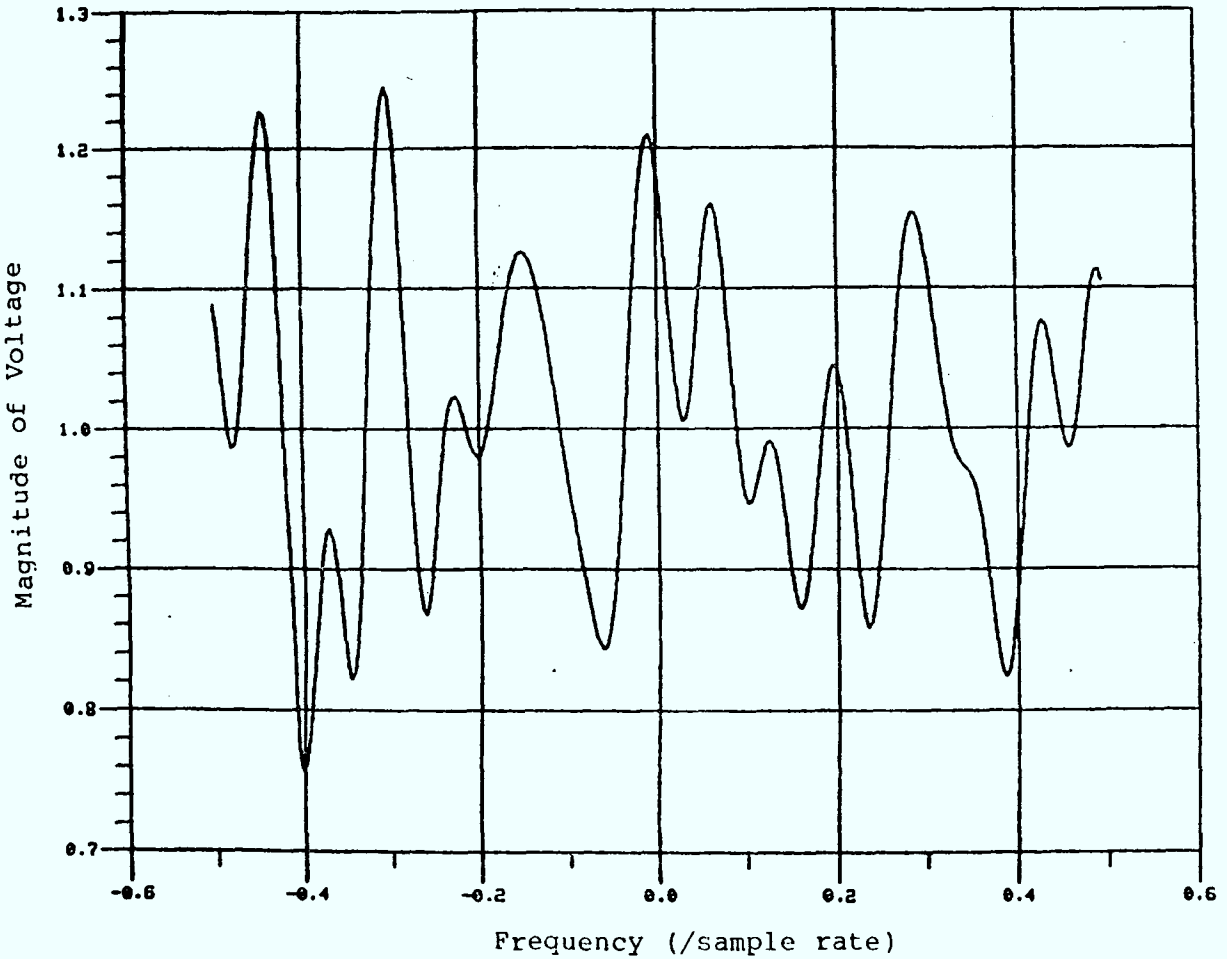


Figure B10 Frequency Response of Filter Adapted to Noise Only - 496 Products in Correlation Estimate, 16 Coefficients.

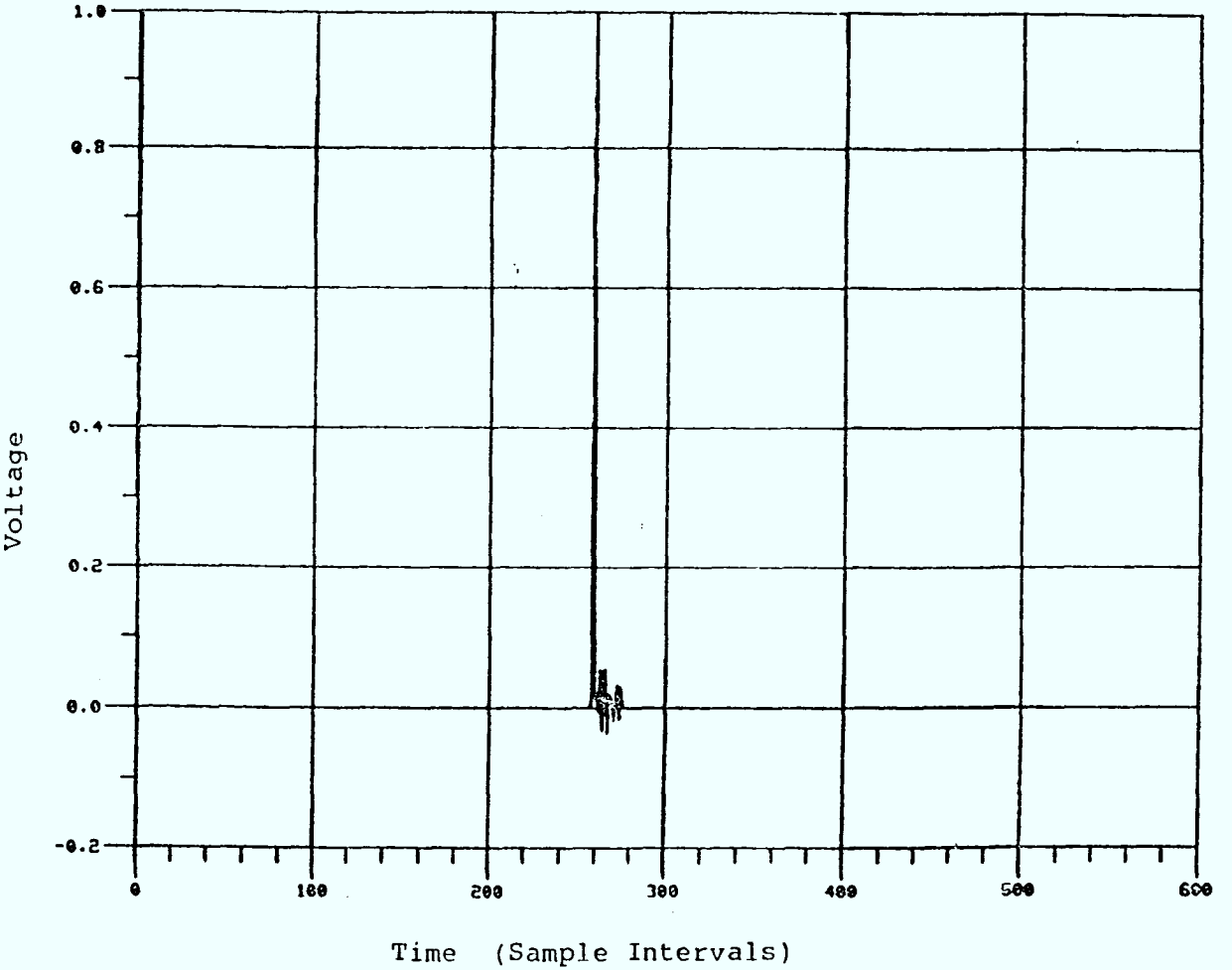


Figure B11 Impulse Response of Filter of Figure 10.

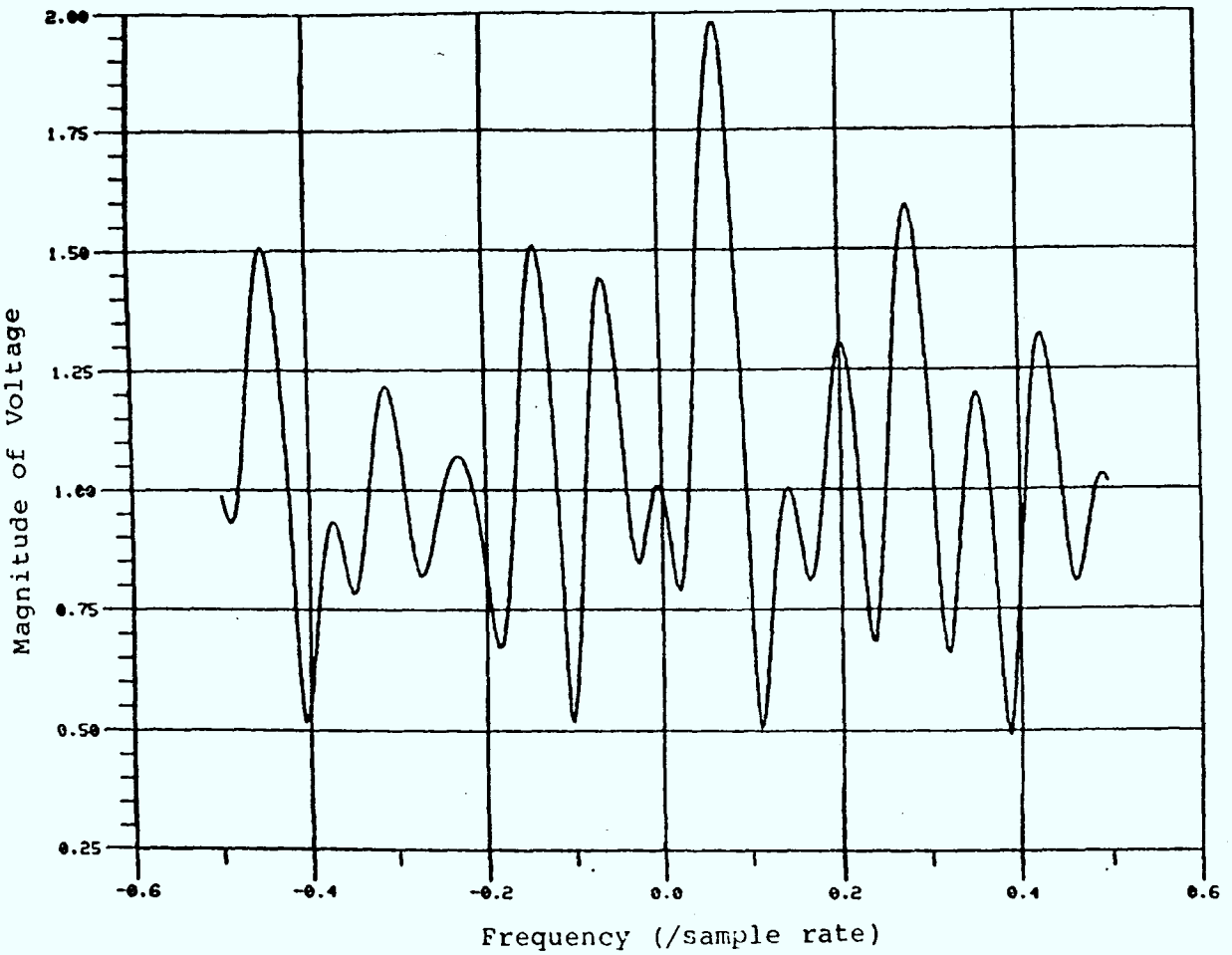


Figure B12 Frequency Response of Filter Adapted to Noise Only - 124 products in Correlation Estimate, 16 Coefficients.

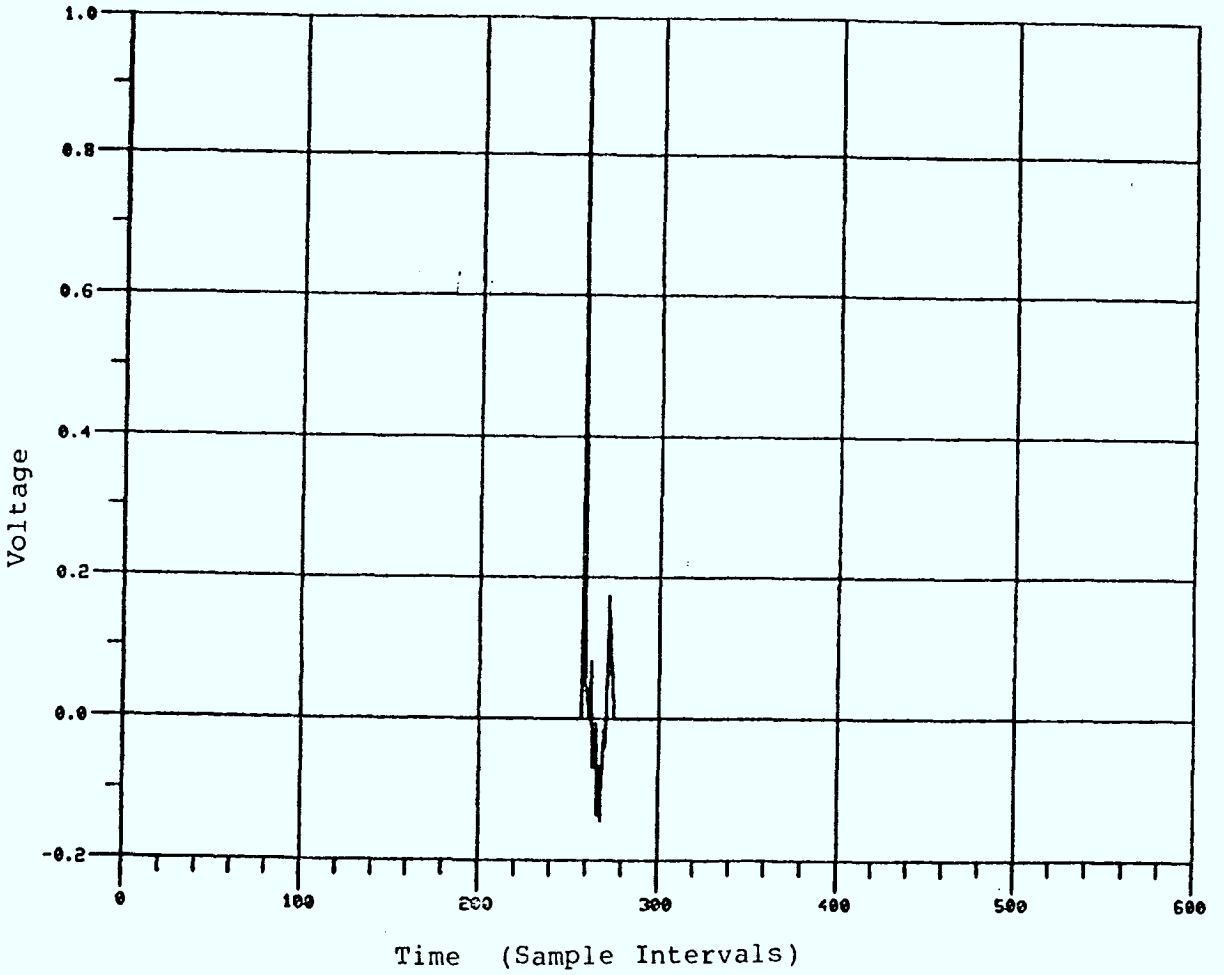


Figure B13 Impulse Response of Filter of Figure 12.

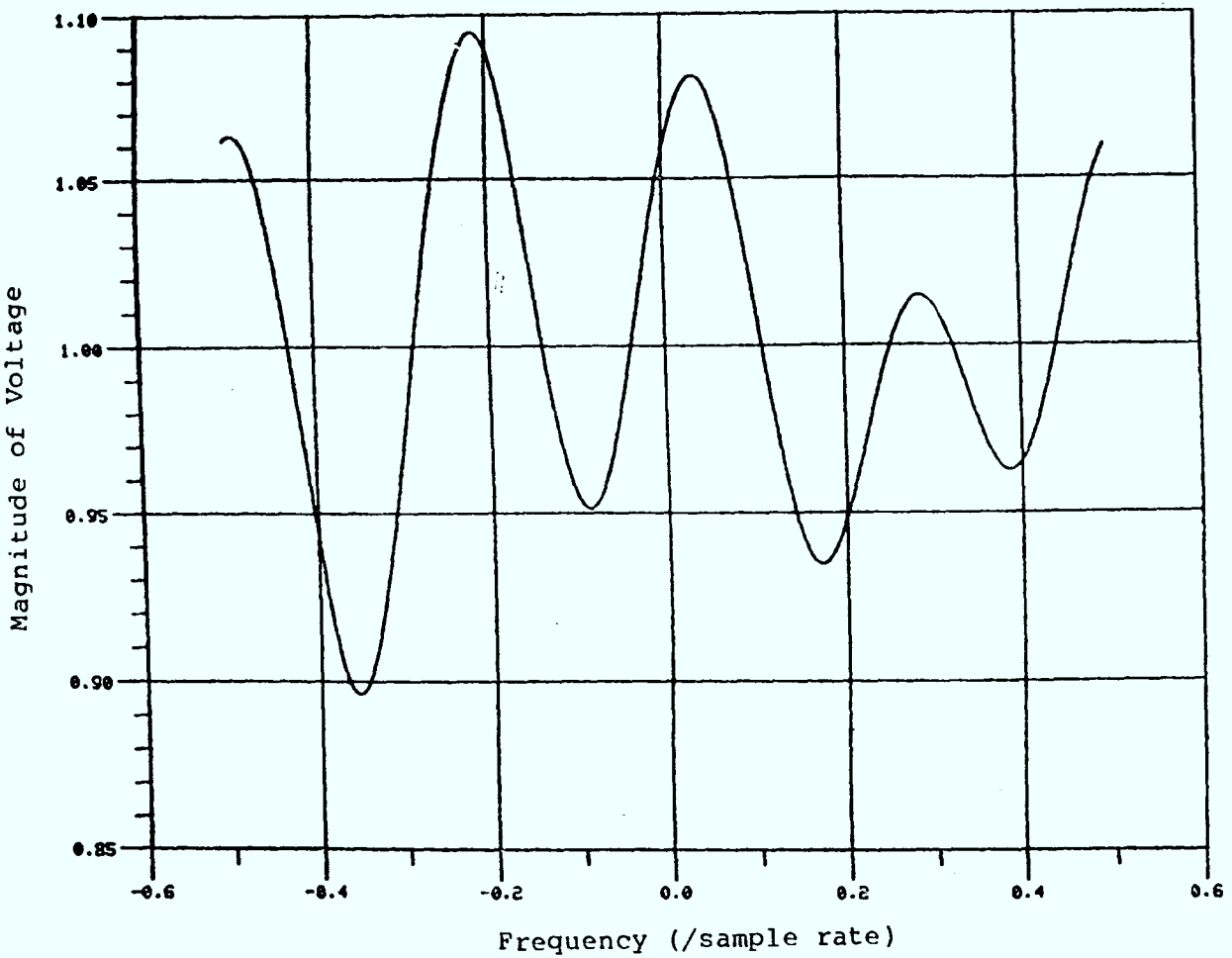


Figure B14 Frequency Response of Filter Adapted to Noise Only - 496 Products in Correlation Estimate, 4 Coefficients.

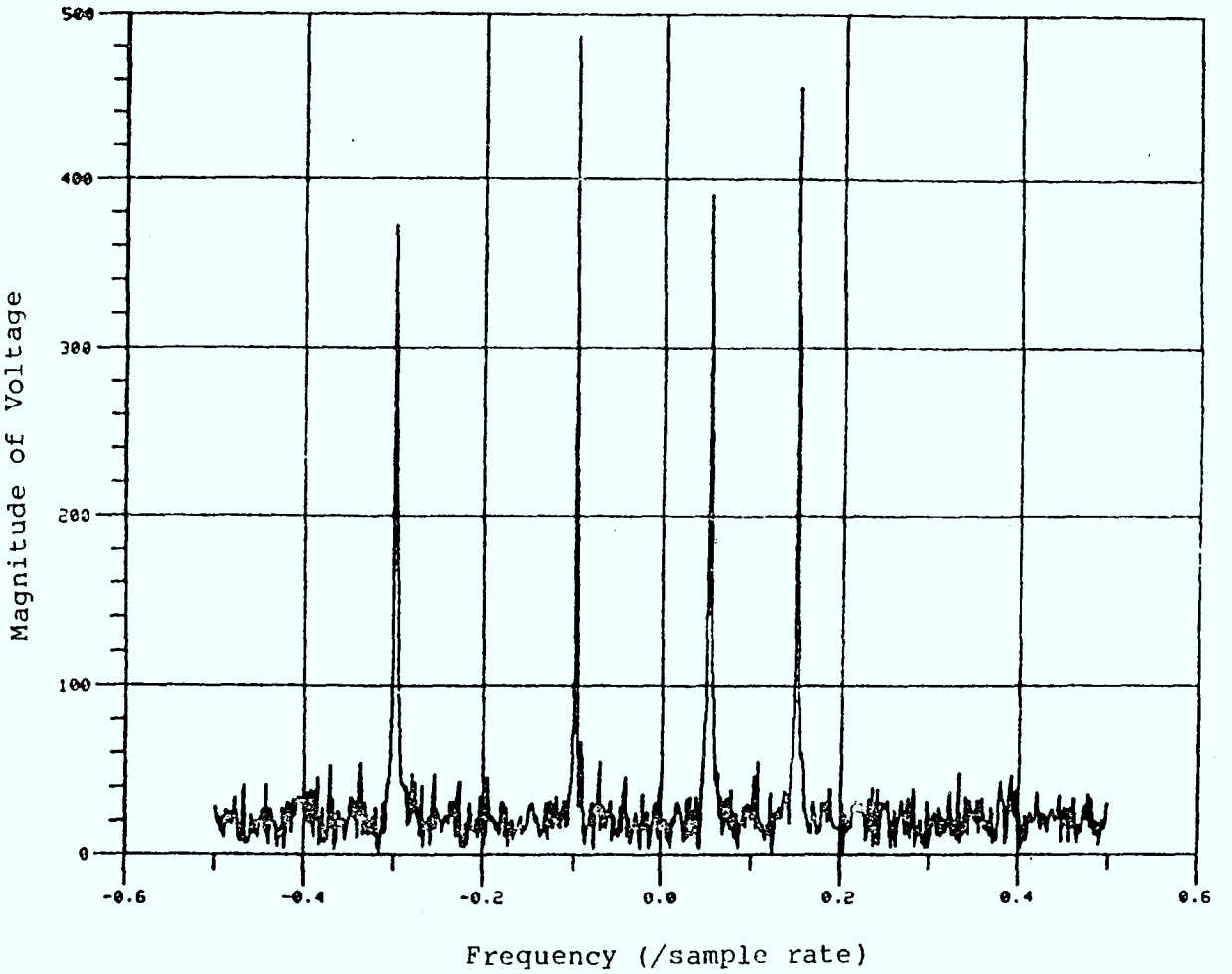


Figure B15 Spectrum of Input Signal - Four Widely-Spaced Tones plus Noise.

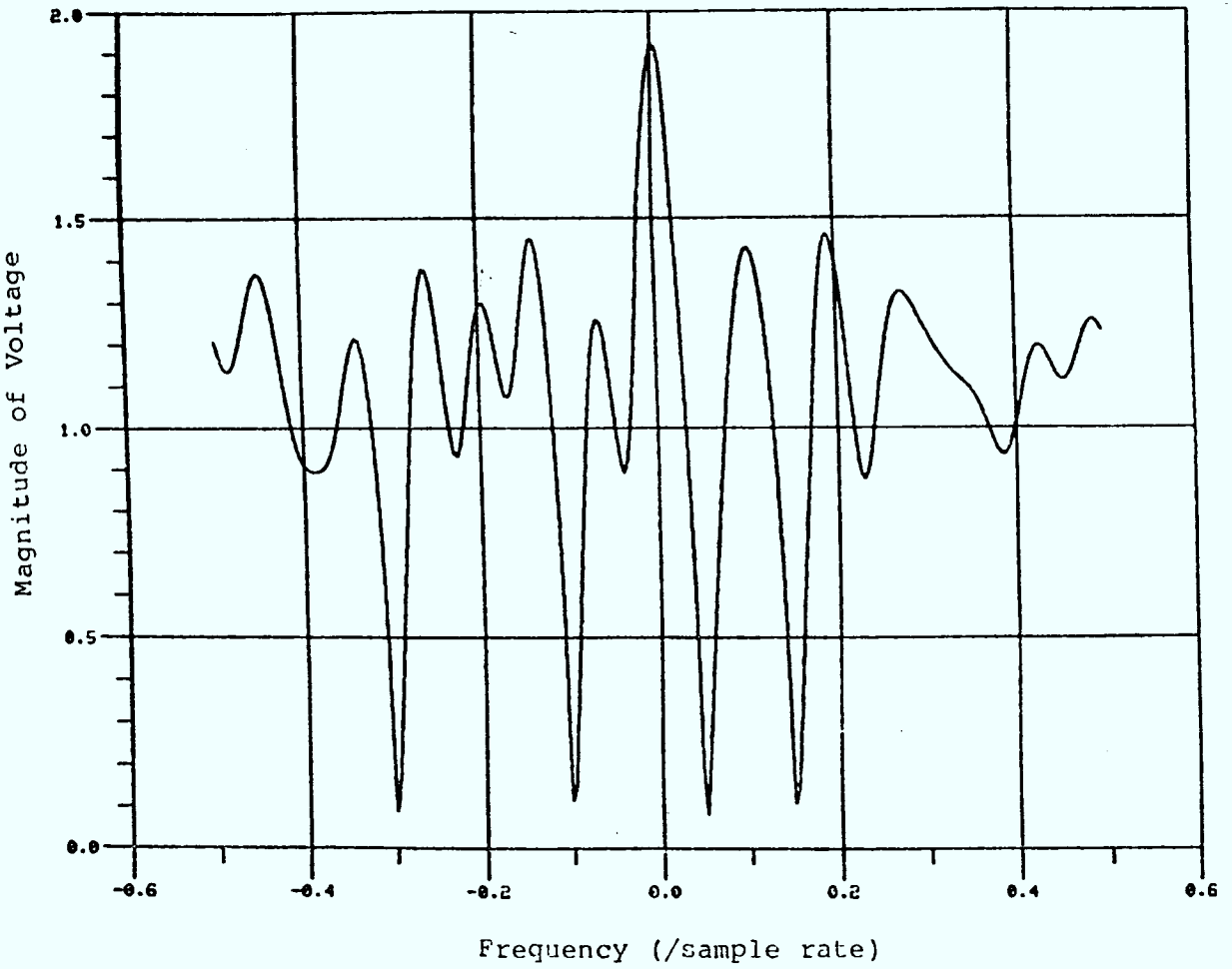


Figure B16 Frequency Response of Filter Adapted to Signal of Figure 15 - 16 Coefficients.

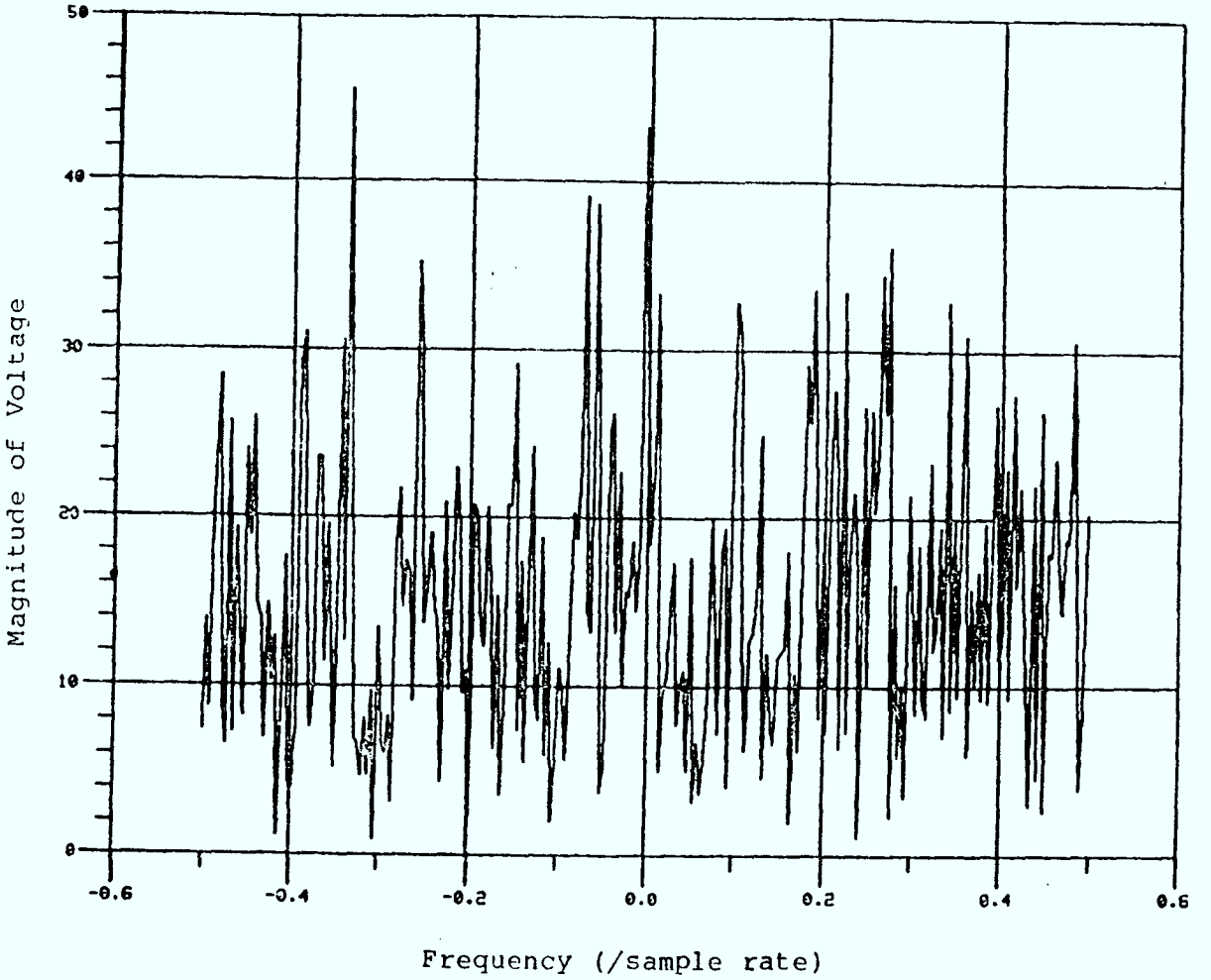


FIGURE B17 Spectrum of Output of Filter of Figure 16 when Signal of Figure 15 is Input.

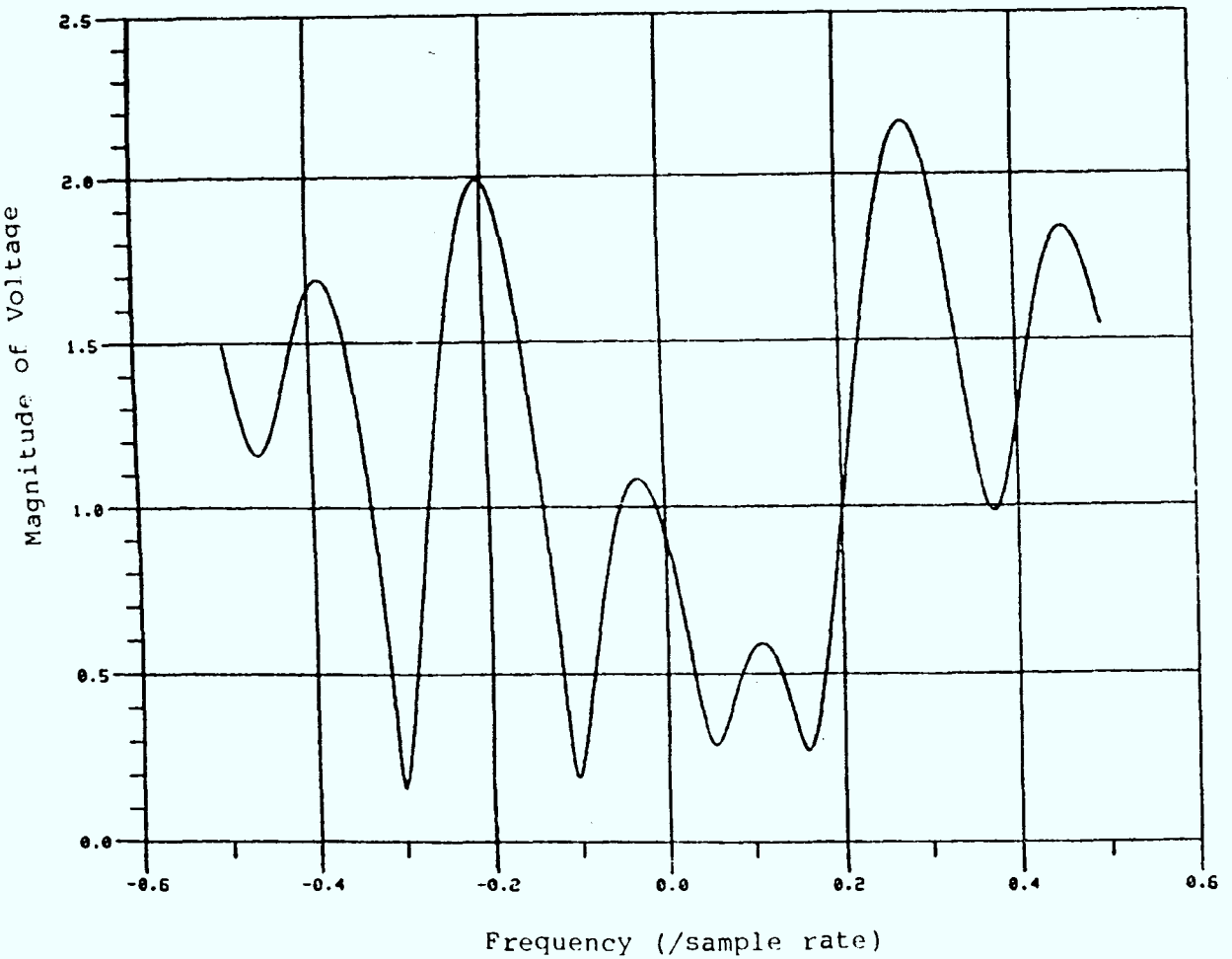


Figure B18 Frequency Response of Filter Adapted to Signal of Figure 15 - 6 Coefficients.

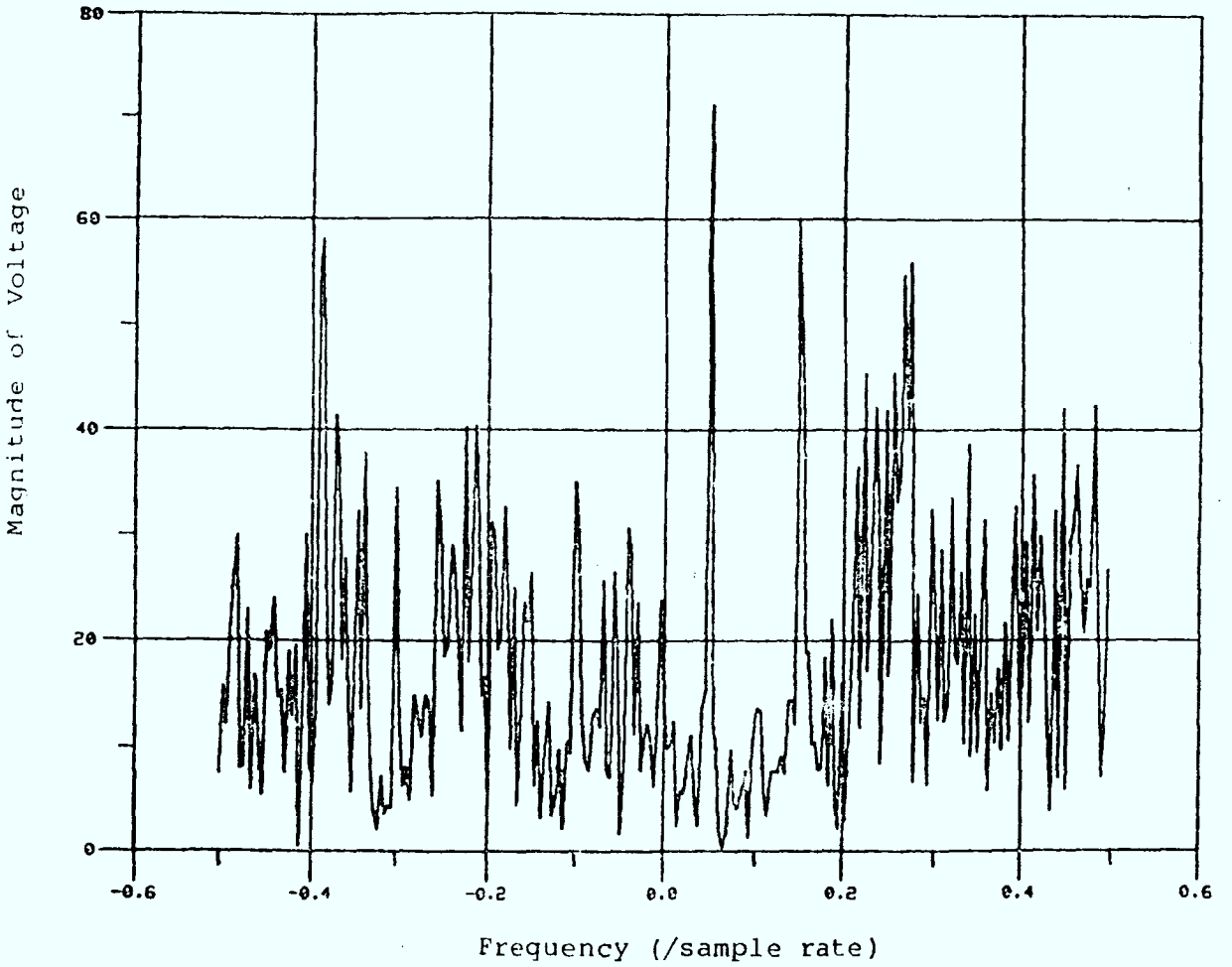


Figure B19 Spectrum of Output of Filter of Figure 18 when Signal of Figure 15 is Input.

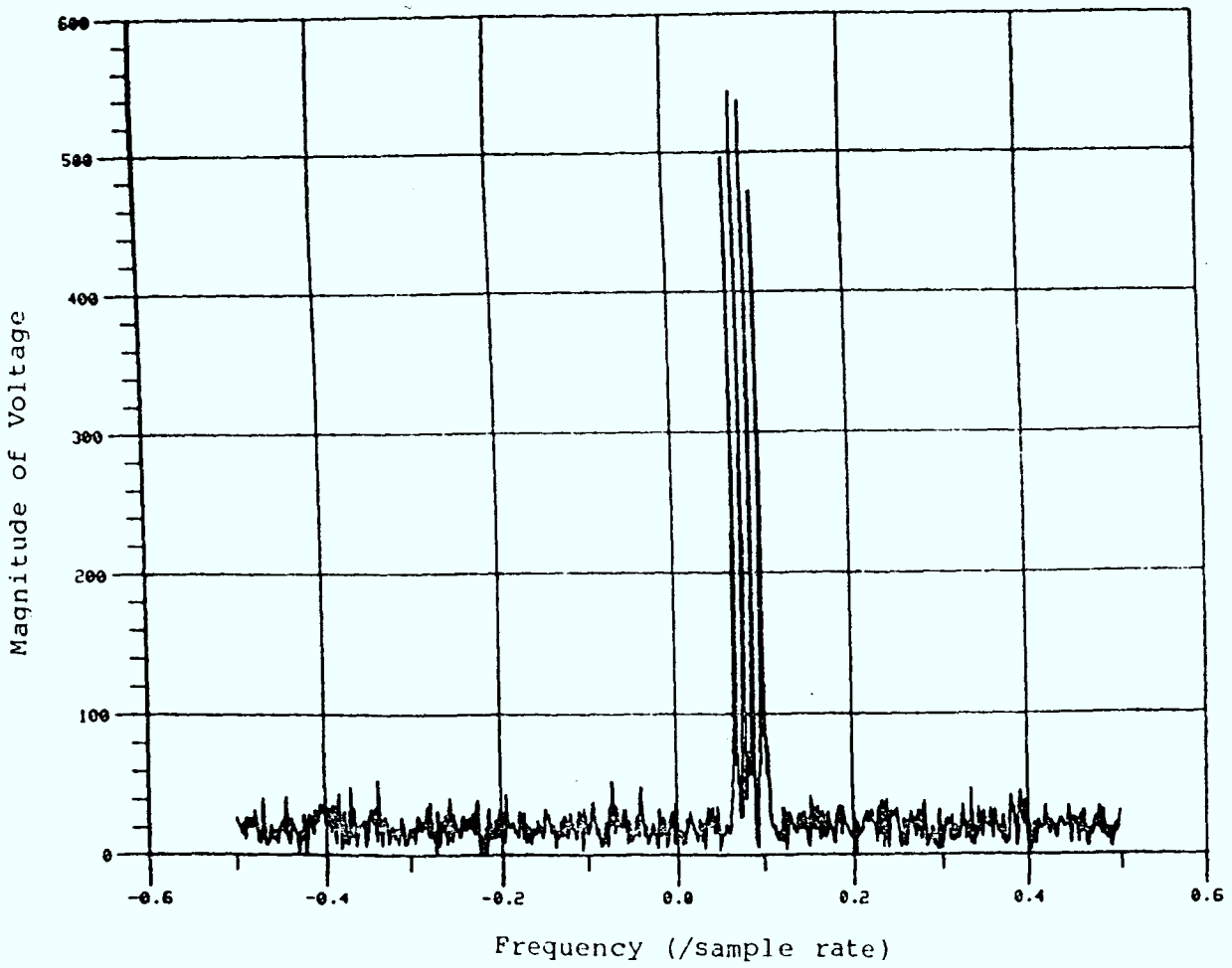


Figure B20 Spectrum of Input Signal - Four Closely Spaced Tones plus Noise.

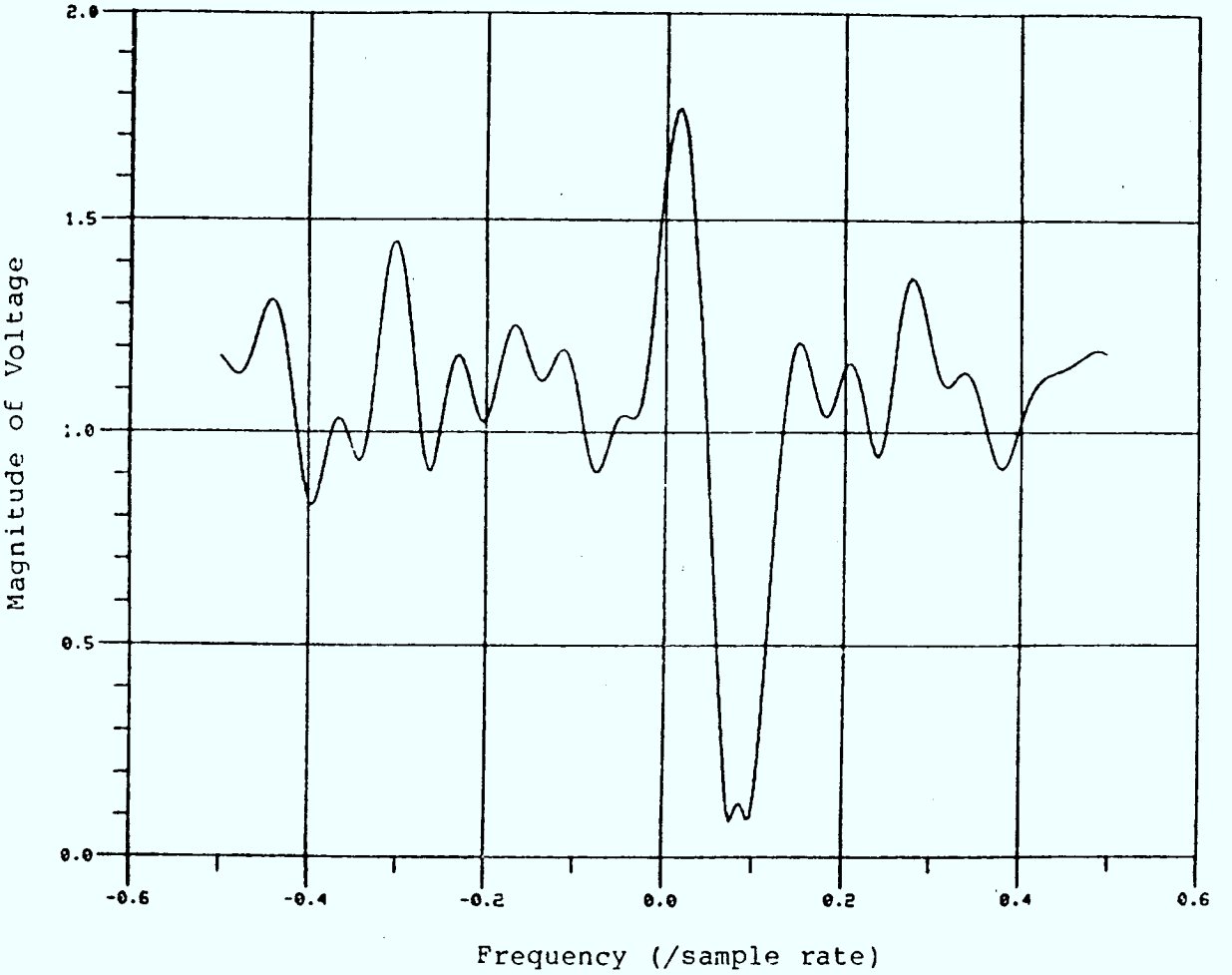


Figure B21 Frequency Response of Filter Adapted to Signal of Figure 20 - 16 Coefficients.

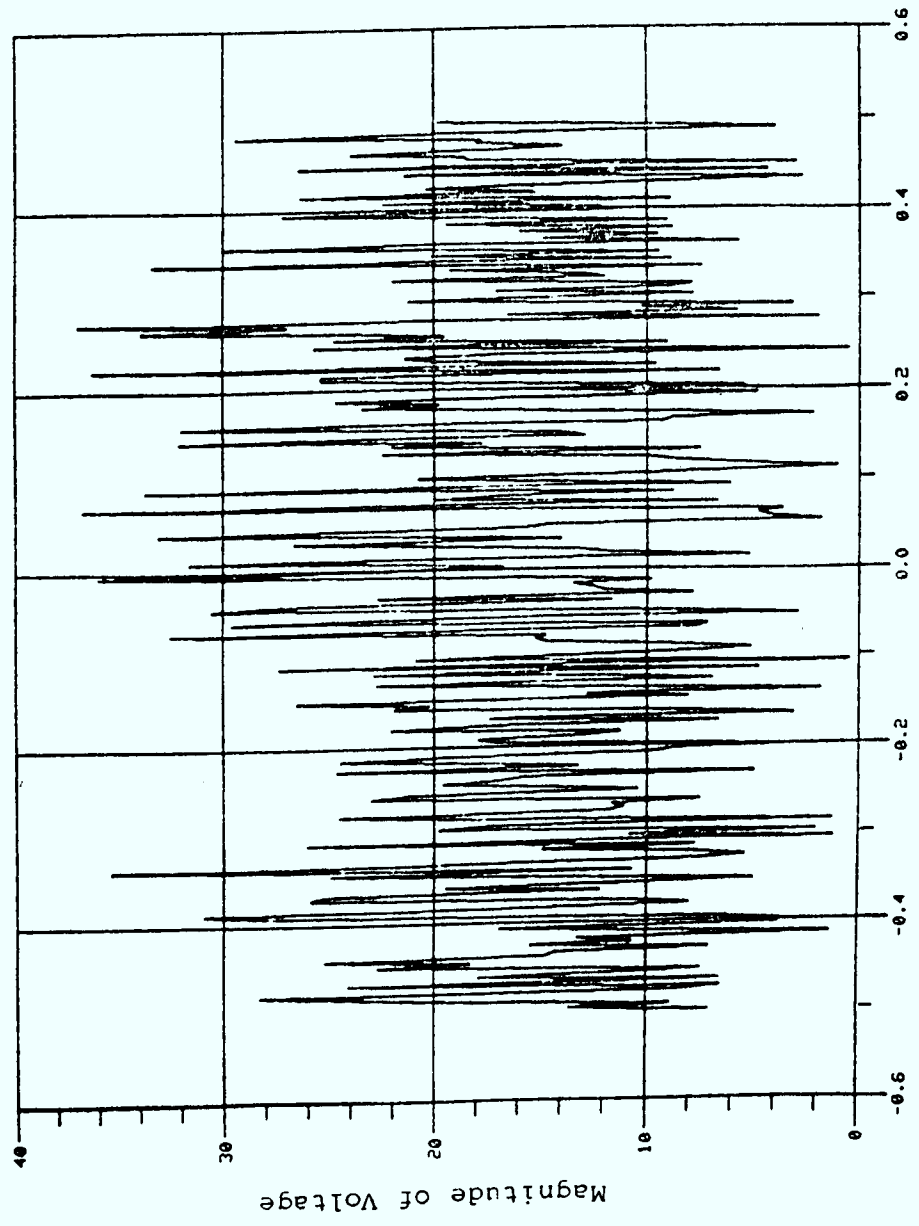


Figure B22 Spectrum of Output of Filter of Figure 21 when Signal of Figure 20 is Input.

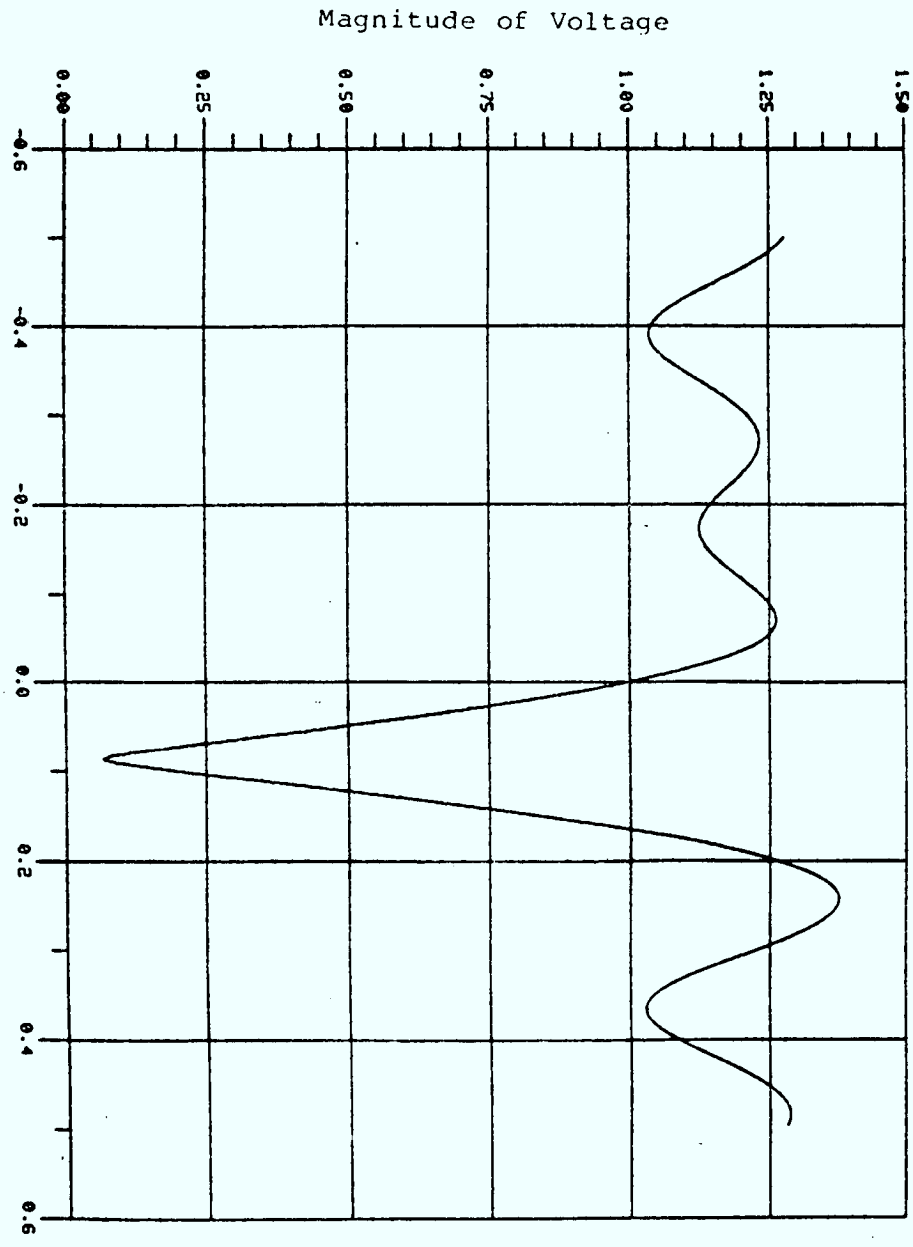


Figure B23 Frequency Response of Filter Adapted to Signal of Figure 20 - 4 Coefficients.

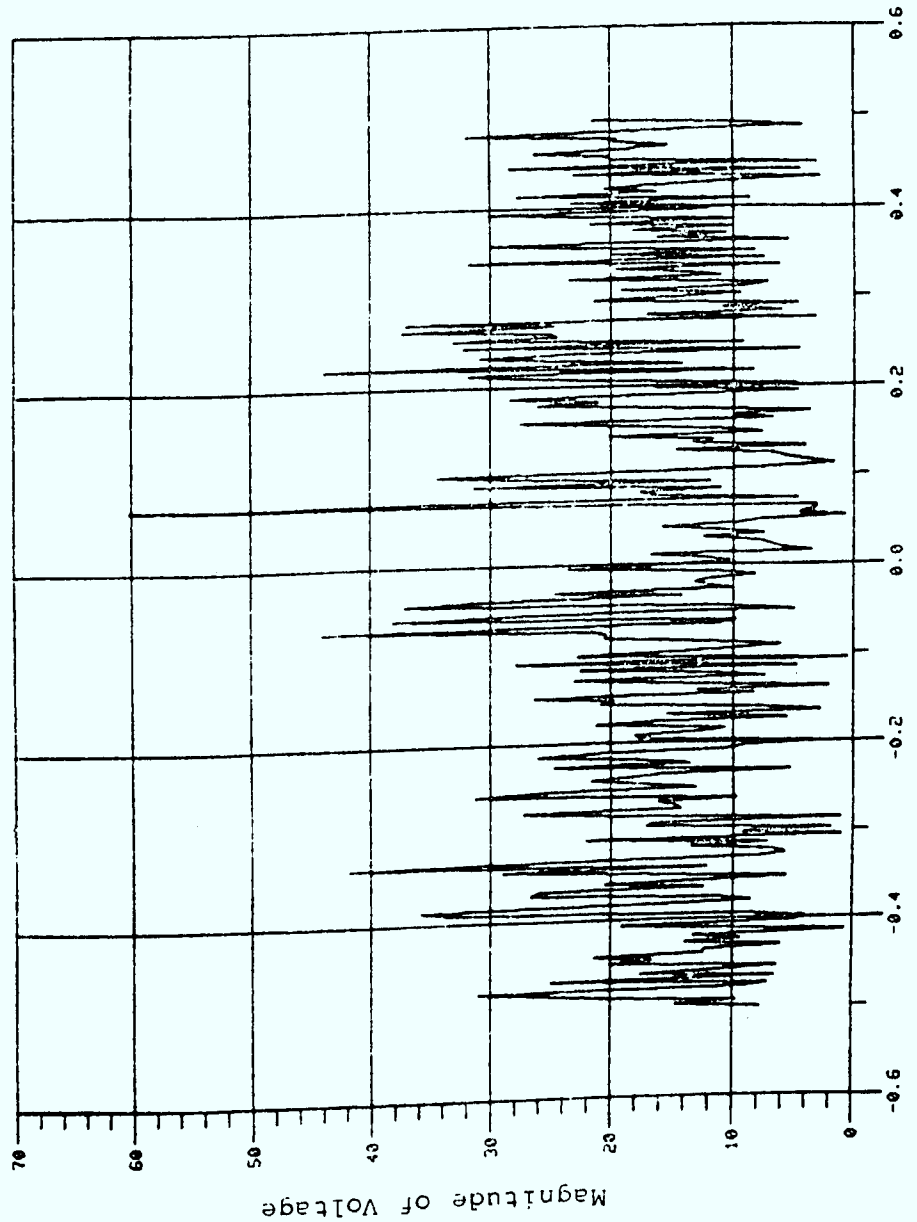


Figure B24 Spectrum of Output of Filter of Figure 23 when Signal of Figure 20 is Input.

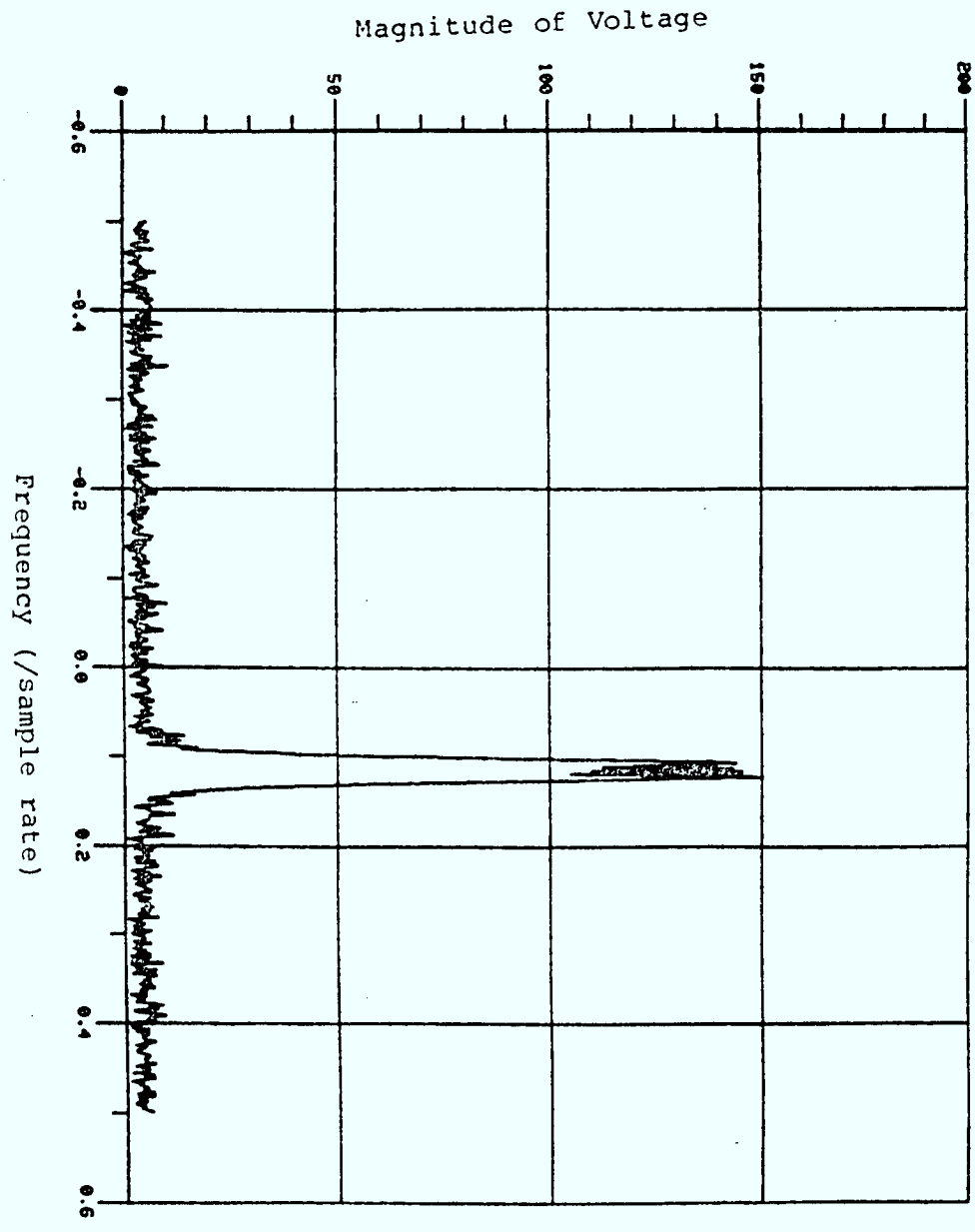


Figure B25 Spectrum of Input Signal - Swept-Frequency Interference plus Noise Signal.

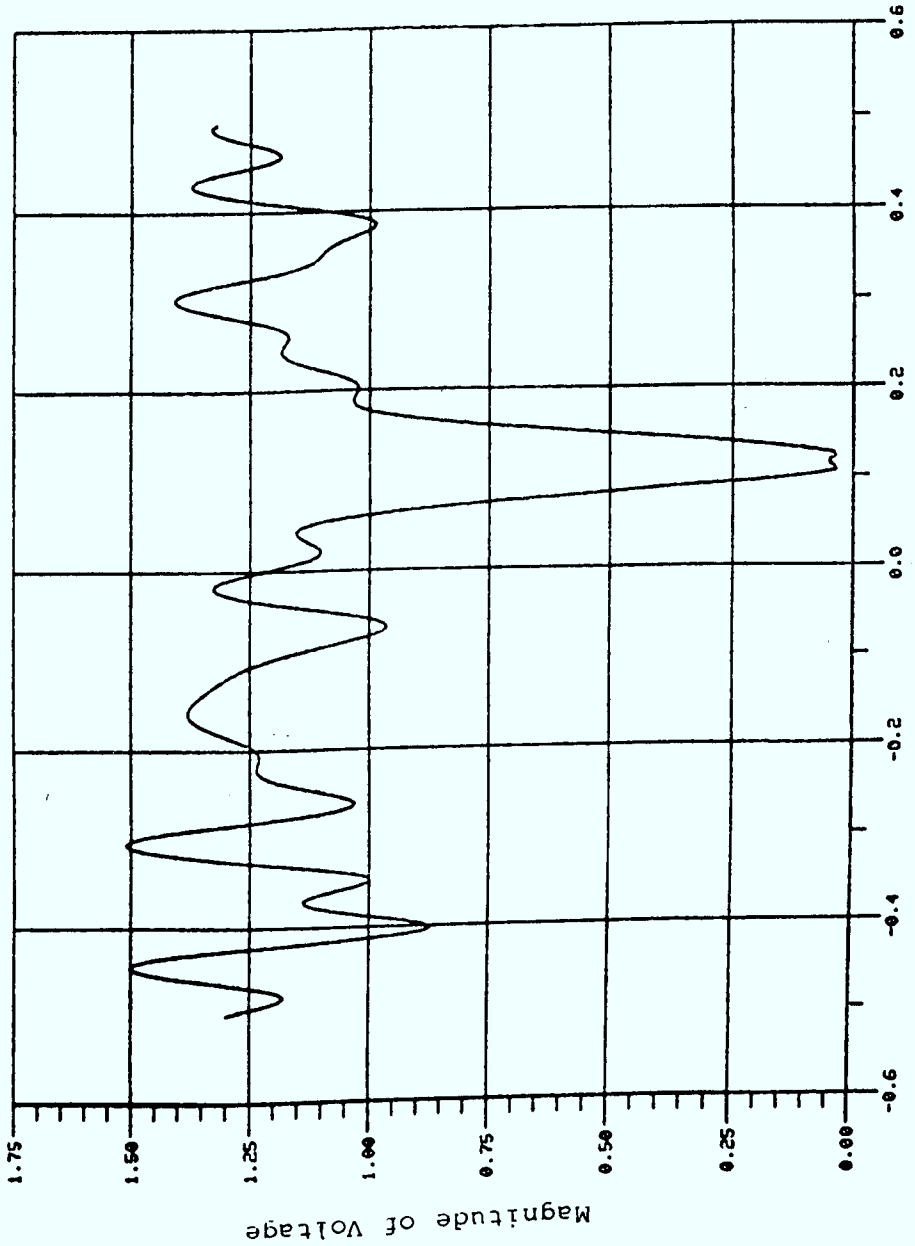


Figure B26 Frequency Response of Filter Adapted to Signal of Figure 25 - 16 Coefficients, 496 Products in Correlation Estimate.

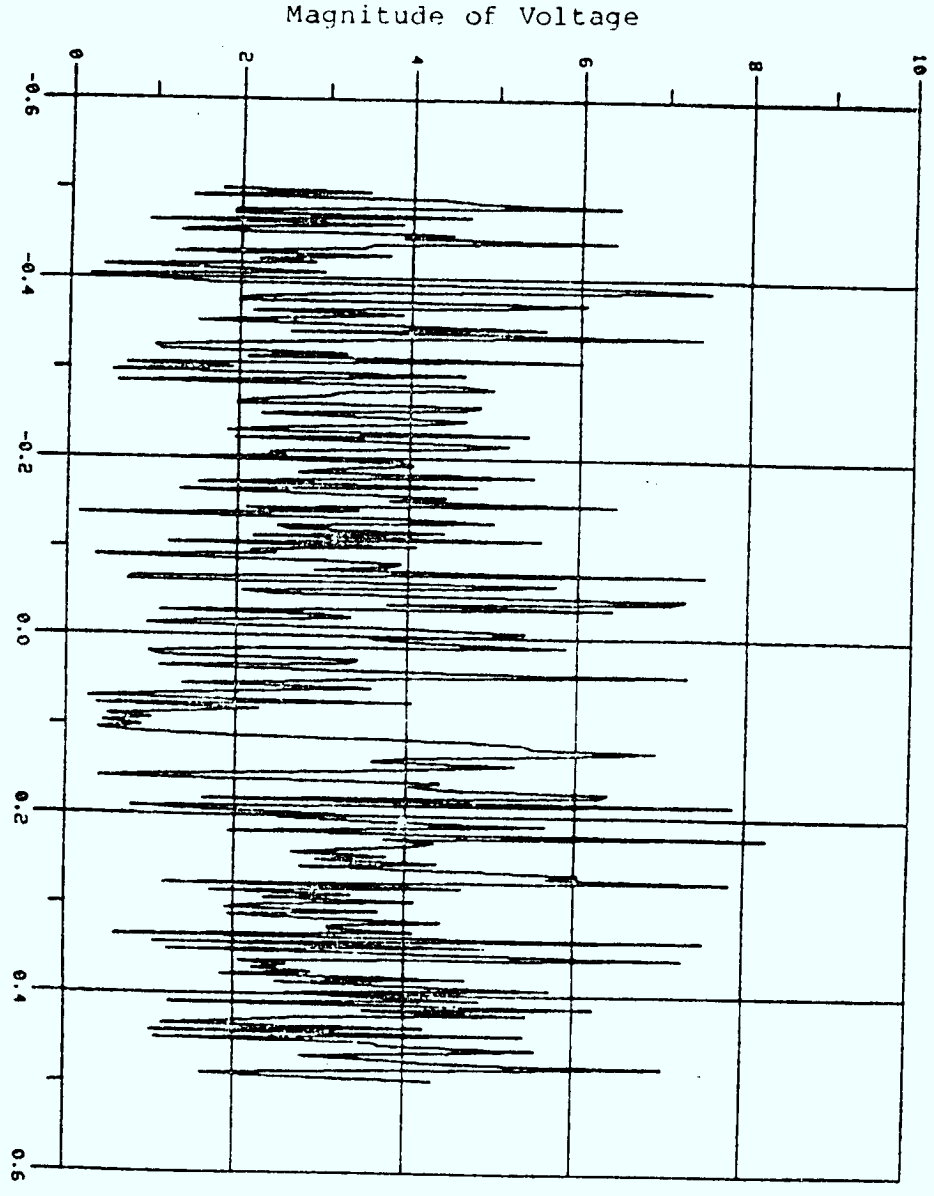


Figure B27 Spectrum of Output of Filter of Figure 26 for Input of Figure 25.

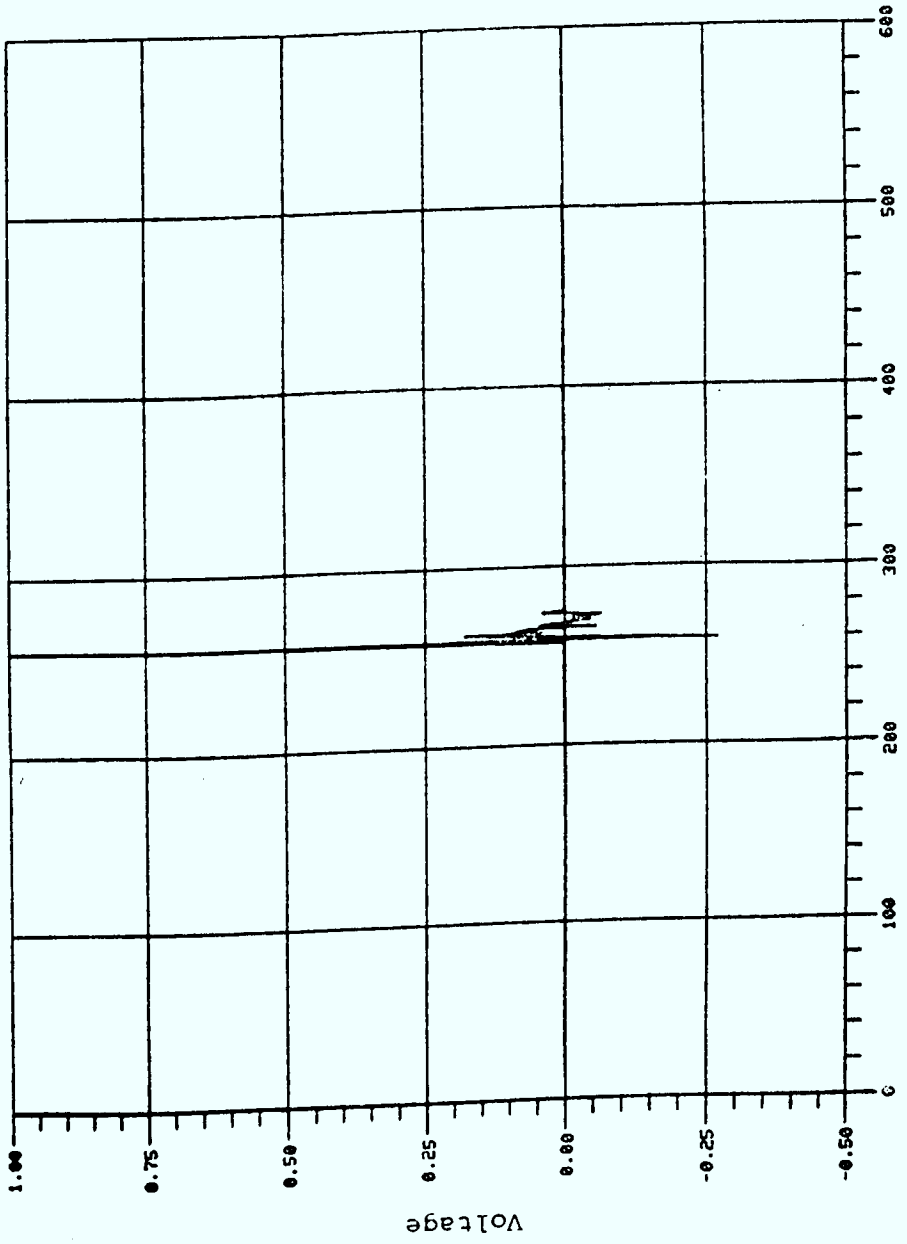


Figure B28 Impulse Response of Filter of Figure 26.

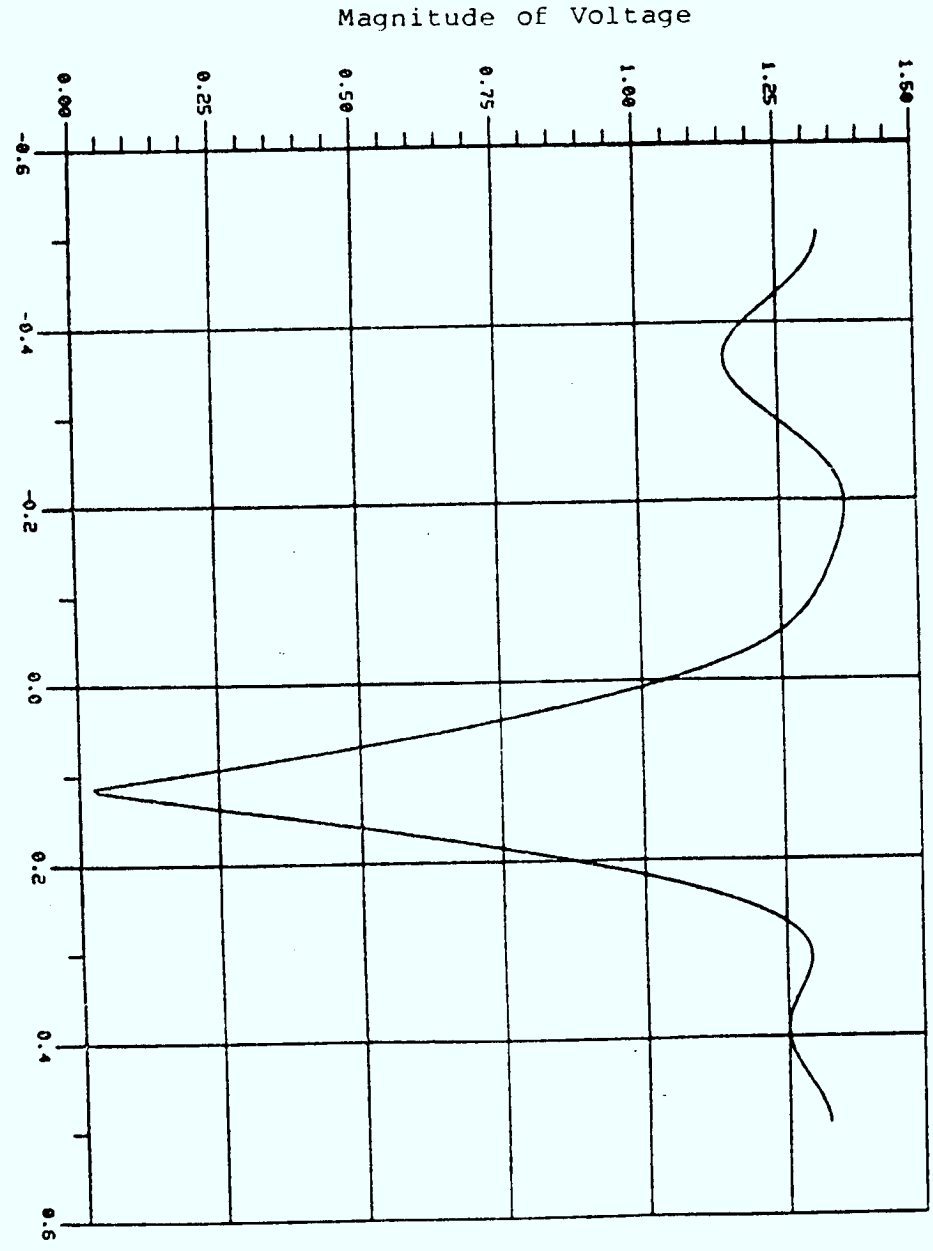


Figure B29 Frequency Response of Filter Adapted to Signal of Figure 25 - 4 Coefficients, 496 Products in Correlation Estimate.

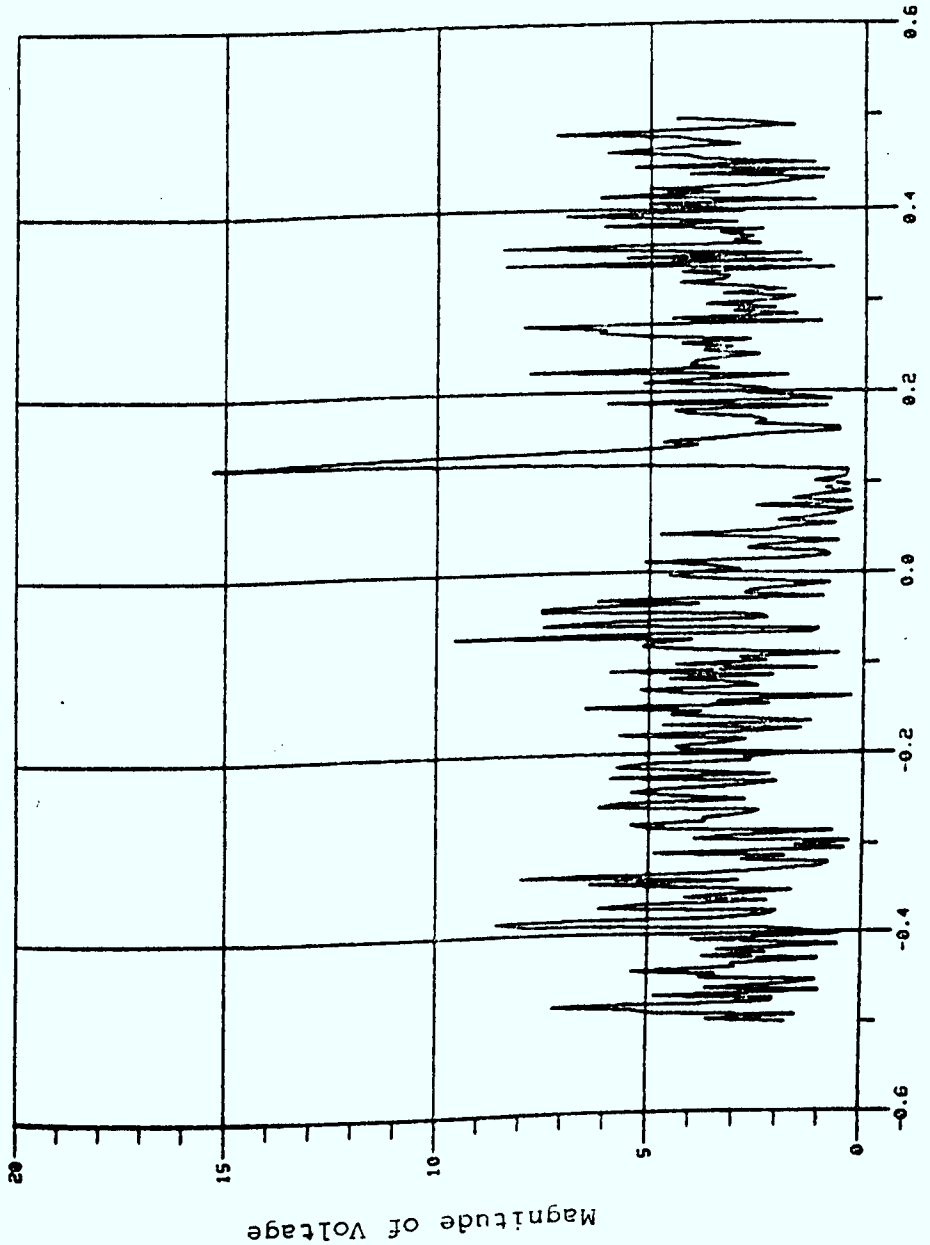


Figure B30 Spectrum of Output of Filter of Figure 29 for Input of Figure 25.

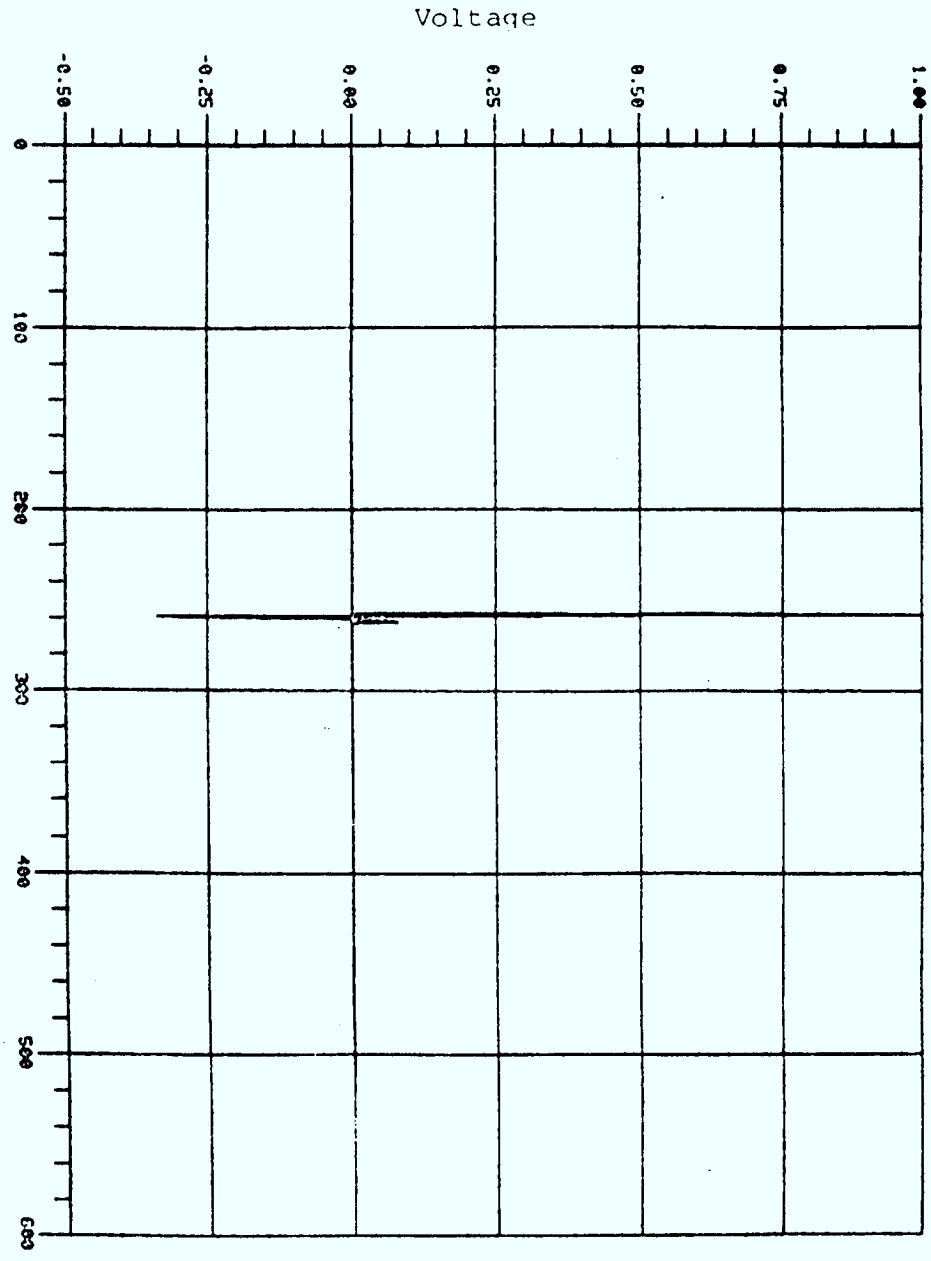


Figure B31 Impulse Response of Filter of Figure 29.

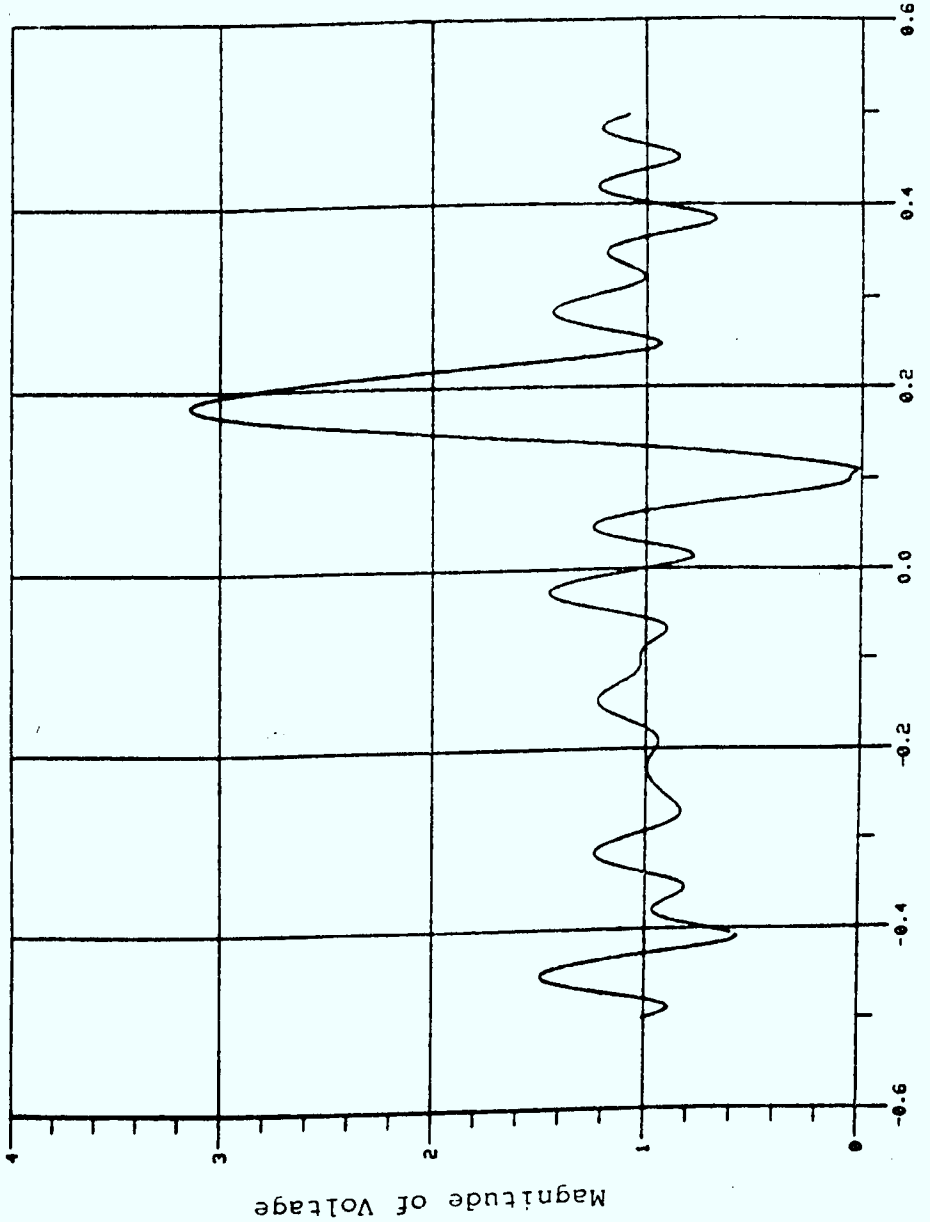


Figure B32 Frequency Response of Filter Adapted to Signal of Figure 25 - 16 Coefficients, 248 Products in Correlation Estimate.

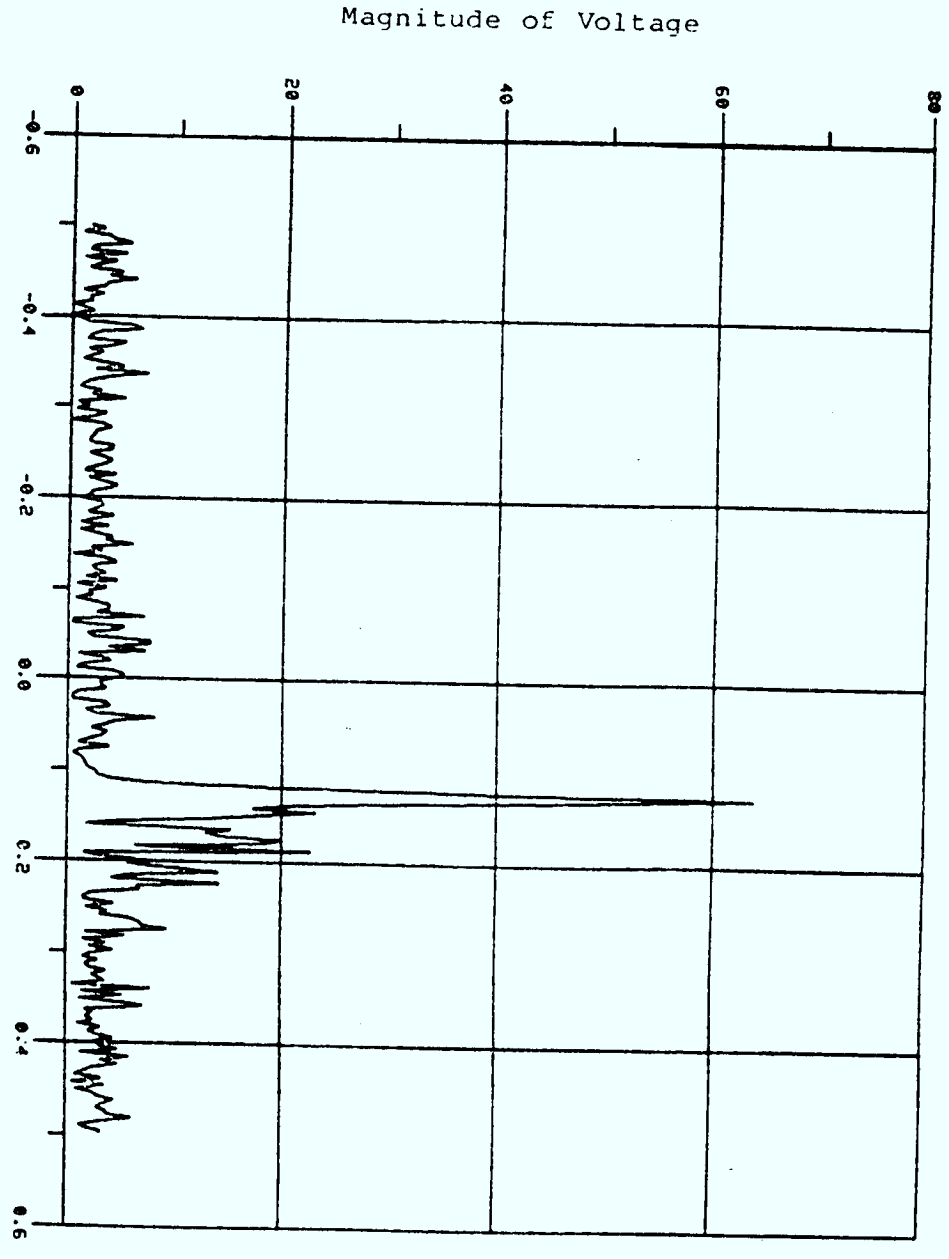


Figure B33 Spectrum of Output of Filter of Figure 32 for Input of Figure 25.

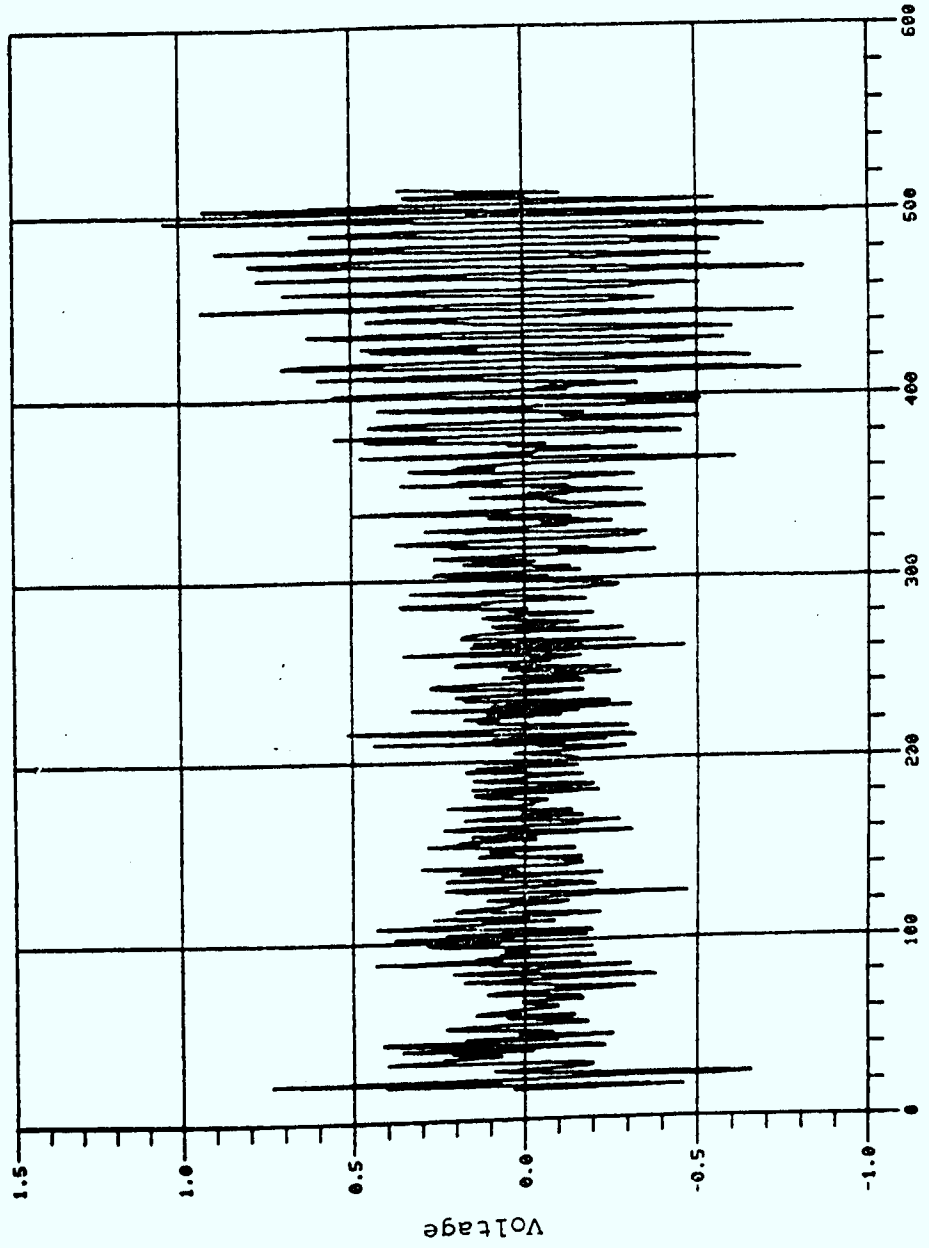


Figure B34 Time-Domain Output of Filter of Figure 32 for Input of Figure 25.

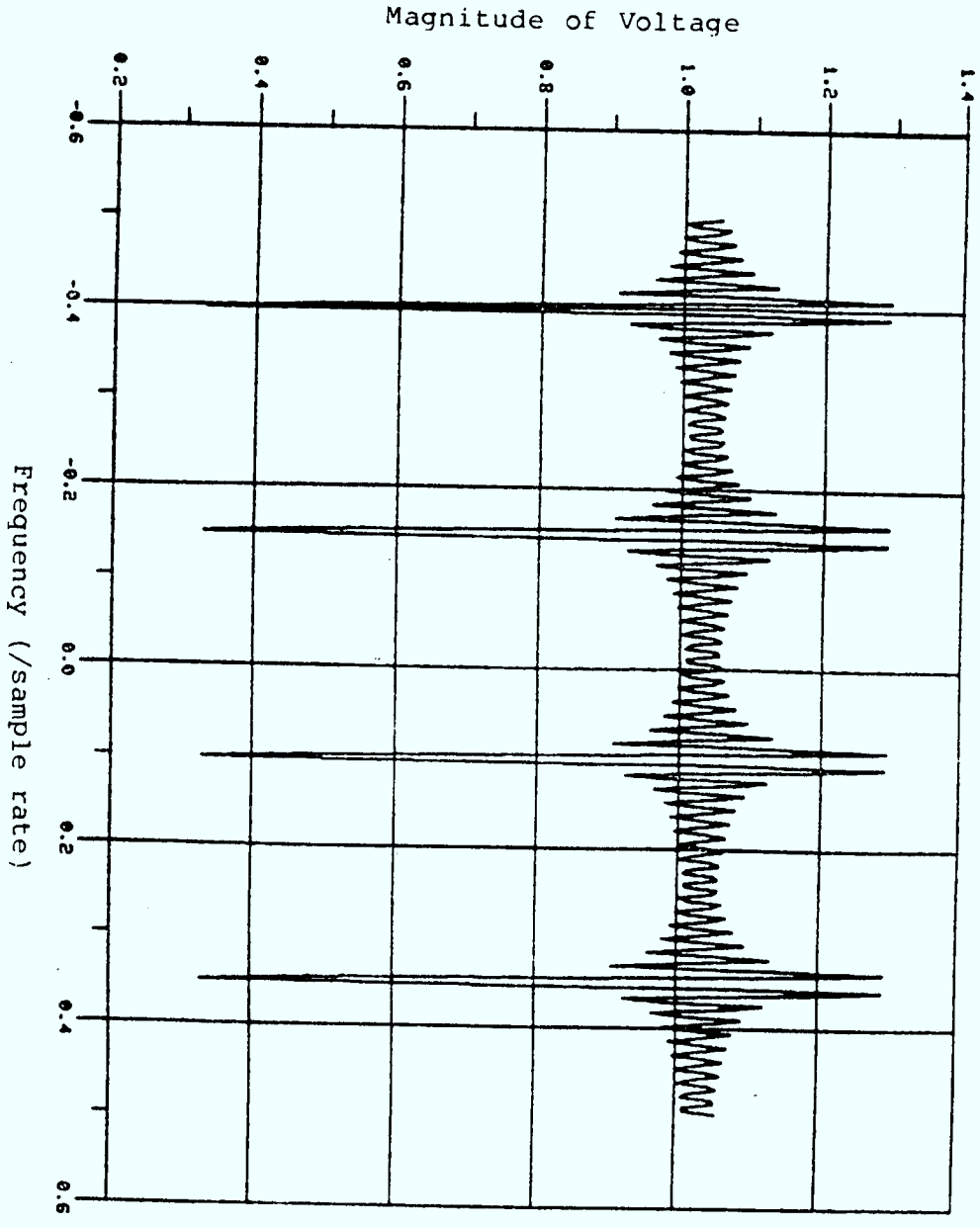


Figure B35 Frequency Response of Filter Adapted to Noise plus Sine Wave at Frequency of 0.1 - 16 Coefficients, Tap Spacing = 4.

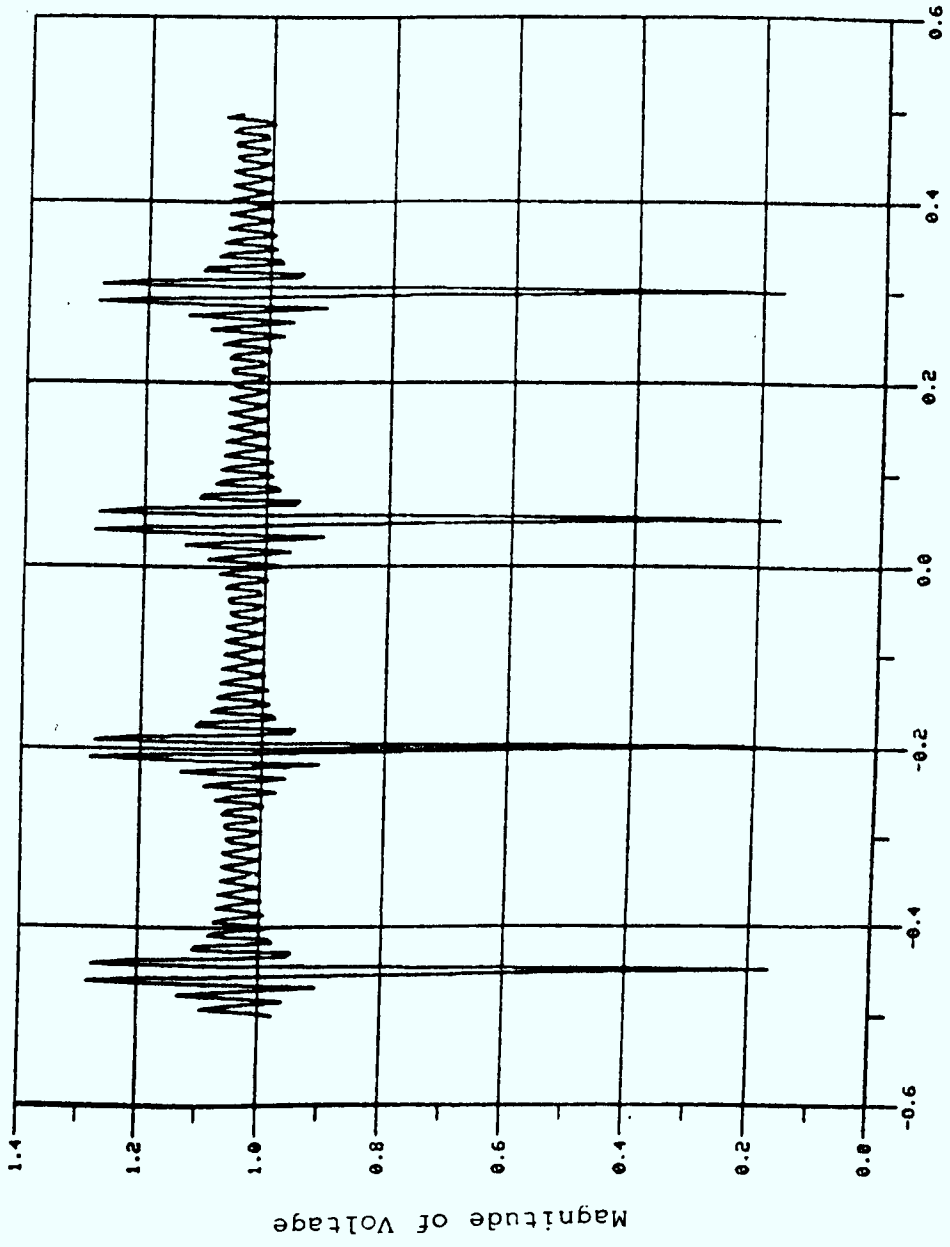


Figure B36 Frequency Response of Filter Adapted to Noise Plus Sine Wave at Frequency of 0.3 - 16 Coefficients, Tap Spacing = 4.

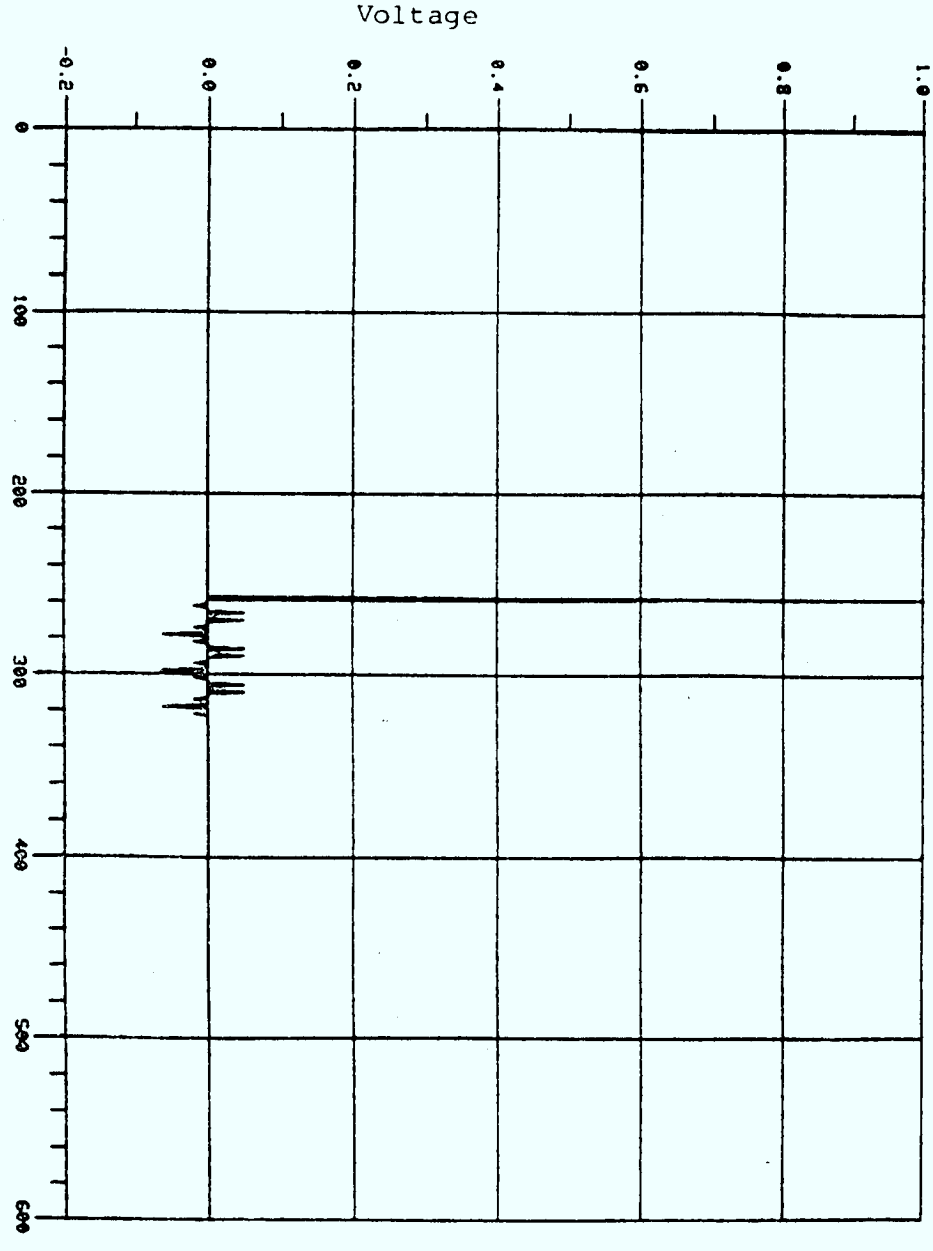


Figure B37 Impulse Response of Filter of Figure 36.

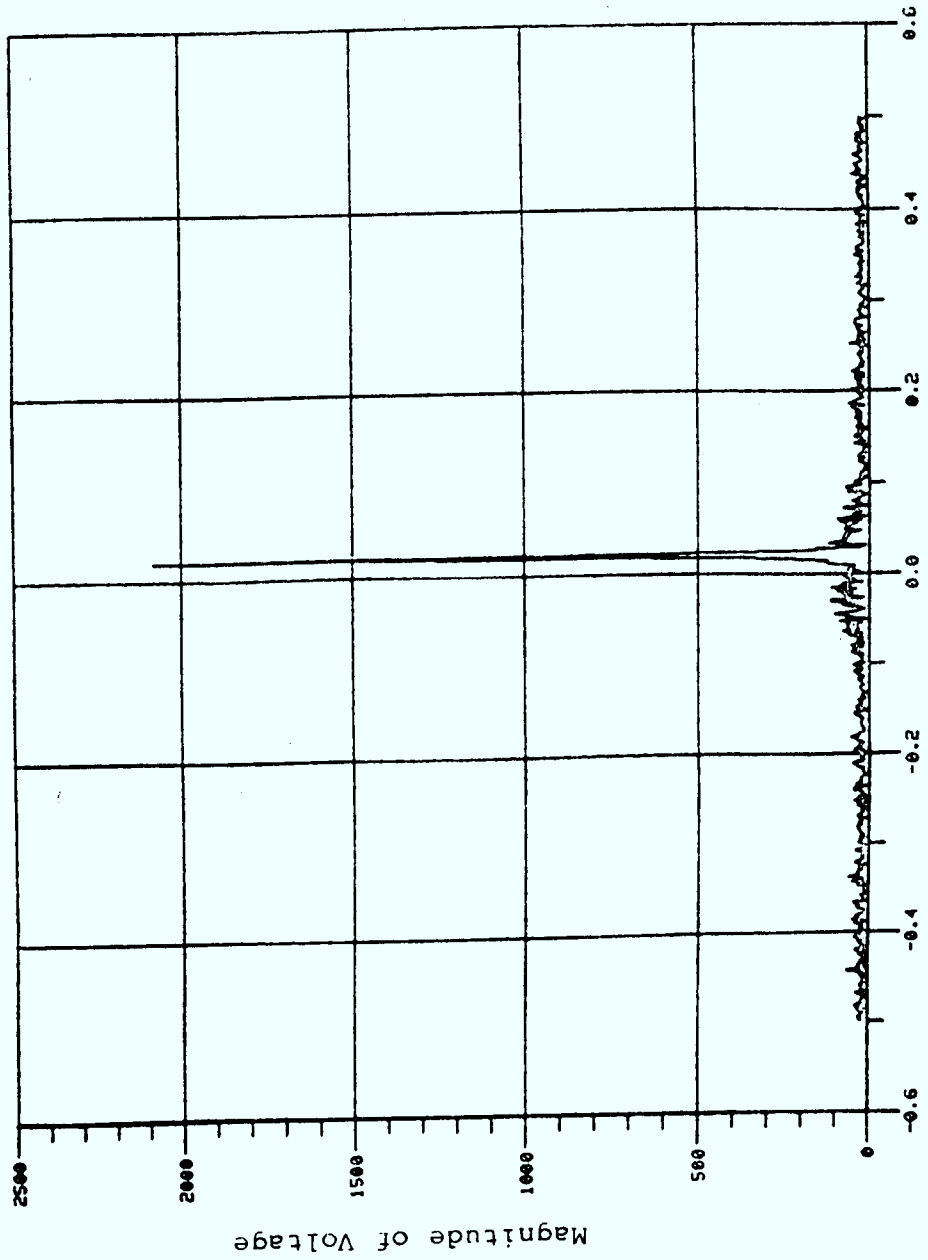


Figure B38 Spectrum of Input Signal M-Sequence + Sine wave + Noise.

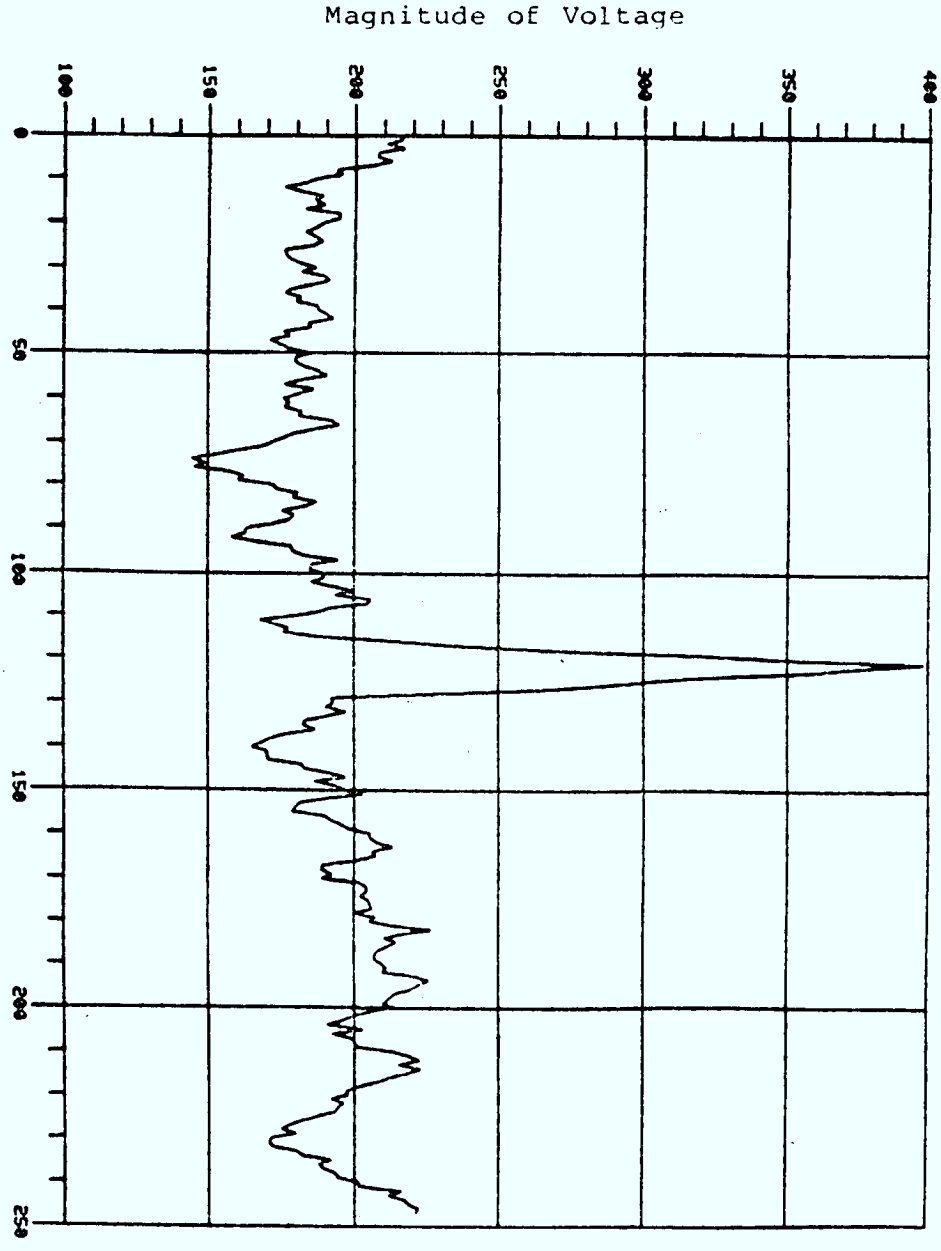


Figure B39 Output of Matched Filter (no excision) for Input of Figure 38.

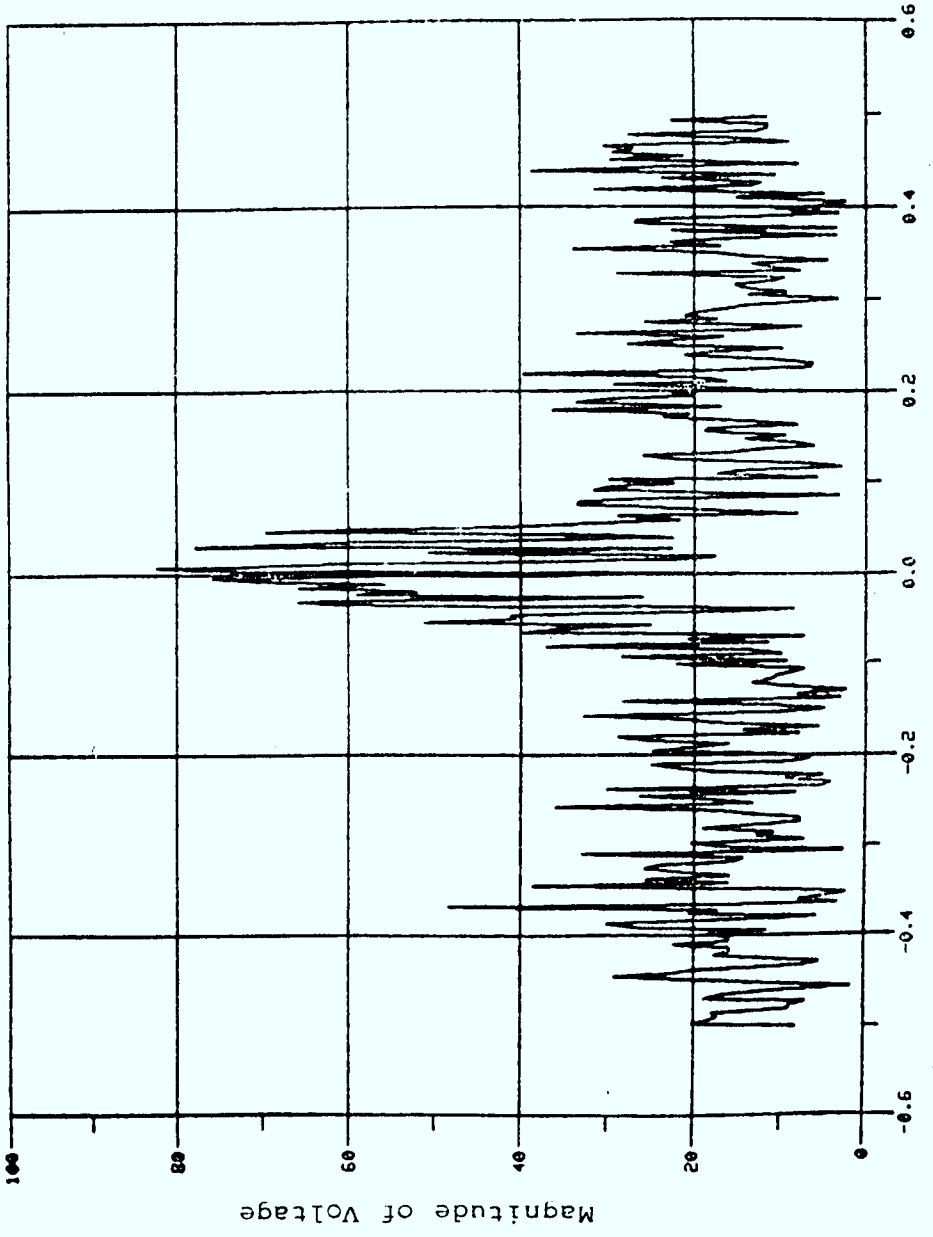


Figure B40 Spectrum of Output of Excision Filter Adapted to Signal of Figure 38 - 8 Coefficients, Tap Spacing = 8.

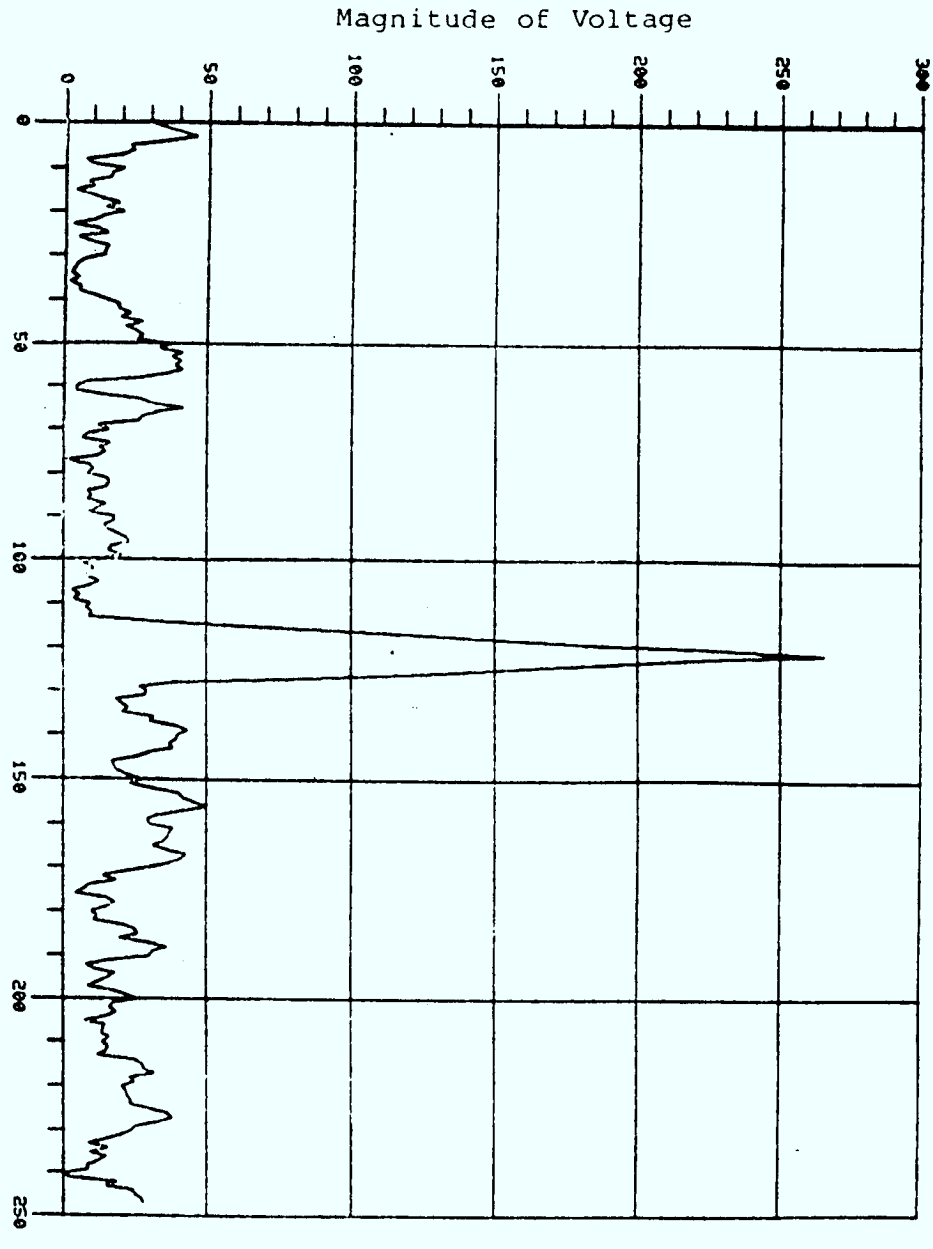


Figure B41 Output of Excision Filter followed by Matched Filter for Input of Figure 38.

APPENDIX C

ACQUISITION THRESHOLD CALCULATIONS

APPENDIX C

ACQUISITION THRESHOLD CALCULATIONS

In the frequency-hop and direct-sequence acquisition systems the squared magnitude of the received signal is integrated to determine when the signal has been acquired. This process is shown in Figure C1. The signal and rms noise levels V_s and V_n are specified at the input of this figure. The signal may or may not be filtered as shown. If not, the bandwidth W, defined as a fraction of the sample rate at that point, is taken as unity. For the following calculations the noise is assumed to be Gaussian at the input to the absolute-value block, but if the filter bandwidth is much less than the the bandwidth of the input noise the output of the filter can usually be assumed to be Gaussian even if the input is not. The integration consists of summing N samples, taking the square root, and comparing the result with a threshold, δ_a; acquisition (or a particular stage of it) is declared when δ_a is exceeded. If this occurs when only noise is present we have a "false alarm"; if it fails to occur when the desired signal is present along with the noise we have a "miss". We wish to compute the threshold required to provide a specified probability of false alarm, and the probability of a miss for this threshold for any specified signal level. It will not change the result, if, instead of making the comparison after the square-root block, we use a threshold of δ_a^2 at the output of the summation block, and this will simplify the computation.

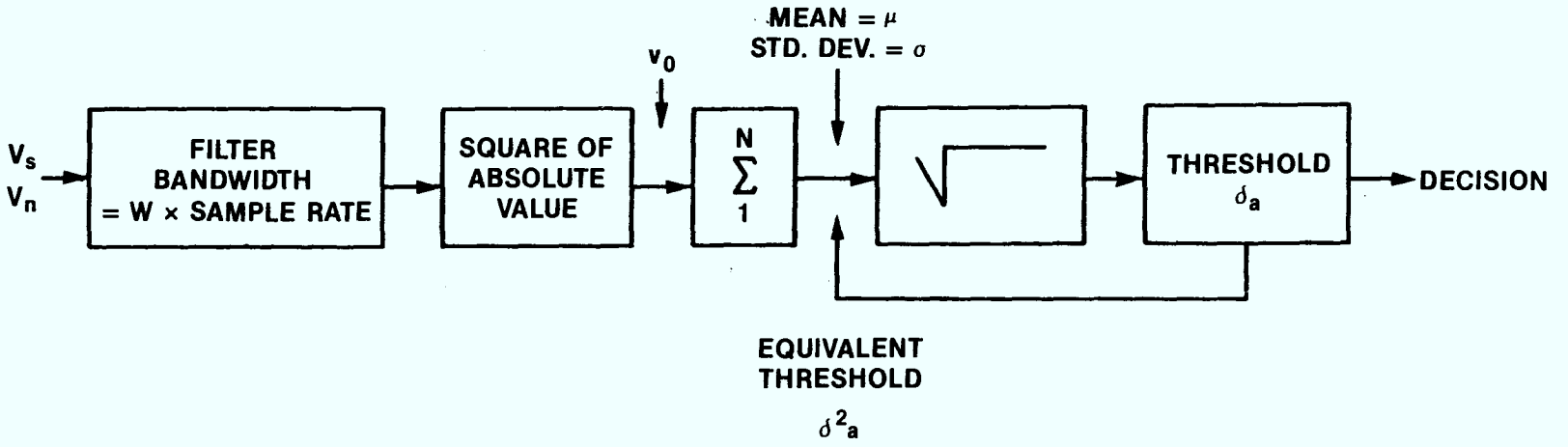


Figure C1 Definition of signals for Acquisition

Let N = number of samples summed,

V_s = signal voltage magnitude (root of mean power) at input,

V_n = rms noise voltage,

μ_n = mean voltage after summer for noise-only input,

μ_s = mean voltage after summer for signal-plus-noise input,

σ_n = standard deviation after summer for noise-only input,

σ_s = standard deviation after summer for signal-plus-noise input,

W = filter bandwidth as a fraction of the sample rate (2 times low-pass bandwidth),

δ_a = acquisition threshold,

υ_s = time-dependent input signal voltage,
$= \upsilon_{sr} + j0$

υ_n = time-dependent input noise voltage,
$= \upsilon_{nr} + j\upsilon_{ni}$,

υ_o = time-dependent voltage at the output of the squared magnitude block,

P_{fa} = probability of false alarm,

P_m = probability of a miss.

Consider first the case of noise only, which determines the false-alarm probability. The effect of the filter is to reduce the noise level by a factor of W in power; the signal is assumed to be unaffected by the filter. After the absolute value is taken the noise will have a Rayleigh distribution with a mean square of WV_n^2. After summing N values of the squared noise we get a Gaussian distribution (for large enough N) with a mean of:

$$\mu_n = NWV_n^2 \qquad (C1)$$

The filter will cause the noise to be correlated over 1/W samples. Thus, in the sum there will be, in effect, NW independent samples, each consisting of the sum of 1/W samples added in phase. Therefore, the standard deviation of the sum will be $\sqrt{N/W}$ times the mean square of the absolute value of the noise samples. That is,

$$\sigma_n = \sqrt{NW}\ V_n^2 \qquad (C2)$$

The probability of false alarm, P_{fa}, is the probability that a sample of a Gaussian distribution of mean, μ_n, and standard deviation, σ_n, will exceed the threshold, δ_a. Thus,

$$P_{fa} = \frac{1}{2}\ \mathrm{erfc}\ \left(\frac{\delta_a - \mu_n}{\sigma_n}\right), \qquad (C3)$$

where erfc is the complementary error function.

If P_{fa} is specified we can compute the required threshold from:

$$\delta_a = \mu_n + \sigma_n\ \mathrm{erfc}^{-1}(2\ P_{fa}). \qquad (C4)$$

In the case of direct-sequence acquisition, when the signal is not synchronized (This is the noise-only case) the noise level will be increased by the "sidelobes" of the direct-sequence autocorrelation

function, which exist at all delays except zero. The variance of this noise can be estimated from:

$$\text{variance} = V_s^2 / N_c \qquad \text{(C5)}$$

where N_c is the number of spreading-code elements per data symbol. The total noise is found by adding this variance to the variance of the noise at the input.

Now consider the case of signal plus noise in order to find the probability of a miss. The following analysis is not entirely rigorous; some intuitive steps are taken, but the result agrees well with tests made using the simulator. No loss of generality is suffered by making the complex input signal voltage, υ_s, have zero imaginary part. Since the noise is random we can take our reference so that it is always in phase with the signal. This will not affect the noise characteristics.

After the square-of-absolute-value block the voltage is:

$$\upsilon_o = \left|\upsilon_s + \sqrt{W}\,\upsilon_n\right|^2$$

$$= (\upsilon_{sr} + \sqrt{W}\,\upsilon_{nr})^2 + W\upsilon_{ni}^2$$

$$= \upsilon_{sr}^2 + W(\upsilon_{nr}^2 + \upsilon_{ni}^2) + 2\sqrt{W}\,\upsilon_{sr}\,\upsilon_{nr}$$

$$= \left|\upsilon_s\right|^2 + W\left|\upsilon_n\right|^2 + 2\sqrt{W}\,\upsilon_{sr}\,\upsilon_{nr}. \qquad \text{(C6)}$$

The last term will have a mean of zero and therefore the mean of υ_o is:

$$\overline{\upsilon_o} = \overline{\left|\upsilon_s\right|^2} + W\overline{\left|\upsilon_n\right|^2}$$

$$= V_s^2 + WV_n^2. \qquad \text{(C7)}$$

After N values are summed the mean value is:

$$\mu_s = N(V_s^2 + WV_n^2) \cdot \qquad (C8)$$

After the summation the output should have a Gaussian distribution which is symmetrical about the mean, and its variance should be increased by a factor N. We also note that the last term of (C6), although having a factor υ_{nr}, is essentially uncorrelated with either component of $|\upsilon_n|^2$ since it has a negative sign about half the time. Therefore, we can compute the variance from (C6) by ignoring the first term, which is constant, and treating the others as independent; that is we can sum their variances to obtain:

$$\sigma_s^2 = N/W \left(W^2V_n^4 + 2V_s^2WV_n^2\right) \qquad (C9)$$

and $$\sigma_s = \sqrt{N\left(WV_n^4 + 2V_s^2V_n^2\right)} \qquad (C10)$$

The probability of a miss is the probability that a sample from a Gaussian distribution with mean, μ_s, and standard deviation, σ_s, will fall below the threshold, δ_a. Thus:

$$P_m = \frac{1}{2}\,\mathrm{erfc}\left(\frac{\mu_s - \delta_a}{\sigma_s}\right) \cdot \qquad (C11)$$

The above applies to the probabilities for a single integration threshold. When more than one threshold crossing is required, as in most cases, the probabilities for overall acquisition can be found from the the individual probabilities.

APPENDIX D

EXAMPLE OF COMMAND FILE FOR BATCH OPERATION

```
$ SET NOVERIFY
$
$ ! SUPPRESS THE LISTING OF COMMAND FILES IN THE LOG FILE.
$
$ ! FILE DPSK0123.COM          14-JAN-1986
$
$ ! VMS BATCH COMMAND FILE TO RUN PROGRAM MODEM.EXE AS A BATCH JOB. THE
$ ! DEFAULT DIRECTORY MUST BE PASSED BY THE SUBMIT COMMAND TO PARAMETER P1
$ ! DPSK WITH 2 RAYLEIGH FADING PATHS AND CCIR NOISE Vd=4.2 Eb/Nc = 18 DB
$
$ ! P1 = DEFAULT DIRECTORY
$
$ !***** SETUP SECTION *****
$
$ SET DEFAULT ['P1']
$
$ ! SET THE DEFAULT DIRECTORY. AFTER PLACING A HEADER IN THE LOG FILE RUN THE
$ ! PROGRAM FEEDING IT COMMANDS.
$
$ WRITE SYS$OUTPUT " "
$ WRITE SYS$OUTPUT "BATCH JOB DPSK0123.COM RUNNING PROGRAM MODEM.EXE"
$ WRITE SYS$OUTPUT "DPSK, 2 RAYLEIGH PATHS, CCIR NOISE Vd=4.2, Eb/Nc=18 DB"
$
$ !***** RUN SECTION *****
$
$ RUN [VENIER.FREYSENG.MODEM]MODEM.EXE
BAT
FILE
DPSK 2 RAYLEIGH CCIR4.2 Eb/Nc=18
YES
DPSK0123
DATEN
ORG
3495
20
GENBIN
YES
```

```
PROCES
MODCOD
1
YES
NO
NO
YES
NO
YES
PSK
600
23
1.6330
NO
YES
HAN
MEDIUM
TERM
NO
NO
17837
NO
4
YES
3
SPC
1.999023080,-.9990233183,0.,0.,.1188570000E-05
YES
0.,0.,0.,0.,0.
YES
1.999023080,-.9990233183,0.,0.,.1188570000E-05
YES
0.,0.,0.,0.,0.
YES
1.999023080,-.9990233183,0.,0.,.1188570000E-05
YES
0.,0.,0.,0.,0.
```

```
YES
2
0
10
.7071,.7071
YES
0.,0.
YES
NO
HOPPER
NO
YES
100
NO
NO .
RECVR
NO
518995245
NO
YES
1
PSK
6
147
NO
NO
NO
NO
YES
2
YES
NO
.3084
4.2
NO
NO
NO
```

```
YES
IIR
1
BUT
.12
SIN
ZPF
NO
YES
YES
NO
NO
NO
NO
NO
NO
0
1
0
YES
YES
0
19
6
HQU
3400
95
0.
.1
52
YES
BYE
RETURN
ANAL
ERRCOM
3400
95
```

```
95
4000
YES
RETURN
TIME
STOP
$
$ !***** CLEANUP SECTION *****
$ .
$ WRITE SYS$OUTPUT "CLEANING UP FILES"
$ WRITE SYS$OUTPUT " "
$ PURGE/LOG DPSK0123.DAT
$ PURGE/LOG DPSK0123.LOG
$ PRINT/NOTIFY DPSK0123.DAT
$
$ WRITE SYS$OUTPUT " "
$ WRITE SYS$OUTPUT "FINISHED BATCH COMMAND FILE EXECUTION"
$ WRITE SYS$OUTPUT " "
$
$ EXIT
```

APPENDIX E

EXAMPLE OF OUTPUT FILE FROM BATCH OPERATION

```
Output file DPSK0123
Wideband H.F. Communication Simulation Program
14-JAN-86.                    Version number  7
"DPSK 2 RAYLEIGH CCIR4.2 Eb/No=18"

The seed used for the 31 bit data generation shift register.
Register = 11101011011010000111110110111100
The number of original binary values generated is      3495
The bit rate for this data in bits/second is    20.00000
The contents of the shift register after binary generation are
Register = 10100000100111001010101001000

Using the MODCOD process
The modulation symbol size is      1 bits resulting
in       2 possible symbol states

Modulation symbols formed
Ordinary symbol formation chosen
No complex zeros are being added to the input data
Modulation symbols being differentially encoded
The number of integer symbols produced is      3495
The new data rate is    20.00000      symbols/second

Modulation samples being generated.
Phase shift keying chosen.
The fixed initial modulation phase is    23.00000      degrees
The number of samples per modulation symbol is        600
The peak envelope voltage is    1.633000      volts
The energy per modulation symbol before post-modulation
processing is  0.1333345     joules
The energy is constant for all symbols
Pulse shaping selected.
Hanning pulse shaping chosen
The energy per modulation symbol after pulse shaping
is a constant  0.5000042E-01 joules
The total number of complex samples after modulation is    2097000
```

```
The new data rate is    12000.00      samples/second

Using the MEDIUM process

Medium parameters taken from the terminal
The data is not frequency hop encoded
The random number seed is        17837

The medium parameters for path number      1:
Rayleigh fading
IIR filtering being done on the gaussian factors

The number of recursive filter sections is      3
Coefficient no.     real part           imaginary part

The coefficients for section number     1
        1          1.999023080         0.0000000000E+00
        2         -0.9990233183        0.0000000000E+00
        3          0.0000000000E+00    0.0000000000E+00
        4          0.0000000000E+00    0.0000000000E+00
        5          0.1188569963E-05    0.0000000000E+00

The coefficients for section number     2
        6          1.999023080         0.0000000000E+00
        7         -0.9990233183        0.0000000000E+00
        8          0.0000000000E+00    0.0000000000E+00
        9          0.0000000000E+00    0.0000000000E+00
       10          0.1188569968E-05    0.0000000000E+00

The coefficients for section number     3
       11          1.999023080         0.0000000000E+00
       12         -0.9990233183        0.0000000000E+00
       13          0.0000000000E+00    0.0000000000E+00
       14          0.0000000000E+00    0.0000000000E+00
       15          0.1188569963E-05    0.0000000000E+00

The initial delay is      0 samples or  0.0000000E+00 seconds
```

```
The delay increment is     10 samples or  0.8333334E-03 seconds

Table of      2 tap multipliers and doppler shifts
Tap      Amplitude        Doppler shift     Doppler frequency
Number   Multiplier       degrees/sample       hertz
    1     0.7071000        0.0000000E+00     0.0000000E+00
    2     0.7071000        0.0000000E+00     0.0000000E+00

The number of medium paths was      1
The number of complex values after passing through the medium is  2097010
The data rate remains    12000.00

Using the HOPPER process

Decimation being done
The decimation rate is         100
The total number of values after decimation is      20970
The new data rate is    120.0000      values/second

Using the RECVR process
The data is not frequency hop encoded
The random number seed is    518995245
The input data was modulated
The modulation symbol size is      1 bits resulting
in         2 possible symbol states
Phase shift keying chosen.
The fixed initial modulation phase is    147.0000      degrees
The number of samples per modulation symbol is          6
Direct sequence spreading has not been used on the input data

Noise signal samples being added.
There are          1 noise addition frequency ranges

The          1 noise addition frequency range has a lowest
frequency of  0.0000000E+00 kilohertz.
Complex noise signal samples being added for this frequency range
CCIR noise to be generated
```

```
The CCIR noise rms voltage is  0.3084000     volts
The ratio of the rms to the mean is   4.200000     decibels
The CCIR noise power spectral density is  0.7925380E-03 watts/hertz

Front end filtering being done
Recursive filtering chosen.
Butterworth recursive filter chosen.
The filter cutoff frequency is  0.1200000     times the data rate
or   14.40000     hertz
The order of the filter is          1
The filter gain factor is   1.000000
The filter group delay is   1.262856     in terms of the
sample period and  0.1052380E-01 in seconds

The number of recursive filter sections is      1
Coefficient no.     real part           imaginary part

The coefficients for section number      1
        1          0.4327386320         0.0000000000E+00
        2          0.0000000000E+00     0.0000000000E+00
        3           1.000000000         0.0000000000E+00
        4          0.0000000000E+00     0.0000000000E+00
        5          0.2836306691         0.0000000000E+00


Ordinary demodulation being used
Differential phase shift keying demodulation is being used

Mode 1 - Nc acquisition and tracking picked

The receiver propogation delay is  0.0000000E+00 samples
The receiver front end delay is   1.000000     samples
The receiver bandwidth reducer delay is  0.0000000E+00 samples
The initial sample reference delay is   1.000000     samples

During the      1 run, the receiver processed     20970 input samples,
resulting in      3494 output values
```

Samples were taken every 6 run input sample
The total number of samples for the run was 3495
Starting with the 95 sample 3400 samples are being looked at

Histogram of samples of the demodulator tracking voltage

Cell number	lower limit	upper limit	number of values	% of total
1	-00	0.0000000E+00	0	0.00
2	0.0000000E+00	0.1000000	36	1.06
3	0.1000000	0.2000000	128	3.76
4	0.2000000	0.3000000	243	7.15
5	0.3000000	0.4000000	291	8.56
6	0.4000000	0.5000000	337	9.91
7	0.5000000	0.6000000	376	11.06
8	0.6000000	0.7000000	389	11.44
9	0.7000000	0.8000000	345	10.15
10	0.8000000	0.9000000	267	7.85
11	0.9000000	1.000000	232	6.82
12	1.000000	1.100000	183	5.38
13	1.100000	1.200000	159	4.68
14	1.200000	1.300000	138	4.06
15	1.300000	1.400000	102	3.00
16	1.400000	1.500000	70	2.06
17	1.500000	1.600000	39	1.15
18	1.600000	1.700000	29	0.85
19	1.700000	1.800000	19	0.56
20	1.800000	1.900000	9	0.26
21	1.900000	2.000000	7	0.21
22	2.000000	2.100000	1	0.03
23	2.100000	2.200000	0	0.00
24	2.200000	2.300000	0	0.00
25	2.300000	2.400000	0	0.00
26	2.400000	2.500000	0	0.00
27	2.500000	2.600000	0	0.00
28	2.600000	2.700000	0	0.00
29	2.700000	2.800000	0	0.00

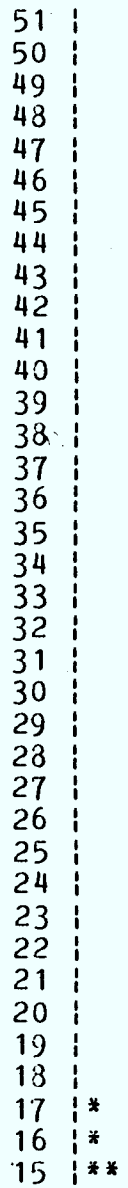
30	2.800000	2.900000	0	0.00
31	2.900000	3.000000	0	0.00
32	3.000000	3.100000	0	0.00
33	3.100000	3.200000	0	0.00
34	3.200000	3.300000	0	0.00
35	3.300000	3.400000	0	0.00
36	3.400000	3.500000	0	0.00
37	3.500000	3.600000	0	0.00
38	3.600000	3.700000	0	0.00
39	3.700000	3.800000	0	0.00
40	3.800000	3.900000	0	0.00
41	3.900000	4.000000	0	0.00
42	4.000000	4.100000	0	0.00
43	4.100000	4.200000	0	0.00
44	4.200000	4.300000	0	0.00
45	4.300000	4.400000	0	0.00
46	4.400000	4.500000	0	0.00
47	4.500000	4.600000	0	0.00
48	4.600000	4.700000	0	0.00
49	4.700000	4.800000	0	0.00
50	4.800000	4.900000	0	0.00
51	4.900000	5.000000	0	0.00
52	5.000000	+00	0	0.00

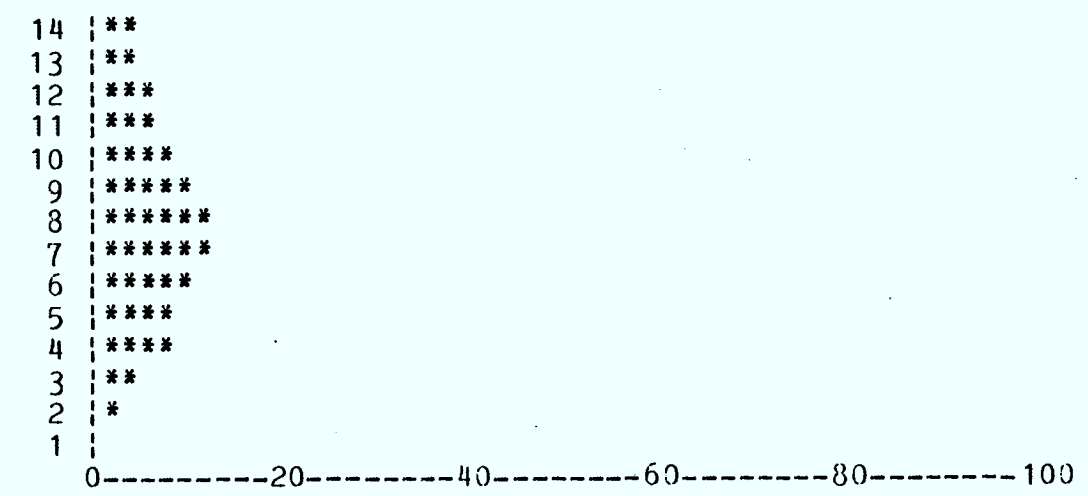
The histogram contains 3400 values in 52 cells
The first cell contains values less than 0.0000000E+00
The last cell contains values equal to or greater than 5.000000
The size of the remaining cells is 0.1000000

The mean of the values is 0.7240425
The standard deviation of the values is 0.3706947
The mean of the squares of the values is 0.6616116
The maximum is 2.041387 while the minimum is 0.2141030E-01

Bar graph corresponding to the histogram

52 |


51
50
49
48
47
46
45
44
43
42
41
40
39
38
37
36
35
34
33
32
31
30
29
28
27
26
25
24
23
22
21
20
19
18
17
16
15
*
*
**



The vertical numbers are cell numbers, while the
horizontal numbers are percentage
Each * represents 2% of the total number of values

In 1 runs, a total of 20970 input samples out
of 20970 possible samples was sent through the receiver
This resulted in 3494 final values with a data
rate of 20.00000 values/second.

Error analysis of selected original and final binary data bits

Number of data bits compared - 3400
The total number of original data bits - 3495
The original data bit rate in bits/second - 20.00000
Original data comparison starting point - 95
The total number of final data bits - 3494
The final data bit rate in bits/second - 20.00000
Final data comparison starting point - 95
The number of comparison windows of size 4000 is 1

The largest period between errors measurable is - 3999

Window number 1 with 3400 values
The original data window starting index - 95
The final data window starting index - 95

Indices for bits in error

Original Index	Original Value	Final Index	Window Index
123	1	123	29
170	1	170	76
172	0	172	78
186	1	186	92
188	0	188	94
189	0	189	95
220	0	220	126
310	1	310	216
636	0	636	542
712	1	712	618
1333	0	1333	1239
1414	0	1414	1320
1415	0	1415	1321
1503	0	1503	1409
1656	1	1656	1562
1657	0	1657	1563
1778	1	1778	1684
1795	0	1795	1701
2374	0	2374	2280
2642	1	2642	2548
2658	0	2658	2564
2712	1	2712	2618
2867	1	2867	2773
2868	1	2868	2774
2877	1	2877	2783
2878	1	2878	2784
2879	1	2879	2785

2880	1	2880	2786
2882	1	2882	2788
2922	0	2922	2828
3230	0	3230	3136

The number of bits in error in the window - 31
The fractional bit error rate for the window - 0.009118
The total number of periods in the window - 30

The periods between errors in the window and their frequency

Period	Occurrence	Percentage
1	7	23.33
2	3	10.00
9	1	3.33
14	1	3.33
16	1	3.33
17	1	3.33
31	1	3.33
40	1	3.33
47	1	3.33
54	1	3.33
76	1	3.33
81	1	3.33
88	1	3.33
90	1	3.33
121	1	3.33
153	1	3.33
155	1	3.33
268	1	3.33
308	1	3.33
326	1	3.33
579	1	3.33
621	1	3.33

% Clock time taken by selected routines

Routine	% Time	Routine	% Time	Routine	% Time

GENBIN	0.00	INTERL	0.00	VIEW	0.00
JAM	0.00	BLOCK	0.00	ERRCOM	0.01
NOISE	0.00	CONVL	0.00	COMPRB	0.00
FILOUT	0.00	OUTCOD	0.00	COMPRC	0.00
FILIN	0.00	MODUL	9.31	FFTCOM	0.00
REPEAT	0.00	MODADD	0.00	MEDIUM	86.99
ADD	0.00	GROUP	0.01	RECVR	3.17
MOVE	0.00	ENCOD	0.00	FILTER	0.00
DECIM	0.51	CLIP	0.00	INGRP	0.00
DEBLCK	0.00	DETCOD	0.00	DECONV	0.00
DEINTL	0.00	MULT	0.00	HISTO	0.00
INSERT	0.00	ABSOL	0.00		

Program stopped

APPENDIX F

EXAMPLE OF LOG FILE FROM BATCH OPERATION

```
$ SET NOVERIFY

BATCH JOB DPSK0123.COM RUNNING PROGRAM MODEM.EXE
DPSK, 2 RAYLEIGH PATHS, CCIR NOISE Vd=4.2, Eb/No=18 DB
Wideband H.F. Communication Simulation Program
14-JAN-86.                    Version number  7

The maximum amount of data is  2097152

Is the program running from the terminal or in batch?
BAT - in batch
TRM - from the terminal
Mode? -
Choose the output device
Terminal or batch job log file - TRM
Ascii output file - FIL
Choice? -
File description (up to 32 characters)? -
"DPSK 2 RAYLEIGH CCIR4.2 Eb/No=18"
Correct? - YES or NO? -
Filename? -

Output file DPSK0123          successfully created

Command              Function

HELP    - Give a menu of available main level commands
STOP    - Stop the program
FORM    - Determine the output device
TIME    - Check the time that processing routines are taking
FILE    - Perform file I/O
DATEN   - Enter or generate original or final data
ANAL    - Analyze or display data
PROCES  - Send original data along the communication link
MODIFY  - Modify data
TELL    - Type the amount of original and final data
DESCRP  - Type a description of the program
```

```
Main level command? -
Original or final data?
ORG or FIN -
Enter the number of data values -
Enter the data rate in values/second -

Command             Function

GENBIN - Generate binary data using a pseudorandom sequence
ENBIN  - Enter binary data from the keyboard
NOISE  - Generate complex noise samples
JAM    - Generate complex jamming signal samples
COMPEN - Enter complex samples from the terminal
CANCEL - Abort and return to the main command level

Data entry command? -
Is the default 31 bit shift register seed to be used?
YES or NO? -

Main level command? -

Process             Function

HELP    -  Give a menu of available processes
RETURN  -  Go back to the main command level
NULL    -  Nothing process
TELL    -  Type the amount of original and final data
BITSRC  -  Preliminary bit processing
MODCOD  -  Generation of modulation samples
HOPPER  -  Filtering, decimation, clipping, and hop encoding
MEDIUM  -  Channel propagation
RECVR   -  Reception of data
BITSNK  -  Post-reception processing of bits

Process? -
```

```
The modulation symbol size in bits raised to the power
two determines the number of possible symbol states.
This is not to be confused with the number of bits encoded
in multiple code shift keying symbols.
Modulation symbol size? - maximum of    24 -

Symbols to be formed from bits?
YES or NO? -
Multiple code shift keying desired?
YES or NO? -


Inverse gray encoding to be applied?
YES or NO? -

Differential encoding of symbols desired?
YES or NO? -


Direct sequence symbols to be added?
YES or NO? -

Modulation to be done?
YES or NO? -

The avaliable modulation types are:
PSK = phase shift keying
FSK = single-tone frequency shift keying (one frequency set)
MTK = multi-tone frequency shift keying (two or more one frequency sets)
MSK = minimum shift keying
Modulation type? -

The number of samples per modulation symbol?
minimum of 2 maximum of      600 -
Initial modulation phase in degrees? -
```

```
The peak envelope voltage along with the data rate and
the type of modulation chosen determines the energy per
modulation symbol. Post modulation processing can modify
the symbol energy.
Peak envelope voltage in volts? -

Selected transitions to be applied?
YES or NO? -
Pulse shaping to be done?
YES or NO? -

The type of shaping?
HAN = Hanning or sine squared
MDH = Modified hanning
HAM = Hamming
SIN = Sine of a sine of a sine
-



Process? -

Source of medium parameters input?
Terminal - TERM
Ascii file - FILE
Choice? -

Medium file to be created or not?
YES or NO? -
Is the default random number generator seed to be used?
YES or NO? -
Enter a seed? (perferably a large odd integer) -
Is the data frequency hop encoded?
YES or NO? -
```

```
Enter the medium parameters for path number      1:

Enter the parameters for the      1 frequency range:
The transmission mode?
1 = Perfect transmission (no delays, multipliers etc.)
2 = Complete blockage (nothing transmitted)
3 = Fixed transmission (fixed multipliers)
4 = Rayleigh fading transmission (gaussian multipliers)
-
Rayleigh fading multipliers to be filtered?
YES or NO? -
The number of filter sections? - maximum of      8 -

The type of recursive filter?
BUT = butterworth filter
CHB = chebyshev filter
RES = resonant filter
SPC = special filter (user enters the coefficients)
-

A row of 5 complex coefficients is entered for each of the      3
IIR filter sections. The entered coefficients should be
separated by commas

Enter a row of 5 real coefficient components for the      1 section
-
The row of entered real coefficient components:
  1.999023     -0.9990233      0.0000000E+00  0.0000000E+00  0.1188570E-05
Correct? - YES or NO? -

Enter a row of 5 imaginary coefficient components for the      1 section
-
The row of entered imaginary coefficient components:
 0.0000000E+00  0.0000000E+00  0.0000000E+00  0.0000000E+00  0.0000000E+00
Correct? - YES or NO? -

Enter a row of 5 real coefficient components for the      2 section
```

```
-
The row of entered real coefficient components:
  1.999023     -0.9990233      0.0000000E+00  0.0000000E+00  0.1188570E-05
Correct? - YES or NO? -

Enter a row of 5 imaginary coefficient components for the     2 section
-
The row of entered imaginary coefficient components:
 0.0000000E+00  0.0000000E+00  0.0000000E+00  0.0000000E+00  0.0000000E+00
Correct? - YES or NO? -

Enter a row of 5 real coefficient components for the     3 section
-
The row of entered real coefficient components:
  1.999023     -0.9990233      0.0000000E+00  0.0000000E+00  0.1188570E-05
Correct? - YES or NO? -

Enter a row of 5 imaginary coefficient components for the     3 section
-
The row of entered imaginary coefficient components:
 0.0000000E+00  0.0000000E+00  0.0000000E+00  0.0000000E+00  0.0000000E+00
Correct? - YES or NO? -
The number of taps for this frequency range?
maximum of    64 taps -
The initial delay in terms of the number of samples?
maximum of     1024 -
The delay increment in terms of the number of samples?
maximum of       16 -

Tap amplitude multipliers and doppler shifts are entered
in rows separated by commas

Enter a row of  2 real tap amplitude multipliers
The multipliers should be in ratio form <= 1
-
The row of entered tap amplitude multipliers:
 0.7071000      0.7071000
```

```
Correct? - YES or NO? -

Enter a row of  2 real tap doppler shifts
The doppler shifts represent the phase change per sample interval
Enter in degrees -
The row of entered tap doppler shifts:
 0.0000000E+00  0.0000000E+00
Correct? - YES or NO? -

More paths to be used?
YES or NO? -

Process? -

Filtering to be done?
YES or NO? -

Decimation to be done?
YES or NO? -

The decimation rate? ie.
1 = All values preserved
3 = 2, 5, 8, 11, 14 ... values preserved
5 = 3, 8, 13, 18, 23 ... values preserved
10 = 5, 15, 25, 35, 45 ... values preserved
-


Saturation amplification (clipping) to be done?
YES or NO? -

Frequency hop encoding to be done?
YES or NO? -

Process? -
Is the default random number generator seed to be used?
YES or NO? -
```

```
Enter a seed? (perferably a large odd integer) -
Is the data frequency hop encoded?
YES or NO? -


Has the data been modulated?
YES or NO? -

The modulation symbol size in bits raised to the power
two determines the number of possible symbol states.
This is not to be confused with the number of bits encoded
in multiple code shift keying symbols.
Modulation symbol size? - maximum of    24 -

The avaliable modulation types are:
PSK = phase shift keying
FSK = single-tone frequency shift keying (one frequency set)
MTK = multi-tone frequency shift keying (two or more one frequency sets)
MSK = minimum shift keying
Modulation type? -

The number of samples per modulation symbol?
minimum of 2 maximum of      600 -
Initial modulation phase in degrees? -


Selected transitions to be applied?
YES or NO? -
Pulse shaping to be done?
YES or NO? -

Please wait receiver sample delay vector initialization
taking place

Is multiple code shift keying being used?
YES or NO? -
```

```
Has direct sequence spreading been used on the input data?
YES or NO? -

Complex noise signal samples to be added?
YES or NO? -

Enter the noise addition parameters:

Enter the parameters for the     1 frequency range:
Noise to be added for the frequency range?
1 = No noise signal samples to be added
2 = Noise signal samples to be added
-

CCIR noise to be generated?
YES or NO? -

CCIR noise to be filtered?
YES or NO? -
CCIR noise rms voltage in volts? -
The ratio of the rms to the mean? Enter in decibels,
maximum of   52.22640     minimum of   1.050000     -


Impulse noise to be generated?
YES or NO? -

Gaussian noise to be generated?
YES or NO? -

Complex jamming signal samples to be added?
YES or NO? -

Front end filtering to be applied?
YES or NO? -
```

```
The avaliable filters types:
IIR = recursive
FIR = nonrecursive
Filter type? -

The number of filter sections? - maximum of      8 -

The type of recursive filter?
BUT = butterworth filter
CHB = chebyshev filter
RES = resonant filter
SPC = special filter (user enters the coefficients)
-

The filter cutoff frequency?
Specify in terms of the data rate -


Is the last section of single or double order
DOU = double, SIN = single
-

Type of filter gain?
ZPF - unity gain at zero frequency
IPG - unity integrated power gain
-

Front end decimation to be applied?
YES or NO? -

Is a demodulator to be used?
YES or NO? -

Differential phase shift keying demodulation to be done?
YES or NO? -
```

```
Independent synchronization to be used?
YES or NO? -

Is an excision filter to be used?
YES or NO? -

Wide band automatic gain control to be used?
YES or NO? -

Bandwidth reducer filtering to be applied?
YES or NO? -

Bandwidth reducer decimation to be applied?
YES or NO? -

Narrow band automatic gain control to be used?
YES or NO? -

Delays in the receiver mean delays of reference
signals, not input signals.
Enter a propogation delay in terms of samples -
Enter a receiver front end delay in terms of samples -
Enter a receiver bandwidth reducer delay in terms of samples -


Out of    20970 total samples        0 have already been processed
Are all the remaining input samples to be processed?
YES or NO -

Is the display feature wanted for this run?
YES or NO? -
Enter the receiver display quantity for the run
This quantity will have samples taken of it
1 = No display quantity
2 = Direct sequence reference delay
3 = Sample delay
4 = Frequency hop reference delay
```

```
5 = Narrow band AGC gain
6 = Wide band AGC gain
7 = Voltage before any narrow band AGC
8 = Matched filter lock indication
9 = Symbol synchronization lock indication
10 = Frequency hop acquisition lock indication
11 = Direct sequence acquisition lock indication
12 = Symbol synch voltage - intermediate stage
0 = Next part of the display quantity menu
-
13 = Symbol synch voltage - final stage
14 = Integrated symbol synchronization magnitude
15 = Transmission hop frequency
16 = Reference hop frequency
17 = Direct sequence tracking voltage
18 = Direct sequence tracking integrated lock voltage
19 = Demodulator tracking voltage
-
Up to    65536 samples of the display quantity can be stored
How often are samples to be taken of the display quantity?
In terms of receiver input samples -

Enter the type of display for the run.
OQU - Output of samples of the receiver display quantity
OFD - Output of final data components
HQU - Histogram of receiver display quantity samples
HFD - Histogram of final data components
BYE - Terminate display and go on
-
The number of display quantity samples to be looked at?
maximum of     3495 -
Sample number of the first sample to be looked at? -
The histogram lower limit? -
The histogram cell size? -
The number of histogram cells?
minimum of 3, maximum of     1024 -
```

```
Is a bar graph of the histogram to be produced?
YES or NO? -

Enter the type of display for the run.
OQU - Output of samples of the receiver display quantity
OFD - Output of final data components
HQU - Histogram of receiver display quantity samples
HFD - Histogram of final data components
BYE - Terminate display and go on
-

Process? -

Main level command? -

Command            Function

HELP   - Give a menu of available analysis commands
RETURN - Return to the main command level
TELL   - Type the amount of original and final data
VIEW   - Look at complex data components
HISTO  - Form a histogram of complex data components
COMPRB - Compare original and final binary data
COMPRC - Compare components of original and final complex data
ERRCOM - Analyse bit errors
FFT    - Perform a fast fourier transform or inverse transform on complex data

Analysis command? -
Number of values to be analyzed? -
Original data starting point? -
Final data starting point? -
The data window size? - maximum of    131072 -
Indices of bits in error to be output?
YES or NO? -

Analysis command? -
```

```
Main level command? -

Main level command? -
Bye for now

CLEANING UP FILES

%PURGE-I-NOFILPURG, no files purged
%PURGE-I-NOFILPURG, no files purged
Job DPSK0123 (queue SYS$PRINT, entry 1204) started on SYS$PRINT

FINISHED BATCH COMMAND FILE EXECUTION

  VENIER       job terminated at 14-JAN-1986 16:40:07.80

  Accounting information:
  Buffered I/O count:           70      Peak working set size:  2600
  Direct I/O count:            103      Peak page file size:  159944
  Page faults:               78970      Mounted volumes:           0
  Charged CPU time:     0 04:26:58.95   Elapsed time:     0 04:41:13.82
```

APPENDIX G

A GENERAL-PURPOSE COMMAND FILE FOR SUBMITTING BATCH JOBS

```
! FILE BATCH.COM       FEB-28-1984.
!
! VMS FILE TO SUBMIT BATCH JOBS. THE BATCH JOB SHOULD HAVE EXTENSION .COM. AND
! IS PASSED THE DEFAULT SUB-DIRECTORY (P2 IN BATCH.COM). IT IS UP TO THE
! BATCH JOB TO MAKE USE OF THE SUB-DIRECTORY NAME (P1 IN THE BATCH JOB COMMAND
! FILE). THE LOG FILE HAS THE SAME NAME AS THE BATCH JOB BUT WITH
! EXTENSION .LOG. PARAMETERS P1, P2, AND P3 CAN BE DETERMINED BY BATCH OR
! PASSED TO BATCH.
!
! P1 = THE NAME OF THE BATCH JOB AND CONTROLLING COMMAND FILE. THE COMMAND
! FILE SHOULD HAVE EXTENSION .COM.
!
! P2 = THE DEFAULT SUB-DIRECTORY TO BE USED. (IE FREYSENG.MODEM)
! P3 = PRIORITY OF THE BATCH JOB. (IE 1)
$
$ IF P1 .EQS. "" THEN -
  INQUIRE P1 "BATCH COMMAND FILE? ASSUMED EXTENSION OF .COM"
$
$ !IF WE DON'T KNOW ALREADY FIND OUT WHAT THE BATCH COMMAND FILE IS. IT
$ !SHOULD HAVE EXTENSION .COM
$
$ IF P2 .EQS. "" THEN -
  INQUIRE P2 "DEFAULT SUB-DIRECTORY FOR BATCH JOB? (SPECIFY WITHOUT [])"
$
$ !IF WE DON'T KNOW ALREADY FIND OUT WHAT THE DEFAULT SUB-DIRECTORY IS.
$
$ IF P3 .EQS. "" THEN -
  INQUIRE P3 "PRIORITY OF BATCH JOB?"
$
$ !IF WE DON'T KNOW ALREADY FIND OUT WHAT PRIORITY THE BATCH JOB IS TO RUN AT.
$
$ SUBMIT/NOTIFY/NOPRINT/LOG_FILE=['P2']'P1'.LOG/PRIOR='P3' -
  /PARAMETER='P2' ['P2']'P1'.COM
$
$ !SUBMIT THE JOB WITH NAME 'P1'.COM. WHEN THE JOB STOPS NOTIFY THE USER.
$ !DO NOT PRINT THE LOG FILE. THE LOG FILE HAS THE SAME NAME AS THE BATCH
$ !JOB NAME ONLY WITH EXTENSION .LOG. THE PRIORIY IS GIVEN BY 'P3', WHILE
```

```
$ !THE DEFAULT SUB-DIRECTORY PASSED TO THE BATCH JOB IS 'P2'.
$
$ EXIT
```

UNCLASSFIED
Security Classification

DOCUMENT CONTROL DATA - R & D
(Security classification of title, body of abstract and indexing annotation must be entered when the overall document is classified)

1. ORIGINATING ACTIVITY
Communications Research Centre

2a. DOCUMENT SECURITY CLASSIFICATION
UNCLASSIFIED

2b. GROUP

3. DOCUMENT TITLE
USER'S GUIDE FOR THE DRL SPREAD-SPECTRUM SIMULATION FACILITY

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)
CRC Report

5. AUTHOR(S) (Last name, first name, middle initial)
G.O. Venier

6. DOCUMENT DATE

7a. TOTAL NO. OF PAGES
271

7b. NO. OF REFS
14

8a. PROJECT OR GRANT NO.
32B76 (0117C11)

8b. CONTRACT NO.

9a. ORIGINATOR'S DOCUMENT NUMBER(S)
CRC Report 1403

9b. OTHER DOCUMENT NO.(S) (Any other numbers that may be assigned this document)

10. DISTRIBUTION STATEMENT
Unlimited

11. SUPPLEMENTARY NOTES

12. SPONSORING ACTIVITY
Directorate of Maritime Combat Systems

13. ABSTRACT

This report describes, from a user's point of view, the spread-spectrum simulation facility developed in the Directorate of Radio Propagation and Systems (DRL) at the Communication Research Centre. This facility simulates, on a VAX-11/750 digital computer, the operation of complete spread-spectrum systems including transmitter, HF propagation path, interference, and receiver, and provides all the data and signal generation and analysis capabilities necessary to determine the performance of the simulated systems. Both direct-sequence and frequency-hopping systems may be simulated. The simulator was made as flexible as possible, permitting the user to select from a number of subsystems to simulate a wide range of existing and proposed communication systems. This report provides detailed information on the simulator that is essential for understanding it and operating it in both interactive and batch modes.

DSIS
77-065

UNCLASSIFIED
Security Classification

KEY WORDS

Simulation
Spread Spectrum
Digital Computer
High Frequency
Communications
Direct Sequence
Frequency Hopping
Multipath
Excision
Non-Gaussion Noise

INSTRUCTIONS

1. ORIGINATING ACTIVITY: Enter the name and address of the organization issuing the document.

2a. DOCUMENT SECURITY CLASSIFICATION: Enter the overall security classification of the document including special warning terms whenever applicable.

2b. GROUP: Enter security reclassification group number. The three groups are defined in Appendix 'M' of the DRB Security Regulations.

3. DOCUMENT TITLE: Enter the complete document title in all capital letters. Titles in all cases should be unclassified. If a sufficiently descriptive title cannot be selected without classification, show title classification with the usual one-capital-letter abbreviation in parentheses immediately following the title.

4. DESCRIPTIVE NOTES: Enter the category of document, e.g. technical report, technical note or technical letter. If appropriate, enter the type of document, e.g. interim, progress, summery, annual or final. Give the inclusive detes when a specific reporting period is covered.

5. AUTHOR(S): Enter the name(s) of author(s) as shown on or in the document. Enter last name, first name, middle initial. If military, show rank. The name of the principal author is an absolute minimum requirement.

6. DOCUMENT DATE: Enter the date (month, year) of Establishment approval for publication of the document.

7a. TOTAL NUMBER OF PAGES: The total page count should follow normal pagination procedures, i.e., enter the number of pages containing information.

7b. NUMBER OF REFERENCES: Enter the total number of references cited in the document.

8a. PROJECT OR GRANT NUMBER: If appropriate, enter the applicable research and development project or grant number under which the document was written.

8b. CONTRACT NUMBER: If appropriate, enter the applicable number under which the document was written.

9a. ORIGINATOR'S DOCUMENT NUMBER(S): Enter the official document number by which the document will be identified and controlled by the originating activity. This number must be unique to this document.

9b. OTHER DOCUMENT NUMBER(S): If the document has been assigned any other document numbers (either by the originator or by the sponsor), also enter this number(s).

10. DISTRIBUTION STATEMENT: Enter any limitations on further dissemination of the document, other than those imposed by security classification, using standard statements such as:

    (1) "Qualified requesters may obtain copies of this document from their defence documentation center."

    (2) "Announcement and dissemination of this document is not authorized without prior approval from originating activity."

11. SUPPLEMENTARY NOTES: Use for additional explanatory notes.

12. SPONSORING ACTIVITY: Enter the name of the departmental project office or laboratory sponsoring the research and development. Include address.

13. ABSTRACT: Enter an abstract giving a brief and factual summary of the document, even though it may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall end with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (TS), (S), (C), (R), or (U).

    The length of the abstract should be limited to 20 single-spaced standard typewritten lines; 7½ inches long.

14. KEY WORDS: Key words are technically meaningful terms or short phrases that characterize a document and could be helpful in cataloging the document. Key words should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location, may be used as key words but will be followed by an indication of technical context.